INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




Desenvolvimento de um Kit Formativo para a Aprendizagem de Robótica Colaborativa

MARGARIDA SOFIA DE SOUSA PINTO
setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Desenvolvimento de um Kit Formativo para a Aprendizagem de Robótica Colaborativa

Margarida Sofia de Sousa Pinto

Dissertação para obtenção do Grau de Mestre em Engenharia Mecânica, Área de Especialização em Construções Mecânicas

Orientador: Professor Doutor Adriano A. Santos

Co-orientador: Professor Doutor Filipe Alexandre Pereira

Júri:

Presidente:

Hernâni Miguel Reis Lopes, Professor Coordenador, ISEP

Vogais:

Eurico Augusto Rodrigues Seabra, Professor Auxiliar, UMinho

Adriano Manuel da Almeida Santos, Professor Adjunto , ISEP

Porto, setembro 2025

Agradecimentos

Gostaria de agradecer a todos os que acompanharam o processo de realização desta etapa.

Agradeço aos meus pais, Nelson e Rosa, por me acompanharem, apoiarem e acreditarem em mim e por me terem incutido todos os valores fundamentais de ética de trabalho e social. No meio deste turbilhão de emoções agradeço por tudo o que fizeram e fazem por mim.

Agradeço à minha irmã, Daniela, por me ter acompanhado e por ter tido paciência comigo quando não estava nos meus melhores dias. Obrigado por fazeres a minha vida mais feliz. Amo-te.

Agradeço ao meu namorado, Diogo, que me apoiou, aturou e ajudou durante todo o processo. Obrigado por estares sempre ao meu lado, a tentar fazer-me ver o lado positivo das coisas, obrigado por todos os momentos em que tornaste a minha vida mais bonita. Sem ti esta jornada teria sido bem mais difícil. Amo-te muito.

À restante família, em especial os meus avós, Manuel, Alzira, Joaquim e Agostinha, e tia, Fatinha, por me apoiarem, terem acreditado em mim e por me terem transmitido palavras encorajadoras ao longo deste percurso.

À minha colega, Inês, que se tornou numa amiga que quero levar para a vida. Obrigado pelo companheirismo e por me acompanhares no desenvolvimento, construção e montagem das estruturas. A tua companhia tornou os meus dias muito mais sorridentes. Obrigado por tudo.

Ao meu orientador, Engenheiro Adriano Santos, por me ter ajudado e orientado durante todo o processo, por me proporcionar oportunidades de crescimento pessoal e profissional e por todas as conversas que ajudaram a controlar a ansiedade em momentos críticos. Muito obrigado.

Ao meu co-orientador, Engenheiro Filipe Pereira, por ter disponibilizado o seu tempo para me orientar, ajudar e esclarecer dúvidas. A sua ajuda foi importante para a realização deste trabalho.

Resumo

A robótica colaborativa tem vindo a assumir um papel de destaque no panorama industrial, ao permitir a interação segura entre humanos e robôs em estações de trabalho partilhadas. Esta evolução tecnológica, alinhada com os princípios das Indústrias 4.0 e 5.0, exige profissionais que apresentem competências técnicas e pessoais adequadas às necessidades atuais. Para responder a este desafio, torna-se essencial desenvolver ferramentas didáticas que aproximem o ensino académico da realidade industrial.

O objetivo desta dissertação consistiu na conceção, implementação e validação de uma bancada didática baseada no robô colaborativo UR3e, orientada para o ensino de robótica colaborativa, em contexto académico. A metodologia aplicada integrou a revisão bibliográfica, a definição dos fundamentos teóricos, a modelação e construção da estrutura, a integração dos equipamentos, o desenvolvimento de exercícios com níveis de dificuldade progressivos e a elaboração de material de apoio.

A discussão dos resultados evidenciou que a qualidade das instruções fornecidas, a modularidade da bancada e a progressividade do nível de dificuldade dos exercícios constituem fatores determinantes para a eficácia pedagógica da solução proposta. A validação experimental com estudantes e docentes confirmou a utilidade da bancada e evidenciou benefícios no desenvolvimento de competências técnicas, nomeadamente na consolidação de conhecimentos de programação e automação. No domínio das competências pessoais, verificou-se um contributo significativo no reforço da autonomia, da confiança e da capacidade de resolução de problemas.

Os resultados obtidos permitem concluir que a bancada didática desenvolvida representa uma estrutura funcional, replicável e alinhada com as exigências das Indústrias 4.0 e 5.0. Apesar da ausência de acesso remoto e da integração restrita de tecnologias complementares, a bancada permitiu reforçar a ligação entre a prática laboratorial e a realidade industrial. Deste modo, a dissertação constitui uma base sólida para evoluções futuras e contribui significativamente para a inovação pedagógica no ensino de robótica colaborativa.

Palavras-chave: Robótica colaborativa; Ensino; Bancada didática; Automação; Indústria 5.0.

Abstract

Collaborative robotics has been assuming a major role in the industrial landscape, allowing safe interaction between humans and robots in shared workstations. This technological evolution, aligned with the principles of Industries 4.0 and 5.0, requires professionals with technical and personal skills adapted to current demands. To address this challenge, it is essential to develop didactic tools that bring academic education closer to industrial reality.

The objective of this dissertation was to design, implement, and validate a didactic bench based on the UR3e collaborative robot, aimed at teaching collaborative robotics in an academic context. The applied methodology included a literature review, the definition of theoretical foundations, the modeling and construction of the structure, the integration of equipment, the development of exercises with progressive levels of difficulty, and the preparation of supporting materials.

The discussion of the results highlighted that the quality of the provided instructions, the modularity of the workstation, and the progressive difficulty of the exercises are key factors for the pedagogical effectiveness of the proposed solution. The experimental validation with students and teachers confirmed the usefulness of the workstation and demonstrated benefits in the development of technical skills, particularly in the consolidation of programming and automation knowledge. In the domain of personal skills, a significant contribution was observed in strengthening autonomy, confidence, and problem-solving ability.

The results obtained allow us to conclude that the teaching bench developed represents a functional structure that is replicable and aligned with the requirements of Industries 4.0 and 5.0. Despite the absence of remote access and the limited integration of complementary technologies, the didactic bench reinforced the connection between laboratory practice and industrial reality. Thus, this dissertation provides a solid basis for future developments and makes a significant contribution to pedagogical innovation in the teaching of collaborative robotics.

KEYWORDS: Collaborative Robotics; Teaching; Didactic Bench; Automation; Industry 5.0.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

CAD	*Computer Aided Design*
CHR	Colaboração Homem-Robô
CRM	Célula Robótica Multifuncional
DT	*Digital Twin*
EUA	Estados Unidos da América
GDA	Gémeo Digital Aumentado
GDL	Gruas de Liberdade
HMI	Interface Homem-Máquina
IA	Inteligência Artificial
IoT	Internet das Coisas
ISEP	Instituto Superior de Engenharia do Porto
ISO	*International Organization of Standardization*
I4.0	Indústria 4.0
I5.0	Indústria 5.0
M/D/S	Montagem/Desmontagem/Substituição
ML	*Machine Learning*
OCR	*Optical Character Recognition*
PLC	*Programmable Logic Controller*
PMEs	Pequenas e Médias Empresas
P.Porto	Instituto Politécnico do Porto
RC	Robô Colaborativo
RTU	*Remote Terminal Unit*
SMSD	Sistemas de Fabrico Inteligente
TCP/IP	*Transport Control Protocol/Internet Protocol*
UDP	*User Datagram Protocol*
UE	União Europeia
USB	*Universal Serial Port*
VPN	Rede Privada Virtual
VR	Realidade Virtual

I/O	Entrada e saída
TCP	*Tool Center Point*

Lista de Símbolos

α	Alfa de Cronbach	-
X_C	Coordenada em x do centro do círculo	mm
Y_C	Coordenada em y do centro do círculo	mm
K	Número de perguntas	-
R	Raio	mm
σ_X^2	Soma das respostas	-
σ_{Yi}^2	Variância da soma das respostas a cada pergunta	-

1. Introdução

Neste capítulo é apresentado o enquadramento geral, os objetivos, a metodologia de desenvolvimento e a estrutura da dissertação.

1.1. Enquadramento geral

A automação ocupa atualmente um papel central na indústria, com impacto direto na produtividade, na qualidade e na segurança dos processos. A utilização crescente de sistemas automatizados, robotizados e programáveis permite reduzir o tempo de fabrico, aumentar a eficiência global e libertar os operadores de tarefas repetitivas, perigosas ou fisicamente exigentes. Estes avanços tecnológicos, apesar de apresentarem claros benefícios para a produção, criam novos desafios ao nível da formação e requerem profissionais cada vez mais qualificados e preparados para trabalhar com tecnologias complexas e em constante evolução.

Neste contexto, a robótica colaborativa é responsável por transformar o panorama industrial ao possibilitar a interação segura entre humanos e robôs em estações de trabalho partilhadas. O UR3/UR3e constitui um exemplo representativo desta tendência, pois é utilizado em operações de montagem, inspeção e manipulação de materiais, e demonstra ganhos significativos em termos de flexibilidade e eficiência. Esta transformação tecnológica implica, contudo, que a formação de futuros engenheiros não se limite ao domínio técnico das ferramentas, mas inclua também a capacidade de trabalhar em equipa, pensar de forma crítica e resolver problemas em cenários colaborativos.

Os kits e bancadas didáticas de aprendizagem de robótica colaborativa assumem um papel determinante na formação de competências práticas e técnicas dos estudantes. Além de proporcionarem contacto direto com tecnologias utilizadas na indústria, estas ferramentas pedagógicas permitem o desenvolvimento de competências transversais como autonomia, criatividade, inovação e resiliência, que se revelam essenciais num ambiente de Indústria 5.0. A realização de atividades práticas que simulam situações reais aproxima o ensino académico da realidade industrial e prepara os alunos para responder a exigências que combinam domínio tecnológico, bem-estar humano e sustentabilidade.

De modo a contribuir para a formação técnica e pessoal dos alunos e consolidar a colaboração entre humanos e máquinas, torna-se necessário desenvolver soluções práticas que permitam aproximar o ambiente académico ao industrial. O desenvolvimento da bancada didática que incorpora o robô colaborativo UR3e visa responder às necessidades atuais do ensino da robótica colaborativa.

1.2. Objetivos

O trabalho desenvolvido tem como objetivo principal desenvolver, implementar e validar uma bancada didática de robótica colaborativa, orientada para o ensino em contexto laboratorial.

De forma mais detalhada, pretende-se:

- Realizar a revisão bibliográfica sobre o tema da dissertação;
- Estabelecer os fundamentos teóricos sobre os kits de aprendizagem de robótica colaborativa e tecnologias integrantes;
- Projetar uma estrutura modular, flexível, ergonómica e segura, adequada a atividades didáticas;
- Integrar o robô colaborativo com sistemas de entradas e saídas digitais e analógicas e garantir a sua operacionalidade em diferentes cenários;
- Integrar tecnologias de acesso remoto para utilização da bancada à distância, aumentando a flexibilidade e a acessibilidade do sistema;
- Integrar sistema de visão,
- Desenvolver um conjunto de exercícios práticos organizados de forma progressiva, de modo a apoiar a consolidação de competências em programação e operação de robôs colaborativos;
- Elaborar materiais de apoio, nomeadamente um guião laboratorial, que facilite a execução autónoma das atividades pelos estudantes;
- Validar a solução proposta através da sua aplicação real em sessões laboratoriais e da recolha de *feedback* dos utilizadores.

A concretização destes objetivos assegura a criação de uma estrutura didática funcional e a disponibilização de uma ferramenta pedagógica capaz de reproduzir, em contexto académico, cenários próximos da realidade industrial, alinhada com os princípios da Indústria 5.0.

1.3. Metodologia de desenvolvimento

A investigação seguiu uma abordagem prática e experimental, estruturada em fases sucessivas: revisão bibliográfica, desenvolvimento da bancada, construção e integração de componentes, desenvolvimento de exercícios, validação com utilizadores e análise crítica dos resultados.

O trabalho foi desenvolvido no Departamento de Engenharia Mecânica do Instituto Superior de Engenharia do Porto (ISEP), no âmbito das atividades laboratoriais da área de automação e robótica colaborativa, com foco na criação de recursos didáticos inovadores para apoio ao ensino.

Na fase de validação participaram 14 utilizadores, entre estudantes e docentes, que testaram os exercícios propostos e responderam a inquéritos de avaliação. Este grupo permitiu recolher diversas perceções e avaliar a adequação pedagógica da solução.

Durante o desenvolvido do trabalho, foram utilizados o robô colaborativo UR3e, o painel de I/Os digitais e analógicos e componentes auxiliares. A modelação tridimensional foi realizada em SolidWorks, os esquemas elétricos em EPLAN e a programação no software PolyScope do UR3e, com possibilidade de extensão ao URScript. Para tratamento dos dados recorreu-se ao Microsoft Excel.

Após o levantamento dos requisitos pedagógicos e técnicos, concebeu-se a arquitetura da bancada e selecionaram-se os materiais adequados. A estrutura foi construída e integrada com o robô e os periféricos. Seguidamente, desenvolveram-se exercícios laboratoriais que foram testados em ambiente real. Numa fase final, a solução foi aplicada em sessões experimentais com utilizadores, que preencheram inquéritos estruturados.

Os dados analisados resultaram das observações feitas durante a execução dos exercícios e das respostas aos inquéritos aplicados aos participantes. Estes dados permitiram avaliar aspetos como a clareza das instruções, a adequação técnica da bancada, a ergonomia, a organização das atividades e a perceção de desenvolvimento de competências.

A metodologia adotada permitiu assegurar a coerência entre o enquadramento teórico, a conceção técnica da bancada e a validação prática. A metodologia adotada garantiu não só a construção de um protótipo funcional, como também a sua avaliação em contexto real de ensino, possibilitando a identificação de resultados, limitações e perspetivas de evolução.

1.4. Estrutura da dissertação

A presente dissertação encontra-se organizada em oito capítulos principais, estruturados de forma a permitir uma compreensão progressiva do trabalho desenvolvido, desde o enquadramento teórico até às conclusões finais. O Capítulo 1 corresponde à introdução. Neste, é apresentado o enquadramento e a motivação do estudo, explicitam-se os objetivos definidos e descrevem-se as opções metodológicas adotadas.

No Capítulo 2 está presente a revisão bibliográfica. Neste capítulo abordam-se os conceitos fundamentais de automação e de robótica colaborativa, assim como os princípios associados à Indústria 5.0. São igualmente analisadas soluções didáticas já existentes, tanto em contexto académico como em contexto comercial, de forma a evidenciar limitações existentes e justificar a pertinência do projeto.

No Capítulo 3 apresenta-se a metodologia aplicada no desenvolvimento do trabalho. Este capítulo inclui a caracterização do contexto da investigação, o planeamento das etapas, as ferramentas de apoio utilizadas e a forma como se operam as diferentes fases do processo.

No Capítulo 4 descreve-se o desenvolvimento e a construção da bancada. Neste, é detalhado a definição da arquitetura, a modelação tridimensional da estrutura, a seleção dos componentes e o processo de integração do robô UR3e com os restantes elementos. Este capítulo demonstra a forma como a proposta foi materializada num protótipo funcional.

No Capítulo 5 analisa-se a programação do robô colaborativo. Neste, são apresentadas as funcionalidades disponíveis no PolyScope, as instruções principais e os procedimentos utilizados para a implementação dos exercícios laboratoriais.

No Capítulo 6 reúne-se a descrição do protótipo final e dos exercícios realizados. Neste capítulo são apresentadas as funcionalidades da bancada, os testes realizados e os resultados obtidos durante a fase experimental. O capítulo evidencia a eficácia da solução e o contributo que esta representa para o processo de ensino-aprendizagem.

No Capítulo 7 discutem-se os resultados obtidos. Neste, são identificadas as limitações do sistema, analisadas as perceções dos utilizadores e avaliados os contributos do trabalho para o desenvolvimento de competências técnicas e pessoais.

No Capítulo 8 apresentam-se as conclusões e os trabalhos futuros. Neste capítulo são sintetizados os contributos mais relevantes, confirma-se o cumprimento dos objetivos iniciais e sugerem-se melhorias e evoluções capazes de reforçar a aplicabilidade da bancada em contextos académicos e industriais.

2. Revisão Bibliográfica

Neste capítulo, encontra-se a revisão bibliográfica efetuada que permite uma contextualização sobre o tema da presente dissertação. A pesquisa incidiu sobre temas como automação industrial, robótica colaborativa, kits didáticos existentes no mercado e kits didáticos desenvolvidos em instituições de ensino superior que auxiliam na aprendizagem de robótica colaborativa.

2.1. Automação Industrial

Atualmente, a automação tem um impacto significativo nos setores de produção da indústria. Neste subcapítulo são abordados os conceitos e as vantagens associados à automação industrial, a interligação entre a automação e a robótica e ainda o impacto da automação e robótica a nível nacional.

2.1.1. Conceito

A automação é geralmente definida como o processo de seguir uma sequência predeterminada de operações com pouco ou nenhum trabalho humano, através de equipamentos e dispositivos especializados que executam e controlam os processos de fabrico [1]. No seu sentido mais lato, a automação é conseguida a partir da utilização de vários dispositivos, sensores, atuadores e equipamentos capazes de observar o processo de fabrico, tomar decisões sobre as alterações a introduzir na operação e controlar todos os aspetos da mesma [1]. A automação industrial traduz-se em qualquer sistema que utilize computação, como máquinas e equipamentos programáveis, que controlam e operam de forma autónoma processos industriais, com o objetivo de aumentar a velocidade e a qualidade dos processos de fabrico [2], [3].

Automação em sistemas de produção

O sistema de produção de uma empresa apresenta duas componentes: automatizada e manual. Os elementos automatizados do sistema de produção podem ser divididos em duas categorias: (1) automatização dos sistemas de produção na fábrica e (2) informatização dos sistemas de apoio à produção. Nos sistemas de produção modernos, as duas categorias (Figura 1) estão intimamente relacionadas, uma vez que os sistemas de fabrico automatizados no chão de fábrica são geralmente implementados por sistemas informáticos [2].

Os sistemas de fabrico automatizados operam na fábrica sobre o produto físico e realizam operações como o processamento, montagem, inspeção e o manuseamento de materiais. Geralmente, estes sistemas realizam mais do que uma operação na linha de produção.

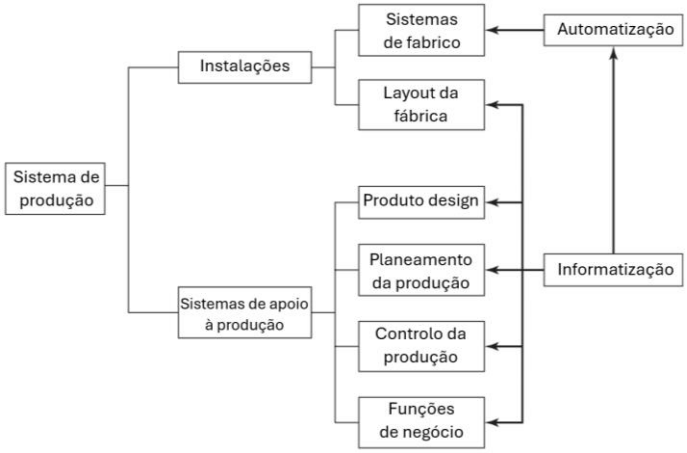


Figura 1 – Esquematização de um esquema de produção (adaptado de [2])

Além disso, são considerados automatizados porque realizam as operações com um nível reduzido de participação humana em comparação com o processo manual. Existem alguns sistemas altamente automatizados onde a ação humana é praticamente nula [2]. Os sistemas de fabrico automatizados podem ser classificados em três tipos: (1) automação fixa, (2) automação programável, e (3) automação flexível [1]. Na Figura 2, está presente um gráfico que relaciona a variedade de produtos com a quantidade produzida. Quando aplicada a automação programável, por exemplo, tem-se elevada variedade de produtos, mas quantidades de produção reduzida.

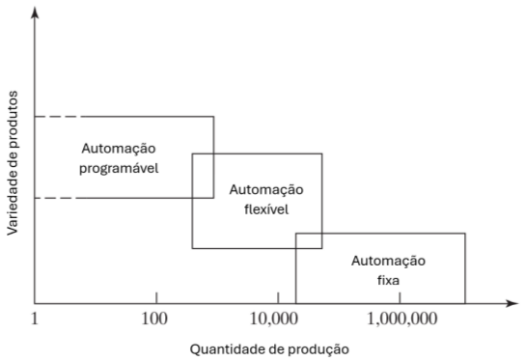


Figura 2 – Impacto dos diferentes tipos de automação, adaptado de [2]

A **automação fixa** ou rígida refere-se à utilização de equipamento para automatizar uma sequência fixa de operações de processamento ou montagem. Esta está tipicamente associada a elevadas taxas de produção e à dificuldade em realizar alterações no *design* do produto. A automação fixa aplica-se maioritariamente quando a conceção do produto é estável e apresenta um ciclo de vida contínuo e longo [1]. No caso da *GE Lighting,* que fabrica cerca de 2 mil milhões de lâmpadas por ano, é utilizada uma metodologia de automação fixa. Seguem-se as principais vantagens da aplicação de automação fixa em linhas de produção.

- Máxima eficiência, e por isso, alta taxa de produção;

- Custo por unidade reduzido;
- Redução do desperdício na produção;

Seguidamente são apresentados inconvenientes da automação fixa:

- Investimento inicial elevado;
- Inflexibilidade na variedade de produtos a produzir.

Na **automação programável**, o equipamento é concebido com a capacidade de alterar a sequência de operações, para se adaptar a diferentes configurações de produto. É particularmente adequado para a "produção em lote" de baixo e médio volume [2]. Um exemplo de aplicação deste tipo de automação é o torno CNC que produz um produto específico numa certa classe de produtos de acordo com o programa de entrada. Neste tipo de automação, a reconfiguração do sistema para um novo produto é demorada porque envolve a reprogramação e a configuração das máquinas, bem como dos dispositivos e ferramentas [1]. Seguem-se as principais vantagens deste tipo de automação:

- Flexibilidade para lidar com variações e alterações na configuração do produto;
- Preço reduzido por unidade na produção de lotes de médio ou grande volume.
- Muito adequado à produção por lotes.

De seguida, são expostas desvantagens da automação programável:

- Um novo produto requer um longo tempo de preparação e reconfiguração dos equipamentos;
- Taxas de produção mais baixas do que a automação fixa;
- Custo elevado quando comparado com a automação fixa.

A **automação flexível** é uma extensão da automação programável. Na automação flexível, o equipamento é concebido para fabricar uma variedade de produtos ou peças praticamente sem perda de tempo na mudança do *design*. É possível incorporar rapidamente alterações no produto ou introduzir rapidamente uma nova linha de produtos. Assim, não existe perda de tempo de produção durante a reprogramação do sistema. A automação flexível proporciona ao fabricante a capacidade de produzir vários produtos de forma mais económica. Por exemplo, a Honda é amplamente reconhecida por utilizar a tecnologia de automação flexível para introduzir 11 alterações na sua linha de motociclos na década de 1970 [1]. O que tona este tipo de automação possível é o facto de as diferenças entre as peças processadas pelo sistema não serem significativas, pelo que a mudança do *design* é mínima [2]. Seguem-se as principais vantagens da automação flexível:

- Produção contínua de várias configurações de peças ou produtos;
- Flexibilidade em lidar com alterações no design dos produtos.
- Produtos personalizados.

As principais desvantagens deste sistema são expostas de seguida:

- Grande investimento inicial;
- Custo de produção elevado em comparação com os outros dois tipos de automação.

2.1.2. Vantagens

As empresas de produção de praticamente todos os setores estão a obter aumentos de produtividade significativos quando tiram proveito das tecnologias de automação nos processos de fabrico [4]. O seu principal objetivo é aumentar a eficiência e a segurança nos processos de fabrico, além de melhorar a qualidade dos produtos e reduzir custos. Seguem-se as principais vantagens da automação [1], [2]:

- Aumento da produtividade do trabalho, mais produção por hora;
- Redução dos custos de produção através da redução da mão-de-obra, economia de material e energia. As máquinas estão cada vez mais a substituir o trabalho humano para reduzir o custo unitário do produto.
- Minimização da fadiga humana;
- Maior flexibilidade: capacidade de rápida adaptação à produção de novos produtos;
- Aumento da qualidade e uniformidade dos produtos;
- Maior segurança, logo melhoria das condições de trabalho: eliminação de trabalhos perigosos e penosos;
- Reduz o tempo de fabrico;
- Integração: obtenção de sistemas de produção onde todos os sectores funcionais estão interligados;
- Realização de tarefas impossíveis de efetuar manualmente ou intelectualmente. Processos que requerem montagem de peças de reduzidas dimensões, precisão, realização de tarefas de alta complexidade, coordenação e velocidade são exemplos de processos que não podem ser realizados manualmente.

2.1.3. Automação VS Robótica

A palavra robô, segundo Gupta et al. [1] refere-se a um manipulador multifuncional automatizado que funciona através de energia, para executar uma variedade de tarefas. A automação está intrinsecamente interligada e presente na robótica. O primeiro braço robótico, o *Unimated*, foi desenvolvido pelo americano *George Devol*, em colaboração com *Joseph Engelberger*, que é conhecido como o "pai dos robôs" [5]. Este robô foi usado numa fábrica da *General Motors* (indústria automóvel) para as operações de fabrico em 1960 e não só tinha um computador digital primitivo como cérebro, mas também utilizava sensores de movimento e motores de corrente contínua como atuadores [6].

A aplicação prática dos robôs teve início entre 1950 e 1960 com o desenvolvimento dos dispositivos programáveis para fábricas e linhas de montagem com automação flexível. Um exemplo da aplicação da robótica na automação flexível, são os robôs de soldadura utilizados em fábricas automóveis (Figura 3). Nestas, o modelo de veículo fabricado pode ser alterado de forma rápida e fácil, através da alteração do programa [7]. No entanto, a reconfiguração dos equipamentos para um novo produto é demorada e complexa. A robótica no processo da automação tem sido a chave das fábricas inteligentes, com os robôs a assumirem tarefas repetitivas e/ou perigosas como a montagem, soldadura e manuseamento de materiais [8]. A

tecnologia associada à robótica permite executar tarefas repetitivas de forma mais rápida e com maior grau de exatidão e precisão, o que reduz os defeitos e, por isso, melhora a qualidade dos produtos [9]. A automatização de processos robóticos melhora a eficiência do sistema, reduz os erros, diminui o tempo de inatividade do equipamento e aumenta a segurança dos trabalhadores nas operações de fabrico. Como vantagem adicional, os robôs podem trabalhar 24 horas por dia, sete dias por semana, o que aumenta significativamente a produtividade [9].

Figura 3 – Exemplo de uma linha de produção automóvel [10]

À medida que a tecnologia da robótica evolui, nomeadamente a Inteligência Artificial (IA), robótica móvel e visão artificial, novos processos são descobertos e novas soluções são adaptadas em diversos setores da indústria. A robótica desempenha um papel decisivo nas empresas modernas, o que reforça a sua competitividade e inovação [5] [4]. Atualmente, com a subsequente investigação, desenvolvimento e reforço curricular em Engenharia e Ciências da Computação, a Robótica Inteligente encontra-se presente em diversas aplicações práticas na indústria, na medicina, no setor doméstico, no setor dos serviços e na sociedade em geral [5].

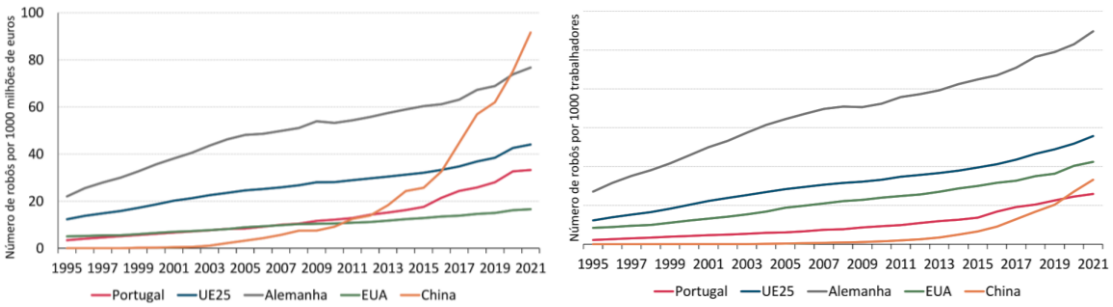
2.1.4. Automação e Robótica a Nível Nacional

O setor industrial é muito importante para a economia de cada país e continua a ser o motor do crescimento e da empregabilidade. A indústria, que neste contexto se centra no fabrico, proporciona valor acrescentado através da transformação de materiais [11].

No estudo realizado por Amador [12], analisou-se e descreveu-se a experiência portuguesa com robôs numa perspetiva agregada e temporal. Além disso, comparou-se a realidade nacional com a dos países da União Europeia (UE), EUA e China. Para comparar internacionalmente, o autor normalizou o número de robôs relativamente a uma medida de dimensão do país.

Na Figura 4 (a) está representado o número de robôs por mil milhões de euros de PIB gerado a preços constantes desde 2015. Verificou-se que de 1995 a 2021, o rácio aumentou quase 10 vezes em Portugal e ronda os três quartos do nível médio da UE (linha UE25 – europa a 25). A China, por outro lado, apresenta um aumento muito rápido neste período e o nível de robôs no PIB é cerca de duas vezes maior do que o da UE em 2021. Na Figura 4 (b) está definido o número de robôs por cada 1000 trabalhadores. Os resultados obtidos neste caso apresentam algumas diferenças devido aos níveis distintos de produtividade de cada país. Analisando o gráfico

verifica-se um claro destaque da Alemanha, com 5,5 robôs por cada mil trabalhadores em 2021. Este valor resulta de um aumento constante desde 1995, o que caracteriza o quão automatizada é a indústria alemã. Por outro lado, em Portugal, o número de robôs por cada mil trabalhadores era de praticamente zero em 1995 e aumentou para 1,3 em 2021.



(a) (b)

Figura 4 - Robôs no mundo: (a) Robôs por unidade de PIB (preços constantes de 2015); (b) Robôs por trabalhador [12]

Na Figura 5 está presente a tabela de distribuição de robôs por setores em 2012 e 2021. De um modo geral, verifica-se uma clara dominância por parte da indústria transformadora, com cotas superiores a 80 por cento em praticamente todos os países. Em Portugal, o setor automóvel (com cota de 47,1 por cento) é o maior setor industrial a nível da utilização de robôs, seguindo-se dos setores da borracha e dos plásticos, que ocupam o segundo e o terceiro lugar, com cotas de 12,4 e 11,1 por cento em 2021, respetivamente.

	Portugal		UE		Alemanha		EUA		China	
SETOR	2012	2021	2012	2021	2012	2021	2012	2021	2012	2021
Transformadora	75,8	89,8	87,7	81,6	89,6	82,4	70,9	86,5	67,3	80,5
Alimentação e bebidas	3,7	4,3	5,2	6,2	3,7	2,8	3,4	5,7	1,2	1,7
Plástico e borracha (não-auto)	4,7	12,4	8,9	8,0	7,7	7,3	4,9	4,9	11,5	3,5
Produtos metálicos (não-auto)	22,1	11,1	8,2	8,6	5,5	6,7	5,4	2,7	2,7	4,5
Maquinaria industrial	1,9	5,2	4,6	6,3	4,3	5,5	1,0	1,6	0,7	5,9
Automóvel	36,8	47,1	45,6	39,6	53,3	48,8	39,2	41,6	33,3	26,3
Elétrico e eletrónica	2,1	1,0	3,9	4,0	4,9	4,8	12,3	15,3	10,8	30,1
Outros setores	1.9	0,9	1,3	2,3	1,4	1,4	0,3	1,2	0,3	2,0
Não especificado	22,3	9,2	11,0	16,1	9,0	16,1	28,8	12,3	32,3	17,4
Total	100	100	100	100	100	100	100	100	100	100

Figura 5 - Distribuição dos robôs na economia, por setor de atividade (2012 e 2021) [12]

2.2. Robótica Colaborativa

Neste subcapítulo são abordados temas como a definição de robô colaborativo, protocolos de comunicação e conceitos da Indústria 5.0 usados na robótica colaborativa. Faz-se ainda referência às tecnologias integradas como o sistema de visão, inteligência artificial e acesso remoto. Consta ainda um estado da arte focado na aplicação destes conceitos.

2.2.1. Robô Colaborativo

Os robôs colaborativos ou *cobots* são concebidos para trabalhar, interagir e colaborar com seres humanos num espaço partilhado, sem um sistema de proteção físico como uma vedação de segurança [13][14]. Os robôs colaborativos estão cada vez mais presentes na indústria. Este

crescimento deve-se essencialmente à possibilidade de poderem trabalhar em condições de segurança em estreita proximidade com o operador (Figura 6) [15]. Para além disso, apresentam uma programação simplificada, o que possibilita a sua fácil implementação [16].

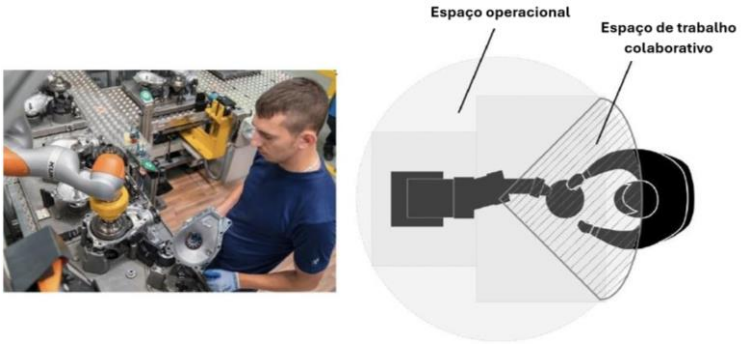


Figura 6 - Proximidade de trabalho entre o *cobot* e o operador [17]

Em consequência destes avanços, a *International Organization for Standardization* (ISO) criou a ISO/TS 15066:2016, que especifica os requisitos de segurança para robôs industriais colaborativos e o respetivo ambiente de trabalho [18].

Existem muitas áreas de investigação sobre as diferentes aplicações dos *cobots* na indústria (Figura 7). A forma como estes robôs e o seu trabalho estão a ser aplicados varia com o aumento da complexidade dos processos de fabrico, o que culmina em sistemas "multirobóticos" [19]. Estes sistemas permitem melhorar a eficiência dos processos produtivos, reduzir o tempo de fabrico e melhorar o acabamento e a qualidade das peças ou produtos.

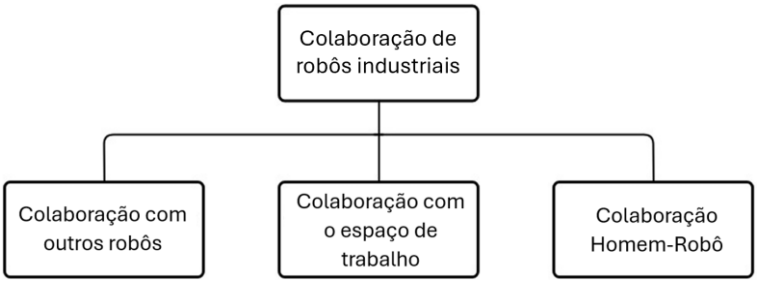


Figura 7 - Diferentes aplicações de cobots na indústria [19]

Para estabelecer a colaboração entre robôs e a ligação entre os controladores e a plataforma *online*, é necessário aplicar sistemas de comunicação adequados [19]. Na colaboração com o espaço de trabalho, são necessários equipamentos como sensores de velocidade, força, posição e visão, ligados ao controlador para coordenar o trabalho sem que se afetem mutuamente [19].

Existem diferentes tipos de *cobots* na indústria, entre os quais se salientam o ***Baxter***, um robô amigo do ser humano que ajuda a aumentar a flexibilidade da produção (Figura 8(a)); o ***Yumi***, um robô colaborativo com dois braços utilizado para a montagem de pequenas peças (Figura 8(b)) e o ***UR3***, um robô flexível para tarefas ligeiras em estações de montagem automatizadas (Figura 8(c)).

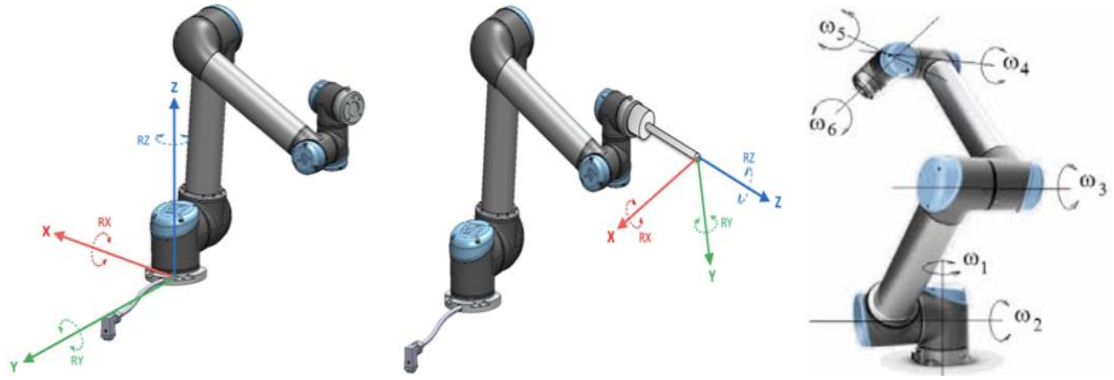
(a) (b) (c)

Figura 8 - Exemplos de robôs colaborativos: (a) Schneider Electric, Baxter [20], (b) ABB, Yumi [21]; (c) Universal Robots, UR3e [22]

Os *cobots* apresentados são soluções especiais para a colaboração homem-robô (CHR) [19]. Uma CHR bem-sucedida pode aumentar a produtividade, eficiência e qualidade de um processo de fabrico, ao mesmo tempo que reduz os custos operacionais, uma vez que a fraqueza de um pode ser compensada pelos pontos fortes do outro [23]. Para além disso, esta parceria contribui para uma resposta mais flexível às procuras do mercado. De um modo geral, os *cobots* são aptos para realizar tarefas repetitivas e monótonas que substituem ou auxiliam os operadores industriais, como a inspeção visual de peças ou produtos, aparafusamento, soldadura, entre outras (Figura 9).

Figura 9 – Exemplos de tarefas industriais realizadas por robôs colaborativos, adaptado de [24]

Os braços robóticos dos *cobots* possuem juntas rotativas, o que lhes permite ter um maior número de graus de liberdade (GDL), entre 6 e 7 graus. Na Figura 10 está presente um exemplo de um robô colaborativo de 6 GDL, o sentido de rotação de cada junta e a diferença entre o sistema de eixos da base e o sistema de eixos da ferramenta.



(a) (b) (c)

Figura 10 - Exemplo de um cobot de 6 GDL: (a) representação do sistema de eixos e rotação da base; (b) representação do sistema de eixos e rotação da ferramenta; (c) orientação da rotação de cada junta [25] [26]

Exemplos de utilização e aplicação de *cobots* na indústria

Os *cobots* estão a ser implementados em diversas indústrias destacando-se as seguintes:

1. **Indústria automóvel** – Empresas como a *Nissan* implementam robôs colaborativos nos ambientes de produção para realizarem tarefas como aparafusamento, montagem, manipulação de materiais e controlo de qualidade [27].
2. **Indústria alimentar** – Os *cobots* estão a ser utilizados no embalamento de produtos alimentares com o objetivo de garantir a qualidade e a higiene dos mesmos [28].
3. **Indústria farmacêutica** – Os *cobots* realizam a inspeção de frascos e o embalamento de medicamentos, o que aumenta a eficiência e minimiza os erros humanos [29].
4. **Indústria de logística e armazenamento** – Os *cobots* são usados para transportar produtos dentro de armazéns, como caixas ou paletes. Neste contexto, a integração de robôs colaborativos facilita o processo de armazenamento e melhora a eficiência logística, como é descrito pela *Amazon Robotics* [30].

2.2.2. Protocolos de comunicação usados na robótica

Os protocolos de comunicação são conjuntos de normas e regras que definem a forma como os dados são transmitidos e recebidos entre diferentes dispositivos numa rede [31]. Estes consistem na "linguagem" comum que diferentes dispositivos usam para comunicarem entre si de forma precisa e eficiente. Sem estes protocolos, a comunicação entre dispositivos de diferentes fabricantes e com diferentes finalidades seria difícil ou até mesmo impossível [32].

Os principais elementos do protocolo de comunicação são [31]:

- **Transmissor** – Dispositivo que envia a informação;
- **Recetor** – Dispositivo que recebe a informação;
- **Meio** – Caminho físico ou lógico por onde a informação é transmitida (por exemplo: cabos, ondas de rádio, entre outros);

- **Regras de transmissão** – Conjunto de normas que definem quando e como os dados são enviados, incluindo códigos de erro, formatos de mensagem e procedimentos de correção.

Atualmente a Ethernet é uma das tecnologias de comunicação mais importantes no setor industrial. Por esse motivo, os protocolos de comunicação industrial baseados em Ethernet são cada vez mais utilizados na automação industrial [33]. Através desta tecnologia, os *cobots* podem trabalhar em conjunto com os utilizadores para realizar o seu trabalho mesmo sem ligação à rede. No entanto, quando é estabelecida uma comunicação, o trabalho colaborativo pode ser realizado através de dispositivos eletrónicos (como telemóveis, computadores portáteis e tablets) comandados por um *software* e por um canal de transmissão de dados [19]. Assim, os *cobots* comunicam e transmitem informações para dispositivos e redes numa condição industrial específica, o que torna o trabalho colaborativo o mais simples possível para o utilizador (Figura 11) [19].

Figura 11 - Interface de comunicação de um robô colaborativo [34]

Existem diversos protocolos de comunicação utilizados na indústria. No entanto, são destacados três: Modbus, Profinet e EtherNet IP.

- **MODBUS**

O Modbus é um dos protocolos de comunicação mais antigo e utilizado na automação industrial. Foi desenvolvido pela *Modicon* (atualmente *Schneider Eletric*) em 1979 para utilização com PLCs (do inglês *Programmable Logic Controller* – Controlador lógico programável). É conhecido pela sua simplicidade e eficácia [33]. Este protocolo fornece soluções de comunicação em série e em Ethernet. O Modbus RTU (*Remote Terminal Unit*) é utilizado para uma comunicação em série, ao passo que o Modbus TCP/IP (*Transport Control Protocol/ Internet Protocol*) é utilizado para a comunicação Ethernet [33].

- **PROFINET**

O protocolo PROFINET foi concebido para facilitar a integração e comunicação eficiente, e viabilizar a troca de dados em tempo real entre os dispositivos do chão de fábrica com os setores administrativos. É baseado na comunicação padrão Ethernet e adota a arquitetura TCP/IP [35]. A adoção deste protocolo acarta diversos benefícios que incluem maior flexibilidade, eficiência operacional, interoperabilidade e redução de custos. Além disso, a compatibilidade entre a otimização da eficiência temporal das atividades com a elevada

produtividade do sistema possibilita a integração de dispositivos distintos na mesma rede, o que culmina numa amplificação de oportunidades de interconexão e interoperabilidade [35].

- **EtherNet/IP**

O protocolo EtherNet/IP apresenta uma comunicação baseada no padrão Ethernet e permite a comunicação entre PLCs e sistemas de supervisão em tempo real. A utilização do EtherNet/IP está relacionada com a necessidade de interligar diversos níveis de cadeia de suplementos (*Supply Chain*) a uma rede padrão TCP/IP. Esta rede utiliza os protocolos TCP para a transmissão de dados explícitos (não é em tempo real), garantindo a ordem e a integridade dos pacotes. No entanto, para a transmissão de dados implícitos (em tempo real) utiliza o protocolo UDP (*User Datagram Protocol*), que proporciona maior velocidade de comunicação [36].

2.2.3. Conceitos da indústria 5.0 na robótica colaborativa

A União Europeia [37] define Indústria 5.0 (I5.0) como uma indústria centralizada no ser humano, sustentável e resiliente (Figura 12), na qual o fabrico inteligente tem um peso menor que a colaboração eficaz entre o homem e a máquina. Os autores defendem que uma indústria puramente focalizada no lucro não considera corretamente os custos e benefícios ambientais e sociais. Para que a indústria se torne o fornecedor da verdadeira prosperidade, é necessário equilibrar as necessidades atuais e futuras dos trabalhadores e da sociedade com a otimização sustentável do consumo de energia, do processamento de materiais e dos ciclos de vida dos produtos [37]. Segundo Barata et al. [38], a prioridade mais recente da I5.0 é a implementação de estratégias de fabrico circular apoiadas na digitalização amiga do ser humano, capaz de antecipar e agir proactivamente sobre os impactos.

Figura 12 - Pilares da I5.0, adaptado de [37]

O conceito de I5.0 aplicado à robótica colaborativa refere-se à integração mais profunda entre os seres humanos e as máquinas, com foco na promoção da CHR. Diferente da Indústria 4.0 (I4.0), que prioriza a automação e a conectividade digital, a I5.0 coloca o ser humano no centro dos processos industriais, destacando valores como o bem-estar no ambiente de trabalho. A melhoria da segurança e do bem-estar dos funcionários, juntamente com o aumento da rentabilidade e da produtividade, são os principais objetivos da CHR.

Existem inúmeras investigações no âmbito da CHR na indústria, que visam melhorar a ergonomia física e cognitiva, através da minimização do desconforto psicológico dos operadores, provocado pela partilha do espaço de trabalho com robôs. Neste contexto, Proia et al. [39] realizaram uma pesquisa pormenorizada sobre os principais métodos de controlo que têm como objetivo central a robótica no contexto da produção com foco na segurança, ergonomia e eficiência (Figura 13). Os autores concluíram que uma CHR segura é obtida através do desenvolvimento de controladores que monitorizam a velocidade dos robôs, o que permite definir zonas seguras (sem contactos indesejados entre o homem e o robô). Outra solução passa por ajustes na velocidade de proximidade do robô, limitando a potência e a força do movimento do mesmo. Os problemas de otimização do planeamento da trajetória são resolvidos para reagir a mudanças súbitas no ambiente e para evitar lesões no operador. Por último, a melhoria ergonómica da CHR ocorre com a perfeita atribuição das tarefas e programação das atividades.

Figura 13 - Pilares da CHR, adaptado de [39]

2.2.4. Tecnologias integradas

As tecnologias integradas na robótica colaborativa desempenham um papel central pois permitem que os robôs trabalhem de forma segura e eficiente ao lado do ser humano, promovendo flexibilidade, produtividade e inovação. Alguns exemplos destas tecnologias são sistemas de visão computacional, inteligência artificial e acesso remoto.

Sistema de visão

Um robô colaborativo pode executar várias tarefas, mas necessita de condições bem definidas para funcionar sem falhas. Por exemplo, numa tarefa de recolha e colocação, o robô tem de ter a garantia que o objeto a ser recolhido está numa posição e orientação predefinidas. De acordo com Silveira et al. [40] a integração de um sistema de visão no *cobot* permite-lhe perceber a orientação, a posição e até a cor do objeto a ser recolhido. Assim, adicionar um sistema de visão a um robô oferece vantagens significativas, não só no reconhecimento do ambiente circundante [41], como também na identificação, inspeção e localização de objetos [42]. Os sistemas de visão integrados em robôs colaborativos permitem a realização de tarefas complexas, como a inspeção de qualidade e a montagem de pequenas peças, aumentando a precisão, eficiência e a segurança nos processos industriais [43].

O sistema de visão, especialmente a tecnologia de deteção e classificação de objetos, torna o robô capaz de identificar objetos, reconhecer as mudanças do ambiente de trabalho e aumentar a sua flexibilidade e adaptabilidade às incertezas [44]. Para criar os métodos de classificação baseados na aprendizagem da máquina (*machine learning* - ML*),* é necessária uma

grande quantidade de amostras para processar os resultados, extrair as características e criar um padrão de reconhecimento [45].

Por fim, a leitura de códigos 1D/2D/OCR (Figura 14) também pode ser feita pelo sistema de visão. O código 1D é conhecido como código de barras, contém dados apenas numa direção (horizontal) e é utilizado na identificação de produtos. Para ser possível identificar este código, é necessário um verificador de código de barras equipado com iluminação ótica [46]. O código 2D utiliza duas direções (horizontal e vertical) para codificar dados numa área reduzida. Existem algumas variáveis deste código, no entanto a mais conhecida no mercado é o *QR Code* [47]. Por último, o código OCR (*Optical Character Recognition*) é uma tecnologia que permite reconhecer caracteres a partir de um mapa de bits ou arquivo de imagem. Os caracteres podem ser escritos à mão, impressos, datilografados ou digitalizados. Através deste código é também possível obter um arquivo de texto editável por um computador [48].

(a) (b) (c)

Figura 14 - Imagem dos códigos: (a) 1D [46]; (b) 2D [47]; (c) OCR [48]

Inteligência artificial

Os robôs colaborativos têm surgido no mercado para dar resposta a uma variedade crescente de aplicações industriais e de serviços, sendo que a sua aplicação depende do local e do tipo de colaboração estabelecida com os seres humanos [49]. Esta evolução é impulsionada pela integração de tecnologias avançadas, como a IA e os sistemas de ML, que aumentam significativamente a capacidade de perceção e interação dos *cobots* com o ambiente em que operam, tornando-os ferramentas essenciais em ambientes industriais. A crescente utilização da IA na CHR permite a integração de algoritmos inteligentes em sistemas de robótica colaborativa para melhorar a segurança, prever falhas e otimizar tarefas [50].

A IA, em conjunto com a robótica, torna possível encontrar respostas criativas para os problemas encontrados pela generalidade das empresas. Os robôs alimentados por IA estão a ser utilizados pelas indústrias para colmatar o fosso entre os seres humanos e a tecnologia, resolver problemas e adaptar as estratégias empresariais à evolução das expectativas dos clientes. Os robôs com capacidades de IA funcionam em ambientes partilhados para manter os funcionários seguros nos locais de trabalho. Além disso, operam de forma independente para concluir operações complexas, como o corte, a retificação, a soldadura e a inspeção. A implementação da ML é fundamental para que os *cobots* aprendam e melhorem a execução das tarefas, se adaptem ao ambiente e resolvam imprevistos, baseando-se em dados em tempo real e recorrendo aos conhecimentos adquiridos através da experiência [51]. O tipo mais sofisticado de aprendizagem automática é designado por aprendizagem profunda (*deep learning*) e, utiliza redes neuronais para modelar algoritmos inspirados no funcionamento do cérebro humano. Esta aprendizagem permite a resolução de tarefas complexas e uma adaptação contínua a novas situações [50].

A Figura 15 permite visualizar que a ML é uma subcategoria da IA e a aprendizagem profunda é uma subcategoria da aprendizagem automática, o que significa que ambas são formas de IA. A IA consiste na ideia geral de que as máquinas podem executar tarefas de forma inteligente, imitando os comportamentos e os processos de pensamento humanos [50].

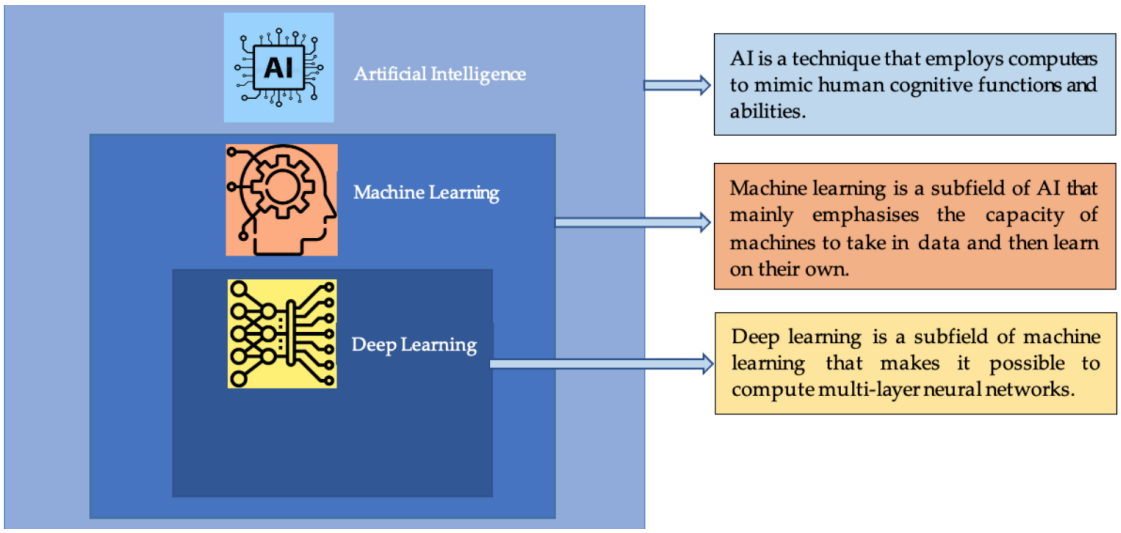


Figura 15 - Relação entre IA, *machine learning* e *deep learning* [50]

Acesso remoto

O acesso remoto é uma tecnologia imprescindível para a indústria, em particular para a robótica colaborativa. A possibilidade de aceder remotamente ao ambiente de trabalho, principalmente durante a pandemia, foi fundamental. Esta tecnologia permite que os robôs sejam controlados, monitorizados e programados à distância através de redes normalmente ligadas à internet (Figura 16)[52]. Na ligação à distância é garantida a segurança dos dados em ambas as direções.

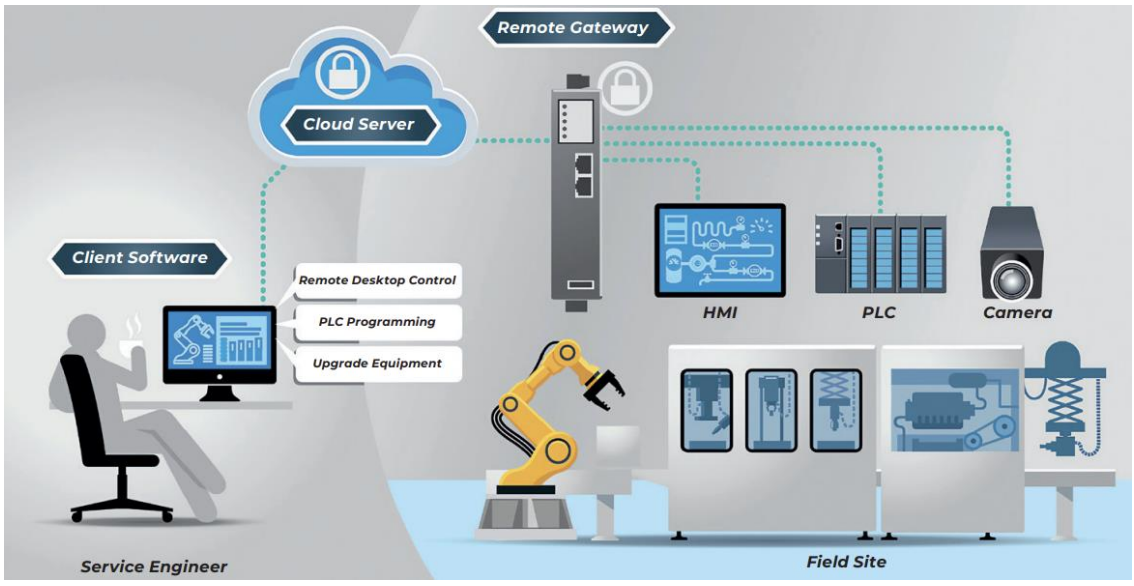


Figura 16 - Acesso Remoto [52]

A utilização adequada das ferramentas permite aceder a computadores com qualquer sistema operativo, em todas as redes, a partir de um telemóvel, computador ou tablet conectado. A forma mais comum de utilizar o acesso remoto é através de uma Rede Privada Virtual (VPN), que estabelece uma ligação direta e segura entre o computador e o servidor de destino [52].

2.2.5. Fabricantes

A indústria dos robôs colaborativos tem vindo a expandir-se ao longo dos anos. No que diz respeito aos fabricantes, só serão apresentados dois dos maiores fabricantes de robôs colaborativos do mundo. De acordo com o estudo de tendências e previsões de crescimento do mercado de robôs colaborativos [53], os cinco líderes de mercado de robôs colaborativos são a *Universal Robots*, *Fanuc*, *TechMan Robot*, *Rethink Robotics GmbH* e *AUBO Robotics USA*.

Universal Robots

A *Universal Robots* (*UR*) é uma empresa dinamarquesa fundada em 2005, por três estudantes universitários. Foi a pioneira no fabrico de robôs colaborativos comercialmente viáveis tornando-se líder mundial neste mercado. Em 2018 foi adquirida pela *Teradyne*, uma empresa de automação. Os braços robóticos desenvolvidos por eles são caracterizados por serem ferramentas avançadas, mas fáceis de utilizar, que tem impactado e ajudado as empresas e organizações a enfrentar a volatilidade do mercado. Acreditam que as soluções *cobot* da UR oferecem flexibilidade e o retorno financeiro que os fabricantes necessitam para competir e vencer em qualquer condição de mercado [54]. Na Tabela 1, estão descritos alguns *cobots* produzidos pela empresa e principais características dos mesmos.

Tabela 1 - Comparação entre UR, adaptado de [55]

					
	UR3e	**UR5e**	**UR10e**	**UR16e**	**UR20**
Alcance	500 mm	850 mm	1300 mm	900 mm	1750 mm
Carga útil	3 kgf	5kgf	12,5 kgf	16 kgf	20 kgf
Repetibilidade	+/- 0,03 mm	+/- 0,03 mm	+/- 0,05 mm	+/- 0,05 mm	+/- 0,05 mm
Peso	11,2 kgf	20,6 kgf	33,5 kgf	33,1 kgf	64kgf

TechMan ROBOT

A *TechMan Robot* (TM Robot) é uma empresa multinacional taiwanesa, fundada em 2015 que se tornou a segunda maior marca de robôs colaborativos do mundo em apenas três anos. Dedicam-se ao desenvolvimento, produção e fabrico de tecnologias de visão e robôs colaborativos. Acreditam que as aplicações tecnológicas (robóticas) corretas podem impactar de forma significativa o desempenho, a eficiência e a produtividade, e ainda criar um impacto positivo e duradouro em todas as indústrias [56]. Em 2018, formaram uma aliança estratégica

com a *Omron,* lidar mundial na indústria da automação, com o objetivo de promover a distribuição e a exposição dos robôs colaborativos da série TM nos mercados globais [57].

Na Tabela 2, encontram-se destacados alguns robôs colaborativos produzidos pela TM Robot bem como algumas características dos mesmos. Existe ainda uma gama de robôs colaborativos produzidos pela empresa que integram sistemas de IA.

Tabela 2 - Comparação entre TM [58][59]

					
	TM5-700	**TM12**	**TM14**	**TM16**	**TM20**
Alcance	746 mm	1300 mm	1100 mm	917 mm	1300 mm
Carga útil	6 kgf	12 kgf	14 kgf	16 kgf	20 kgf
Repetibilidade	+/- 0,05 mm	+/- 0,1 mm	+/- 0,1 mm	+/- 0,1 mm	+/- 0,1 mm
Peso	22,1 kgf	32,8 kgf	32,5 kgf	32 kgf	32,8 kgf

2.2.6. Trabalhos complementares

A título de resumo apresenta-se na Tabela 3 alguns estudos recentes e relevantes relacionados com os avanços e perspetivas da robótica colaborativa, a implementação das tecnologias associadas e benefícios e metodologias para a implementação de uma CHR segura e ergonómica.

Tabela 3 - Compilação de estudos relevantes relacionados com robótica colaborativa

Montini et al. [60] **(2024)**	Este artigo apresenta uma revisão exaustiva da literatura sobre as aplicações de robôs colaborativos (RC) na automação industrial, detalhando diversas operações como montagem, manuseamento de materiais, manutenção de máquinas, inspeção de qualidade e soldadura. A análise revelou que os RCs são maioritariamente utilizados para tarefas simples, com interação limitada com os seres humanos, devido à falta de confiança na CHR. Apesar destes desafios, as empresas da indústria transformadora demonstram forte interesse em melhorar o nível de colaboração para aumentar a produtividade e desempenho das linhas de produção. Este estudo aponta ainda para uma necessidade de interligar as investigações académicas com as necessidades industriais para ser possível maximizar o potencial da robótica colaborativa. Este caminho abre portas para soluções inovadoras que complementam as capacidades humanas e impulsionam a transformação industrial.

Tabela 3 - Compilação de estudos relevantes relacionados com robótica colaborativa (continuação)

Amin et al. [61] **(2020)**	Neste estudo propõe-se um sistema de segurança baseado em perceção mista para melhorar a CHR em termos de segurança e produtividade. Este sistema combina perceção visual para deteção de ações humanas no espaço de trabalho, com perceção tátil, para identificar contactos entre seres humanos e robôs. O sistema de perceção visual utiliza imagens RGB combinadas com deteção de esqueleto humano, atingindo uma precisão de 99,7% no reconhecimento de ações humanas em tempo real. A perceção tátil faz a distinção entre contactos intencionais e acidentais com uma precisão de 99% e um tempo de resposta de 80 ms. Os autores concluíram que este modelo permite que o robô detete as ações humanas no espaço de trabalho e avalie os níveis de segurança. O ajuste da sua velocidade e comportamento promovem a segurança e a produtividade da CHR. Embora o modelo inicial de perceção mista demonstre resultados promissores, existe a necessidade de aumentar a inteligência dos robôs na deteção de múltiplos contactos.
Peruzzini et al. [62] **(2023)**	Neste artigo exploraram-se as limitações da I4.0. Casos de estudo revelaram que a I4.0 subestima o papel humano nas fábricas. Apresentou-se um quadro de conceção de sistemas de fabrico inteligentes (SMSD) que introduzem a I5.0, com base na simbiose homem-automação. Para o desenvolvimento do SMSD, os autores basearam-se nos conceitos Gémeo Digital Aumentado (GDA) e Interfaces Homem-Máquina (HMI). O GDA coleta dados de máquinas, robôs, seres humanos e do ambiente para modelar e simular o comportamento do sistema, proporcionando um suporte adaptativo em tempo real. As HMIs ajustam as informações e instruções fornecidas aos operadores, o que promove uma colaboração eficaz e uma aprendizagem mútua entre os humanos e as máquinas. Os resultados demonstraram que o SMSD supera as limitações da I4.0 e permite a implementação prática da I5.0, na qual humanos e máquinas co evoluem e colaboram de forma eficaz. Além disso, a I5.0 apoia a criação de cenários industriais mais resilientes e sustentáveis, alinhados aos princípios da economia circular.
Santos et al. [63] **(2024)**	Neste estudo propõe-se um sistema robótico colaborativo para automação de tarefas Pick and Place, baseado na integração de visão artificial com câmaras de baixo custo. O sistema recorre a um robô UR3e, a uma câmara Intel D435i e ao software Niop, onde são aplicados algoritmos de pré-processamento e segmentação para reconhecer objetos com precisão, independentemente da cor, posição ou orientação. A integração de iluminação uniforme e algoritmos de ML demonstrou precisão na deteção de objetos e a fiabilidade das operações, apesar de pequenas margens de erro na localização. Os autores concluíram que a solução apresenta vantagens económicas e operacionais, com potencial de aplicação em PMEs. Além disso, o sistema mostrou potencial para ser utilizado em tarefas complexas como inspeção de qualidade e montagem de peças.

Tabela 3 - Compilação de estudos relevantes relacionados com robótica colaborativa (continuação)

Asaas et al. [64] **(2024)**	O presente artigo teve como objetivo explorar o impacto da inteligência artificial na CHR. Para tal, os autores realizaram uma análise literária sobre o tema. Após análise, concluiu-se que a IA desempenha um papel crucial na CHR pois melhora a eficiência, segurança e produtividade. Com algoritmos de ML, a IA otimiza tarefas colaborativas, deteta falhas e processa dados em tempo real. Esta permite que os *cobots* se adaptem ao comportamento humano e tomem decisões éticas, contribuindo para a transição da I5.0. Concluiu-se ainda que a IA é uma tecnologia importante para melhorar a CHR, pois garante segurança e eficiência. A IA permite que os humanos e os robôs aprendam em conjunto, o que melhora a produtividade, reduz os erros, fomenta a inovação e apoia ambientes de fabrico inteligentes.
Peterhansl [65] **(2023)**	Neste estudo propõe-se a adaptação de robôs colaborativos para operar em redes 5G, de forma a ultrapassar as dificuldades das PMEs na adoção de novas tecnologias. O sistema desenvolvido utiliza um robô UR5, óculos de realidade virtual HTC Focus 3, joystick, câmara USB e routers 5G, com comunicação em TCP/IP devido à baixa latência. Foram definidos dois modos de controlo: por óculos de realidade virtual, com transmissão direta de sinais, e por joystick e computador, com feedback visual através de vídeo. Para comparar redes privada, privada com VPN e pública, realizaram-se 12.000 medições de latência em cada uma. Os resultados indicaram que o acesso remoto via 5G reduz atrasos e aumenta a precisão, constituindo uma solução prática e acessível para PMEs. O autor recomenda redes privadas em aplicações industriais que exigem baixa latência e segurança, e redes públicas em cenários em que o custo e a rapidez de implementação assumem maior relevância.

2.3. Kits didáticos de aprendizagem de robótica colaborativa

No presente subcapítulo, faz-se uma análise aprofundada e mais focada no tema da dissertação. Faz-se referência a três kits formativos comercializados e a quatro instituições de ensino superior que desenvolveram o seu próprio kit didático de aprendizagem de robótica colaborativa. Para cada conjunto é realizada uma análise comparativa com o intuito de avaliar os pontos fortes e as limitações de cada kit. Por fim, efetua-se uma revisão sobre o impacto e a importância da inserção de kits didáticos de aprendizagem de robótica colaborativa no ensino.

2.3.1. Kits didáticos existentes no mercado

De modo a evidenciar as características distintivas de kits didáticos relevantes existentes no mercado, apresentam-se de seguida as particularidades de três kits comerciais: RTS-200, ROBOT Training Kit e Kit didático UR.

RTS-200 – Sistema de formação em robótica

O sistema RTS-200 [66] é uma estação didática (desenvolvida pela *SMC Corporation* [67]) constituída por um robô colaborativo, um sistema de *gripper*, um controlador e uma interface (Figura 17). O sistema permite realizar movimentos de paletização e manuseamento de pequenas peças [66].

Figura 17 - Sistema RTS-200 [68]

Esta bancada didática apresenta um *design* flexível que permite a configuração e escolha dos equipamentos que melhor se adaptam às necessidades do utilizador. No *website* da empresa [69], existe a opção de criar a própria bancada com uma lista de robôs colaborativos e sistemas *gripper*. É também possível incorporar equipamentos adicionais como sistemas de visão, sistemas pneumáticos, entre outros. Deste modo, o RTS-200 permite que os formandos evoluam da programação básica de robótica colaborativa para a integração de várias tecnologias, como controlo elétrico, pneumático, controladores programáveis, visão artificial e sensores. Na Tabela 4, encontra-se de forma mais concisa e detalhada tudo o que o sistema inclui e todas as aplicações básicas que ele possui.

Tabela 4 - Sistema RTS-200

RTS-200		
Inclui	• Estrutura móvel de alumínio. • Robô e controlador. **Painel de controlo do operador:** • Paragem de emergência. • Botões (iniciar/parar) e seletor.	**Pacote Pneumático Integrado:** • Compressor de ar silencioso. • Regulador e Válvula de corte. • Distribuidor de alimentação de pressão. • Luz indicadora multicolorida. • *Gripper* Pneumático SMC.
Funções	**Braço robótico:** • Movimentos simples. • Desenhar e escrever. • Recolher e colocação. • Paletização básica.	

O equipamento didático dispõe de uma série de aplicações opcionais que alargam os objetivos de aprendizagem como a aplicação pneumática e transferência de peças; sensores integrados

para deteção de material, tamanho, cor e qualidade da peça; transportador de velocidade variável com sensores; alimentação de máquinas CNC (CNC Simulado); inspeção de qualidade.

ROBOT Training Kit

A *TM Robot* [56] tem disponível um kit formativo de aprendizagem de robótica colaborativa que integra um *cobot* TM5, sistema de visão, uma câmara externa e um *gripper* (Figura 18) [70]. Este sistema permite realizar movimentos de paletização, movimentos livres do braço robótico e inspeção visual das peças de trabalho. A inspeção das peças de trabalho está associada à leitura de códigos AOI/OCR. Deste modo, e com a placa de referência incorporada no sistema (interface), este é capaz de detetar os códigos gravados nas peças, capturar imagens e verificar alguns parâmetros como a cor e a presença de defeitos.

Figura 18 – ROBOT Training kit [70]

Na Tabela 5 estão descritos os componentes que constituem o sistema *ROBOT Training Kit*, bem como as funções que podem ser realizadas.

Tabela 5 - Robot Training Kit [70]

Robot Training Kit		
Inclui	• Estrutura fixa de alumínio; • Robô TM5-900 e controlador; • Sistema de Visão; • Tapete.	
Funções	**Braço robótico:** • Orientação manual livre; • Movimentos de paletização; • Posicionamento de ponto fixo.	**Inspeção ótica automatizada (leitura de códigos):** • AOI/OCR; • 1D; • 2D.

Kit didático UR

A bancada robótica da *Grow Skills* [71](Figura 19) é constituída por um *cobot* UR3e da *UR*, um sistema de *gripper Robotiq Hand-E* e um sistema de visão artificial *Robotiq Wrist camera*. Este sistema permite manipular, montar e verificar peças de trabalho [72].

Figura 19 - Kit didático UR [72]

A incorporação do sistema de visão no *cobot* UR3e permite que o robô realize ações como montagem de peças, de acordo com o requisito do utilizador. Para além disso, o sistema de visão serve de interface de montagem mecânica e elétrica com o *cobot* e com o *gripper Hand-e*. A *Robotiq Hand-e* é um *gripper* colaborativa de 2 dedos, com controlo e *feedback* de posição, força e velocidade. Os dedos do *gripper* adaptam-se à geometria do objeto. Na Tabela 6 está presente uma descrição pormenorizada dos constituintes e aplicações do kit.

Tabela 6 - Kit didático UR

Kit didático UR		
Inclui	• Mesa em alumínio perfilado; • Robô colaborativo UR3e e controlador; • Tapete e motor; • Sensores de presença de peça no início e no fim do tapete; • Cilindro de duplo efeito; • Electroválvula; • *Gripper Robotiq hand-e*; • Sistema de visão Robotic wrist camera.	
Funções	**Braço robótico:** • Manipular, montar e verificar peças de trabalho; • Montagem de peças; • Empilhamento de peças.	**Sistema de visão:** • Leitura de códigos 1D e 2D; • Importação de ficheiros CAD; • Gravação de imagens; • Calibração automática; • Offset visual através de códigos matriciais; • Auto-Pick.

2.3.2. Kits didáticos desenvolvidos por instituições de ensino superior

A área da automação, nomeadamente a robótica colaborativa, está em constante crescimento, pelo que é necessário começar a corresponder a essa indústria a nível institucional. Algumas instituições de ensino superior dispõem de laboratórios de automação, equipados com robôs colaborativos que estão interligados a bancadas didáticas. Normalmente, estas bancadas são desenvolvidas por docentes e/ou discentes, o que potencializa a aprendizagem da engenharia

[73]. É imprescindível que estes laboratórios disponham de acesso remoto, permitindo a utilização dos equipamentos à distância e, assim, retirar máximo proveito de todas as funcionalidades.

Universidade Autónoma de Querétaro, México (2014)

Espinosa et al. [74] apresentam o desenvolvimento e a construção de uma bancada didática que incorpora um robô colaborativo e um sistema de *gripper* (Figura 20). O principal objetivo deste trabalho foi permitir que os alunos desenvolvessem um sistema mecatrónico robusto, com materiais de baixo custo. Além disso, o braço robótico deveria ter um alcance mínimo de 40 cm, uma precisão de movimento de +/- 0.5 cm, suportar cargas cujo peso deve ser pelo menos 10 % do peso próprio e ainda possuir três graus de liberdade. Quanto ao *gripper*, esta tinha de possuir um grau de liberdade e ser capaz de suportar o peso definido anteriormente.

Figura 20 - Exemplo de dois protótipos desenvolvidos [74]

Os autores consideram que a plataforma didática e a metodologia proposta podem ser utilizadas para introduzir aos estudantes os procedimentos de *design* industrial. Deste modo, e atendendo às ferramentas disponíveis, incorporaram-se diferentes processos de fabrico e arquiteturas de robôs. Para além disso, os alunos expandem o seu conhecimento em tópicos como a cinemática, a dinâmica e a mecatrónica.

Universidade Nordakademie Hochschule der Wirtschaft, Alemanha (2023)

O trabalho de Kobras et al. [75] reflete o desenvolvimento de um laboratório educativo para o ensino da tecnologia da robótica colaborativa, desde a conceção até à sua implementação. Para a conceção da bancada didática, que intitularam como CoBot, utilizaram um robô colaborativo UR5e em combinação com uma câmara de pulso e diferentes tecnologias de pinças da *Robotiq,* para simular diferentes cenários de paletização, recolha e colocação. Os alunos podem trabalhar diretamente com a consola de ensino ligada ao robô ou remotamente, através do *software* ArtiMinds RPS3. Para além disso, o sistema está equipado com uma câmara que providencia imagens em direto. Na Figura 21, observa-se a estação de trabalho do CoBot, que estava pronto a recolher os objetos colocados sobre a mesa.

Figura 21 - Estação de trabalho do CoBot [75]

Este laboratório de robótica colaborativa foi desenvolvido no âmbito do mestrado em engenharia industrial. Através de máquinas virtuais, os alunos tinham acesso a um simulador do controlador do robô, fornecido pelo fabricante. Após a simulação, o programa desenvolvido pelos estudantes é transferido para o robô e testado em ambiente físico.

Instituto Politécnico de Bragança, Portugal (2018)

Nakashima [76] apresenta o estudo e o desenvolvimento de uma bancada de trabalho de inspeção industrial automatizada de consolas HMI. O projeto apresentado nesta dissertação tem como objetivo a criação de uma bancada de trabalho padrão, incluindo os conceitos da Indústria 4.0 na inspeção da qualidade das consolas (botões, displays TFT e LCD). Para a realização da inspeção (Figura 22), é utilizado um robô colaborativo UR3, um sensor de força e binário FT300 da *Robotiq*, uma câmara industrial Mako G-125B, um Arduino Uno, uma iluminação LED e uma bancada para a instalação do sistema. Os mecanismos comunicam por *Ethernet* e *Universal Serial Port* (USB), de modo a conseguir um processo simples, permitir a expansão do sistema e a comunicação com outros dispositivos externos. Posto isto, o sistema é capaz de inspecionar quatro tipos diferentes de consolas HMI, sendo que os botões e o ecrã são inspecionados através da utilização de visão artificial e IA.

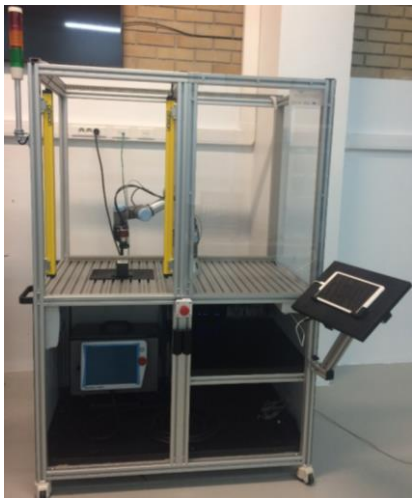


Figura 22 - Bancada de trabalho completamente montada [76]

A inspeção da qualidade é uma tarefa existente em todas as fábricas de produção. Quando esta inspeção é realizada, os produtos valorizam comercialmente. Além disso, a introdução das

tecnologias inerentes a esta nova indústria aumenta a adaptabilidade dos sistemas, a qualidade dos produtos, a eficiência do processo e simplifica a rastreabilidade dos produtos.

Universidade do Galati, Roménia (2024)

No artigo desenvolvido por Filipescu et al. [77], propõe-se a conceção e implementação da monitorização e controlo remoto ou local de um sistema mecatrónico de laboratório com base na tecnologia Internet das Coisas – nuvem (*IoT-cloud*), VPN e gémeo digital (DT – do inglês *digital twin*). O sistema mecatrónico é o de célula robótica multifuncional (CRM), equipado com um manipulador robótico industrial, ABB IRB 120, que realiza a montagem flexível de duas peças de trabalho, cada uma composta por cinco componentes (Figura 23). O sistema também pode efetuar a desmontagem e a substituição, permitindo que as peças sejam recuperadas, para reutilização ou revenda. Se a peça montada estiver completamente comprometida, é devolvida ao CRM no tapete rolante para desmontagem completa ou substituição. O ABB 120 IRM, integrado no CRM, executa tarefas complexas de montagem/desmontagem/substituição (M/D/S). O sistema adapta-se em tempo real a qualquer variação, utilizando informações baseadas em dados de modelos de IA (ML).

Figura 23 - Sistema mecatrónico desenvolvido no laboratório do Galati [77]

Os autores concluíram que a plataforma *IoT-cloud* permite o acesso remoto a dados em tempo real e a capacidades de controlo como a análise de dados para programação otimizada de tarefas. A VPN garante uma comunicação segura entre o centro de controlo e o sistema robótico, o que permite que os utilizadores autorizados acedam e controlem o sistema remotamente. Isto permite que os alunos pratiquem com segurança tarefas complexas (como montagem, desmontagem, substituição e manutenção) virtualmente antes de trabalharem na célula robótica física. Esta experiência prática alinha-se com a Educação 5.0 na preparação dos alunos para as carreiras da I4.0, integrando tecnologias avançadas na aprendizagem.

2.3.3. Comparação entre kits

A diversidade de tecnologias integradas nos diferentes kits influencia diretamente a sua relevância para o ensino de robótica colaborativa. A comparação entre os kits didáticos apresentados anteriormente revela-se particularmente relevante no contexto académico, pois permite identificar as soluções que melhor conciliam aplicabilidade, inovação tecnológica e adequação pedagógica.

Na Tabela 7 apresenta-se a comparação entre os sete kits, dos quais três correspondem a soluções comerciais e quatro foram desenvolvidos em instituições de ensino superior. A avaliação foi realizada com base num conjunto de tecnologias de referência na área da robótica colaborativa, nomeadamente a presença de sistemas de visão, o acesso remoto, a utilização de IA e a capacidade de deteção de códigos (1D, 2D e OCR).

Tabela 7 - Comparação entre as tecnologias integradas em cada kit

Kit\Tecnologia	Sistema de visão	Acesso remoto	Tecnologia IA	Deteção de código		
				1D	2D	OCR
RTS-200 [66]	Não	Não	Não	Não	Não	Não
ROBOT Training kit [70]	Sim	Sim	Não	Sim	Sim	Sim
Kit didático UR [72]	Sim	Não	Não	Sim	Sim	Não
México [74]	Não	Não	Não	Não	Não	Não
Alemanha [75]	Sim	Sim	Sim	Não	Não	Não
Portugal [76]	Sim	Não	Sim	Não	Não	Não
Roménia [77]	Sim	Sim	Sim	Não	Não	Sim

A análise da Tabela 7 evidencia diferenças significativas entre os kits. O ROBOT Training Kit, de origem comercial, é uma solução completa a nível de tecnologias integradas e que oferece a possibilidade de aquisição com IA incorporada. As soluções desenvolvidas na Alemanha e na Roménia destacam-se por integrarem um conjunto alargado de tecnologias, incluindo sistema de visão, acesso remoto e IA, o que os posiciona como soluções mais avançadas e alinhadas com os princípios da Indústria 5.0. Por outro lado, o RTS-200 e o kit desenvolvido no México não apresentam nenhuma das tecnologias avaliadas, o que revela fortes limitações para utilização em contextos de ensino avançado ou de investigação.

Como kits de nível intermédio no que respeita à quantidade de tecnologias integradas, consideraram-se o kit didático UR e o kit desenvolvido em Portugal. Estes incluem tecnologias relevantes como o sistema de visão, mas não possuem acesso remoto, o que compromete a sua utilização em ambientes de ensino à distância.

Comparação dos custos

No que respeita aos custos de aquisição, apenas os kits comerciais apresentam valores públicos. O RTS-200 constitui a opção de menor custo (34.000€), seguido pelo Kit didático UR (36.000€) e pelo ROBOT Training Kit (38.000€). Embora as diferenças de preço não sejam significativas, as disparidades nas funcionalidades evidenciam que o critério de seleção não se deve centrar exclusivamente no custo, mas sobretudo na relação custo-benefício.

Conclusão

A análise realizada reflete que o ROBOT Training Kit e os kits da Alemanha e da Roménia constituem as soluções mais completas e adequadas para o ensino e para a investigação de robótica colaborativa, dado o seu alinhamento com as exigências tecnológicas atuais. Em contrapartida, o RTS-200 e o kit desenvolvido no México revelam limitações significativas, restringindo a sua aplicabilidade a cenários básicos de aprendizagem. Os kits intermédios, apesar de úteis em determinados contextos, não oferecem um conjunto suficientemente abrangente de tecnologias para responder às necessidades da formação em robótica colaborativa avançada.

No caso dos kits comerciais, destaca-se a possibilidade de integração de tecnologias adicionais, como inteligência artificial, o que aumenta a versatilidade das soluções. Contudo, essa expansão tecnológica implica um acréscimo significativo no valor final do kit, o que pode dificultar a sua aquisição em instituições com restrições orçamentais.

Com base no que foi apresentado, conclui-se que nenhum dos kits analisados apresenta uma solução verdadeiramente completa. Esta lacuna constitui um desafio relevante para o ensino, uma vez que limita a criação de experiências práticas alinhadas com os requisitos da Indústria 5.0, evidenciando a necessidade de desenvolvimento de kits mais equilibrados, acessíveis e tecnologicamente completos.

2.3.4. Impacto dos Kits didáticos no ensino

A crescente importância da robótica colaborativa e da automação nos processos industriais, acompanhada da constante evolução tecnológica, acarreta uma procura, por parte das entidades empregadoras, de operários qualificados e familiarizados com estes conceitos. Este avanço tecnológico exige uma necessidade de aprendizagem destes conceitos em contexto académico apoiada pelos laboratórios das instituições de ensino superior. Segundo Pereira et al. [78], a presença de simuladores reais nestes laboratórios representa um auxílio essencial na perceção da realidade funcional. Poder experimentar e observar os resultados, conciliando conhecimentos teóricos e práticos, são vantagens inegáveis.

A aprendizagem através de kits didáticos proporciona maior sensibilidade, experiência e capacidade para a inovação no domínio da robótica colaborativa em contexto industrial [79]. As *soft skills* ganham maior relevância comparativamente às *hard skills*, pois, devido à natureza multidisciplinar das soluções, há uma intensificação das competências de trabalho em equipa, pensamento crítico e resolução de problemas [78]. Os kits didáticos, concebidos à semelhança

do que se encontra no mercado de trabalho, são, por isso, soluções importantes e vantajosas para esta necessidade urgente de uma formação adequada [78].

Na Tabela 8, estão presentes alguns estudos relevantes sobre a importância da aplicação de kits formativos e colaborativos no ensino, bem como os desafios que podem surgir neste contexto.

Tabela 8 – Compilação de estudos importantes relacionados com a aprendizagem através de kits colaborativos

Mercier et al. [80] **(2024)**	O artigo tem como objetivo dar resposta à necessidade de melhorar a eficácia do ensino de engenharia através de uma aprendizagem colaborativa onde os alunos trabalhem em equipas multidisciplinares que refletem as exigências do ambiente profissional. O estudo foi desenvolvido com base numa extensa revisão literária e análise de estudos publicados entre 2010 e 2022. Para categorizar os artigos, utilizou-se a abordagem "4Ts" (equipas, tarefas, ferramentas e professores) para estruturar a análise e identificar os fatores cruciais para o sucesso da aprendizagem colaborativa. Para validar as suas conclusões, utilizaram-se estudos empíricos que demonstram os efeitos positivos de intervenções em equipa. Os autores concluíram que o ensino colaborativo promove melhores resultados de aprendizagem, um maior empenho dos alunos e o desenvolvimento de competências como o trabalho em equipa e a resolução de problemas complexos, que são importantes para o ambiente profissional.
Caetano e Felgueiras [81] **(2019)**	Neste estudo, estão presentes as reflexões de um docente em relação ao uso e aplicação didática de um laboratório remoto no ensino superior. O trabalho recai num contexto educativo onde o acesso remoto é cada mais utilizado para facilitar o acesso a recursos tecnológicos e a simulação. Desta forma, é promovida a inovação no ensino, especialmente em disciplinas práticas de engenharia. O principal objetivo é examinar como a introdução de um laboratório remoto influencia o planeamento, a implementação e a evolução das estratégias pedagógicas do docente. Para tal, adotou-se uma abordagem qualitativa, centrada na autoanálise dos professores sobre as suas ações, decisões e resultados para identificar pontos de melhoria. O estudo considerou três perspetivas principais como uma revisão bibliográfica da literatura que influenciou o *design* inicial da implementação didática, o *feedback* dos alunos e as dificuldades encontradas ajudaram os docentes a refinar continuamente as suas abordagens. Os autores concluíram que a experiência com laboratórios remotos evidencia como a tecnologia pode enriquecer o processo de aprendizagem, de forma flexível e acessível. Esta abordagem exige que os professores estejam preparados para adaptar as suas metodologias e enfrentar os desafios associados à transição digital. Como benefícios observados destacam-se o estímulo à autonomia dos alunos no processo de aprendizagem e a melhoria na interação e participação dos estudantes com os conteúdos práticos.

Tabela 8 – Compilação de estudos importantes relacionados com a aprendizagem através de kits colaborativos (continuação)

Pinto et al. [82] **(2025)**	O artigo tem como objetivo avaliar a relevância dos kits de aprendizagem de robótica colaborativa como ferramentas pedagógicas fundamentais para a formação de engenheiros preparados para os desafios da Indústria 5.0. O estudo foi desenvolvido através de uma revisão literária e da comparação entre soluções comerciais disponíveis no mercado e kits desenvolvidos em instituições de ensino superior. Verificou-se que os kits comerciais se destacam pela sua versatilidade e completude tecnológica, mas apresentam custos de aquisição demasiado elevados. Já os kits académicos, apesar de constituírem alternativas mais económicas e flexíveis, revelam lacunas importantes, nomeadamente a ausência de acesso remoto, inteligência artificial e sistemas avançados de visão. A análise permitiu concluir que a integração destas tecnologias é essencial para aproximar os processos de ensino das necessidades reais da indústria, o que potencia uma aprendizagem mais prática, inovadora e eficaz. Os autores defendem que o futuro passa pelo desenvolvimento de kits mais acessíveis, mas com integração tecnológica suficiente para simular cenários industriais e promover competências como pensamento crítico, resolução de problemas complexos e trabalho em equipa. Neste sentido, reforçam a importância de alinhar a formação em engenharia com as exigências de um mercado em rápida transformação, onde a interação entre humanos e robôs assume um papel central.

3. Metodologia

Este capítulo apresenta a metodologia seguida para o desenvolvimento do projeto, incluindo a abordagem adotada, as ferramentas e tecnologias utilizadas, o planeamento das etapas de execução e a estratégia definida para testes e validação, de forma a garantir um processo estruturado e eficiente.

3.1. Abordagem adotada

A abordagem seguida neste projeto baseou-se numa metodologia prática-experimental, orientada para o desenvolvimento de uma bancada didática com integração de tecnologias de automação e robótica colaborativa. A estratégia combinou fases de conceção, montagem, integração de componentes, desenvolvimento de *software* e validação funcional, com foco na aplicação em contexto educativo.

Para a construção do protótipo foram considerados diferentes aspetos, tais como, a facilidade de replicação, modularidade da estrutura, segurança de operação e flexibilidade na execução de exercícios laboratoriais. Todas as decisões técnicas foram orientadas por critérios de aplicabilidade didática, resistência e compatibilidade entre os sistemas.

A metodologia seguiu os seguintes passos fundamentais:

- Levantamento de requisitos pedagógicos e técnicos;
- Definição da arquitetura da bancada;
- Seleção e aquisição dos componentes e equipamentos;
- Montagem da estrutura e instalação dos dispositivos;
- Integração dos sistemas de hardware e software;
- Testes e validação em ambiente real de utilização.

O processo foi iterativo, o que permitiu realizar ajustes conforme os resultados dos testes e o *feedback* recolhido junto dos utilizadores.

3.2. Ferramentas e tecnologias utilizadas

Para a execução do projeto foram utilizadas diversas ferramentas e tecnologias, tanto a nível físico como digital. No que respeita o *hardware*, a bancada inclui o robô colaborativo UR3e, o

painel de I/Os digitais e analógicos, o Teach Pendant com o *software* PolyScope integrado, o controlador dedicado, o potenciómetro e o voltímetro.

Do ponto de vista de *software*, o principal ambiente de desenvolvimento foi o PolyScope, que permitiu a programação do robô e a criação de fluxos lógicos de trabalho. Foram ainda utilizados *softwares* auxiliares como:

- SolidWorks: Modelação 3D da estrutura;
- PolyScope: Interface gráfica de programação do UR3;
- EPLAN: Elaboração dos esquemas elétricos;
- Excel: Tratamento dos dados recolhidos no inquérito realizado.

Foram também utilizados instrumentos de medição e ferramentas manuais durante a montagem da estrutura. Este processo teve como referência a documentação técnica dos fabricantes, o que garante precisão na construção e a correta instalação dos componentes. A integração entre os módulos foi viabilizada por ligações elétricas diretas ao controlador do robô, sem necessidade de dispositivos adicionais de interligação, o que simplificou o sistema e o manteve acessível para fins didáticos.

3.3. Planeamento do desenvolvimento

O planeamento do desenvolvimento foi essencial para garantir que o projeto decorresse de forma organizada, evitando contratempos e minimizando imprevistos. Ao definir antecipadamente cada etapa e os seus objetivos, foi possível otimizar recursos e assegurar a integração harmoniosa entre as diferentes fases do trabalho.

O projeto foi dividido nas seguintes etapas:

1. Definição dos requisitos;
2. Conceção da estrutura mecânica;
3. Seleção e integração dos componentes eletrónicos;
4. Modelação 3D;
5. Montagem física do protótipo;
6. Programação e validação funcional;
7. Testes finais e elaboração de documentação.

3.4. Estratégia de testes e validação

A validação do sistema foi realizada através de uma abordagem faseada, com o objetivo de assegurar o correto funcionamento de todos os componentes físicos e lógicos da bancada didática, bem como a sua aplicabilidade ao contexto educativo. As estratégias de teste

abrangeram tanto a verificação técnica individual dos elementos, como a sua avaliação em ambiente real de utilização com utilizadores finais.

Num primeiro momento, foram conduzidos testes funcionais dirigidos aos principais elementos de controlo, nomeadamente:

- Verificação do correto funcionamento dos botões de entrada digital, simulando sinais típicos de controlo;
- Testes aos LEDs, assegurando o acionamento conforme os comandos enviados pelo controlador;
- Leitura e interpretação do potenciómetro como entrada analógica no controlador;
- Monitorização do voltímetro digital para visualização de valores de tensão simulada.

Seguidamente, foram realizados testes de comunicação entre o robô colaborativo UR3e e os periféricos, com o objetivo de garantir a integração lógica do sistema. Para tal, criaram-se programas-teste no ambiente PolyScope, que permitiram verificar a resposta do robô às ações nos dispositivos de entrada e a sua interação com os elementos de saída.

Após a validação técnica individual, foi conduzida uma fase de testes com utilizadores reais, composta por alunos e professores. Os participantes foram convidados a executar um conjunto de seis exercícios práticos definidos num guião laboratorial desenvolvido especificamente para este fim. Os exercícios envolvem tarefas como movimentos programados do robô, resposta a sinais digitais, leitura de entradas analógicas e realização de sequências de automação integradas.

Concluída a execução dos exercícios, os utilizadores responderam a um inquérito de avaliação, desenvolvido para recolher perceções sobre:

- Facilidade de utilização da bancada;
- Clareza e aplicabilidade dos exercícios propostos;
- Ergonomia da estrutura;
- Relevância pedagógica da experiência prática;
- Sugestões de melhoria.

Estas estratégias permitiram não só validar tecnicamente a solução desenvolvida, como também recolher dados objetivos e subjetivos que suportam a análise da sua eficácia didática e operacional, apresentada no capítulo 5.

4. Projeto e desenvolvimento da bancada didática

Neste capítulo, são apresentados os principais aspetos relacionados com a conceção e construção da bancada didática destinada à integração de um robô colaborativo UR3e. São abordados temas como as especificações técnicas e funcionais da bancada, a arquitetura do sistema e a integração dos componentes. Apresenta-se ainda uma modelação tridimensional, em SolidWorks, do projeto que permite obter uma representação precisa da estrutura e uma pré-visualização da disposição dos elementos.

4.1. Especificações técnicas

A bancada foi projetada com base num conjunto de especificações técnicas que asseguram a segurança, resistência, funcionalidade e adaptabilidade ao contexto de ensino e aprendizagem em robótica colaborativa. Na Tabela 9 expõe-se as especificações técnicas da estrutura da bancada e do robô colaborativo UR3e (Anexo A).

Tabela 9 - Especificações técnicas do sistema

Estrutura da bancada	• **Dimensões gerais da estrutura:** 610 x 605 x 765 mm^3; • **Peso:** aproximadamente 23 kgf; • **Tipo de perfil estrutural:** perfil técnico de alumínio 45 x 45 mm^2; • **Superfície de trabalho:** madeira e alumínio; • **Capacidade de carga:** adequada para suportar o robô UR3e e periféricos (aproximadamente 15-20 kgf); • **Mobilidade:** quatro rodízios, dois com travão para transporte e fixação segura.
Robô colaborativo UR3e	• **Carga útil:** 3 kgf; • **Alcance:** 500 mm; • **Precisão:** ± 0,1 mm; • **Número de juntas (eixos):** 6; • **Interface de Programação:** PolyScope (interface gráfica).

4.2. Especificações funcionais

O sistema desenvolvido pretende suportar o ensino, mais precisamente a automação e robótica colaborativa. As especificações funcionais do sistema são:

- **Sistema de controlo:** controlador CB3.1 com interface PolyScope;
- **Expansão e elementos de controlo:** Entradas e saídas (I/O) digitais e analógicas (5 botões, 5 lâmpadas/*Leds*, 1 potenciómetro e 1 voltímetro) para integração com sensores e atuadores externos;
- **Segurança:** cablagem organizada, painel de controlo acessível e estrutura estável;
- **Aplicações:** ensino e investigação em automação e robótica colaborativa.

Estas especificações garantem uma solução equilibrada no que respeita funcionalidade, segurança e ensino, o que promove um ambiente de aprendizagem prático e eficiente.

4.3. Arquitetura do sistema

O presente subcapítulo descreve a arquitetura global do sistema, contemplando a estrutura mecânica, os elementos que a compõem, os equipamentos utilizados, incluindo o robô UR3e e seus periféricos, a caixa de I/O digitais e analógicas, bem como os *softwares* associados ao funcionamento do conjunto.

4.3.1. Estrutura mecânica

A estrutura da bancada foi desenvolvida com base em perfis técnicos de alumínio 45 x 45 L, amplamente utilizados em sistemas de automação industrial devido à sua versatilidade, facilidade de montagem e excelente relação entre resistência e peso. A escolha deste material teve como base critérios de durabilidade, resistência à corrosão e compatibilidade com sistemas de fixação normalizados.

Os perfis são unidos entre si (a 90°) através de parafusos de conexão M12x30, o que garante a resistência da ligação e a torna adequada às exigências mecânicas do sistema. A estrutura foi dimensionada para suportar não só o peso estático do robô UR3e e dos seus periféricos, mas também as cargas induzidas durante a operação do robô.

Na base da estrutura foram incorporados quatro rodízios giratórios, dos quais dois possuem travão, o que possibilita a mobilidade da bancada e a sua imobilização segura durante o funcionamento do sistema. A superfície superior é composta por um tampo de madeira de 610 x 605 x 16 mm^3, fixo à estrutura através de oito parafusos DIN 7996 M8x45. Esta superfície é responsável por acoplar o robô UR3e e o seu espaço de trabalho. Além disso, existe uma superfície intermédia constituída por um tampo de madeira de 605 x 504 x 10 mm^3, reforçado por travessas longitudinais e transversais que conferem rigidez ao conjunto. Este último é fixo à estrutura e é responsável por suportar os periféricos do robô, como o controlador e Teach Pendant. Este tampo intermédio é fixo à estrutura através de oito parafusos DIN 7996 M4x35.

Na Figura 24 apresenta-se uma representação tridimensional da bancada didática. De notar que foram considerados aspetos ergonómicos e de acessibilidade, como a altura da bancada e a disposição dos componentes, para garantir uma interação confortável do equipamento com o utilizador.

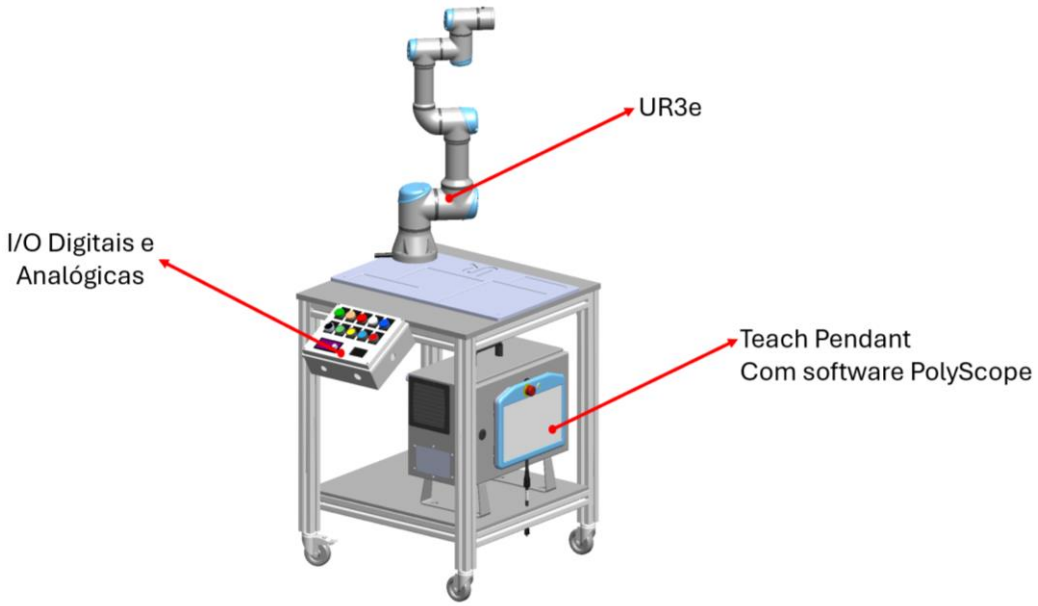


Figura 24 - Bancada didática com identificação do sistema UR3e + Teach Pendant + I/O

4.3.2. Elementos da estrutura

A estrutura principal da bancada foi adquirida ao fornecedor One Profile [83]. Este tipo de estrutura foi utilizado para garantir estabilidade, durabilidade e facilidade na montagem. O seu desenho modular permite a integração de diferentes componentes de forma funcional, assegurando a resistência mecânica necessária para suportar os equipamentos instalados. A Figura 25 ilustra a configuração geral da estrutura principal.

Figura 25 - Estrutura principal

Para a construção da estrutura foram selecionados elementos normalizados que garantem resistência, modularidade e facilidade de manutenção. A Tabela 10 apresenta uma descrição dos componentes utilizados, incluindo os perfis estruturais, elementos de fixação e acessórios que compõem o conjunto.

Tabela 10 - Elementos da estrutura

Designação	Descrição	Representação
Perfil técnico alumínio 45 x 45 L	É um perfil quadrado leve da série 45, com 45 x 45 mm² e quatro ranhuras em T abertas, uma em cada face. O perfil é liso, o que o torna resistente à acumulação de sujidade e detritos. Este perfil é leve, adequado para aplicações em salas limpas e estações de trabalho (como bancadas didáticas). As quatro ranhuras em T abertas são úteis para a montagem de acessórios.	 [84]
Parafuso M12 X 30	Elemento de ligação de baixo custo, utilizado para unir os perfis de alumínio a 90°. Parafuso auto-roscante, pode ser montado sem acabamento do perfil.	 [85]
Rodízio Ø75 giratório com eixo central roscado M12x30 mm	Rodízio giratório utilizado para assegurar a mobilidade da bancada.	 [86]
Rodízio Ø75 giratório com travão e eixo central roscado M12x30 mm	Rodízio giratório utilizado para assegurar a mobilidade da bancada e a sua imobilização segura durante utilização.	 [87]
Tampo reto em madeira 605x504x10 mm **Tampo reto em madeira 610x605x16 mm**	Tampo em madeira compensada revestido com laminado decorativo cinzento.	 [88]
Parafuso DIN 7996 M4x35 **Parafuso DIN 7996 M8x45**	Elementos de ligação de baixo custo, utilizados para unir os tampos de madeira aos perfis.	 [89]

4.3.3. Equipamentos

Este subcapítulo apresenta os equipamentos que integram o sistema, incluindo o robô UR3, os seus periféricos e a caixa de I/O digitais e analógicas, detalhados nas secções seguintes.

4.3.3.1. Robô UR3e e periféricos

O cobot UR3e foi montado numa base dedicada, com posicionamento preciso para assegurar a estabilidade e repetição de tarefas. A sua fixação à base de trabalho do robô e ao tampo superior da bancada foi realizada através de quatro parafusos M6 x 35. A base de trabalho encontra-se fixa ao tampo superior da bancada através de três parafusos M6 x 25. O controlador do robô e o Teach Pendant (equipado com o software PolyScope) localizam-se na prateleira inferior da bancada, com fácil acesso para manutenção, configuração e manuseamento. O controlador encontra-se fixo ao tampo inferior através de dois parafusos M10 x 35, duas anilhas ISO 7089 M10 e duas porcas hexagonais ISO 4032 M10 x 8.

Acoplado ao UR3e encontra-se o *gripper* HCR-03-118505 Zimmer (Anexo B) e o sistema de visão Intel D435i. O *gripper* é de atuação elétrica e é o componente responsável pela manipulação de objetos.

A Tabela 11 apresenta uma descrição dos elementos utilizados, incluindo o robô colaborativo UR3e, os seus periféricos e os elementos de ligação necessários para a fixação destes componentes.

Tabela 11 – Cobot UR3e, periféricos e modos de fixação

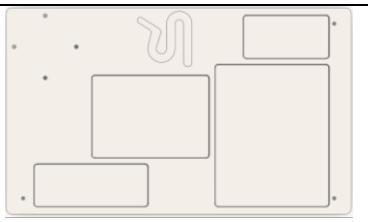
Designação	Descrição	Representação
Robô colaborativo UR3e	Robô colaborativo fácil de utilizar e programar, rápido de implementar e seguro para trabalhar lado a lado com o operador.	 [90]
Base de trabalho (em alumínio) 610 x 365 x 10 mm³	Base de trabalho do robô colaborativo, situado entre o robô e o tampo da mesa. Contribui para a fixação do robô. Este componente é caracterizado por apresentar diferentes zonas de trabalho do robô colaborativo.	
4 Parafusos M6x35	Elemento de ligação utilizado para fixar o robô à base de trabalho e ao tampo superior da bancada.	 [91]
3 Parafusos M6x25	Elemento de ligação utilizado para fixar a base de trabalho ao tampo superior da bancada.	

Tabela 11 – Cobot UR3e, periféricos e modos de fixação (continuação)

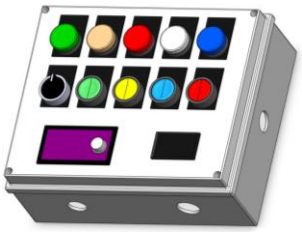
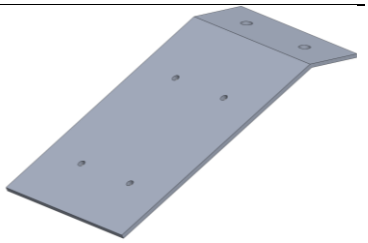
2-JAW PARALLEL GRIPPERS HRC-03-118505	O *gripper* permite agarrar peças com precisão e segurança, garantindo um posicionamento estável durante as operações. A sua conceção compacta e robusta assegura versatilidade e fiabilidade na execução de tarefas de montagem, movimentação ou inspeção.	 [92]
Controlador CB3.1	O controlador é um componente essencial do sistema global. É o centro nevrálgico para gerir os movimentos do robô, as interações com dispositivos externos e a integração de sistemas de automação mais amplos.	
3PE Teach Pendant e-Series	Interface de comunicação equipada com o *software* PolyScope. A partir do PolyScope é possível desenvolver a programação e execução dos exercícios/programas pretendidos. Integra-se na estrutura através do encaixe nos dois suportes circulares presentes na parte frontal do controlador.	 [93]
2 Parafusos de cabeça hexagonal M10x35		 [94]
2 Anilhas ISO 7089 M10	Elementos de ligação responsáveis por fixar o controlador ao tampo inferior da bancada.	 [95]
2 Porcas hexagonais ISO 4032 M10x8		 [96]

4.3.3.2. Caixa de I/O digitais e analógicas

A caixa de I/O digitais e analógicas é fixa à estrutura com auxílio de uma chapa em alumínio, cujas dimensões são 215 x 120 x 3 mm^3. A chapa é quinada a 45°, para conferir uma inclinação ergonómica da caixa durante utilização. A ligação entre a caixa e a chapa é realizada por quatro parafusos M4X25, quatro anilhas ISO 7089 M4 e quatro porcas hexagonais ISO 4032 M4x3. Por fim, a ligação entre a chapa e a estrutura é feita por dois parafusos M6x35 e por duas porcas martelo M6.

A Tabela 12 apresenta uma descrição dos elementos utilizados, incluindo a caixa de I/O digitais e analógicos e os elementos de ligação responsáveis pela fixação da caixa à estrutura.

Tabela 12 - Modo de fixação da caixa de I/O digitais e analógicas

Designação	Descrição	Representação
Caixa de I/O digitais e analógicas	Caixa que integra todos os componentes de entradas e saídas digitais e analógicas.	
Chapa 215 x 120 x 3 mm³	Chapa de alumínio que confere suporte à caixa de I/O, quinada a 45°.	
Porca martelo M6	Elemento de ligação utilizado para fixar a chapa ao perfil técnico de alumínio (à estrutura).	 [97]

A Tabela 13 apresenta os elementos que compõem a caixa de I/O digitais e analógicas, incluindo botões de pressão e sinalizadores luminosos (LEDs). Na coluna de representação encontra-se apenas um exemplo de botão e um exemplo de LED, uma vez que os restantes modelos diferem apenas na cor da superfície visível do atuador ou do difusor luminoso. Todos os botões e *Leds* foram adquiridos ao fornecedor Soflight, pelo que se apresenta a designação utilizada pelo mesmo.

Tabela 13 – Elementos inseridos na caixa de I/O digitais e analógicas

Designação	Descrição	Representação
SL-CPB01-AA31	Botão de pressão verde	
SL-CPB01-AA42	Botão de pressão vermelho	
SL-CPB01-AA51	Botão de pressão amarelo	
SL-CPB01-AA61	Botão de pressão azul	
SL-CPB01-AD21	Seletor de 2 posições fixas	

Tabela 13 – Elementos inseridos na caixa de I/O digitais e analógicas (continuação)

SL-CPB01-22DS-GR-024	Sinalizador compacto Ø22mm *Led* cor verde 24V	
SL-CPB01-22DS-RD-024	Sinalizador compacto Ø22mm *Led* cor vermelho 24V	
SL-CPB01-22DS-YE-024	Sinalizador compacto Ø22mm *Led* cor amarelo 24V	
SL-CPB01-22DS-BL-024	Sinalizador compacto Ø22mm *Led* cor azul 24V	
SL-CPB01-22DS-WH-024	Sinalizador compacto Ø22mm *Led* cor branco 24V	
JOY-IT VM433	Voltímetro de painel digital 0..33VDC (4 dígitos) - vermelho	
0-10V Signal Generator DM-HOD-1AO-010V	Potenciómetro analógico com geração de sinal 0-10V	

4.3.4. Softwares

O sistema utiliza o *software* PolyScope, incorporado no Teach Pendant do robô UR3e, como interface principal para a programação e controlo do cobot. Este ambiente gráfico permite a criação de sequências de movimento e lógica sem necessidade de programação avançada, sendo ideal para contextos didáticos.

4.4. Integração de componentes

Neste subcapítulo são abordados os aspetos relacionados com a integração dos componentes do sistema, incluindo a organização dos equipamentos na bancada, a integração física dos componentes e os protocolos de comunicação. Estes elementos, quando corretamente implementados, asseguram a operação eficiente, segura e otimizada da bancada experimental.

4.4.1. Organização dos equipamentos na bancada

Os equipamentos foram organizados de modo a otimizar o espaço disponível e garantir a segurança e o conforto dos utilizadores. O robô UR3e foi instalado na parte central da superfície superior, sobre a base de alumínio (área de trabalho do robô), garantindo uma área de utilização livre de obstáculos. Na zona frontal da estrutura encontra-se o painel de I/O digitais e analógicas, inclinado ergonomicamente para facilitar a operação dos botões e do potenciómetro, bem como a visualização dos *LEDs* e do voltímetro. Na parte inferior direita encontra-se o controlador do robô e o Teach Pendant.

A integração de I/O digitais e analógicas confere à bancada um elevado grau de versatilidade e expansão. Esta capacidade permite a conexão com sensores, atuadores e outros dispositivos externos, o que torna possível a simulação de sistemas industriais mais complexos. Assim, os utilizadores podem aprofundar os seus conhecimentos não só em robótica colaborativa, como também explorar conceitos associados à automação. Esta abordagem multidisciplinar contribui para o desenvolvimento de competências técnicas em diferentes áreas. Além disso, estes elementos permitem uma interação intuitiva com o sistema e facilitam o controlo de parâmetros como a velocidade e estado da operação. A disposição e organização dos componentes na caixa (Figura 26) pretende proporcionar uma utilização simples e eficiente para o utilizador.

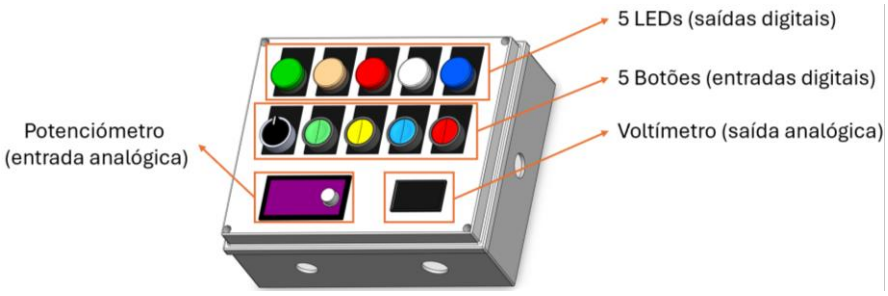


Figura 26 - Caixa de I/Os com representação da disposição dos componentes

4.4.2. Integração física dos componentes

A integração dos componentes na bancada foi realizada de forma a assegurar a compatibilidade elétrica e lógica entre os dispositivos. Na Figura 27 apresenta-se o diagrama de blocos com as conexões do sistema entre o controlador, a caixa de I/Os e o robô UR3e.

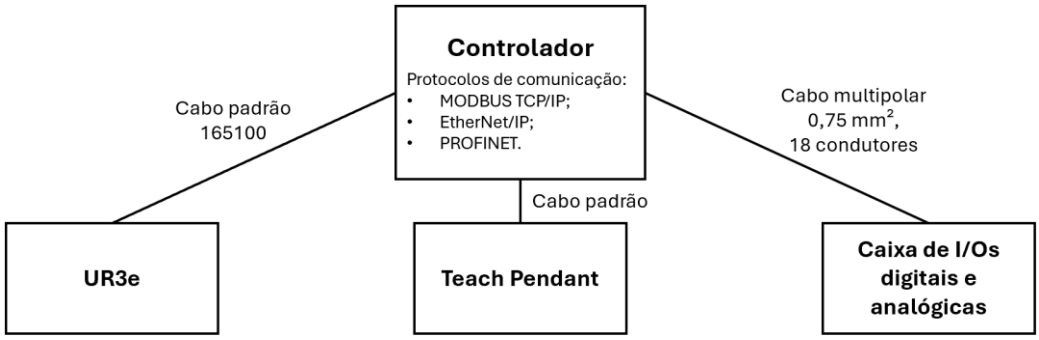


Figura 27 - Conexões do sistema

Para melhor compreensão, as conexões do sistema são apresentadas em dois subconjuntos:

- Controlador, robô colaborativo UR3e e Teach Pendant;
- Controlador e caixa de I/Os digitais e analógicas.

Controlador, robô colaborativo UR3e e Teach Pendant

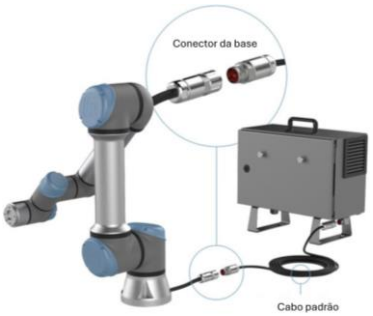
O robô colaborativo UR3e está ligado diretamente ao controlador Universal Robots, que integra a fonte de alimentação, os terminais de comunicação e os módulos de I/O (Figura 28). A comunicação entre o robô e o controlador é assegurada pelo cabo padrão 165100.

Figura 28 - Controlador com representação dos terminais [98]

O Teach Pendant, responsável pela interface homem-máquina, liga-se ao controlador através de um conector. Este elemento permite o desenvolvimento, simulação e execução de programas no ambiente PolyScope.

A Tabela 14 apresenta a designação, descrição e representação gráfica dos elementos, incluindo ainda uma explicação da sua respetiva função no sistema.

Tabela 14 - Designação, descrição e representação dos elementos de ligação

Designação	Descrição	Representação
Cabo padrão 165100	Cabo que realiza a ligação do robô colaborativo UR3e ao controlador.	 [99]  [100]
3PE Teach Pendant e-Series	A ligação do Teach Pendant é feita ao controlador através de um cabo padrão de 4,5 m de comprimento que está acoplado no componente.	 [93]

Controlador e caixa de I/Os digitais e analógicas

O controlador disponibiliza entradas e saídas digitais (DI/DO), bem como entradas e saídas analógicas (AI/AO), que permitem a ligação de sensores e atuadores externos. Foi através

destes terminais que se estabeleceu a comunicação com a caixa de I/O digitais e analógicas. A Figura 29 apresenta a disposição e categorização dos blocos de entradas e saídas que o controlador do robô UR3e dispõe.

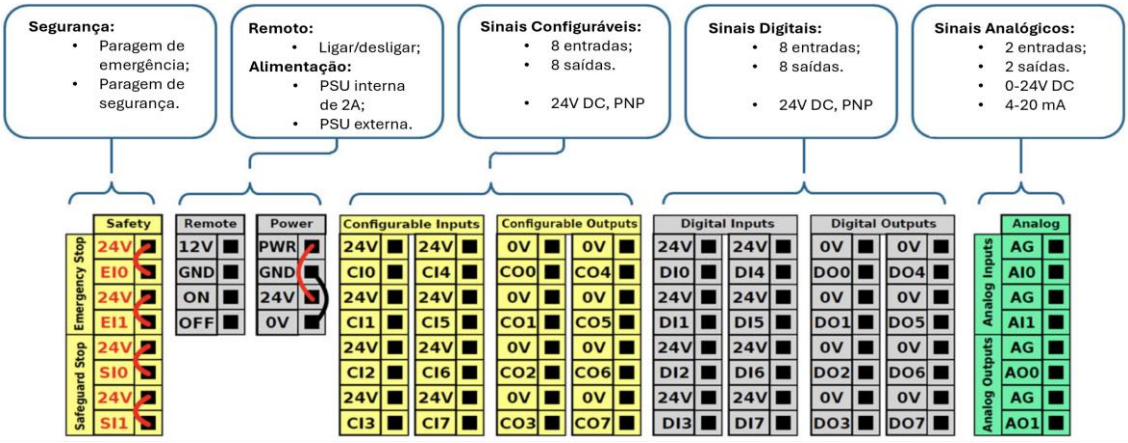


Figura 29 - Representação dos blocos do controlador do UR3e

Para a realização das ligações elétricas foram desenvolvidos esquemas elétricos no *software* EPLAN Education (projeto completo no Apêndice A). Para a ligação entre o controlador e a caixa de I/Os utiliza-se um cabo multipolar de 0,75 mm², composto por 17 condutores e um condutor terra. Cada condutor é numerado, o que facilita a identificação e a correspondência direta entre os componentes e o controlador. Desta forma, garante-se praticidade e organização na ligação.

Na Figura 30 apresenta-se um exemplo de um esquema elétrico desenvolvido para as ligações das saídas digitais do controlador CB3.1 do robô UR3. O diagrama mostra a conexão de quatro saídas digitais (D00 a D03), cada uma associada a um *LED* sinalizador: verde (H1), amarelo (H2), vermelho (H3) e branco (H4). Cada canal de saída é ligado em paralelo a um terminal de 0V, completando o circuito de alimentação. Este arranjo permite o acionamento individual de cada *LED* a partir dos sinais digitais emitidos pelo controlador, com o objetivo de indicar diferentes estados operacionais do sistema. Além disso, observa-se que o condutor número 11 estabelece a ligação entre o *LED* verde e a saída digital 0 do controlador, seguindo-se o mesmo princípio para os restantes condutores correspondentes aos outros *LEDs*.

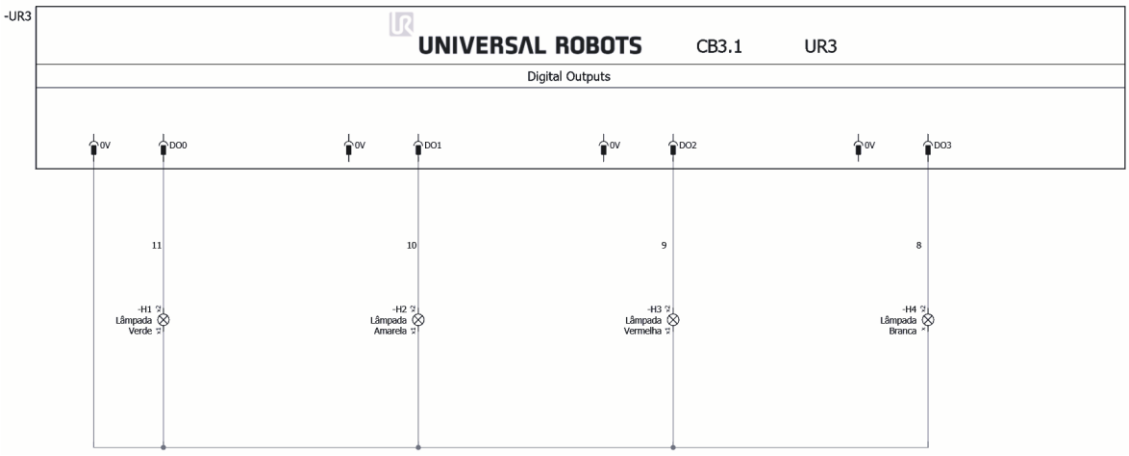


Figura 30 - Esquema das ligações elétricas das saídas digitais

Os cabos e respetivas ligações foram organizados de forma limpa e segura, para minimizar riscos de interferência e facilitar futuras manutenções. A estrutura modular da bancada permitiu adaptar e reposicionar os elementos conforme necessário durante a fase de desenvolvimento.

Após a ligação dos componentes, foi estabelecida a correspondência entre cada dispositivo e a respetiva entrada ou saída digital no controlador, conforme apresentado na Tabela 15. Esta tabela organiza de forma clara quais os dispositivos conectados às entradas digitais (botões) e analógicas (potenciómetro) e às saídas digitais (LEDs) e analógicas (voltímetro), permitindo identificar rapidamente a função de cada canal.

Tabela 15 - Correspondência entre as I/O digitais e analógicas e respetiva ligação no controlador

Entradas digitais	
Botão de 2 posições	DI0
Botão Verde	DI1
Botão Amarelo	DI2
Botão Azul	DI3
Botão Vermelho	DI4
Saídas digitais	
LED Verde	DO0
LED Amarelo	DO1
LED Vermelho	DO2
LED Branco	DO3
LED Azul	DO4
Entrada analógica	
Potenciómetro	DAI1
Saída analógica	
Voltímetro	DAO1

4.4.3. Protocolos de comunicação

O sistema desenvolvido está preparado para operar com diferentes protocolos de comunicação industrial, o que permite a ligação do robô colaborativo UR3e a controladores externos, como PLCs, módulos de expansão e dispositivos periféricos habitualmente utilizados em ambientes de automação. Esta abordagem assegura a interoperabilidade entre os diferentes elementos do sistema e permite a construção de células robóticas didáticas com elevado grau de complexidade.

O sistema desenvolvido está preparado para operar com os protocolos MODBUS TCP e EtherNet/IP, dispositivos IIoT ou plataformas SCADA (referentes a sistemas de supervisão, controlo e aquisição de dados que monitorizam e gerem processos industriais).

- **MODBUS TCP/IP**

O protocolo MODBUS TCP/IP está disponível no controlador do cobot UR3e e permite a integração direta com controladores lógicos programáveis (PLCs) e outros equipamentos. Trata-se de um protocolo aberto, amplamente utilizado na indústria, que possibilita a troca de dados através de uma rede Ethernet de forma estruturada e fiável.

- **EtherNet/IP**

O protocolo EtherNet/IP é um protocolo determinístico (os dados são enviados em intervalos de tempo previsíveis e consistentes) e orientado a objetos, ideal para comunicação em tempo real com módulos de I/O, sensores ou controladores de segurança.

Módulos e Dispositivos Compatíveis

A utilização destes protocolos assegura uma comunicação fiável, o que possibilita a ligação a uma vasta gama de dispositivos industriais. Entre os principais módulos e equipamentos que podem ser integrados no sistema destacam-se:

- **PLCs** – Lógica de controlo, comunicação com sensores/atuadores, e sincronização de tarefas com o robô;
- **HMIs** – Visualização em tempo real do estado do sistema e interface de operação;
- **Variadores de Velocidade** – Controlo de motores;
- **Servomotores e Servo Drives** – Operações de posicionamento com elevada precisão;
- **Cilindros pneumáticos e válvulas eletropneumáticas** – Atuação mecânica de objetos, acionados por sinais digitais provenientes do PLC ou do robô.
- **Módulos de I/O digitais e analógicos** – Expansões para sensores e atuadores diversos;
- **Sensores industriais** (óticos, proximidade, pressão, temperatura) – *Feedback* sobre condições do processo;
- **Relés, contatores e botoneiras** – Elementos auxiliares de comando e proteção;
- **Módulos de segurança** – Dispositivos que asseguram o cumprimento das normas de segurança funcionais (ISO 13849, IEC 62061);
- ***Gateways* industriais** (ex.: eWON Flexy 205) – Acesso remoto seguro ao sistema, com funcionalidades de recolha de dados, monitorização remota e diagnóstico em tempo real.

Alguns dos módulos e dispositivos anteriormente descritos não estabelecem comunicação direta com o controlador do robô colaborativo UR3e. Nestes casos, é necessária a presença de um dispositivo intermédio, geralmente um PLC, que atua como elemento de controlo e interface. É o que acontece, por exemplo, com variadores de velocidade, cilindros pneumáticos e *servo drivers*, que são comandados diretamente pelo PLC, que por sua vez comunica com o robô através dos protocolos industriais MODBUS TCP ou EtherNet/IP.

Acesso remoto

Um dos principais benefícios da adoção de protocolos industriais abertos e de arquitetura modular é a capacidade de aceder remotamente ao sistema. Através da utilização de *gateways* como o eWON Flexy 205, torna-se possível a ligação à *cloud*, o acesso remoto seguro ao controlador do robô, a recolha contínua de dados e a execução de comandos à distância. Estas funcionalidades permitem trabalhar remotamente no robô e, desta forma, alinhar o sistema

com os princípios da Indústria 4.0, tornando-o altamente relevante para contextos didáticos e industriais onde a conectividade e flexibilidade são fatores diferenciadores.

Deste modo, a capacidade de comunicação com múltiplos dispositivos torna a bancada um sistema altamente didático, o que permite simular arquiteturas industriais completas com vários níveis de complexidade, como a bancada didática proposta por Azevedo et al. [73].

5. Programação e interface com o utilizador

Este capítulo apresenta a interface de programação utilizada ao longo de todo o processo, o PolyScope, *software* incorporado no Teach Pendant do robô. Descrevem-se as suas principais funcionalidades, desde a configuração inicial e os modos de movimentação até à criação e execução de programas. São ainda abordados diferentes tipos de instruções, posteriormente aplicadas em exercícios práticos que replicam a utilização dos conceitos em cenários reais.

O manual de utilizador desenvolvido (Apêndice B) expõe as instruções de programação disponíveis na interface, a totalidade dos exercícios desenvolvidos e a metodologia de resolução passo a passo dos mesmos.

5.1. Ambiente inicial da interface

A compreensão do ambiente inicial do PolyScope é fundamental para a utilização eficiente do robô, uma vez que esta interface constitui o ponto de partida para as operações de configuração, programação e controlo. Na Figura 31 apresenta-se a página inicial do *software*, com destaque para as diferentes áreas e funcionalidades disponíveis.

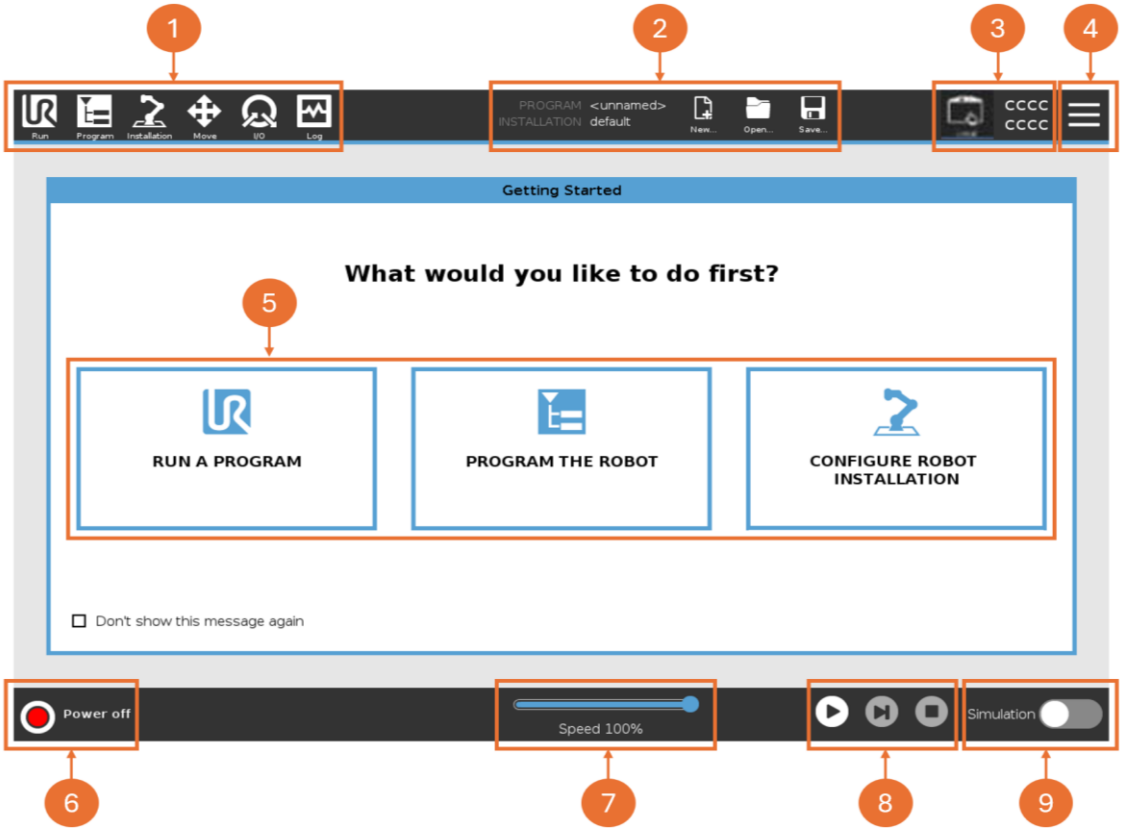


Figura 31 - Ambiente inicial da interface

As zonas destacadas na Figura 31 apresentam as seguintes funcionalidades:

1. Conjunto de ícones relacionados com a configuração, programação, movimento e configuração do robô;
2. Conjunto de ícones de gestão de arquivos. Permitem criar, guardar e gerir programas;
3. Ícones relacionados com o controlo local ou remoto do controlador, verificação de segurança, modos de operação do sistema (manual/automático) e ferramentas URCaps;
4. Ícone do menu. Dá acesso a várias configurações do sistema operacional como definição de senha, atribuição de endereço IP, escolha do idioma da interface, entre outros;
5. Conjunto de ícones de acessibilidade rápida para executar um programa já existente, criar um novo programa e configurar parâmetros de instalação;
6. Botão de ligar e desligar. Elemento que permite ativar ou desativar o robô, com indicação do estado atual do sistema;
7. Controlador de velocidade. Permite ajustar a velocidade de movimento do robô;
8. Ícones de controlo dos programas. Permitem iniciar, pausar e parar um programa;
9. Botão deslizante que ativa a simulação no sistema operacional e impede os movimentos reais do robô.

5.2. Ativação do robô

O braço robótico, mesmo após ativação do seu controlador, permanece no estado desativo. Para que o robô possa ser configurado, movido e programado, é necessário ativá-lo. Para tal, deve ser acionado o botão que se encontra no canto inferior esquerdo da tela inicial do PolyScope (assinalado com o número 6 na Figura 31). De seguida, é necessário seguir uma sequência de três passos até que o botão associado ao estado do robô passe para o estado **Normal**. Segue-se a referida sequência:

1. Selecionar o botão de ativação que demonstra o estado do robô ***Power off***;
2. Selecionar o botão ***ON***, assinalado com o número 2 na Figura 32;
3. Selecionar o botão ***START***, assinalado com o número 3 na Figura 32;

Após esta sequência, o botão que demonstra o estado do robô passa a **Normal** e os cinco estados do robô devem aparecer a verde (ver secção assinalada com o número 4 na Figura 32).

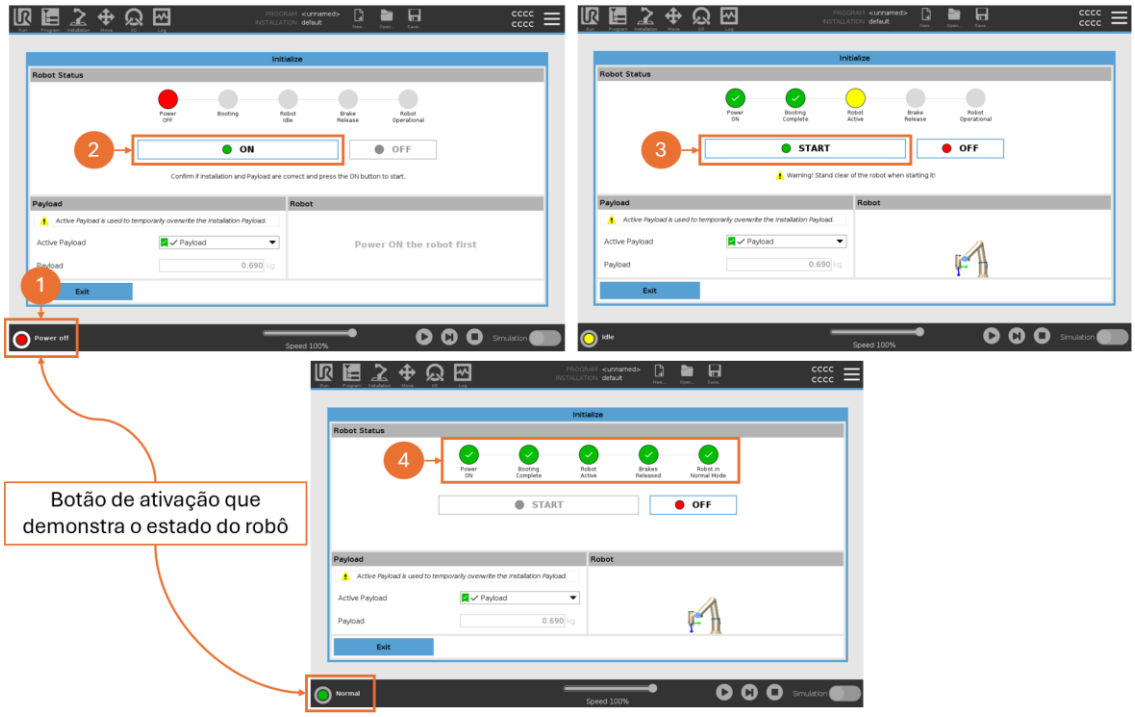


Figura 32 - Ativação do robô

5.3. Instalação básica

Numa frase prévia à programação do cobot, é necessário configurar os parâmetros essenciais do *Tool Center Point* (TCP), correspondente ao ponto central entre garras do *gripper* (Figura 33). É ainda necessário configurar a carga útil em quilogramas (kgf).

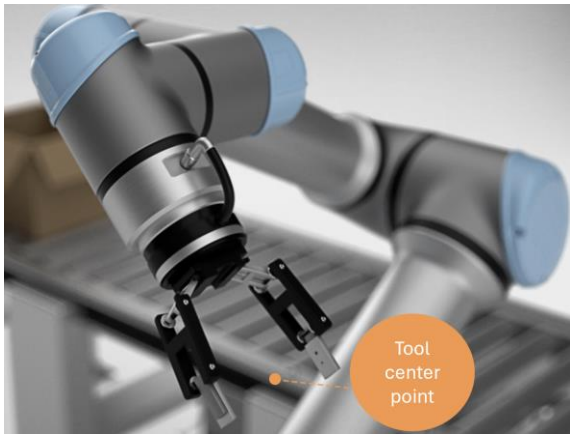


Figura 33 - Localização do TCP, adaptado de [101]

De modo a facilitar a compreensão da configuração do TCP e da carga útil, expõem-se um caso prático da parametrização resolvido o passo a passo. De referir que cada ferramenta tem o seu TCP, pelo que é necessário considerar diferentes aspetos.

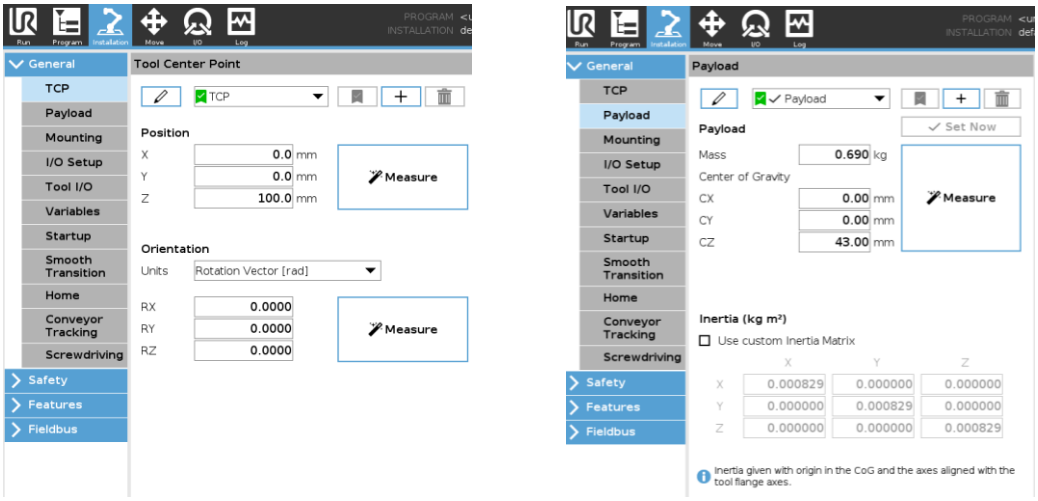
Caso prático resolvido

Realizar a configuração de uma ferramenta com os seguintes dados:

- TCP: X=0 mm; Y=0 mm; Z=100 mm.
- Carga: 0,690 kgf; centro de gravidade CX=0, CY=0, CZ=43 mm.

Resolução:

Inicialmente, para garantir uma correta parametrização do robô, é necessário atribuir um nome à configuração desejada, preencher os parâmetros relativos à posição (no TCP, Figura 34 (a)) e à *payload* (para a carga útil, Figura 34 (b)). De seguida, é necessário proceder à ativação dos referidos parâmetros através da opção “Set Now”, presente na interface do Teach Pendant. A presença de um ícone verde junto ao nome da configuração indica que a definição do TCP e da carga útil foi aplicada com sucesso, como se verifica na Figura 34.



(a) (b)

Figura 34 - Parametrização do robô: (a) Definição do TCP; (b) Definição da carga útil

Após a configuração, o Teach Pendant apresenta uma visualização tridimensional (3D) do robô, refletindo os dados introduzidos. Após analise da Figura 35 verifica-se que o sistema de coordenadas TCP é representado graficamente, com o eixo z, a azul, orientado segundo o eixo da flange do robô, o que facilita a interpretação espacial da carga. A carga útil (*payload*) é ilustrada por um cubo 3D, cujo volume é proporcional ao valor atribuído durante a configuração. O centro de gravidade da carga é evidenciado pela posição relativa do cubo face à flange do robô, permitindo uma análise visual da distribuição de massa.

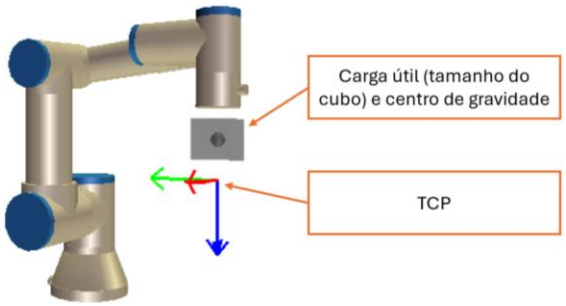


Figura 35 - Representação dos dados configurados

5.4. Movimento do robô

O movimento do robô constitui um elemento central na programação e operação do UR3e, sendo a base para a execução de qualquer tarefa. A compreensão dos diferentes modos de movimentação e das ferramentas de controlo disponíveis no PolyScope é essencial para garantir precisão, repetibilidade e segurança nas operações. Nesta secção são descritos os tipos de movimentos disponíveis na aba superior *Move* da interface.

5.4.1. Movimento em modo Freedrive

O movimento em modo *Freedrive* só pode ser realizado fisicamente (modo manual). Na Figura 36 ilustra-se o processo de utilização deste modo. O processo inicia com a seleção do botão ***Freedrive*** (1) localizado no painel inferior da interface. Após esta seleção, é apresentado no lado direito do ecrã um menu (2) no qual o botão ***Freedrive - Press & Hold*** (3) deve ser mantido pressionado enquanto se movem manualmente as juntas do robô. Para facilitar o seu posicionamento, é possível limitar o movimento a determinados eixos cartesianos ou à rotação.

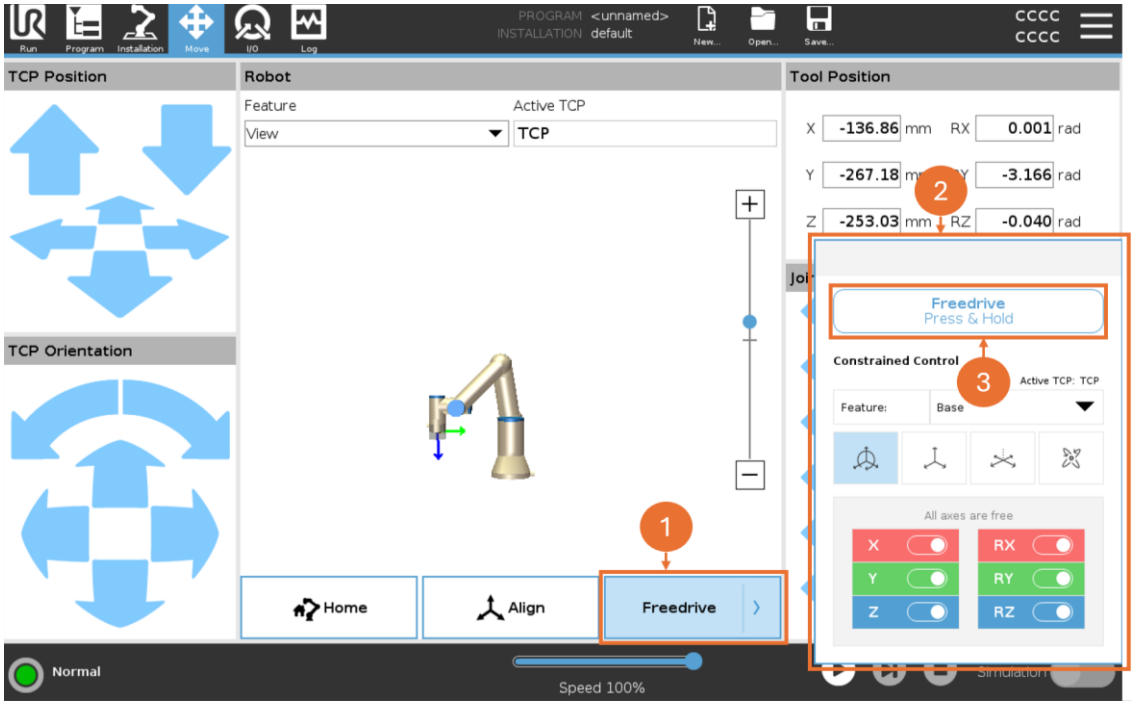


Figura 36 - Movimento do robô em modo Freedrive

5.4.2. Movimento a partir do Teach Pendant

Através do Teach Pendant o movimento pode ser feito de duas maneiras:

1. Janela de movimento

A janela de movimento permite ao utilizador realizar os diferentes tipos de movimentação do robô, nomeadamente movimentos das juntas, lineares e de reorientação. Como ilustrado na Figura 37, esta janela inclui um conjunto de botões gráficos que facilitam a execução de movimentos lineares do TCP ao longo dos eixos cartesianos, bem como movimentos de reorientação baseados na rotação em torno desses mesmos eixos. Adicionalmente, é disponibilizada uma área dedicada ao controlo direto das juntas do robô, onde é possível ajustar individualmente a posição de cada junta. Esta interface permite ainda a seleção do sistema de coordenadas de referência (por exemplo, base ou ferramenta), em função do tipo de movimento realizado, o que garante maior precisão e controlo nas operações de manipulação.

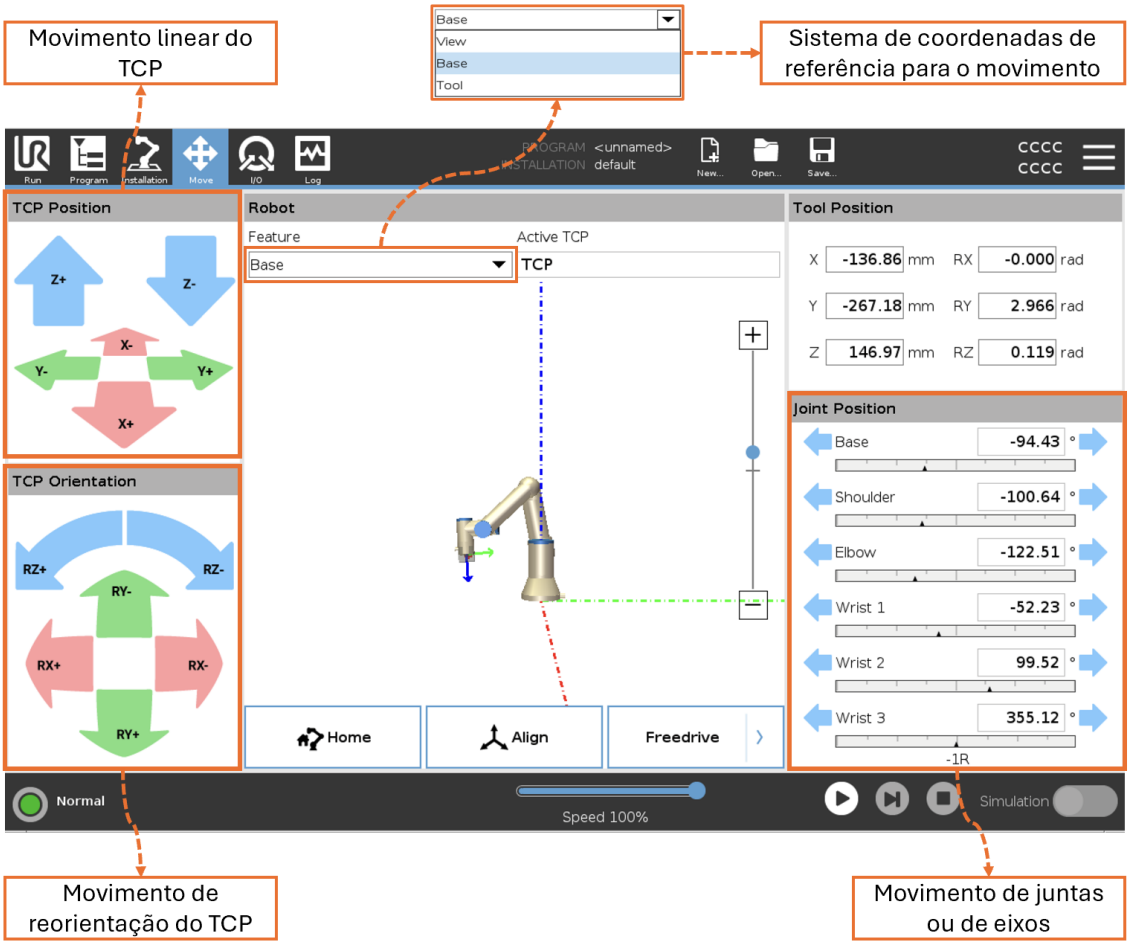


Figura 37 - Janela de movimento

2. Janela de ajuste numérico de posição

Na Figura 38 é possível observar a **janela de ajuste numérico de posição**, que permite a inserção precisa dos dados de posição do robô. Esta funcionalidade encontra-se disponível ao selecionar um dos campos numéricos correspondentes ao TCP. Após aceder à janela e pressionar sobre um dos campos numéricos de posição, é exibido um teclado flutuante, que possibilita a introdução direta dos valores pretendidos. Adicionalmente, encontram-se disponíveis botões identificados com os símbolos “+” e “–” junto de cada coordenada, permitindo o aumento ou diminuição incremental dos valores apresentados. A partir desta janela, é também possível escolher o sistema de coordenadas onde os dados são modificados.

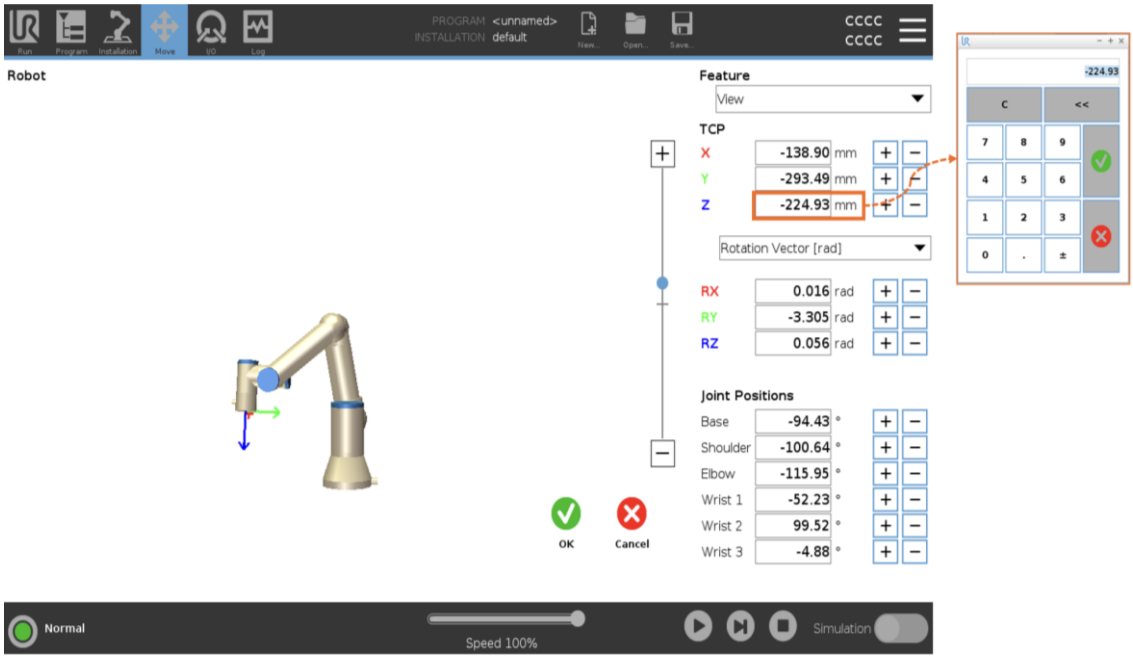


Figura 38 - Janela de ajuste numérico de posição

5.5. Ambiente de programação

O ambiente de programação do PolyScope apresenta uma interface gráfica intuitiva e funcional. Este ambiente dispõe de diferentes áreas estruturadas que facilitam a criação e edição de programas, destinados ao controlo do cobot UR3e.

A Figura 39 ilustra a página inicial do ambiente de programação, organizada em várias secções numeradas, cada uma com uma função específica:

1. **Menu de comandos de instruções**: Situado no lado esquerdo da interface. Esta área contém os blocos de programação disponíveis, organizados por categorias, tais como *Basic*, *Advanced*, *Templates* e URCaps.
2. **Árvore do programa**: Esta secção corresponde à zona de elaboração do programa. Nesta são introduzidas todas as secções, instruções e subprogramas que constituem o programa.
3. **Barra de ferramentas do programa**: Localizada na parte inferior da árvore do programa, esta barra disponibiliza funcionalidades como mover, eliminar, cortar, copiar e colar linhas de código do programa. Estas ferramentas permitem gerir eficientemente a estrutura do programa.
4. **Área de comando, parametrização e informação**: Situada no lado direito da árvore do programa. Esta secção corresponde à zona principal de edição, onde se visualiza e edita as secções/instruções da árvore principal do programa. Esta área inclui separadores como *Command*, *Graphics* e *Variables*, que permitem aceder a diferentes aspetos como comandos específicos, configuração de variáveis e visualização gráfica.

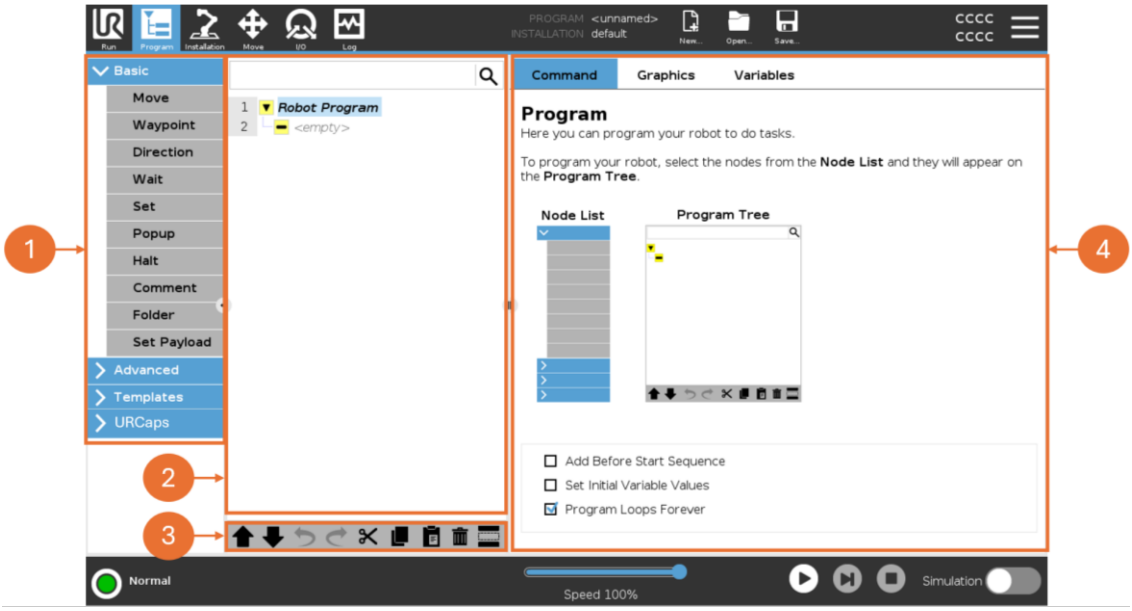


Figura 39 - Página inicial do ambiente de programação

Na Figura 40 apresenta-se de forma detalhada a barra de ferramentas do programa.

Figura 40 - Barra de ferramentas do programa

Através da análise da Figura 40, seguindo a ordem da esquerda para a direita, verificam-se as seguintes funções:

1. Mover linha para cima
2. Mover linha para baixo
3. Desfazer
4. Refazer
5. Cortar
6. Copiar
7. Colar
8. Eliminar
9. Ocultar

5.6. Programação dos robôs UR

A programação dos robôs UR realiza-se a partir do Teach Pendant, através de dois modos diferentes:

- **Interface gráfica PolyScope**: Baseia-se na criação de uma árvore de programação, na qual o ajuste de todos os parâmetros é feito por comandos interativos, semelhantes a

formulários. Esta forma de programação destaca-se pela sua intuitividade e acessibilidade, o que simplifica a criação, edição e teste de programas.

- **URScript**: Consiste em escrever código em texto simples baseado em *scripts*. Esta escrita pode ser feita diretamente na consola ou num editor externo. Os *scripts* devem ser chamados a partir da interface gráfica.

5.6.1. Estrutura do programa principal

O programa principal do robô executa-se linha a linha, com sentido descendente, e de forma cíclica. Para restringir a ciclicidade de execução, o utilizador deve inserir restrições. Na estrutura do programa podem ser inseridas várias secções, tais como:

- **Programa principal** (***Robot Program***): Corresponde ao corpo principal do programa, no qual as instruções são executadas sequencialmente, linha a linha.
- **Antes de começar** (***Before Start***): Secção dedicada à execução de instruções iniciais, antes do início do ciclo principal do programa.
- **Variáveis iniciais** (***Initial Variables Values***): Área destinada à definição dos valores iniciais das variáveis utilizadas no programa.
- **Pastas** (***Folder***): Utilizam-se para armazenar e organizar partes do programa. Não têm efeito lógico no funcionamento da sequência.
- **Subprogramas** (***SubProg***): Porções de código que são executadas quando são invocados a partir do programa principal. Podem ser guardados como arquivos para reutilização noutros programas.
- **Subtarefas** (***Threads***): Programas paralelos que são executados em simultâneo com o programa principal.
- **Eventos** (***Event***): Programa que é executado de forma semelhante a uma instrução, com a diferença de que não interrompe a execução do programa principal.

Na Figura 41 é apresentada a estrutura em árvore das referidas secções no ambiente de programação do robô.

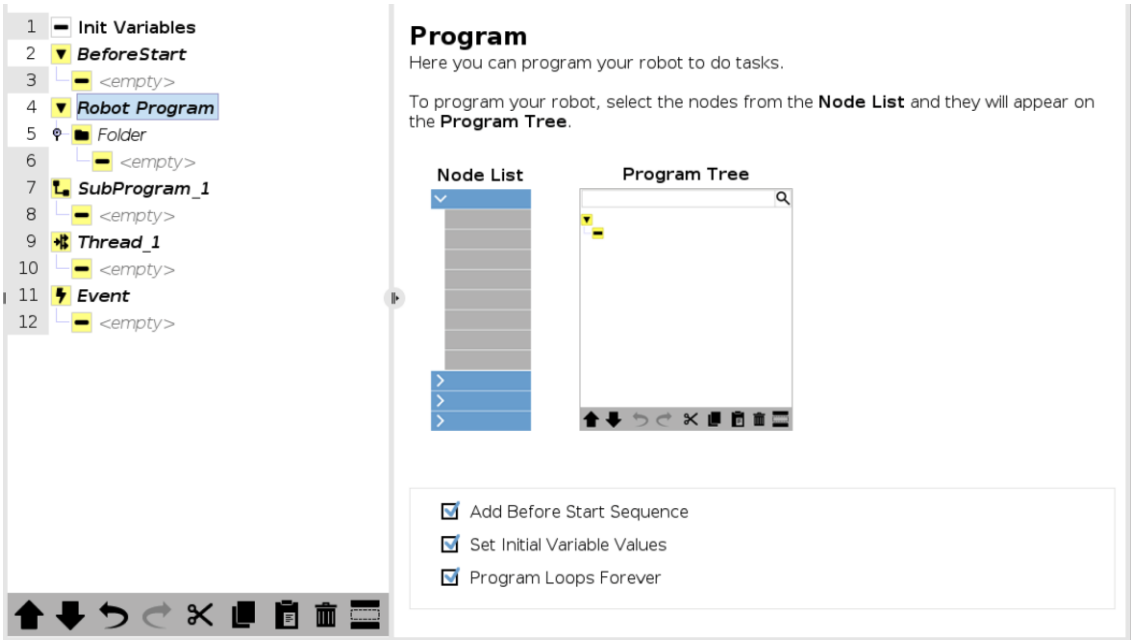


Figura 41 - Estruturação da árvore de um programa

5.6.2. Instruções de programação

A interface gráfica estrutura as instruções de programação em três grupos base:

- ***Basic***: Contém as instruções básicas de programação. Com elas é possível criar, por exemplo, uma sequência de movimento (Figura 42 (a)).
- ***Advanced***: Contém as instruções avançadas de programação. Permite introduzir instruções de controlo de fluxo como condicionais ou ciclos, e blocos lógicos para a estruturação do programa como subprogramas, subtarefas, eventos, entre outros (Figura 42 (b)).
- ***Templates***: Corresponde a uma coleção de modelos pré definidos que facilitam a implementação e programação de um determinado tipo de operação, como paletização, aparafusamento, empilhamento e desempilhamento de peças, entre outros (Figura 42 (c)).

Na Figura 42 é apresentada a estrutura completa das instruções disponíveis nos três grupos de programação do sistema.

(a) (b) (c)

Figura 42 - Instruções de programação: (a) *Basic*, (b) *Advanced*; (c) *Templates*

Na interface de programação dos robôs da Universal Robots, para além dos grupos de instruções base disponibilizados pelo sistema, podem ser adicionadas novas secções através das designadas **URCaps**, que correspondem a extensões de *software* desenvolvidas por fabricantes terceiros. Estas extensões permitem o controlo direto de periféricos instalados no robô como câmaras ou *grippers*. Na Figura 43 encontram-se dois módulos URCap instalados no robô UR3e: **Image Execution**, associado ao funcionamento da câmara, e **Zimmer**, correspondente aos comandos do Gripper. De referir que apesar de ambos os módulos URCpas se encontrarem instalados no robô, apenas se utilizaram os comandos associados ao *gripper*.

Figura 43 - Instruções de programação dos periféricos (câmara e *gripper*) instalados

5.6.3. Instrução *Move*

A instruções ***Move*** corresponde a uma instrução de movimento e está disponível na secção *Basic*. Esta instrução permite a definição de trajetórias que o manipulador deve seguir para mover o ponto TCP entre diferentes posições. Esta função comporta três tipos distintos de movimento:

- ***MoveJ* (Movimento em arco):** O *MoveJ* executa a trajetória do TCP através da interpolação entre as posições articulares (juntas) do robô, o que culmina num movimento em arco. Este tipo de movimento é caracterizado por ser mais rápido e energeticamente eficiente, mas menos preciso em relação à trajetória cartesiana percorrida pelo TCP.

  Indicação: *MoveJ* é utilizado sempre que não seja necessário seguir uma trajetória linear precisa, como em deslocações entre tarefas ou aproximações iniciais.

- ***MoveL* (Movimento Linear):** O *MoveL* garante o deslocamento do TCP ao longo de uma linha reta no espaço cartesiano entre dois pontos. Este tipo de movimento é fundamental

para tarefas que exigem precisão na trajetória como soldadura, colagem ou inserção de componentes.

Indicação: *MoveL* é utilizado em operações que exigem contacto contínuo ou movimento controlado em linha reta.

- ***MoveP* (Movimento Planeado):** O *MoveP* permite a definição de sequências de *Waypoints* que o robô percorre de forma suave e contínua, minimizando paragens entre os pontos intermédios. Através da mistura de trajetórias, garante-se uma transição fluída e ideal para aplicações que requerem velocidade e fluidez como pintura ou polimento.

Indicação: *MoveP* é utilizado quando é necessário um movimento fluido entre múltiplos pontos sem paragens intermédias.

Os parâmetros ajustáveis associados às instruções de movimento encontram-se disponíveis no comando da instrução, localizado na zona direita do ambiente de programação. Nesta área é possível definir o tipo de movimento a executar (MoveJ, MoveL ou MoveP), a velocidade e aceleração das juntas, o TCP a utilizar e o sistema de coordenadas de referência. Na Figura 44 apresenta-se a janela de configuração destes parâmetros.

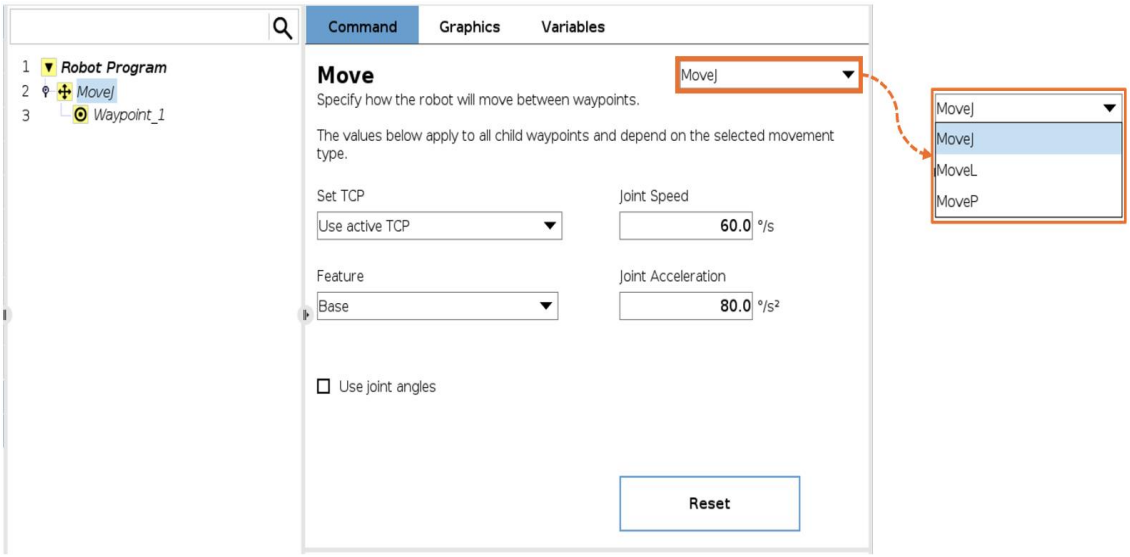


Figura 44 - Janela de ajuste dos parâmetros das instruções de movimento

5.6.4. Instrução Waypoint

A instrução *Waypoint* define uma posição no espaço que o robô deve atingir, podendo incluir orientação, velocidade e aceleração. A forma como essa instrução é usada varia ligeiramente conforme o modo de programação selecionado. De referir que uma instrução deste tipo vem associada a uma instrução *Move*.

A instrução permite definir posições que o robô deve atingir durante a execução do programa. A Figura 45 mostra três formas distintas de configurar essa posição: ***Fixed***, ***Relative*** e ***Variable***.

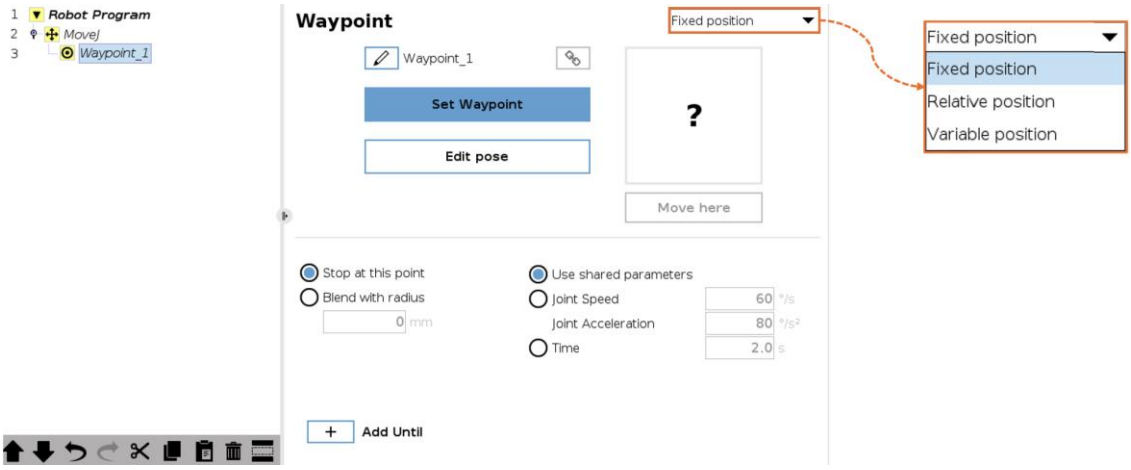


Figura 45 - Apresentação da instrução *Waypoint*

1. *Fixed Position* (Posição Fixa):

- Define uma posição absoluta no espaço cartesiano;
- O robô desloca-se para a posição definida, independentemente da sua localização inicial;
- Ideal para tarefas repetitivas com alta precisão.

2. *Relative Position* (Posição Relativa)

- Define a posição em relação a outro ponto de referência (por exemplo, o ponto anterior);
- Indicada para quando o robô precisa de fazer movimentos em sequência com deslocamentos constantes.

3. *Variable Position* (Posição Variável)

- A posição é definida por uma variável, que pode mudar durante o programa;
- Permite lógica dinâmica e adaptação a diferentes cenários.

5.6.5. Instrução *Direction*

A instrução *Direction* permite realizar movimentos lineares do TCP da ferramenta ao longo de um dos eixos cartesianos, com uma distância definida em milímetros. Este movimento é executado a partir da posição atual do TCP, assim que o programa atinge a linha onde a função está inserida. Para garantir que o movimento é interrompido no momento apropriado, é necessário configurar uma condição de paragem. O sistema disponibiliza quatro opções distintas para definir essa condição: uma expressão lógica, uma distância específica, a deteção de contacto da ferramenta ou a ativação de uma entrada digital ou analógica.

A Figura 46 apresenta a janela de configuração da instrução *Direction,* na qual é possível ajustar os parâmetros de movimento e definir a condição de paragem na secção *Until* da instrução.

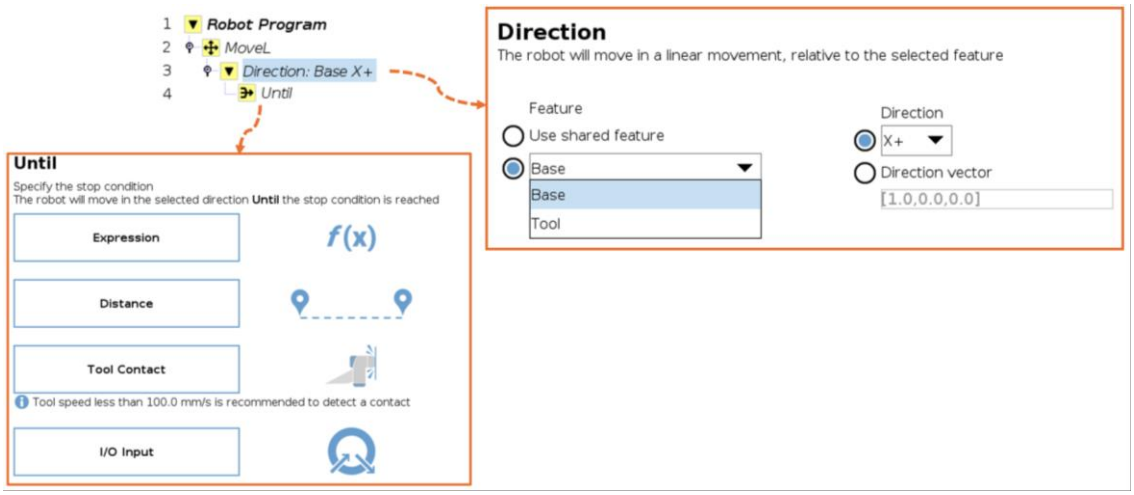


Figura 46 - Janela de ajuste de parâmetros da instrução *Direction*

5.6.6. Variáveis

Uma variável é um elemento de programação que ocupa uma zona de memória no controlador, utilizada para armazenar valores em função do tipo de dados, que posteriormente são processados e modificados no modo de execução.

As variáveis podem apresentar diferentes tipos de dados como booleana, número inteiro, número real, *string* e posição. Na Tabela 16 são apresentados os diferentes tipos de variáveis, o seu valor e um exemplo.

Tabela 16 - Tipos de dados das variáveis

Tipo	Valor	Exemplo
Booleana	*True* ou *False*	Botão_verde = *True*
Número inteiro	Entre -2147483648 e 2147483647	Contador=0
Número real	Números de pontos flutuantes de 32 bits	Posição Y = 0.15
String	---	"Olá, mundo"
Posição	p[X,Y,Z,rX,rY,rZ]	p[0.2,0.3,0,0,0,0]

No ambiente de programação PolyScope e UR, as variáveis podem ser de três tipos:

- **Variáveis de programa:**
  - Criadas diretamente na árvore do programa;
  - Usadas para controlar a lógica e o comportamento do robô durante a execução;
  - Específicas para o programa em que foram criadas.

- **Variáveis de instalação:**
  - Definidas na aba de instalação do PolyScope;
  - Permitem configurar elementos como posições, características de ferramentas, *payloads*, entre outros;
  - Podem ser reutilizadas em diferentes programas dentro da mesma instalação.

- **Variáveis de *Script* (URScript):**

- Utilizadas em *scripts* personalizados e escritas em URScript;
- Permitem maior controlo e flexibilidade como *loops*, condições e chamadas de funções;
- Úteis para programadores mais experientes que pretendem ir além da interface gráfica.

5.6.7. Uso de variáveis como um contador

Para utilizar uma variável como se fosse um contador, utiliza-se uma sintaxe semelhante à das linguagens de programação usuais. Para que a variável seja incrementada (ou decrementada) numa quantidade relativamente ao valor que ela possui, deve-se aplicar a seguinte expressão:

Contador = Contador + 1

Para criar uma variável de programa, é necessário introduzir uma instrução *Assignment,* que se situa na secção *Advanced* de programação (Figura 47). A variável pode ser criada na secção *Before Start* ou na árvore do programa principal.

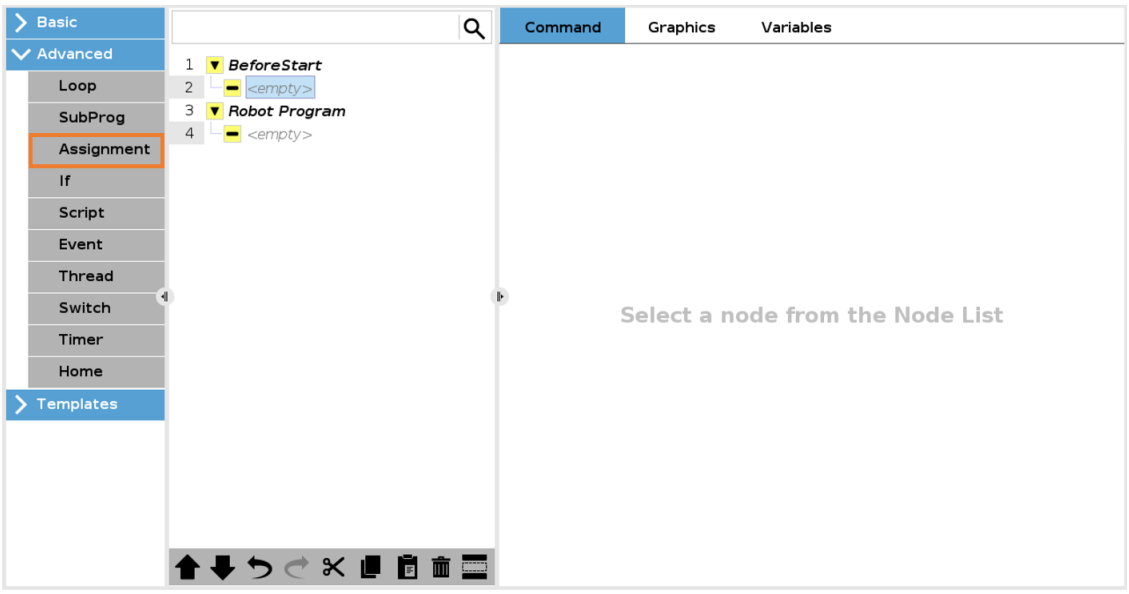


Figura 47 - Inserção da instrução *Assignment*

De seguida, deve ser definido o nome da variável. Para tal segue-se a sequência de três passos presente na Figura 48. Neste caso, atribuiu-se o nome *count* para a variável contador.

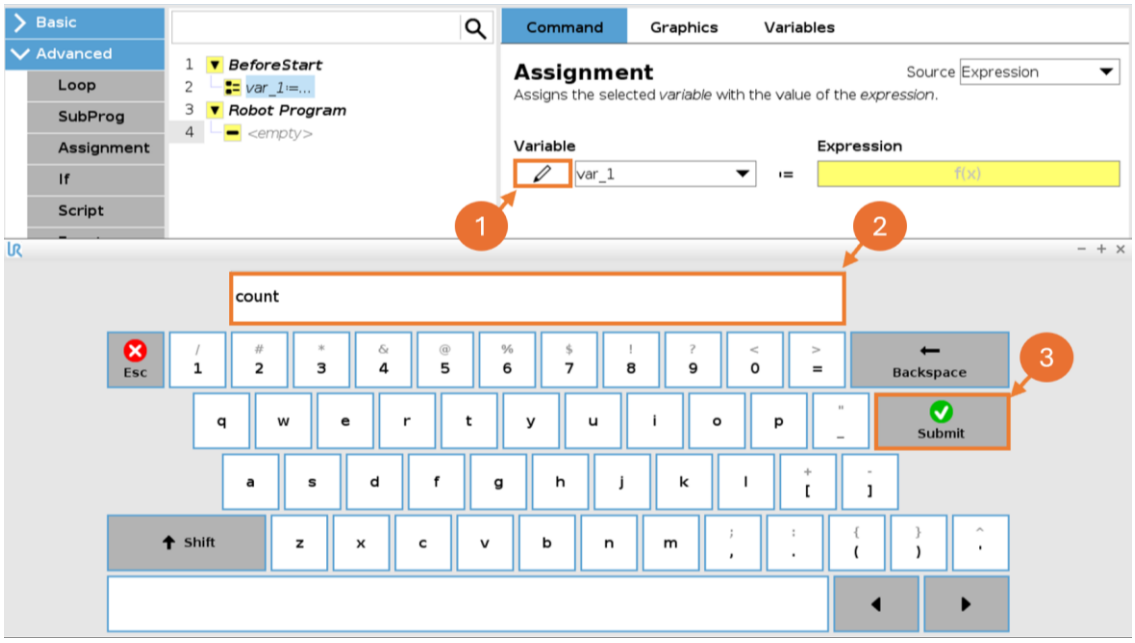


Figura 48 - Atribuição do nome à variável

Após a atribuição do nome à variável, procede-se com a atribuição do valor inicial da variável, que neste caso é zero (Figura 49).

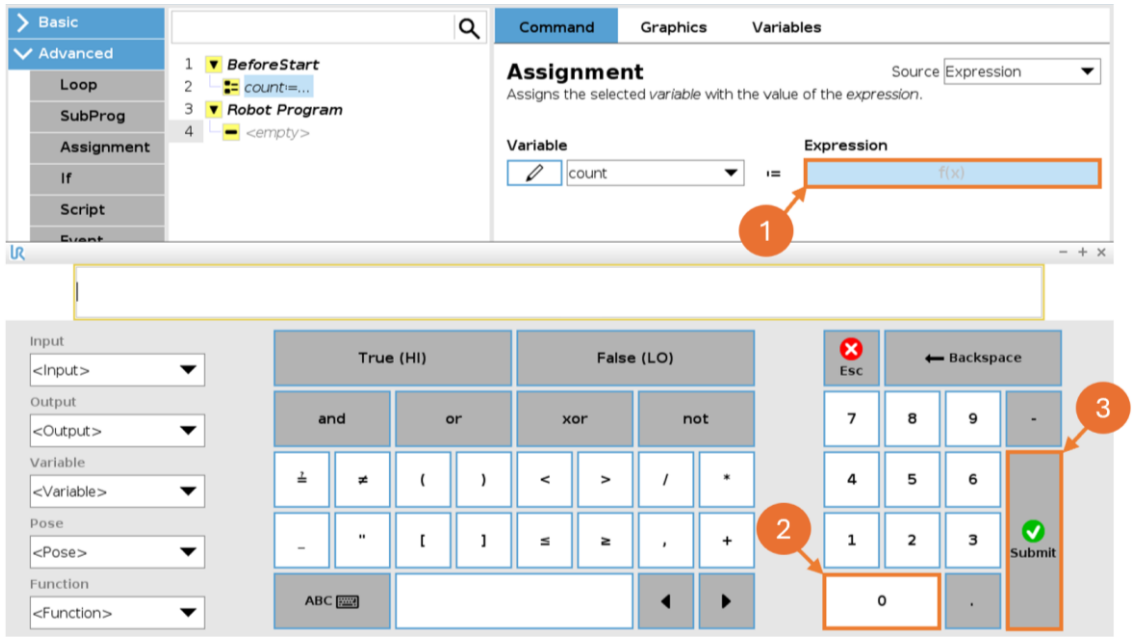


Figura 49 - Atribuição do valor inicial à variável

Nesta fase, a variável contador encontra-se criada. Assim, torna-se possível "chamar" a variável e incrementar o seu valor. Para tal, deve-se inserir uma instrução *Assignment* e, no campo de expressão, colocar a expressão pretendida. Na Figura 50 está presente uma representação do editor de expressões. Independentemente da instrução, o campo de expressões funciona da mesma forma e é composto pela zona de seleção de sinais e variáveis, a zona da expressão e os teclados que permitem editar a expressão.

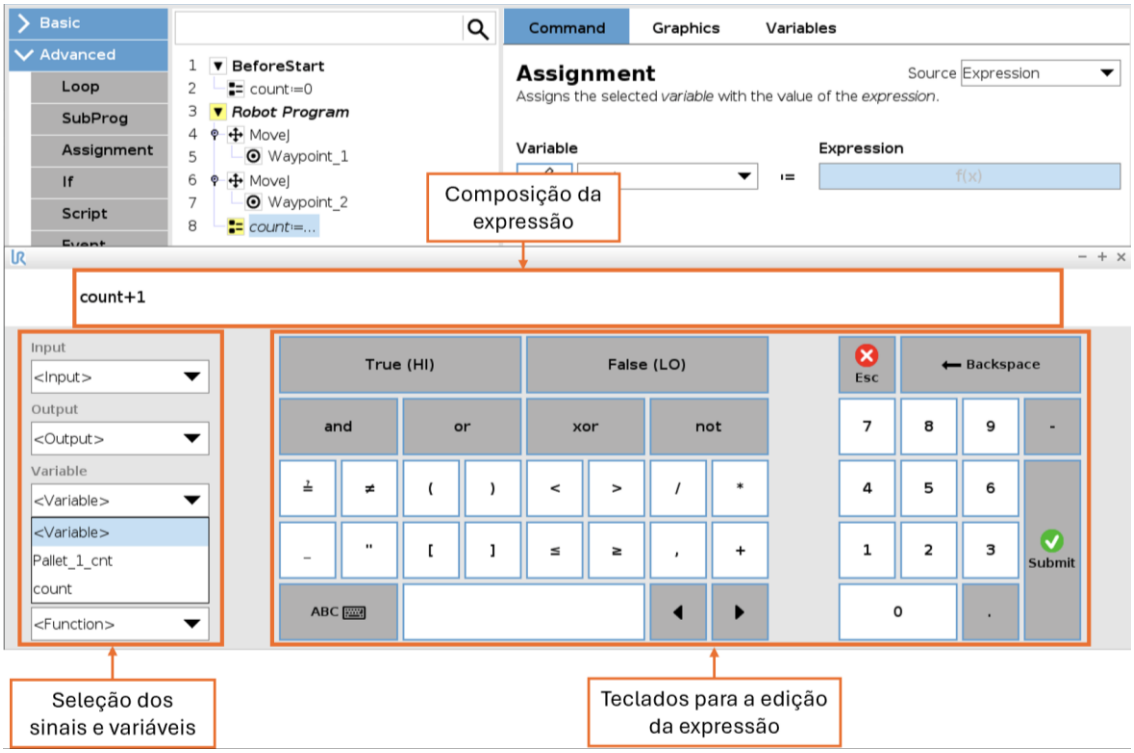


Figura 50 - Editor de expressões

5.6.8. Instrução Popup

A instrução *Popup* permite apresentar mensagens ao operador durante a execução do programa, funcionando como uma ferramenta de comunicação e controlo. Na Figura 51, observa-se a interface de configuração desta instrução, onde o utilizador pode escolher entre dois modos de entrada: **Texto** ou **Variável**. Ao optar por Texto, insere diretamente a mensagem que se pretende exibir. Se selecionar Variável, utiliza uma variável previamente definida para gerar o conteúdo da mensagem, o que permite maior flexibilidade e adaptação a diferentes contextos operacionais.

A interface oferece três tipos de popup: **Mensagem**, **Aviso** e **Erro**. A **Mensagem** serve para informar, o **Aviso** para alertar o operador de uma situação que requer atenção e o **Erro** indica uma condição crítica que pode exigir intervenção imediata. Além disso, esta instrução permite interromper a execução do programa ao apresentar o *popup* "*Halt Program execution at this popup*".

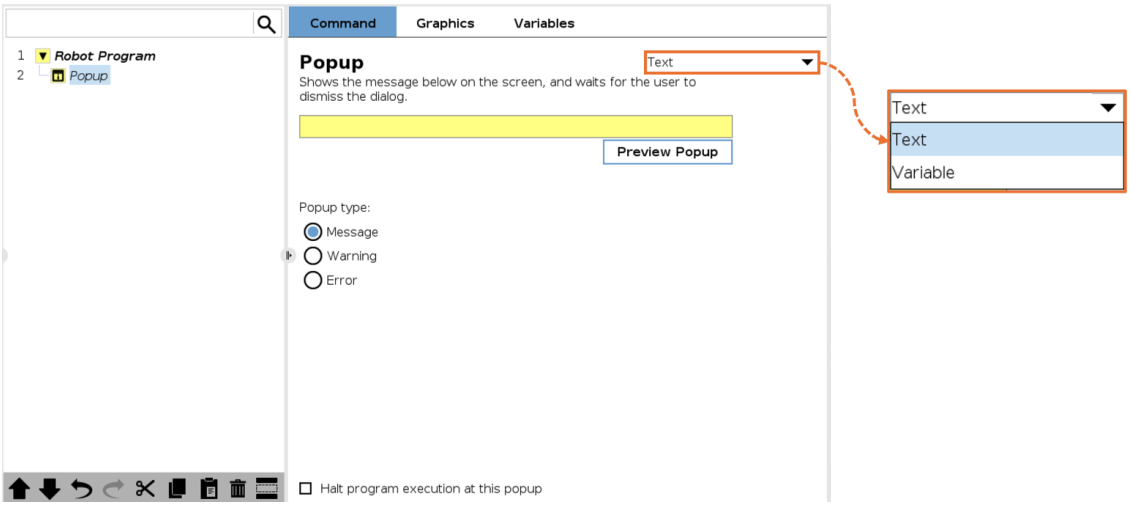


Figura 51 - Apresentação da instrução *Popup*

5.6.9. Instruções de controlo e fluxo do programa

As instruções de controlo e fluxo de um programa são utilizadas para tomar decisões e direcionar a execução para zonas específicas do código, conforme os sinais recebidos ou valores de variáveis.

As instruções condicionais *If*, *ElseIf* e *Else* são geralmente utilizadas na maioria das linguagens de programação. Neste caso, na linguagem da interface PolyScope, são utilizadas da mesma forma. De seguida é descrito o funcionamento de cada uma das instruções:

- ***If***: Avalia o valor de um sinal ou variável. Se esse valor for verdadeiro, as linhas de código associadas são executadas.
- ***ElseIf***: Permite executar determinadas linhas de código se a condição for satisfeita. Junto com a instrução If e/ou outras instruções Elself, é possível criar um programa de seleção com base no valor de uma variável.
- ***Else***: Executa as linhas de código associadas se nenhuma das condições anteriores (If ou Elself) for satisfeita.

Na Figura 52 apresenta-se um exemplo de aplicação deste tipo de instruções de controlo. Nestas, em função do valor de uma variável denominada contador, executa-se o primeiro, segundo ou terceiro subprograma. Se o número não estiver entre 1 e 3, a interface exibe um aviso que indica “Contagem concluída”.

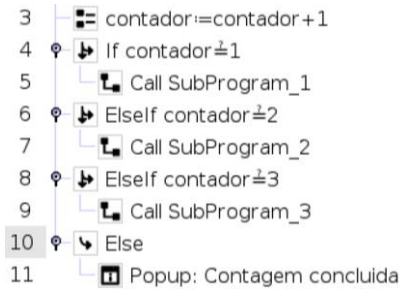


Figura 52 - Exemplo do uso da condicional *If*

A instrução condicional ***Switch*** permite tomar decisões de controlo de fluxo com base no valor de uma variável. O seu modo de utilização é semelhante ao da condicional *If* em conjunto com *ElseIf*, no entanto de forma mais simplificada. A instrução lê o valor de uma variável e, com base nesse valor, executa cada um dos casos atribuídos. Se o valor da variável não corresponder a nenhum deles, é possível executar o *Default Case* que corresponde a um caso padrão. A Figura 53 apresenta o exemplo anteriormente resolvido com a estrutura condicional *If*, porém, neste caso, foi utilizada a instrução *Switch*.

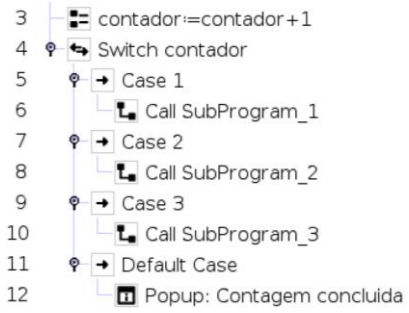


Figura 53 – Exemplo do uso do *Switch*

5.6.10. Instrução *Wait*

A instrução *Wait* é amplamente utilizada na programação de robôs, pois permite introduzir pausas controladas na execução do programa. Esta funcionalidade é essencial para realizar compassos de espera prévios à ocorrência de determinados eventos ou condições, como a receção de sinais externos provenientes de entradas digitais ou analógicas ou pelo decurso de um intervalo de tempo definido. A opção de aguardar um número específico de segundos revela-se particularmente útil em operações que envolvem o funcionamento simultâneo com atuadores externos, como *grippers*. Nestes casos, para garantir que o robô segura a peça antes de prosseguir para a próxima instrução, é comum inserir uma instrução *Wait* de, por exemplo, 1 segundo. Sem essa pausa, a execução seria tão rápida que o robô avançaria para a instrução seguinte antes que o movimento de fecho da garra estivesse concluído. A Figura 54 apresenta o ecrã de configuração desta instrução no ambiente de programação, onde é possível definir o tempo de espera, bem como condições associadas a sinais digitais, analógicos ou expressões lógicas personalizadas.

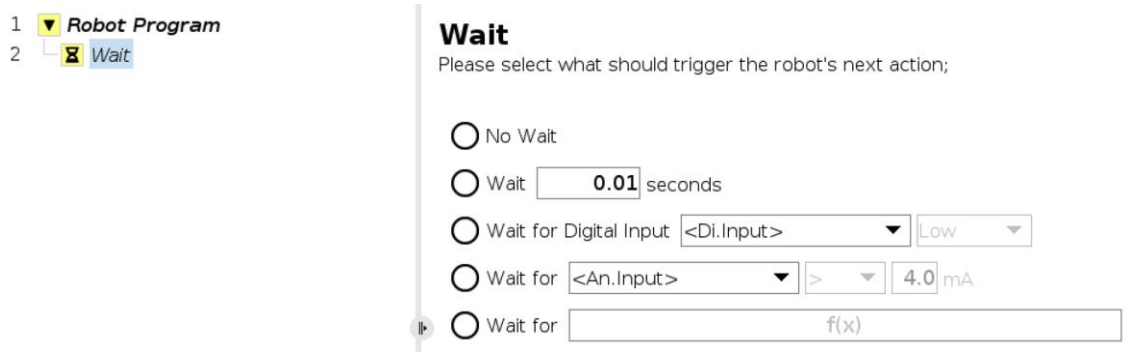


Figura 54 - Instrução Wait

5.6.11. Leitura de entradas digitais

A leitura de entradas digitais na estrutura de um programa pode ser feita de várias formas, das quais se destacam as seguintes:

- **Leitura do sinal em instruções de controlo de fluxo**: Neste caso, se o valor do sinal for verdadeiro (True), o bloco de programa dependente da condição é executado, como por exemplo numa instrução *If*. No exemplo presente na Figura 55, se a entrada digital 1 for ativa, o programa excuta a trajetória 1. Por outro lado, se a entrada digital 2 for ativa, o programa executa a trajetória 2.

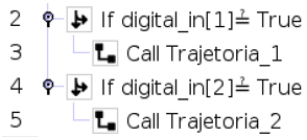


Figura 55 - Leitura de entradas digitais na instrução *If*

- **Espera por um valor na entrada digital:** A execução é interrompida numa linha do programa até que o valor do sinal seja avaliado como LO (Baixo) ou HI (Alto). Neste caso, a execução do programa principal fica a aguardar até que essa linha de espera seja verificada. Este comportamento pode ser observado na Figura 56, que apresenta o modo de utilização da instrução *Wait*, configurada para aguardar um sinal *High* (alto) na entrada digital *digital_in[1].*

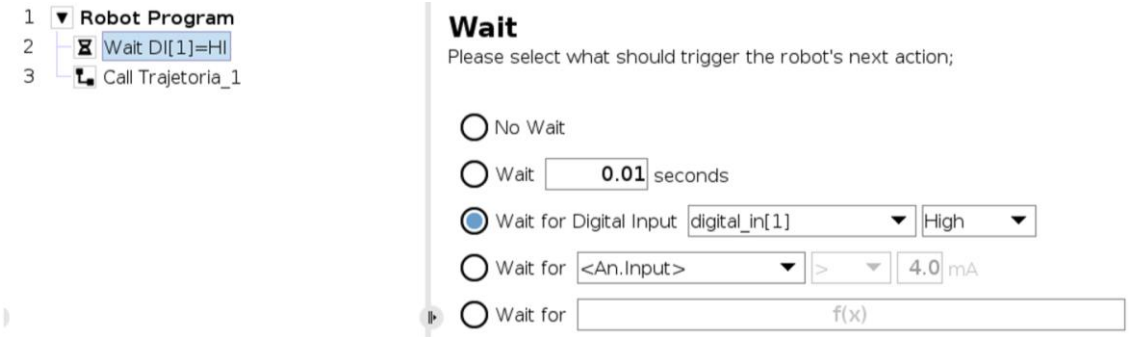


Figura 56 - Leitura de entradas digitais na instrução *Wait*

5.6.12. Leitura de saídas digitais

As saídas digitais podem ser lidas e escritas de forma semelhante às entradas digitais. A sua inserção no programa é realizada através da instrução *Set,* que se encontra no menu básico de programação (Basic). Na Figura 57 é possível visualizar as diferentes opções disponíveis para a configuração desta instrução.

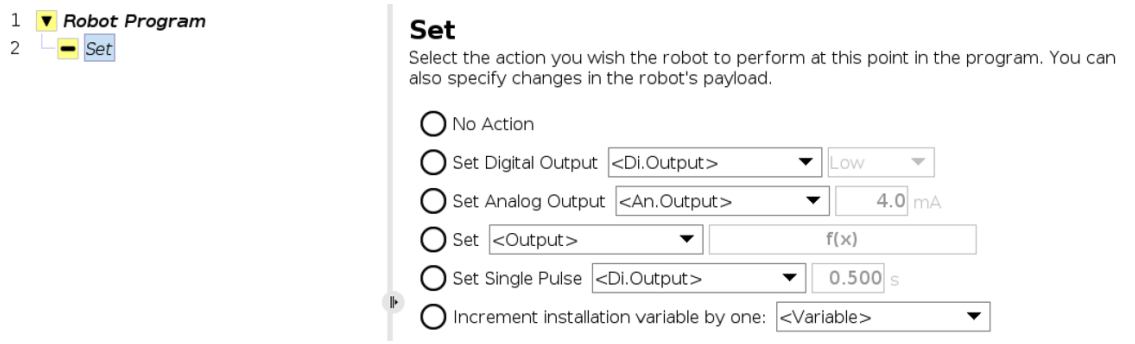


Figura 57 - Leitura de saídas digitais com instrução *Set*

Existem três formas principais de utilização das saídas digitais, designadamente:

- **Set Digital Output** (ajustar saída digital): Neste caso, ativa-se ou desativa-se a saída digital pretendida, utilizando os sinais *High* para ativar e *Low* para desativar.
- **Set** (ajustar saída com função lógica): Controla uma saída com base numa função lógica, por exemplo, ativar uma saída digital somente quando uma determinada entrada digital estiver ativa.
- **Set Single Pulse** (ajustar pulso único): Ativa uma saída com um pulso de flanco positivo, cujo tempo é ajustável.

5.6.13. Leitura de entradas analógicas

A leitura de entradas analógicas no ambiente PolyScope é realizada de forma direta e contínua. Na bancada desenvolvida há um potenciómetro conectado à entrada analógica 1, o que significa que qualquer variação no valor do potenciómetro, seja por ajuste manual ou alteração na resistência, é imediatamente refletida na aba de I/O do sistema.

A leitura em tempo real da entrada analógica permite monitorizar com precisão os sinais analógicos recebidos, facilitando o controlo de processos que dependem de valores variáveis como intensidade de luz, temperatura, posição ou velocidade de movimento do robô (exemplo de exercício apresentado no subcapítulo 6.2).

5.6.14. Leitura de saídas analógicas

No ambiente PolyScope, as saídas analógicas podem ser utilizadas para enviar sinais contínuos ou variáveis para dispositivos externos, como voltímetros. Ao contrário das entradas analógicas, que são lidas diretamente, as saídas analógicas são definidas ou ajustadas por comandos no programa. No entanto, é possível monitorizar o valor atual da saída através da aba de I/O, o

que permite verificar se o sinal enviado corresponde ao valor programado. As saídas analógicas são configuradas através da instrução *Set*. O subcapítulo 6.2 apresenta um exercício no qual o valor de um contador é vinculado à saída analógica zero, que está conectada ao voltímetro do sistema.

5.6.15. Posicionamento com variáveis

A linguagem de programação da UR permite utilizar pontos de passagem, ou destinos, com base nos valores de variáveis. Para isso, é necessário criar uma variável de posição (*pose*) e, posteriormente, atribuir valores a cada um dos elementos dessa variável.

Variável *Pose* (posição)

Uma variável do tipo *Pose* é representada por uma matriz 1x6, na qual as primeiras três colunas são de posição (X, Y, Z) e as três restantes de rotação (rX, rY, rZ). Os valores de posição são dados em metro e os de rotação em radianos. Cada elemento deve manter o formato de número real e pode ser atribuído com variáveis. No subcapítulo 6.2 apresentam-se dois exemplos de exercícios resolvidos que utilizam este tipo de configuração para a movimentação do robô.

5.6.16. Instrução *Seek*

A instrução *Seek* permite realizar operações automáticas de empilhamento e desempilhamento de peças, através de parâmetros previamente definidos como direção, distância máxima e sequência de movimentos, considerando alguns graus de liberdade como força e localização, por exemplo.

5.6.17. Instrução *Palletizing* (paletização)

A instrução *Palletizing* permite realizar a paletização de peças em três padrões diferentes, designadamente linha, grelha e irregular (diferentes orientações). A instrução é constituída por uma estrutura de programação pré-definida que deve ser devidamente parametrizada pelo utilizador. No *software* PolyScope do UR3e é possível programar diferentes estratégias de paletização para otimizar o espaço e a eficiência do processo. A Figura 58 ilustra quatro exemplos de configurações.

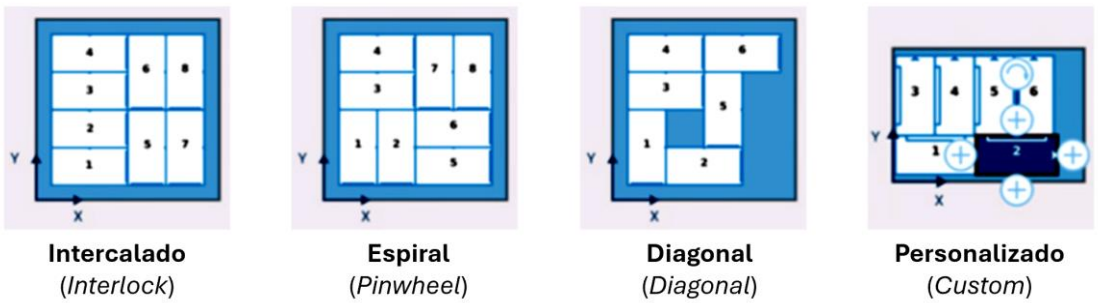


Figura 58 - Exemplos de padrões de paletização

Estas configurações podem ser implementadas diretamente no PolyScope, através da opção de paletização, definindo o número de camadas, filas, colunas, e deslocamentos nos eixos X, Y e Z.

5.6.18. Instruções do *Gripper*

As instruções para o acionamento do *gripper* encontram-se na secção URCaps de programação, mais precisamente no módulo *Zimmer*. As duas operações mais utilizadas correspondem à abertura e ao fecho das garras do *gripper*. Segue-se a apresentação destas instruções:

- **Abrir:**
    - Z_Release: Gripper 1.
- **Fechar:**
    - Z_Grip: Gripper 1.

5.7. Desafios enfrentados e como foram resolvidos

Apesar de o PolyScope apresentar uma interface intuitiva, a falta de formação e de conhecimento na área tornou o processo de aprendizagem totalmente autónomo. Para ultrapassar esta dificuldade, foi necessário recorrer ao manual de utilizador da UR, à documentação técnica disponível no *website*, a vídeos e tutoriais do curso *e-learning*, além de outros recursos digitais que ajudaram a complementar o conhecimento. Aprender a linguagem e a lógica de programação do sistema foi essencial para o desenvolvimento dos exercícios didáticos. Este esforço não só permitiu implementar corretamente as rotinas, como também ajudou a criar soluções mais eficientes, com movimentos mais fluídos, maior segurança e precisão.

Para validar as funcionalidades do protótipo e potenciar a sua aplicação em contexto pedagógico foi desenvolvida uma componente didática constituída por sete exercícios, organizados de forma a abranger os principais tópicos de interação. Estes exercícios foram testados em ambiente real, o que garante a fiabilidade da solução. A descrição detalhada de cada exercício, bem como a análise dos resultados obtidos é apresentada no subcapítulo 6.2. Neste, demonstra-se a forma como a bancada cumpre os objetivos inicialmente propostos.

6. Resultados obtidos

Este capítulo apresenta os principais resultados obtidos ao longo do desenvolvimento do projeto. Inicialmente, é descrito o protótipo construído, com destaque para os objetivos definidos e as funcionalidades implementadas. De seguida, são detalhados os testes realizados, que abrangem desde a execução de diferentes tipos de movimentos até à interação com entradas e saídas digitais e analógicas. São também exploradas operações mais complexas, como ciclos de *Pick and Place*, paletização e empilhamento de peças, recorrendo à utilização de funções específicas. Por fim, são apresentados e analisados os resultados obtidos em cada exercício, evidenciando o desempenho e a eficácia do sistema desenvolvido.

6.1. Protótipo desenvolvido

O protótipo final da bancada didática desenvolvida no âmbito deste projeto corresponde à materialização prática do projeto descrito no capítulo anterior e integra o robô colaborativo UR3e, um conjunto de módulos de controlo e todos os elementos estruturais e elétricos. No Apêndice C apresentam-se imagens reais das fases de montagem da bancada e no Apêndice D apresenta-se um desenho de definição com as cotas principais da bancada.

Na Figura 59 é apresentada a versão final da bancada, totalmente montada e em funcionamento, onde se observa a integração e disposição de todos os componentes.

Figura 59 - Protótipo final da bancada didática

6.1.1. Objetivos do projeto

A definição dos objetivos constitui uma etapa essencial no desenvolvimento de um projeto. Pretendia-se desenvolver uma bancada didática resistente, flexível, modular e de fácil deslocação, capaz de suportar e integrar o robô colaborativo UR3e e diferentes periféricos. Outro objetivo fundamental passou por garantir que a bancada fosse segura, ergonómica e confortável, adequada para contextos laboratoriais e pedagógicos. Paralelamente, procurou-se assegurar a integração completa do UR3e, com acesso às suas interfaces, e preparar a estrutura para a possibilidade de implementação futura de novas tecnologias.

Do ponto de vista pedagógico, estabeleceu-se como meta criar condições para o ensino da programação de robôs colaborativos. Os objetivos centrais deste tópico focam-se na introdução de instruções básicas de programação e na familiarização dos utilizadores (alunos) com a linguagem de programação utilizada no PolyScope. Deste modo, pretende-se capacitar os alunos de competências para projetar, programar e testar diferentes exercícios com o robô UR3e, aplicando conhecimentos de programação robótica, lógica condicional e I/O digitais. Pretende-se que, no final, os alunos desenvolvam competências nas seguintes áreas:

- Definição e organização das posições de recolha e colocação de objetos;
- Controlo de entradas e saídas digitais e analógicas;
- Simulação de tarefas industriais, como operações de manipulação de peças como *Pick and Place* e paletização;
- Organização da programação de forma modular;
- Teste e validação das aplicações.

Com base nestes objetivos, foi desenvolvido o protótipo apresentado na Figura 59. A bancada didática construída cumpre de forma eficaz os requisitos definidos, o robô colaborativo UR3e foi integrado de forma segura, com acesso direto ao controlador e aos respetivos periféricos. A cablagem foi organizada internamente, o que contribuiu para a segurança e arrumação do sistema. A bancada possibilita o uso de entradas e saídas analógicas e digitais, a simulação de diversos cenários industriais e o desenvolvimento da programação, tanto com o *software* PolyScope como com outras plataformas compatíveis com o cobot. Encontra-se ainda preparada para futuras expansões com novos dispositivos e tecnologias. Assim, é possível concluir que todos os objetivos propostos foram atingidos.

6.1.2. Funcionalidades implementadas

O protótipo desenvolvido incorporou diversas funcionalidades corroborando os objetivos inicialmente definidos. Entre elas destaca-se a possibilidade de programação do robô UR3e, quer através da interface gráfica PolyScope, quer através de instruções em URScript. Além disso, a bancada permite o controlo de entradas e saídas digitais e analógicas, o que possibilita a interação com dispositivos externos como *Leds*, botões, potenciómetro e voltímetro.

Foi também assegurada a capacidade de simular tarefas industriais, nomeadamente de *Pick and Place*, paletização e empilhamento, aproximando o ambiente laboratorial a contextos reais da indústria. Outra funcionalidade de destaque é a possibilidade de controlar remotamente o robô através de uma ligação Ethernet e do *software* nativo da UR, o que amplia a flexibilidade do sistema. Por fim, a bancada foi projetada de modo a facilitar a integração de novos módulos, como PLCs ou sistemas de visão, o que reforça a sua característica de plataforma inovadora, modular e flexível.

Segurança da Bancada

A segurança funcional da bancada didática foi uma preocupação central no seu desenvolvimento, de modo a garantir não apenas a integridade dos utilizadores durante as sessões laboratoriais, mas também a proteção dos equipamentos. Para tal, foram implementadas as seguintes medidas:

Paragem de Emergência (E-Stop):

- A bancada integra um botão de emergência tipo cogumelo (Figura 60), com atuação por pressão e bloqueio mecânico, colocado na zona frontal para acesso imediato pelo utilizador e pelo formador;
- O circuito do E-Stop está ligado ao conector de segurança do UR3e, atuando sobre o sinal *Enable/Protective Stop* do controlador;
- Ao ser acionado, o UR3e entra em estado de paragem segura (Safe Stop 1 → STO), interrompendo o movimento de todos os eixos e garantindo ausência de binário nos motores.

Figura 60 - Botão de emergência (E-Stop) presente no Teach Pendant

Para além do sistema de paragem de emergência, a segurança do operador aquando do manuseamento da bancada é garantida quando este se encontra fora do espaço operacional do robô. Na Figura 61 representa-se o espaço operacional do robô, definido pelo alcance máximo do braço robótico (500 mm em cada direção, resultando num diâmetro total de 1000 mm). Este espaço integra a zona de trabalho colaborativo, onde o robô atua em proximidade com o operador. De acordo com a norma ISO/TS 15066 [18], a definição destas zonas constitui um requisito essencial para a avaliação de riscos e para a implementação de medidas de segurança adequadas. Apesar de os robôs colaborativos serem desenvolvidos para operar em

proximidade com os seres humanos, a sua utilização segura depende do cumprimento rigoroso dos limites estabelecidos, o que exige constante atenção durante a execução das tarefas.

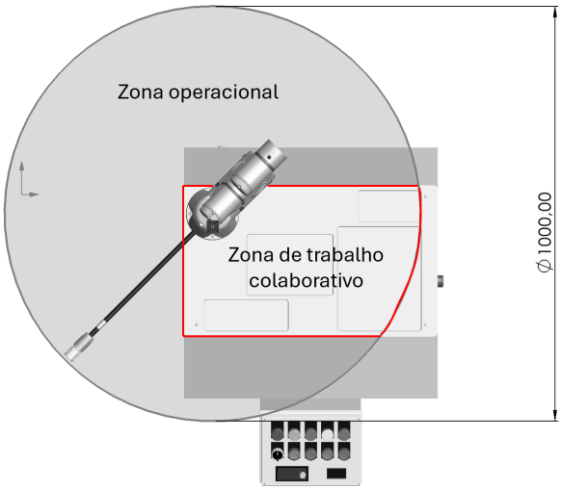


Figura 61 – Representação do espaço operacional do robô e do espaço de trabalho colaborativo

6.2. Testes realizados

Com o objetivo de consolidar os conhecimentos adquiridos e demonstrar as funcionalidades do sistema, foram desenvolvidos diversos exercícios práticos que abrangem desde movimentos básicos do robô até a interação com entradas e saídas digitais e analógicas. Estes exercícios permitem explorar diferentes instruções de programação, testar ciclos condicionais e integrar o funcionamento de periféricos, como o *gripper*. Foram desenvolvidos 13 exercícios didáticos, testados em ambiente real na bancada. No entanto, optou-se por apresentar apenas sete exercícios, que se consideraram suficientes para demonstrar a aplicabilidade e os objetivos pedagógicos do projeto. Estes e os restantes exercícios encontram-se documentados e detalhados no Apêndice B.

Na Tabela 17 apresentam-se as competências técnicas e operacionais a ser adquiridas aquando da realização dos sete exercícios apresentados de seguida.

Tabela 17 - Competência técnicas e operacionais associadas a cada exercício

	Competências Técnicas	Competências Operacionais
Exercício 1	• Compreender e configurar diferentes tipos de movimento (*MoveJ*, *MoveL*, *MoveP*); • Definir *Waypoints*.	• Interpretar de forma clara as diferentes trajetórias.
Exercício 2	• Programar condições operacionais com instruções lógicas (*If/ElseIf*); • Configurar e testar entradas e saídas digitais (DI/DO); • Utilizar instruções de *Popup* para interação com o utilizador; • Gerar mensagens na interface.	• Reconhecer as diferenças entre respostas a sinais externos; • Comunicar de forma clara com o operador durante a execução do programa; • Representar o estado do sistema com sinais visuais.

Tabela 17 - Competência técnicas e operacionais associadas a cada exercício (continuação)

Exercício 3	• Utilizar contadores e variáveis lógicas; • Implementar ciclos de *Pick and Place;* • Programar subprogramas com *Palletizing* e *Seek;* • Utilizar parâmetros de desempilhamento e paletização.	• Automatizar processos com repetição estruturada.
Exercício 4	• Modificar a lógica de funcionamento de programas previamente desenvolvidos; • Configurar e testar saídas analógicas; • Relacionar variáveis internas do programa (contador) com sinais analógicos (voltímetro).	• Efetuar leituras no voltímetro e interpretar os valores medidos; • Confirmar correspondência entre o comportamento do programa e o resultado obtido no instrumento de medida; • Desenvolver autonomia na execução de ajustes e testes ao programa.
Exercício 5	• Definir e configurar variáveis para leitura e manipulação de sinais analógicos; • Implementar funções de conversão de valores contínuos em parâmetros de velocidade; • Programar restrições e limites de segurança em variáveis numéricas; • Utilizar comunicação via *socket* para atualização dinâmica de parâmetros de controlo; • Configurar e testar entradas analógicas; • Executar movimentos repetitivos entre pontos predefinidos (Waypoints) com velocidade variável.	• Ajustar a velocidade do robô em tempo real através da interação com o potenciómetro; • Monitorizar o comportamento do sistema para não ultrapassar os limites de segurança; • Desenvolver sensibilidade para interpretar a resposta do robô face às variações de velocidade; • Demonstrar capacidade de intervenção manual para interromper o processo de forma segura; • Gerir a execução contínua do programa garantindo estabilidade e suavidade no movimento.
Exercício 6	• Programar trajetórias baseadas em funções trigonométricas (sin() e cos()); • Definir pontos sucessivos através de incrementos angulares para percorrer curvas aproximadas; • Implementar variáveis e parâmetros de controlo (ângulo θ, raio, incrementos); • Desenvolver sub-rotinas que permitam desenhar formas geométricas no plano de trabalho.	• Interpretar e validar trajetórias geométricas representadas por pontos discretos; • Controlar a qualidade do percurso realizado, para assegurar correspondência entre o trajeto programado e a geometria prevista; • Avaliar os efeitos da variação do incremento angular e da velocidade de movimento na suavidade e precisão da trajetória.
Exercício 7	• Programar ciclos de empilhamento e desempilhamento com controlo de variáveis; • Desenvolver subprogramas para modularizar operações de recolha e posicionamento de peças; • Implementar rotinas de *Pick and Place* otimizadas para múltiplas zonas de trabalho.	• Organizar fluxos de trabalho estruturados para manipulação de peças; • Gerir diferentes zonas de recolha e posicionamento de forma clara e sistemática; • Demonstrar capacidade de planear tarefas industriais aplicáveis a contextos reais de logística e produção.

Exercício 1 – Tipos de movimentos (*MoveJ, MoveL* e *MoveP*)

Este exercício tem como objetivo analisar e comparar os três tipos de movimento disponíveis na instrução ***Move***: ***MoveJ***, ***MoveL*** e ***MoveP***, sendo que cada um corresponde a um método distinto de interpolação dos eixos do robô. A implementação e execução de cada tipo de movimento permite observar o comportamento prático do robô.

Requisitos do exercício:

- Criar um programa e denominá-lo como *exercício_1*;
- Inserir 3 instruções *Move*;
- Associar os movimentos *MoveJ, MoveL* e *MoveP* às 3 instruções *Move*;
- Definir a posição dos *Waypoints*;
- Renomear os *Waypoints* como *ponto_1*, *ponto_2* e *ponto_3*, respetivamente.

Possível resolução:

A Figura 62 apresenta uma possível solução para o exercício. Neste caso, o robô executa um movimento em arco entre os pontos 1 e 2, um movimento linear entre os pontos 2 e 3 e termina num movimento planeado entre os pontos 3 e 1. O ciclo do programa é contínuo sendo necessário parar o programa de forma manual.

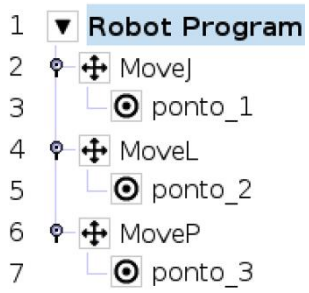


Figura 62 - Resolução do exercício 1

Resultados esperados:

Após a resolução do exercício o esperado é que os utilizadores sejam capazes de compreender e configurar diferentes tipos de movimento, definir *Waypoints* e interpretar de forma clara as diferentes trajetórias.

Exercício 2 – Interação com I/O digitais e mensagens

Este exercício introduz o conceito de interação com o sistema de I/O (entradas/saídas digitais). Pretende-se que o robô reaja a uma entrada digital externa (DI1), ou seja, só pode executar o movimento e ativar o Led verde (DO0) quando o botão verde (DI1) é acionado. Caso o botão verde não seja acionado, deve surgir uma mensagem no ecrã do Teach Pendant a alertar o utilizador. A interação deve ser feita através de uma janela de mensagem com o texto: “Pressione o botão verde”.

Requisitos do exercício:

- Inserir uma secção de lógica condicional que verifique o estado da entrada digital:
    - Se *DI1 = True*:
        - O programa realiza os movimentos definidos pelo utilizador;
        - Ativar o *LED* verde (DO0) como primeira ação.
    - Se *DI1 = False*:
        - Inserir a mensagem "Pressione o botão verde", para alertar o utilizador;
        - O programa espera até ser verdadeiro.

Possível resolução:

Na Figura 63 apresenta-se uma possível solução para o exercício. Neste caso, através de instruções de controlo e fluxo de programa verifica-se o estado da entrada digital 1. Quando a entrada é ativada, o programa liga o *Led* verde, executa dois movimentos e desliga o *Led* verde. Quando a entrada se encontra desativa, surge uma mensagem no Teach Pendant com a mensagem "Pressione o botão verde", seguindo uma instrução de espera que só é superada quando o botão verde é pressionado.

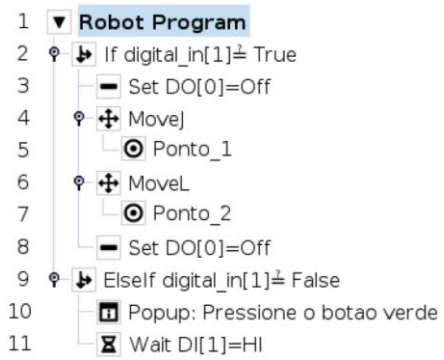


Figura 63 - Resolução do exercício 2

Resultados esperados:

Após a resolução deste exercício, espera-se que os utilizadores desenvolvam competências na programação de condições operacionais com instruções lógicas, bem como na configuração e teste de entradas e saídas digitais. Pretende-se ainda que sejam capazes de reconhecer as diferenças entre respostas a sinais externos, comunicar de forma clara com o operador através da geração de mensagens na interface e representar o estado do sistema através de sinais visuais, como a ativação de *Leds*.

Exercício 3 – Ciclo *Pick and Place* e paletização com contador

O objetivo deste exercício é desenvolver uma aplicação de paletização automática utilizando o robô colaborativo UR3e. O robô deve realizar operações de *Pick and Place* para transferir peças da zona de recolha (zona A) para a zona de paletização (zona B), de forma ordenada e repetitiva.

A tarefa deve ser executada até que cinco peças sejam colocadas na zona B, parando automaticamente após a última colocação. Na Figura 64 encontra-se a representação da superfície de trabalho com as zonas A e B assinaladas. A zona A contém 5 peças empilhadas de diferentes cores. A zona B está subdividida em cinco compartimentos onde as peças deverão ser colocadas.

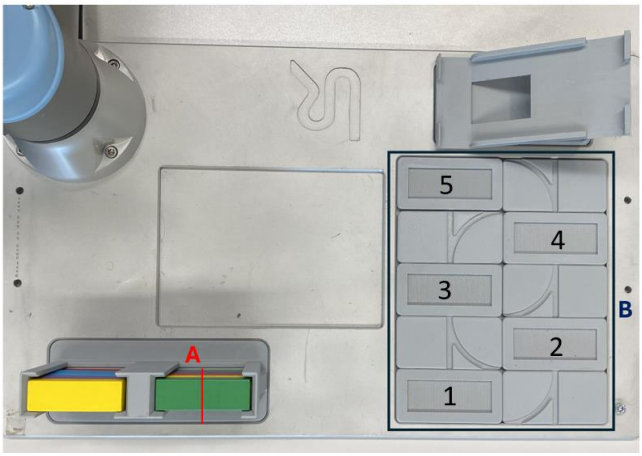


Figura 64 - Sinalização da superfície de trabalho

Requisitos do exercício:

1. A execução do programa deve iniciar apenas quando o botão verde (DI1) for pressionado;
2. Após o sinal de início, o robô deve deslocar-se para a posição acima da zona A, que deve ser definida como *waypoint* *before_start*;
3. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher (definir como *waypoint* *start_point*);
4. Acionar o *gripper* para agarrar a peça;
5. Aguardar 1 segundo;
6. Efetuar o movimento linear inverso até à posição *before_start*;
7. Transportar a peça até à zona B e colocá-la na próxima posição livre (de 1 a 5);
8. Desativar o *gripper* para largar a peça;
9. Aguardar 1 segundo;
10. Incrementar o contador de peças paletizadas;
11. Repetir o ciclo até que cinco peças tenham sido colocadas;
12. Durante todo o processo, manter o LED verde (DO0) ligado;
13. Após a colocação da 5ª peça, o programa deve terminar automaticamente.

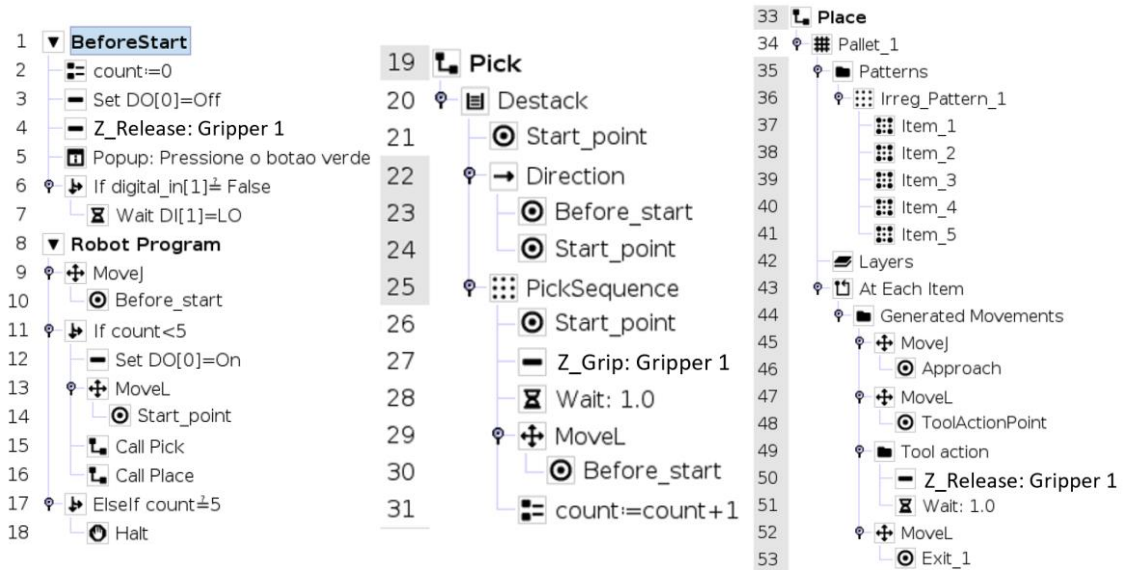
Possível solução 1:

A Figura 65 apresenta uma das duas soluções desenvolvidas para o presente exercício. Nesta implementação, o valor inicial do contador foi definido na secção *Before Start* (a), garantindo-se também que o LED inicia desligado e que o *gripper* possui as garras abertas. De seguida,

através de uma instrução lógica condicional, assegura-se que o programa apenas inicia após o acionamento do botão verde.

No corpo principal do programa, o robô foi posicionado inicialmente acima da zona de recolha, sendo inserida uma estrutura de controlo e fluxo do tipo *If* e *ElseIf* para verificar duas condições distintas. Enquanto o contador apresenta um valor inferior a cinco, executa-se o subprograma *Pick* (b), responsável pela recolha das peças, seguido do subprograma *Place* (c), encarregue pelo seu posicionamento. Quando o contador atinge o valor lógico de cinco, o programa é automaticamente interrompido.

Para a configuração do subprograma *Pick*, foi utilizada a instrução *Seek* com a opção *Destack*, que possibilita a despaletização automática das peças. Já para o posicionamento, recorreu-se à instrução *Palletizing*, que simplifica o processo de paletização de forma prática e eficiente.



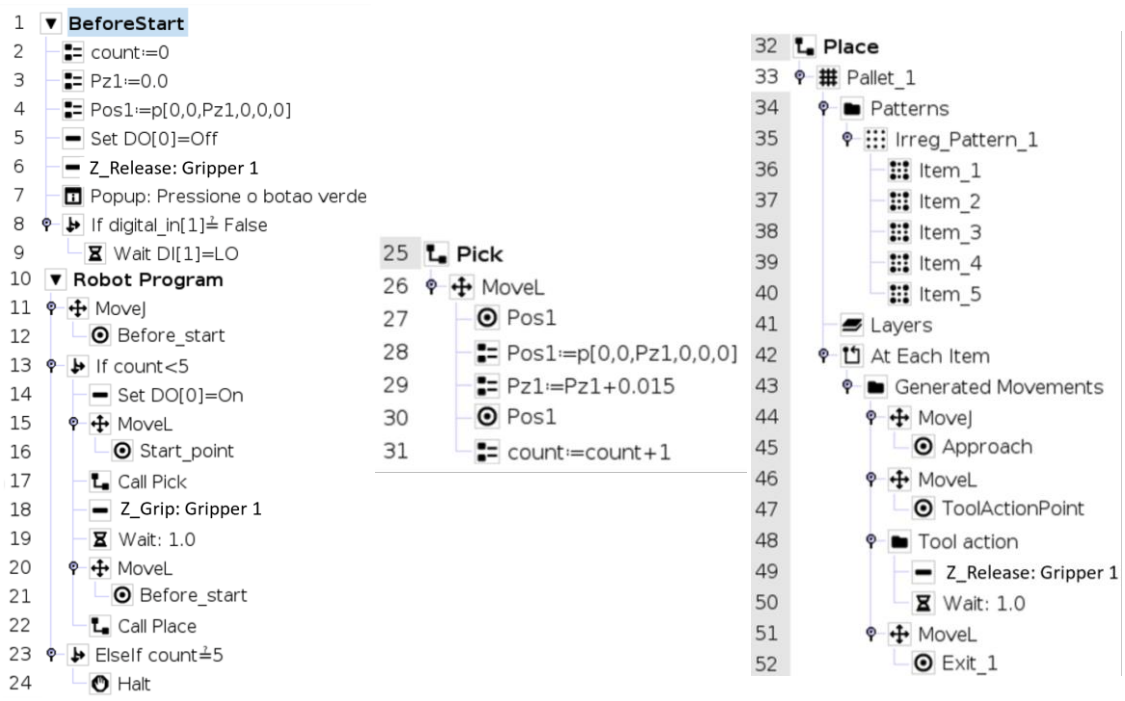
(a) (b) (c)

Figura 65 - Solução 1 do exercício 3: (a) Secção *BeforeStart* e estrutura principal do programa; (b) Subprograma de recolha de peças *Pick*; (c) Subprograma de posicionamento de peças *Place*

Possível solução 2:

Nesta solução, manteve-se a estrutura geral do programa apresentada na solução anterior, alterando-se apenas a configuração do subprograma *Pick*. Nesta abordagem, a despaletização das peças deixa de recorrer à instrução *Seek* e passa a ser realizada a partir de uma posição de recolha previamente definida (Pos1). Esta posição corresponde a uma variável do tipo *pose*, à qual se acrescentam 0,015 m na coordenada Z para cada peça recolhida. Esta alteração implica um controlo mais explícito da trajetória vertical no processo de recolha.

Na Figura 66 apresenta-se a codificação final desenvolvida para o exercício com a nova abordagem na zona de recolha de peças.



(a) (b) (c)

Figura 66 - Solução 2 do exercício 3: (a) Secção *BeforeStart* e estrutura principal do programa; (b) Subprograma de recolha de peças *Pick* editado; (c) Subprograma de posicionamento de peças *Place*

Resultados esperados:

Após a resolução do exercício o esperado é que os utilizadores desenvolvam competências na utilização de contadores e variáveis lógicas, na implementação de ciclos de *Pick and Place* e na programação de subprogramas com as instruções *Seek* e *Palletizing.* Para além disso, pretende-se que sejam capazes de automatizar processos com repetição estruturada.

Exercício 4 – Interação com saídas analógicas

O objetivo deste exercício passa por estabelecer uma relação entre o valor do contador utilizado no exercício 3 e a leitura do voltímetro. Ou seja, à medida que o contador é incrementado, o valor lido no voltímetro deve acompanhar essa variação (ex: contador = 1 → voltímetro = 1.0). Deve ser utilizado o programa desenvolvido no exercício anterior e realizadas as alterações necessárias.

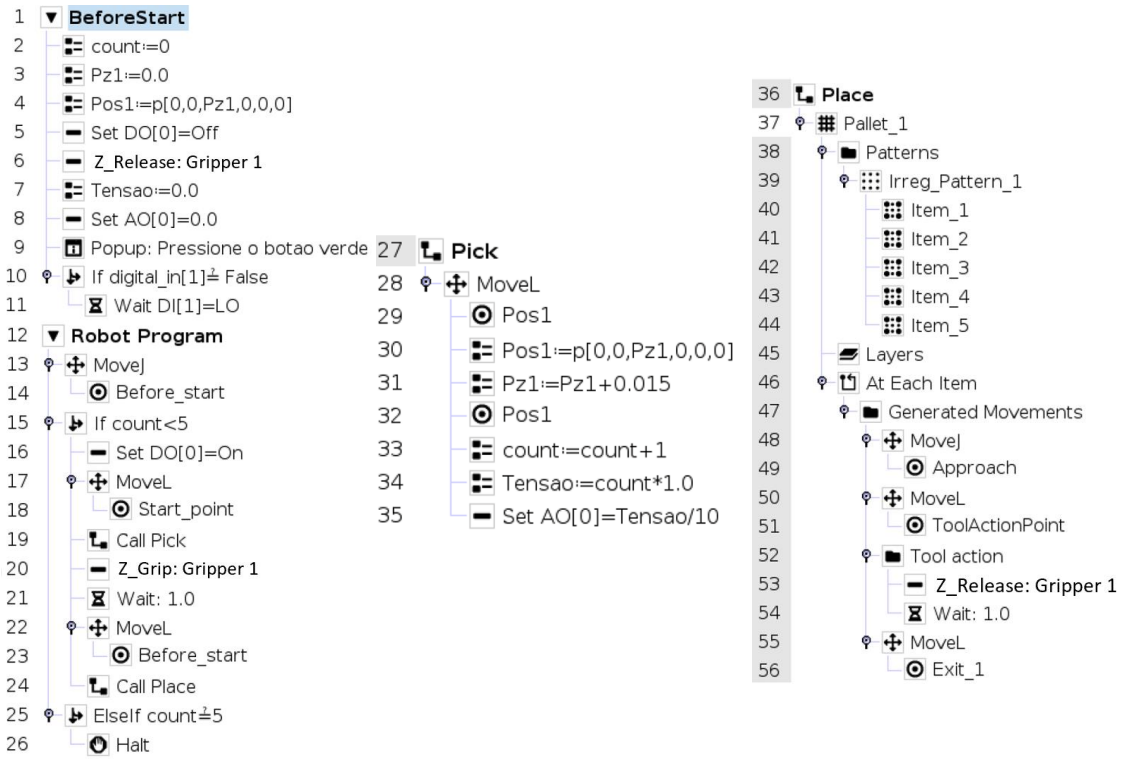
Requisito do exercício:

1. Relacionar o valor do voltímetro com o valor do contador.

Possível solução:

Para a resolução deste exercício utilizou-se como base a segunda resolução do exercício 3. A principal alteração ocorre no subprograma *Pick*, imediatamente após o incremento da variável contador, como se observa na Figura 67. Foram acrescentadas duas instruções: a primeira atribui à variável *Tensao* o valor do contador convertido para número real (Tensao := count * 1.0), enquanto a segunda envia este valor para a saída analógica (Set AO[0] := Tensao/10). A

operação de divisão por 10 assegura que o valor de tensão transmitido ao voltímetro se encontra dentro da faixa operacional adequada, estabelecendo assim a relação matemática pretendida entre o número de ciclos executados e a medição do dispositivo.



(a) (b) (c)

Figura 67 - Solução do exercício 4: (a) Secção *BeforeStart* e corpo do programa principal; (b) Subprograma de recolha *Pick*; (c) Subprograma de posicionamento *Place*

Resultados esperados:

Após a resolução deste exercício, espera-se que os utilizadores desenvolvam competências na configuração e programação de saídas analógicas, bem como na conversão e escalonamento de variáveis internas para gerar sinais compatíveis com a gama de funcionamento do voltímetro. Pretende-se ainda que sejam capazes de calibrar e validar o sinal enviado, de modo a assegurar a correspondência entre o valor registado pelo contador e a leitura obtida no equipamento, prevenir situações de saturação e garantir a fiabilidade da medição.

Exercício 5 – Controlo da velocidade do robô com entrada analógica

O objetivo deste exercício é desenvolver uma aplicação capaz de controlar a velocidade de movimento do robô UR3e através da leitura de um valor analógico proveniente de um potenciómetro. O valor lido deve ser convertido numa fração de velocidade, o que permite ao operador ajustar dinamicamente a velocidade de execução do robô.

O robô deverá realizar um movimento contínuo entre dois pontos predefinidos, variando a sua velocidade conforme o valor do potenciómetro. O valor mínimo de velocidade deve ser de 0.1 (10% da velocidade máxima configurada) e o valor máximo de 1.0 (100% da velocidade máxima).

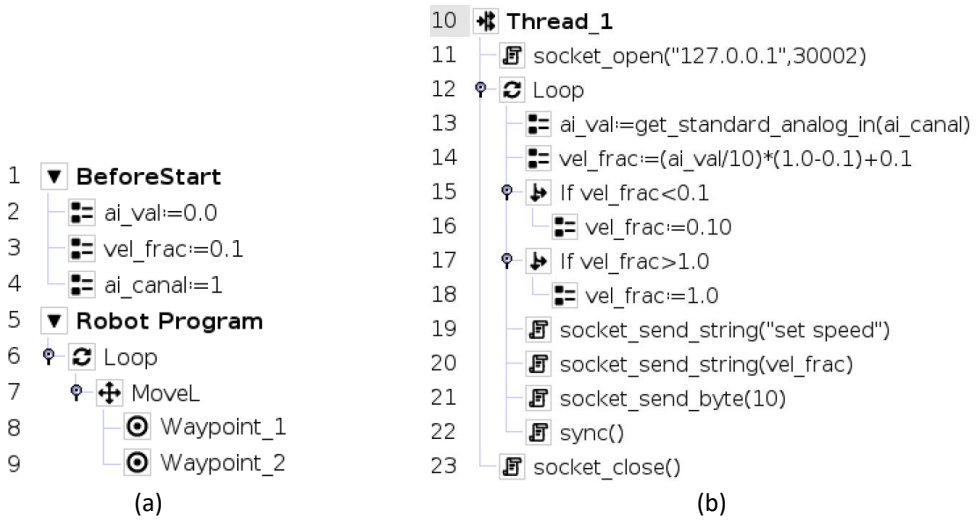
O programa deve estar continuamente a ler o valor do potenciómetro e a atualizar a velocidade em tempo real, de forma suave, garantindo que o robô não ultrapassa os limites de velocidade definidos.

Requisitos do exercício:

1. Definir as variáveis para armazenar o valor analógico lido (*ai_val*), a fração de velocidade (*vel_frac*) e o canal de leitura analógica (*ai_canal*);
2. Ler continuamente o valor do potenciómetro e convertê-lo para uma fração de velocidade no intervalo entre 0.1 e 1.0;
3. Garantir que valores abaixo de 0.1 são corrigidos para 0.1 e valores acima de 1.0 são corrigidos para 1.0;
4. Utilizar uma comunicação via *socket* para enviar o comando de alteração de velocidade ao controlador do robô;
5. Implementar um movimento repetitivo entre dois pontos definidos (*Waypoint 1* e *Waypoint 2*), cuja velocidade seja ajustada conforme a leitura do potenciómetro;
6. O sistema deve permanecer em funcionamento contínuo, atualizando a velocidade em tempo real até que seja interrompido manualmente.

Possível solução:

A solução apresentada para o controlo da velocidade do robô através de um potenciómetro é composta por duas partes principais: o programa de movimento do robô e uma *Thread* dedicada à leitura do valor analógico e ao envio da nova velocidade para o controlador. Na Figura 68 apresenta-se a solução desenvolvida.



(a) (b)

Figura 68 - Solução do exercício 5: (a) Secção *BeforeStart* e estrutura principal do programa; (b) Subtarefa de controlo e atualização da velocidade em tempo real

Através da análise a Figura 68 (a), verifica-se que antes do início do programa são atribuídas as variáveis cuja função é armazenar o valor analógico lido (*ai_val*), a fração de velocidade (*vel_frac*) e o número do canal de entrada analógica (*ai_canal*). Esta preparação garante que o

programa é composto por uma base de dados consistente para receber e processar a informação do potenciómetro.

No programa principal, o robô executa um ciclo de movimentos lineares entre dois pontos definidos (*Waypoint 1* e *Waypoint 2*). A particularidade está no facto da velocidade destes movimentos não ser fixa. Esta é ajustada dinamicamente conforme o valor do potenciómetro.

Para que a velocidade seja atualizada em tempo real, foi implementada uma *Thread*, presente na Figura 68 (b), que executa as seguintes funções:

1. **Abertura de porta de comunicação com o controlador**

   É utilizada a função *socket_open("127.0.0.1", 30002)* para estabelecer uma conexão TCP/IP direta com o controlador do robô na porta 30002. Esta porta é específica para o envio de comandos de *script* ao robô, permitindo alterar parâmetros como a velocidade de movimento sem interromper o ciclo principal.

2. **Leitura do valor analógico**

   A cada ciclo, o valor do potenciómetro é lido através da função *get_standard_analog_in(ai_canal)*. Este valor, entre 0 e 10 V, é convertido para um número decimal (*ai_val*) que serve de base para o cálculo da nova velocidade.

3. **Conversão para fração de velocidade**

   O valor lido é convertido para uma escala compreendida entre 0.1 e 1.0 através de uma expressão matemática. Estes limites são impostos para evitar velocidades extremamente baixas (que podem causar movimentos instáveis) ou demasiado altas (que podem comprometer a segurança).

4. **Envio de comandos via *socket***

   A comunicação com o controlador para alterar a velocidade é feita através do envio de *strings* formatadas através do *socket*. Utiliza-se *socket_send_string* para enviar o comando textual *"set speed"* seguido do valor calculado. Este método é mais avançado do que o uso de variáveis de programa, pois requer conhecimento da linguagem de URScript e da codificação de dados em formato *string* para controlo.

5. **Atualização em tempo real**

   A *thread* repete continuamente a leitura, conversão e envio do valor de velocidade enquanto o programa principal executa o movimento. Desta forma, qualquer alteração no potenciómetro reflete-se instantaneamente no comportamento do robô, sem necessidade de parar ou reiniciar o programa.

6. **Encerramento da comunicação**

   Quando o programa termina ou a *thread* é encerrada, a porta de comunicação é fechada com *socket_close()* para libertar o recurso e evitar conflitos em execuções futuras.

Esta abordagem demonstra como é possível integrar entradas analógicas externas para um controlo dinâmico e flexível do robô, combinando conhecimentos de programação em robótica, comunicação de rede e manipulação de *strings* específicas da Interface de Programação de Aplicações (IPA) do fabricante. Na Figura 69 apresentam-se dois exemplos onde se visualiza a relação entre o valor do potenciómetro e a velocidade de movimento do robô enquanto se executa o programa.

Figura 69 – Demonstração da relação entre o valor do potenciómetro e a velocidade de movimento do robô

Resultados esperados:

Após a resolução deste exercício, o utilizador deve ser capaz de integrar sinais analógicos no controlo dinâmico de um robô, convertendo valores de um potenciómetro em frações de velocidade dentro de limites seguros. Deve demonstrar competências no uso de comunicação TCP/IP para envio de comandos em formato *string* através da IPA do fabricante, bem como estruturar o programa de forma modular, separando a leitura contínua da entrada analógica da execução dos movimentos. Espera-se ainda que compreenda o impacto da variação da velocidade na execução física do robô e que seja capaz de adaptar o código para outros cenários de programação e trajetória, reforçando a autonomia na programação e operação de sistemas robóticos em tempo real.

Exercício 6 – Utilização de funções para traçar percursos

Um robô colaborativo UR3e, equipado com uma ferramenta de gravação (por exemplo, uma caneta), deve realizar um percurso constituído por dois círculos e meio de 50 mm de raio, dispostos horizontalmente no mesmo plano de trabalho, conforme ilustrado na Figura 70.

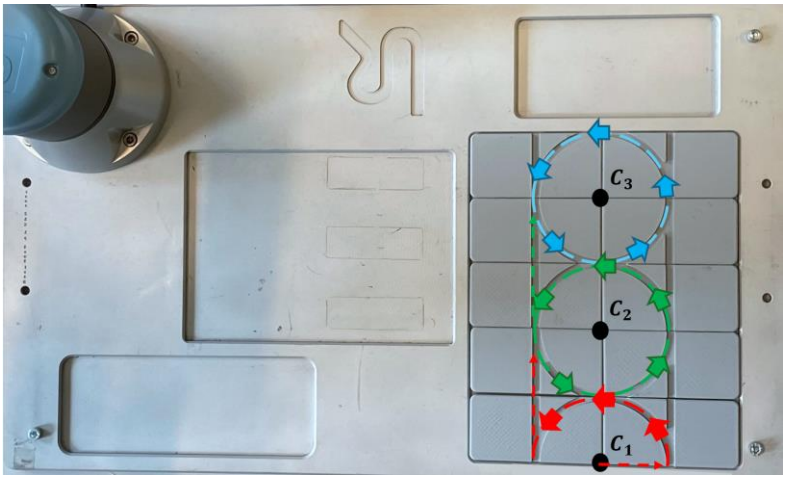


Figura 70 - Trajetória a percorrer

Requisitos do exercício:

1. Desenvolver um programa que permita desenhar dois círculos e meio, começando no C1 e terminando no C3.
2. Cada círculo deve ser aproximado por pontos sucessivos calculados por funções trigonométricas cos() e sin(), incrementando o ângulo θ de 10° em 10° (ou um valor definido pelo utilizador).

Possível solução:

A solução desenvolvida baseia-se na definição matemática das coordenadas de pontos ao longo de um círculo, de forma a permitir que o TCP percorra trajetórias circulares de raio R em torno do *centro_atual*. A determinação da posição de cada ponto do círculo é feita através das expressões matemáticas (1) e (2), que calculam as coordenadas cartesianas x e y a partir do centro do círculo (X_C, Y_C), do raio R e do ângulo θ. O ângulo varia no intervalo $[0, 2\pi]$, sendo incrementado de forma constante com um passo Δθ equivalente a $10°$ (0.1745 rad).

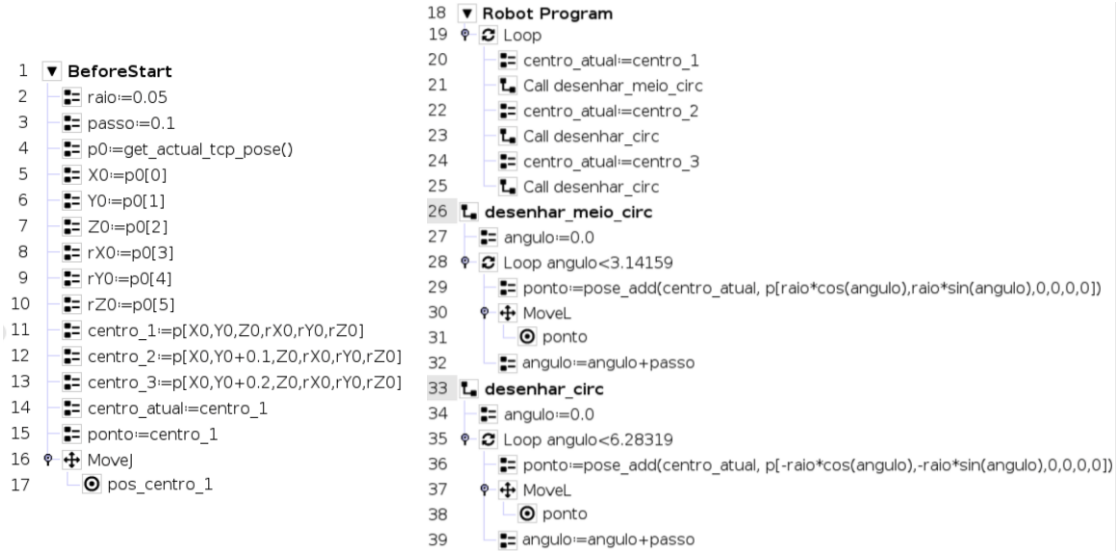
$$x = X_c + R \,.\cos(\theta) \quad (1)$$

$$y = Y_c + R \,.\sin(\theta) \quad (2)$$

Na implementação prática, representada na Figura 71 (a), são inicialmente definidos parâmetros como o raio do círculo e o incremento angular. Seguidamente, é obtida a posição atual do TCP, com a função *get_actual_tcp_pose*, armazenando as suas coordenadas e orientações. São também definidos diferentes centros possíveis para a execução do percurso, sendo selecionado um deles como referência inicial (*centro_atual*).

Os subprogramas ***desenhar_meio_circ*** e ***desenhar_circ***, apresentados na Figura 71 (b), traduzem as expressões matemáticas para o contexto da programação do robô. No subprograma ***desenhar_meio_circ***, o ângulo varia de 0 até π, o que permite percorrer exatamente metade da circunferência. Já no subprograma ***desenhar_circ***, o ângulo percorre todo o intervalo [0,2π], resultando na execução completa do círculo. Em ambos os casos, as coordenadas dos pontos são calculadas com base no centro de referência e no raio definido,

recorrendo à função *pose_add* para gerar a nova posição do TCP, que é depois alcançada através do comando de movimento linear *MoveL*.



(a) (b)

Figura 71 - Solução do exercício 6: (a) Secção *BeforeStart*; (b) Programa principal e subprogramas que definem o trajeto

A utilização de diferentes limites para o ângulo θ permite adaptar o percurso para desenhar arcos de qualquer dimensão, possibilitando a execução de meios círculos, quartos de círculo ou circunferências completas, de acordo com as necessidades da tarefa.

Após a definição das posições e da sequência de movimentos, o programa deverá ser executado para validar a trajetória. Durante esta fase, é recomendado que a operação inicie com uma velocidade de movimento reduzida, de modo a garantir maior precisão na trajetória e minimizar do risco de desvios. Após a validação, deve ser feita a afinação dos parâmetros de velocidade e aceleração para otimizar o tempo de execução, mantendo a precisão.

Resultados esperados:

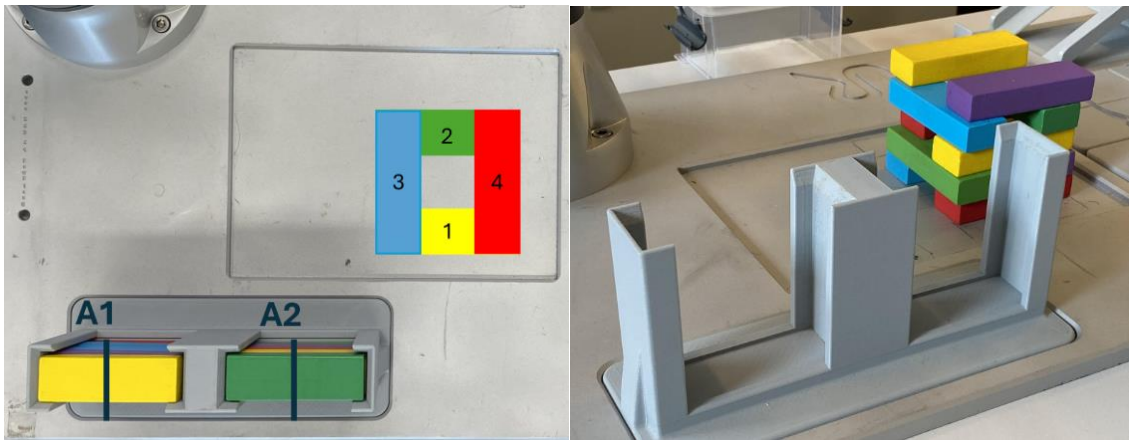
Após a resolução deste exercício, o utilizador deve demonstrar competências que abrangem a aplicação prática de equações para a definição de trajetórias, a manipulação de coordenadas cartesianas para posicionamento relativo do TCP e a utilização de funções específicas do sistema, como *pose_add* e *get_actual_tcp_pose(),* para cálculo dinâmico de posições. Deve ainda evidenciar capacidade para estruturar o programa de forma modular, através da criação de subprogramas reutilizáveis para percursos parciais ou completos, bem como ajustar parâmetros de movimento, como incrementos angulares, velocidades e acelerações, de forma a otimizar a suavidade e precisão dos deslocamentos. Operacionalmente, espera-se que o utilizador seja capaz de configurar e executar movimentos circulares completos ou parciais em torno de diferentes centros, compreender o impacto da variação dos parâmetros na execução física do robô e adaptar o código para diferentes geometrias de trajetória, consolidando assim a autonomia na programação e operação do manipulador.

Exercício 7 – Empilhamento de peças com funções

O objetivo deste exercício é desenvolver uma aplicação de manipulação e empilhamento de peças no UR3e. O robô deve recolher sequencialmente as peças existentes nas zonas A1 e A2, depositando-as nas posições 1 a 4, de modo a criar cinco camadas sobrepostas.

Na primeira fase, o robô deve desempilhar as 5 peças da zona A1 e colocá-las ordenadamente nas posições 1, 2, 3 e 4, criando a base do empilhamento. Na segunda fase, o robô deve desempilhar as 5 peças da zona A2 e colocá-las nas mesmas posições (1, 2, 3 e 4), de forma a completar as cinco camadas sobrepostas. Na Figura 72 (a) é apresentada a superfície de trabalho com as zonas A1 e A2 assinaladas, bem como as posições de empilhamento (1 a 4). Na Figura 72 (b) consta a solução pretendida.

Para a execução da tarefa, devem ser criados subprogramas específicos para as operações de recolha e posicionamento, permitindo modularidade e reutilização do código.



(a) (b)

Figura 72 – Esquematização do exercício 7: (a) representação da superfície de trabalho; (b) solução pretendida

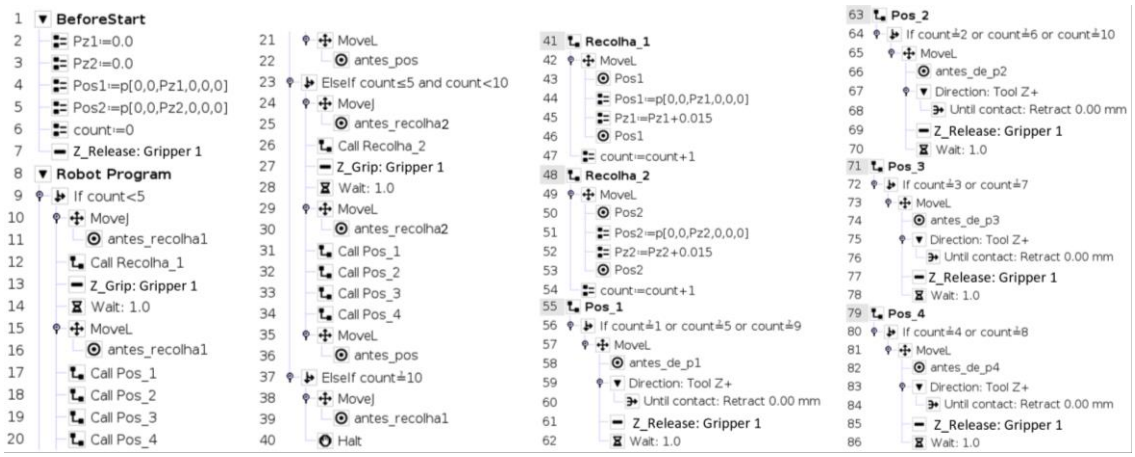
Requisitos do exercício:

1. Inserir as variáveis para a contagem de peças movimentadas e para o registo das alturas (*offsets* no eixo Z) das pilhas em A1 e A2;
2. Criar funções para recolher peças nas zonas A1 e A2, ajustando automaticamente a altura de recolha a cada iteração;
3. Criar funções para posicionar as peças nas posições 1, 2, 3 e 4;
4. Implementar um ciclo que, para cada peça, execute:
    - Movimento para a zona de recolha;
    - Ativação do *gripper* para agarrar a peça;
    - Transporte até à posição de destino;
    - Libertação da peça;
    - Incremento dos contadores de altura e peças colocadas.

5. Garantir que, após movimentar todas as peças de A1, o robô inicia automaticamente o desempilhamento de A2 até completar as 5 camadas finais;
6. Otimizar os movimentos para evitar trajetórias desnecessárias e reduzir o tempo de execução.

Possível solução:

A solução proposta organiza o código em duas secções principais: a fase de recolha e a fase de posicionamento, controladas por um contador (*count*) que determina a zona de recolha e posicionamento da peça. Na Figura 73 apresenta-se a solução desenvolvida.



(a) (b) (c) (d)

Figura 73 - Solução do exercício 7: (a) Secção *BeforeStart*; (a) e (b) programa principal; (c) e (d) Subprogramas de recolha e posicionamento de peças

A análise da Figura 73 permite verificar que o programa é composto pela secção *BeforeStart*, pelo programa principal, por dois subprogramas de recolha e por quatro subprogramas de posicionamento de peças. Segue-se uma descrição de função de cada uma das referidas secções:

- **Secção *BeforeStart*:**

  São definidas as variáveis para armazenar as alturas de recolha em A1 (*Pz1*) e A2 (*Pz2*), bem como as posições base *Pos1* e *Pos2* para cada local. O contador *count* inicia com o valor zero e o *gripper* é aberto para garantir que começa livre de peças.

- **Programa principal:**

  Um conjunto de condições *If* e *Elseif* controlam as diferentes fases:

  - count < 5: recolhe peças de A1 através do subprograma *Recolha_1*, ajustando a altura de recolha a cada iteração;
  - count ≥ 5 e < 10: recolhe peças de A2 através do subprograma *Recolha_2*;
  - Para cada peça recolhida, são chamados sequencialmente os subprogramas *Pos_1*, *Pos_2*, *Pos_3* e *Pos_4*, que depositam as peças nas posições de empilhamento.

- **Subprogramas de recolha (*Recolha_1* e *Recolha_2*):**

  Movem o robô até à posição base da pilha, ajustam o eixo Z para compensar a altura já retirada e atualizam a variável de altura e o contador global.

- **Subprogramas de posicionamento (*Pos_1* a *Pos_4*):**

  Movem o robô até à posição do empilhamento, ajustando a altura de acordo com a camada a ser formada (calculada pelo valor de *count*). A peça é libertada com o comando *Z_Release* quando a base da ferramenta deteta contacto. De seguida, o robô recua para evitar colisões.

Esta abordagem modular permite reutilizar funções tanto na recolha de A1 como de A2, alterando apenas a altura inicial e o valor de *count*. Na execução do programa, é importante utilizar velocidades de movimento reduzidas para garantir maior precisão no posicionamento das peças. Dado que se recorre à função de posicionamento com deteção de contacto pela ferramenta, a utilização de velocidades mais baixas aumenta a sensibilidade na deteção. Desta forma, minimiza-se o risco de dano nas peças e assegura-se um posicionamento suave e controlado.

Resultados esperados:

Após a conclusão do exercício, o utilizador deve ser capaz de programar um robô para executar tarefas de empilhamento de forma modular, recorrendo a funções reutilizáveis para recolha e posicionamento de peças. Deve demonstrar competência no controlo dinâmico de alturas com base em variáveis e contadores, garantindo a correta sobreposição das camadas. Espera-se também que otimize trajetórias para reduzir o tempo de execução e evitar colisões, e que consiga adaptar o código para diferentes quantidades de peças e configurações de empilhamento.

6.3. Resultados dos testes realizados

Os exercícios desenvolvidos foram testados e implementados, verificando-se que o seu funcionamento correspondeu ao esperado. O robô executou as tarefas com precisão de trajetória, a conexão entre os sinais digitais e analógicos com o controlador não demonstrou falhas e o *gripper* acoplado responde aos comandos dados. Para além de constituírem uma ferramenta prática de aprendizagem, estes exercícios revelaram-se igualmente relevantes para a criação de um guião laboratorial e de um manual de utilizador. Estes documentos reúnem de forma organizada o conhecimento adquirido e as boas práticas de programação identificadas ao longo do projeto. Além disso, asseguram que futuros utilizadores possam replicar e expandir os exercícios, com base na experiência acumulada durante o desenvolvimento do projeto

O guião laboratorial, apresentado no Apêndice E, é constituído por seis exercícios selecionados de forma a assegurar uma progressão gradual de dificuldade, favorecendo a consolidação de conhecimentos em diferentes níveis de complexidade. Através deste guião, um conjunto de catorze alunos desenvolveu programas diretamente no robô, cumprindo os objetivos e requisitos de cada exercício.

Após a execução das tarefas propostas, os alunos responderam a um questionário que possibilitou avaliar o impacto do trabalho realizado na aprendizagem, no desenvolvimento de competências técnicas e no crescimento pessoal. A análise das respostas permitiu ainda identificar o potencial da bancada no contexto do ensino, reforçado pela avaliação gráfica da ergonomia do sistema e do envolvimento emocional dos utilizadores durante a sua utilização. A partir destes inquéritos realizou-se uma análise estatística, que se encontra detalhada no subcapítulo 7.1.

7. Discussão de resultados

Neste capítulo apresenta-se a análise crítica e a discussão dos resultados obtidos ao longo do projeto, com foco no impacto pedagógico da bancada didática desenvolvida no ensino de robótica colaborativa. Através da realização de exercícios práticos e da aplicação de um inquérito aos participantes, foi possível avaliar a eficiência da metodologia adotada e a sua relevância para a aquisição de competências técnicas e pessoais. A análise dos resultados incide sobre diferentes temas, nomeadamente a organização da atividade, o conteúdo técnico, o desenvolvimento pessoal, a ergonomia e o conforto da bancada e o envolvimento pessoal dos utilizadores. Por fim, apresentam-se as limitações do sistema e a opinião global dos utilizadores.

7.1. Análise crítica dos resultados

A análise dos resultados tem por base as respostas obtidas através do inquérito (Apêndice F) aplicado a 14 participantes, após a conclusão dos exercícios práticos presentes no guião laboratorial. O inquérito é constituído por 21 questões e encontra-se dividido em quatro áreas principais: organização da atividade, conteúdo técnico, desenvolvimento pessoal e ergonomia e envolvimento emocional. As respostas recolhidas permitem avaliar o impacto da atividade na aquisição de conhecimento e verificar a adequação da bancada didática como recurso pedagógico no ensino de robótica colaborativa. De referir que os inquéritos foram realizados a uma amostra generalizada de participantes, com o consentimento dos mesmos e em anonimidade.

A análise crítica dos resultados inicia com a caracterização do nível de conhecimento dos participantes previamente à realização dos exercícios, seguida pela apresentação das quatro áreas principais do inquérito. Os resultados são expostos com recurso a representações gráficas e a interpretações descritivas, de modo a garantir uma leitura objetiva e fundamentada da experiência pedagógica.

Nível de experiência dos participantes

A segunda questão do inquérito permitiu analisar a distribuição do nível de experiência dos participantes (Figura 74). A maioria dos inquiridos declarou não possuir qualquer experiência prévia com este tipo de sistema (57%, correspondente a 8 participantes), enquanto 36% (5 participantes) indicaram deter conhecimentos básicos. Apenas um participante (7%) afirmou possuir experiência intermédia e nenhum dos inquiridos referiu deter experiência avançada.

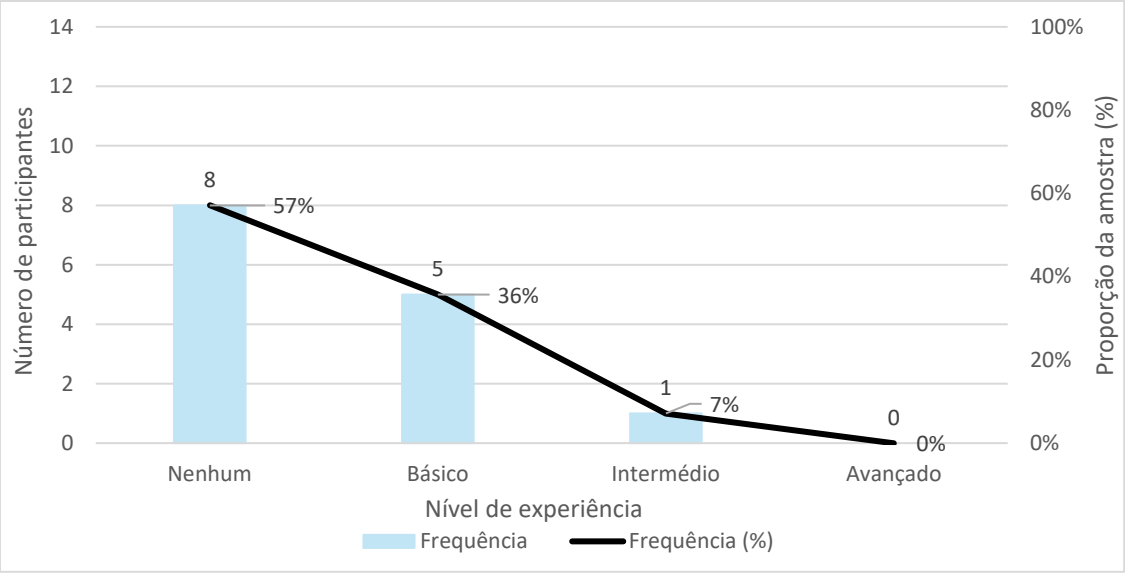


Figura 74 - Distribuição inicial do nível de experiência dos participantes

A distribuição obtida evidencia que a maioria dos utilizadores está numa fase inicial de aprendizagem de robótica colaborativa, o que torna importante apresentar um guião laboratorial bem organizado e progressivo, para ajudar na aprendizagem passo a passo e facilitar o desenvolvimento das competências nesta área.

Organização da atividade

Na presente secção são analisadas as respostas obtidas às questões do inquérito que procuraram avaliar a organização da atividade. O objetivo consistiu em perceber de que forma os participantes interpretaram a clareza dos objetivos propostos, a quantidade e a qualidade dos materiais e instruções disponibilizadas. Estes fatores assumem particular relevância, uma vez que influenciam diretamente a autonomia dos utilizadores e a eficácia da experiência pedagógica.

A questão relativa à clareza dos objetivos dos exercícios práticos permitiu avaliar a perceção dos participantes quanto à forma como os mesmos foram apresentados. Após análise da Figura 75, verifica-se que dos 14 inquiridos, 13 (93%) indicaram que os objetivos foram apresentados com clareza. Apenas 1 participante (7%) considerou que os objetivos estavam apenas parcialmente claros, e nenhum participante assinalou a opção “Não”. Estes resultados demonstram que a formulação dos exercícios foi, na sua maioria, eficaz. A quase unanimidade das respostas positivas evidencia que os enunciados apresentaram uma estrutura compreensível e suficientemente explícita para orientar a realização das tarefas. O facto de apenas um participante ter apontado falta de total clareza pode indicar a necessidade de pequenos ajustes na forma de enunciar ou detalhar os objetivos, de modo a assegurar uniformidade na interpretação. No entanto, a ausência de respostas negativas reforça a adequação da metodologia seguida na conceção dos exercícios.

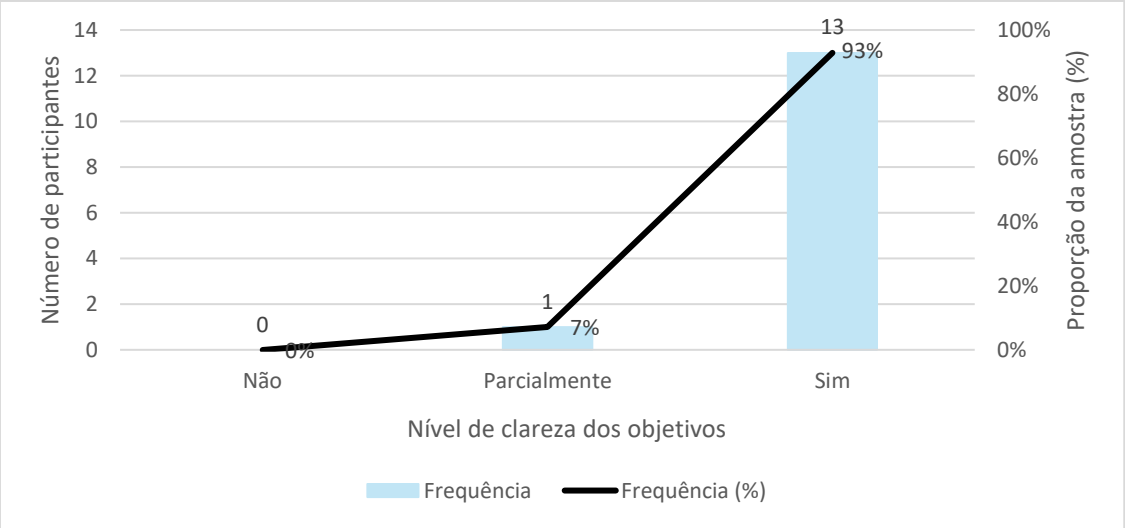


Figura 75 - Avaliação da clareza dos objetivos dos exercícios práticos

Relativamente à quantidade de material e instruções fornecidas (Figura 76), 57% dos participantes classificaram-nos como "Muito suficientes" e 43% como "Totalmente suficientes". Não foram registadas respostas nas categorias "Nada suficientes", "Pouco suficientes" ou "Suficientes". A totalidade da amostra reconheceu, assim, que os recursos disponibilizados asseguraram uma compreensão adequada e permitiram concluir os exercícios com sucesso. A predominância das respostas mais positivas reforça a adequação do material e a eficácia da sua organização.

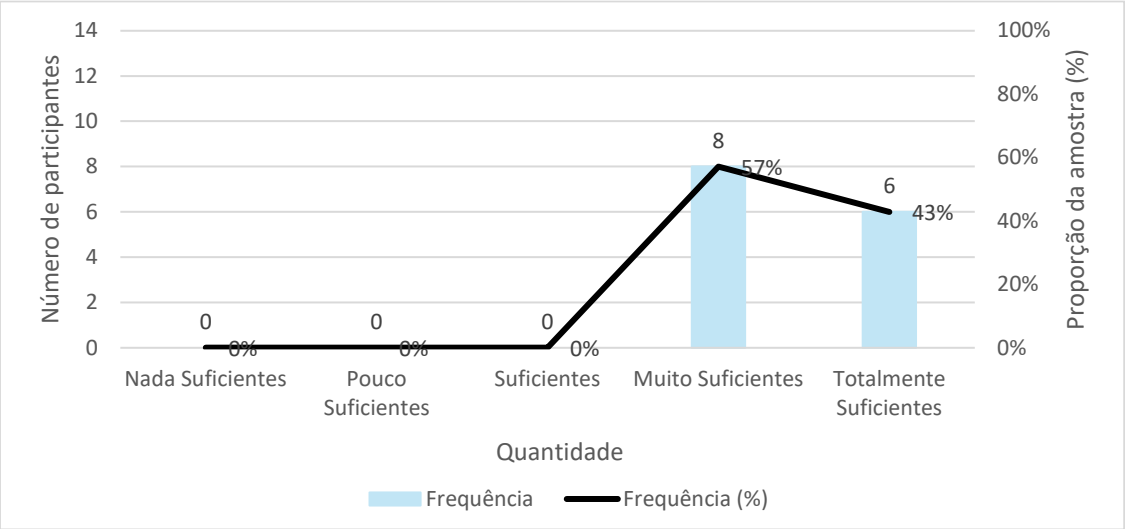


Figura 76 - Avaliação da quantidade do material e das instruções fornecidas

Por último, nesta secção, a análise da clareza das explicações fornecidas para a resolução dos exercícios permitiu avaliar a perceção dos participantes relativamente à forma como os conteúdos foram apresentados. A Figura 77 apresenta a distribuição das respostas à pergunta em análise. Observa-se que 50% dos participantes atribuíram a classificação de "Totalmente claras" às explicações, 43% consideraram-nas como "Muito claras" e 7% optaram pela categoria "Claras", não se registando qualquer resposta nas restantes opções disponíveis.

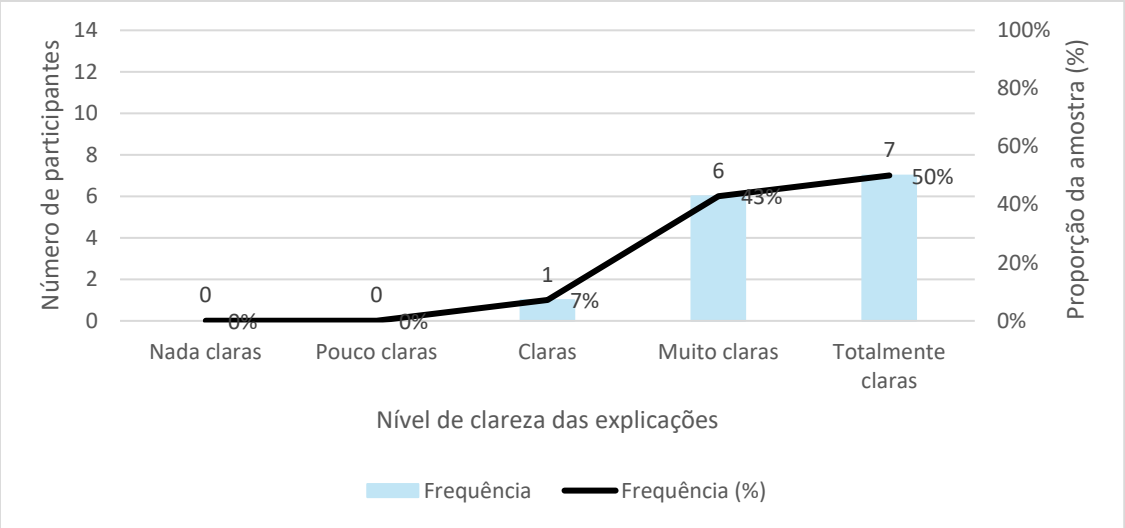


Figura 77 - Avaliação da clareza das explicações fornecidas

A análise conjunta das questões evidencia que a atividade apresentou um elevado nível de organização, refletido tanto na clareza dos objetivos como na quantidade do material disponibilizado. A unanimidade de respostas positivas confirma que os participantes encontraram condições favoráveis para a execução autónoma dos exercícios.

Conteúdo técnico

A secção de conteúdo técnico do inquérito permitiu avaliar tanto o número de participantes que realizou cada exercício como o nível médio de dificuldade atribuído a cada um. Na Figura 78 é apresentado o gráfico relativo à distribuição da resolução dos exercícios. Observa-se que 100% dos participantes completaram os cinco primeiros exercícios, enquanto apenas 50% desenvolveram o exercício 6. Esta diferença significativa pode estar relacionada com o facto de o sexto exercício ser o mais complexo, aliado ao cansaço acumulado após a realização dos cinco exercícios anteriores. Este indicativo pode indicar que o guião laboratorial se revela demasiado extenso ao integrar seis exercícios consecutivos.

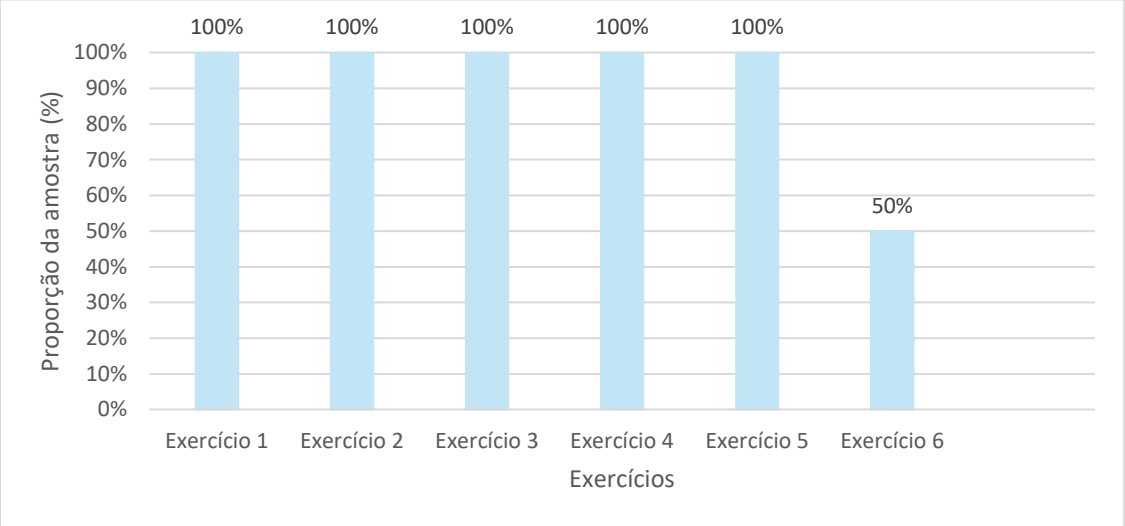


Figura 78 - Distribuição da resolução dos exercícios

A dificuldade média atribuída pelos participantes a cada exercício permitiu verificar se a perceção de dificuldade aumenta conforme a resolução dos exercícios. Para esta secção foi atribuída uma codificação a cada nível de resposta para facilitar a análise gráfica (Tabela 18).

Tabela 18 - Codificação do nível de dificuldade

Nível de dificuldade	Codificação
Muito fácil	1
Fácil	2
Moderado	3
Difícil	4
Muito difícil	5

Na Figura 79 apresentam-se os resultados relativos à distribuição da dificuldade média atribuída a cada exercício. A análise do gráfico revela uma progressão crescente na perceção de dificuldade ao longo dos exercícios. Os exercícios 1, 2 e 3 foram classificados entre os níveis "muito fácil" e "fácil", enquanto os exercícios 4 e 5 situaram-se entre "fácil" e "moderado". O exercício 6 destacou-se como o mais exigente, com uma dificuldade média de 3,5, valor que corresponde exatamente ao ponto intermédio entre os níveis "moderado" e "difícil". O desvio padrão evidencia que a dispersão das respostas aumentou nos exercícios de maior dificuldade, o que sugere perceções menos homogéneas entre os participantes à medida que os desafios se tornaram mais complexos. A dificuldade média atribuída pelos participantes a cada exercício permitiu comprovar que a estruturação do guião laboratorial correspondeu aos objetivos pedagógicos inicialmente definidos, nomeadamente a apresentação de exercícios com grau progressivo de dificuldade.

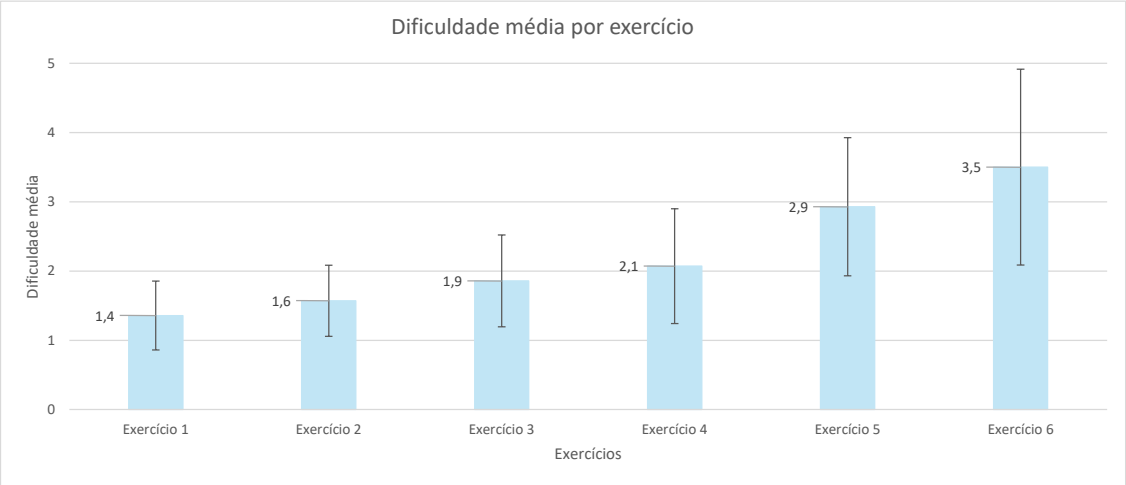


Figura 79 - Distribuição da dificuldade média atribuída pelos participantes a cada exercício

Foi ainda realizada uma análise da relação entre o nível de experiência individual dos participantes e a perceção do grau de dificuldade atribuído aos exercícios. O objetivo desta avaliação consistiu em identificar se a experiência prévia de cada participante influencia a forma como os exercícios são percecionados em termos de complexidade. Na Figura 80 apresenta-se a distribuição do nível médio de dificuldade atribuído a cada exercício, agrupado de acordo com o nível de experiência dos utilizadores (nenhum, básico e intermédio).

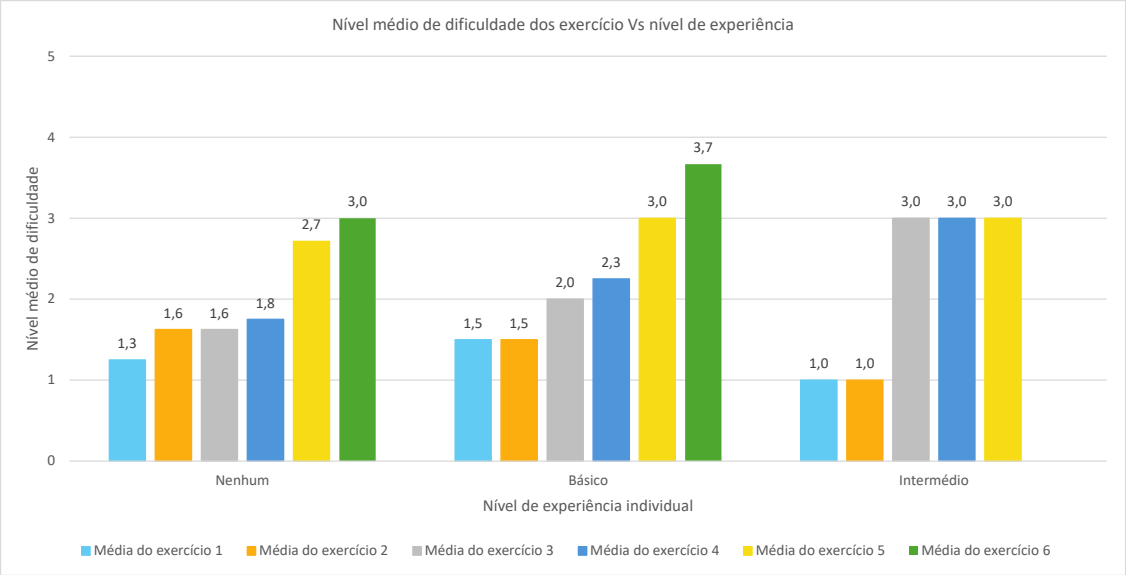


Figura 80 - Distribuição do nível médio de dificuldade dos exercícios relacionado com o nível de experiência dos participantes

A análise da Figura 80 permite observar diferenças significativas na perceção de dificuldade consoante o nível de experiência dos participantes. O grupo sem experiência apresenta uma progressão gradual, com valores a aumentar de 1,3 no exercício 1 até 3,0 no exercício 6, destacando-se uma variação mais acentuada entre os exercícios 4 e 5. O grupo com experiência básica revela uma tendência semelhante, mas atribui valores mais elevados nos exercícios finais, com o valor máximo de 3,7 no exercício 6. Já o grupo com experiência intermédia apresenta uma avaliação distinta: considera os dois primeiros exercícios muito fáceis (1,0), mas atribui 3,0 a todos os restantes, sem diferenciar a dificuldade entre eles.

A análise dos resultados permite concluir que a perceção de dificuldade está diretamente relacionada ao nível de experiência prévia dos participantes. Os participantes sem experiência apresentam uma clara distinção entre os exercícios, o que demonstra que a sequência de exercícios foi bem estruturada e permite uma progressão gradual de dificuldade adequada. Os utilizadores com experiência básica apresentam um padrão de respostas semelhante, embora revelem uma perceção de maior exigência no exercício 6. Por outro lado, os participantes com experiência intermédia uniformizam a avaliação a partir do exercício 3, o que sugere que as tarefas propostas não foram suficientemente desafiantes para este perfil.

Em suma, os resultados demonstram que o guião responde de forma eficaz às necessidades dos participantes com níveis iniciante e básico. Contudo, para garantir a relevância pedagógica em todos os níveis de aprendizagem, devem ser incluídos exercícios complementares com maior grau de complexidade, de modo a desafiar utilizadores mais experientes e potenciar a sua evolução no domínio da programação e operação do robô.

Desenvolvimento pessoal

Para avaliar o impacto da bancada didática e dos exercícios realizados pelos participantes no seu desenvolvimento pessoal, foram incluídas cinco questões no inquérito sobre este tema. A primeira questão teve como objetivo analisar a perceção da utilidade dos exercícios práticos na

aprendizagem de robótica colaborativa. Na Figura 81 é apresentada a distribuição das respostas. Observa-se que 57% dos participantes classificaram os exercícios como “Muito úteis” e 36% como “Úteis”. Apenas 7% registaram uma posição “Neutra” e não se verificaram respostas negativas. Os resultados demonstram uma avaliação global positiva. A ausência de respostas nas categorias “Pouco úteis” e “Nada úteis” confirma que os exercícios foram considerados relevantes e adequados para a aprendizagem de robótica colaborativa. A elevada percentagem de respostas “Muito úteis” evidencia que a abordagem prática acrescentou valor real à experiência dos participantes. Assim, conclui-se que a componente experimental do guião laboratorial contribuiu para a consolidação dos conhecimentos em robótica colaborativa e reforçou a motivação dos utilizadores.

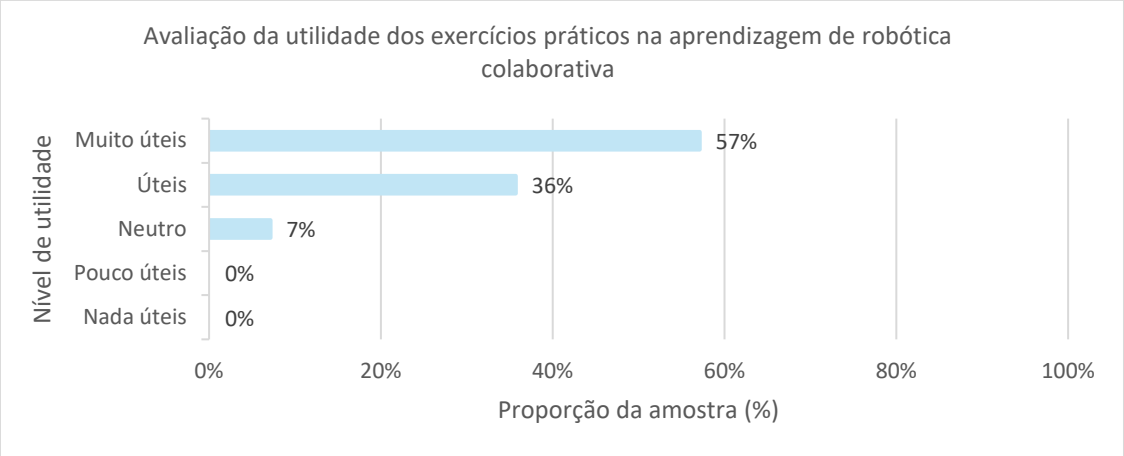


Figura 81 - Distribuição percentual das respostas à questão que relaciona a utilidade dos exercícios práticos na aprendizagem em RC.

A segunda questão teve como objetivo compreender a perceção do nível de desafio associado aos exercícios práticos. Na Figura 82 expõe-se a distribuição das respostas. Verifica-se que a maioria dos participantes (57%) considerou os exercícios “Desafiantes”, 29% avaliou os como “Pouco desafiantes” e 14% como “Muito desafiantes”. Não se registaram respostas na categoria “Nada desafiantes”. Os resultados revelam que a maioria dos participantes considerou existir um equilíbrio adequado entre dificuldade e acessibilidade. A predominância da opção “Desafiantes” demonstra que os exercícios exigiram empenho dos participantes, mas não se tornarem excessivamente complexos. A presença da resposta “Pouco desafiantes” indica que, para uma parte da amostra (4 participantes), o nível de dificuldade poderia ter sido mais elevado. Contudo, a percentagem de respostas “Muito desafiantes” confirma que os exercícios também foram capazes de estimular participantes com maior domínio na área. Desta análise é possível concluir que a estrutura das atividades promoveu um desafio ajustado às necessidades do grupo e favoreceu o desenvolvimento de competências, sem comprometer a motivação dos participantes.

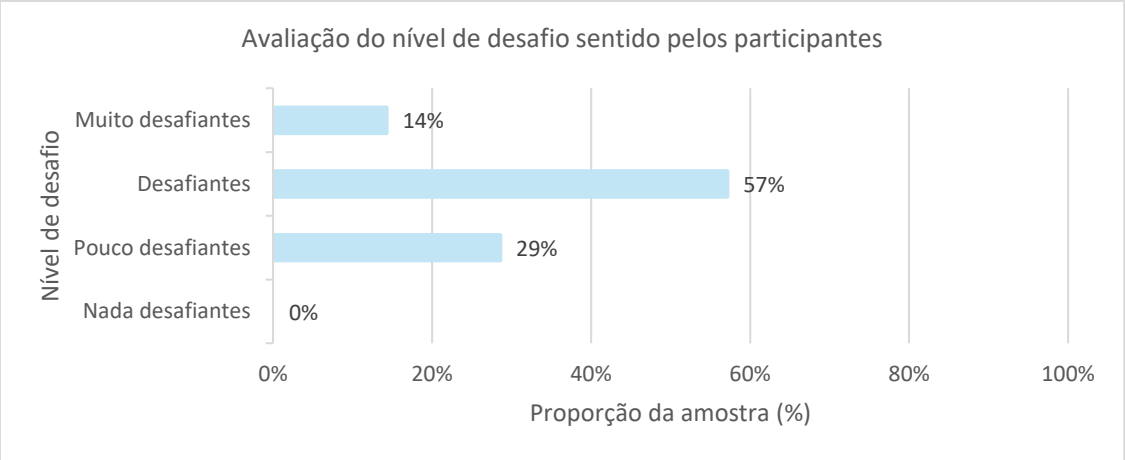


Figura 82 - Distribuição percentual das respostas à questão sobre o nível de desafio dos exercícios

Com o objetivo de aprofundar a análise da perceção do nível de desafio, procedeu-se à análise da relação entre as respostas à questão anterior (questão 12 do inquérito) e o nível de experiência prévia em robótica colaborativa dos participantes. Na Figura 83 é apresentada a distribuição percentual dos resultados. A análise do gráfico permite verificar que nos participantes sem experiência, ocorre uma predominância da opção "Desafiantes" (62,5%), seguida de "Pouco desafiantes"(25,0%). Apenas 12,5% dos participantes consideraram os exercícios "Muito desafiantes". Entre os participantes com nível de experiência "Básico", a distribuição foi equilibrada entre "Pouco desafiantes" (50%) e "Desafiantes" (50%), não se registando respostas na categoria "Muito desafiantes". No grupo com experiência "Intermédia", todos os participantes (100%) classificaram os exercícios como "Muito desafiantes".

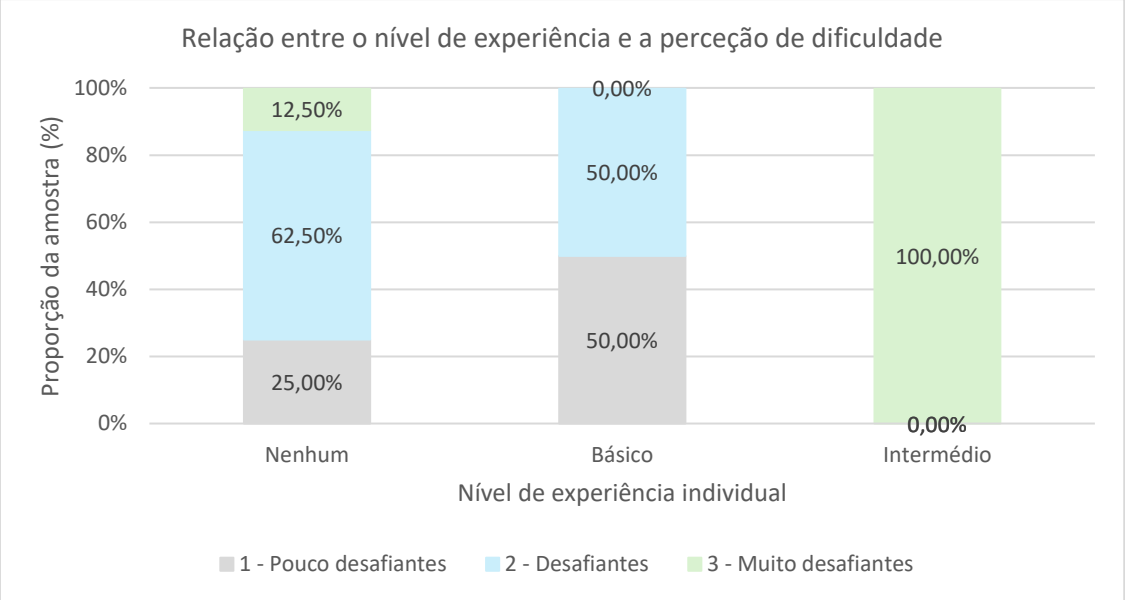


Figura 83 - Distribuição percentual do nível de dificuldade atribuído aos exercícios em função da experiência prévia dos participantes

Os resultados obtidos demonstram que a perceção do nível de desafio dos exercícios encontra-se fortemente associada ao nível de experiência inicial dos utilizadores. Os participantes sem experiência consideraram que os exercícios foram exigentes o suficiente para estimular a sua aprendizagem, sem se tornarem excessivamente complexos. No caso dos participantes com experiência básica, a dificuldade revelou-se adequada e existiu equilíbrio entre acessibilidade e exigência. Por outro lado, os participantes com experiência intermédia identificaram os exercícios como "Muito desafiantes", o que indica que o nível de complexidade foi capaz de corresponder às suas competências e de promover um desafio significativo. Em virtude da análise realizada, é possível concluir que a atividade prática se ajustou harmoniosamente aos diferentes perfis de experiência e proporcionou uma oportunidade de aprendizagem relevante e motivadora a todos os participantes.

A terceira questão pretendia avaliar o contributo dos exercícios práticos na aprendizagem e compreensão da programação de robôs colaborativos através da interface PolyScope. Na Figura 84 está representada a distribuição gráfica desta questão. Verifica-se unanimidade na resposta, uma vez que 100% dos participantes consideraram que os exercícios "Contribuíram muito" para o seu entendimento. Este resultado evidencia que a componente prática da atividade foi determinante para o desenvolvimento das competências de programação. A unanimidade das respostas confirma que os exercícios foram eficazes e adequados e permitiram compreender de forma clara os conceitos associados ao PolyScope.

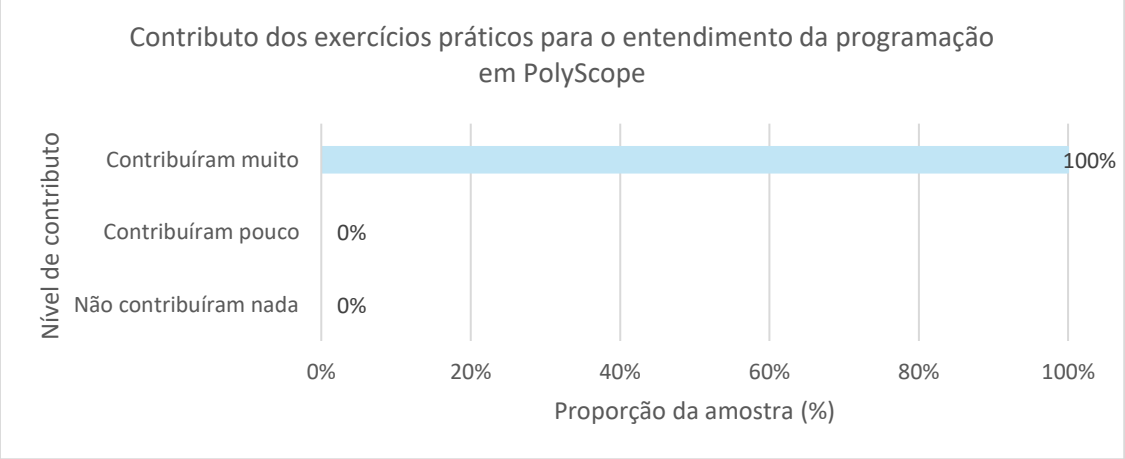


Figura 84 - Distribuição percentual das respostas à questão sobre o contributo dos exercícios práticos para a compreensão da programação de robôs colaborativos via PolyScope

A quarta questão teve como objetivo avaliar em que medida os exercícios práticos contribuíram para o aumento da confiança dos participantes na área da robótica colaborativa. A Figura 85 apresenta a distribuição gráfica das respostas obtidas. Após análise do gráfico, observa-se uma tendência positiva, uma vez que 79% dos participantes consideraram que a execução dos exercícios práticos "Aumentaram" a sua confiança em robótica colaborativa, 14% votou em "Aumentaram muito" e 7% (1 participante) declarou que "Não teve impacto". Pode-se concluir que a realização da atividade promoveu, de forma global, o aumento da confiança dos participantes em robótica colaborativa, o que confirma a relevância pedagógica da abordagem adotada.

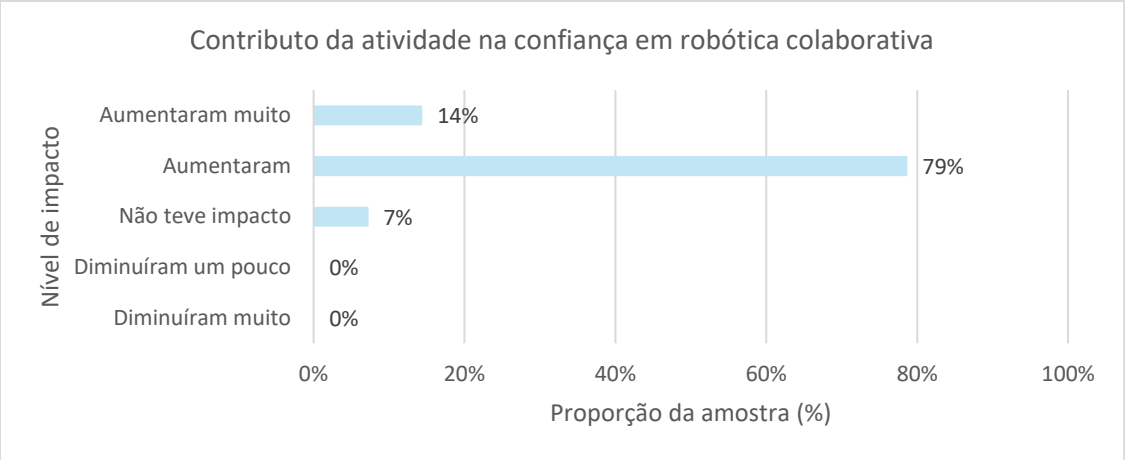


Figura 85 - Distribuição percentual das respostas à questão sobre o contributo da atividade na confiança em robótica colaborativa

A última questão pretendia avaliar o contributo da atividade na capacidade de programação de robôs colaborativos adquirida pelos participantes. Na Figura 86 está presente o gráfico de distribuição percentual das respostas obtidas. A análise dos resultados demonstra que todas as respostas foram positivas, sendo que 21% dos participantes consideraram-se "Totalmente capazes" de programar robôs colaborativos, 50% classificaram-se como "Muito capazes" e 29% como "Capazes". Estes resultados validam os objetivos pedagógicos definidos para a componente didática da bancada. Conclui-se que a realização da atividade permitiu aos participantes adquirir conhecimentos técnicos na área e, simultaneamente, desenvolver o nível de confiança necessário para criar e implementar novos programas de forma autónoma.

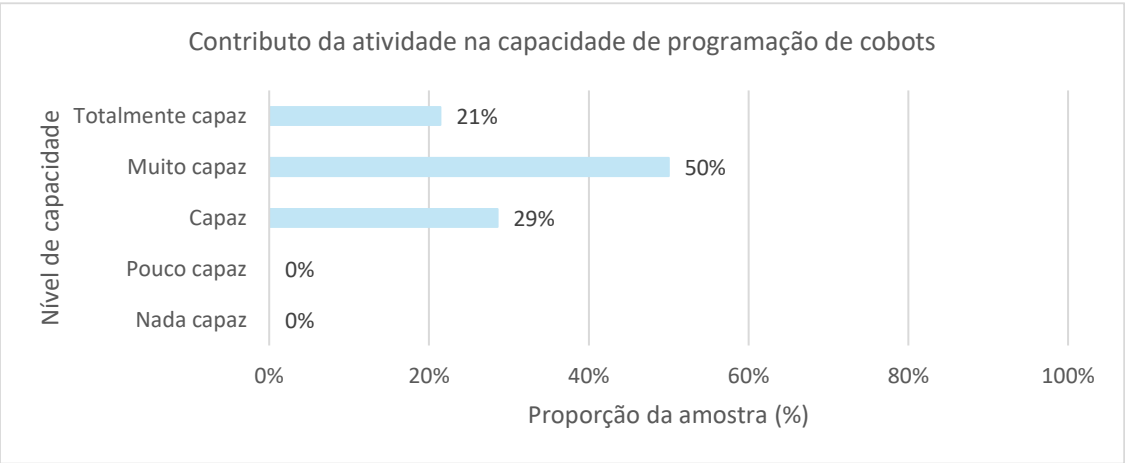


Figura 86 - Distribuição percentual das respostas à questão sobre o contributo da atividade na capacidade de programação de cobots

Com o objetivo de desenvolver uma visão global das cinco questões apresentadas anteriormente, procedeu-se à codificação das opções de resposta do inquérito. Esta metodologia permitiu determinar, numa escala de 1 a 5, a média global de cada questão, o que possibilita uma análise objetiva do contributo da atividade para o desenvolvimento pessoal dos participantes. Na Tabela 19 é apresentada a codificação atribuída a cada opção de resposta.

Tabela 19 - Codificação das opções de resposta da secção “Desenvolvimento pessoal”

Pergunta 11	Pergunta 12	Pergunta 13	Pergunta 14	Pergunta 15	Codificação
Nada úteis	Nada desafiantes	Não contribuíram nada	Diminuíram muito	Nada capaz	1
Pouco úteis	Pouco desafiantes	---	Diminuíram um pouco	Pouco capaz	2
Neutro	---	Contribuíram pouco	Não teve impacto	Capaz	3
Úteis	Desafiantes	---	Aumentaram	Muito capaz	4
Muito úteis	Muito desafiantes	Contribuíram muito	Aumentaram muito	Totalmente capaz	5

A visão global do contributo da atividade no desenvolvimento pessoal dos participantes está presente no gráfico da Figura 87.

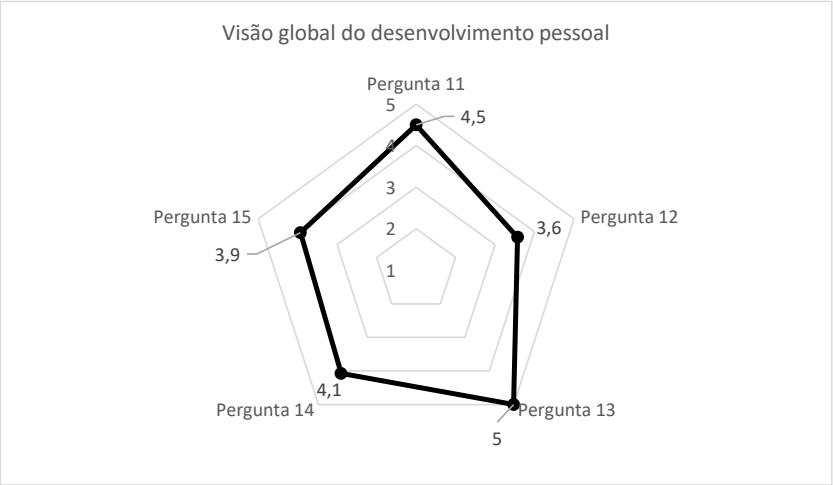


Figura 87 - Visão global do contributo da atividade no desenvolvimento pessoal dos participantes

A análise global da secção “Desenvolvimento pessoal” (Figura 87) evidencia que a atividade teve um impacto positivo na perceção dos participantes. As médias atribuídas às cinco questões situam-se entre 3,6 e 5. De destacar a Pergunta 13, relativa à contribuição dos exercícios práticos para o entendimento em programação de robôs colaborativos, que obteve a classificação máxima. A Pergunta 11, elativa à utilidade dos exercícios, obteve uma média de 4,5. Os resultados indicam que os participantes consideraram os exercícios úteis e desafiantes (Perguntas 11 e 12), reconheceram o seu contributo para o desenvolvimento de competências técnicas e pessoais (Pergunta 13), avaliaram positivamente o aumento da confiança na área da robótica colaborativa (Pergunta 14) e se sentiram mais capazes de programar sistemas simples após a atividade (Pergunta 15). Em termos analíticos, a consistência dos resultados revela uma valorização global da experiência, o que confirma que a metodologia aplicada cumpriu os objetivos pedagógicos estabelecidos, promoveu a aquisição de conhecimento, reforçou a autonomia e apoiou o desenvolvimento profissional na área da robótica colaborativa.

Para aprofundar a análise desta secção, avaliou-se a relação entre a quantidade de exercícios resolvidos e o impacto no nível de confiança final dos participantes. O gráfico da Figura 88 foi elaborado a partir das respostas às perguntas 6 e 14 do inquérito. A análise dos dados revela que os participantes que resolveram cinco exercícios registaram um impacto médio de 4,0 no seu nível de confiança. Já aqueles que completaram seis exercícios apresentaram uma média

ligeiramente superior, de 4,2. Os valores apresentados revelam que o nível de confiança dos participantes tende a aumentar à medida que realizam um maior número de exercícios. Assim, é possível concluir que a realização integral dos exercícios propostos tem um impacto benéfico no desenvolvimento da confiança individual. Este resultado demonstra a importância de assegurar que os participantes concluam todas as atividades previstas.

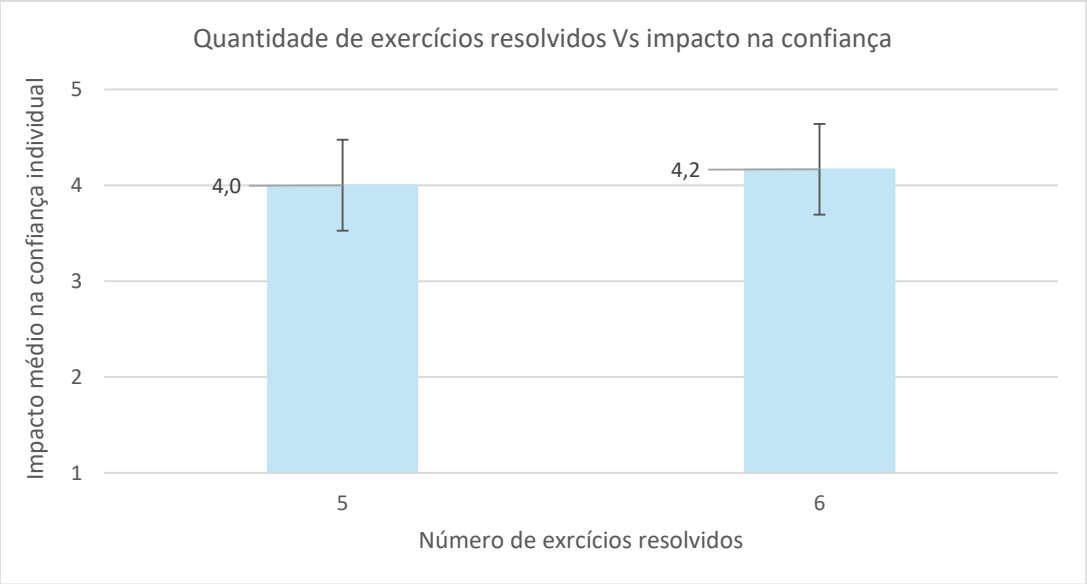


Figura 88 - Análise do impacto da quantidade de exercícios resolvidos com o nível e confiança dos participantes após atividade

Por fim, procedeu-se à avaliação da relação entre a capacidade final média dos participantes e a sua experiência prévia. O objetivo desta análise consistiu em compreender se a experiência anterior na área contribui para a melhoria do desempenho final. O gráfico apresentado na Figura 89 foi construído com base nas respostas às perguntas 2 e 15 do inquérito. A análise do gráfico revela que os participantes com experiência de nível básico apresentam a capacidade final média mais elevada (4,25), seguidos pelos participantes com experiência intermédia (4,0). Os indivíduos sem qualquer experiência prévia registam a capacidade final média mais baixa (3,75). Estes resultados sugerem que a familiaridade inicial com o tema facilita a aquisição de competências, embora a diferença entre grupos não seja acentuada. Conclui-se que a experiência prévia contribui para um desempenho ligeiramente superior. Contudo, o facto de os participantes sem experiência apresentarem uma capacidade média próxima dos restantes demonstra que a metodologia de ensino adotada foi adequada para todos os perfis e promoveu a aprendizagem independentemente do nível de conhecimento inicial.

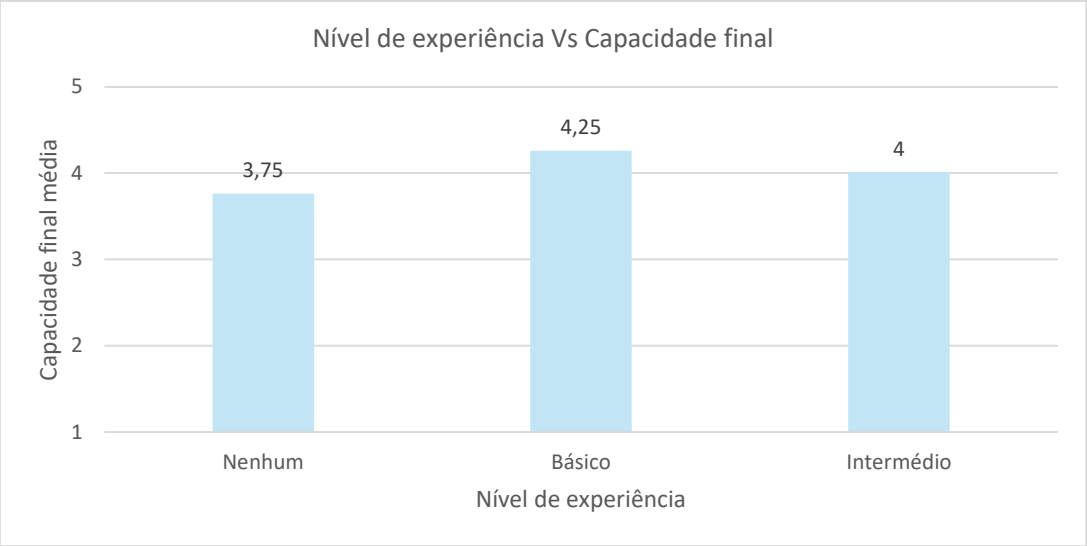


Figura 89 – Análise do impacto do nível de experiência na capacidade final dos participantes

Ergonomia e envolvimento pessoal

Para avaliar a ergonomia, o conforto da bancada didática e o nível de interesse dos participantes na atividade, foram incluídas três questões no inquérito sobre o tema. Estas questões tiveram como objetivo recolher a perceção dos utilizadores relativamente à adequação da bancada, ao grau de envolvimento na experiência e à intenção de participação em futuras atividades de robótica colaborativa.

A primeira questão focou-se na perceção da ergonomia e do conforto da bancada desenvolvida. Do gráfico de respostas presente na Figura 90, observa-se que 57% dos participantes classificaram a bancada como "Ergonómica e confortável" e 43% como "Muito ergonómica e confortável". Este resultado demonstra que a bancada didática apresenta um nível adequado de experiência de uso e conforto, o que assegura condições favoráveis para a execução das tarefas propostas.

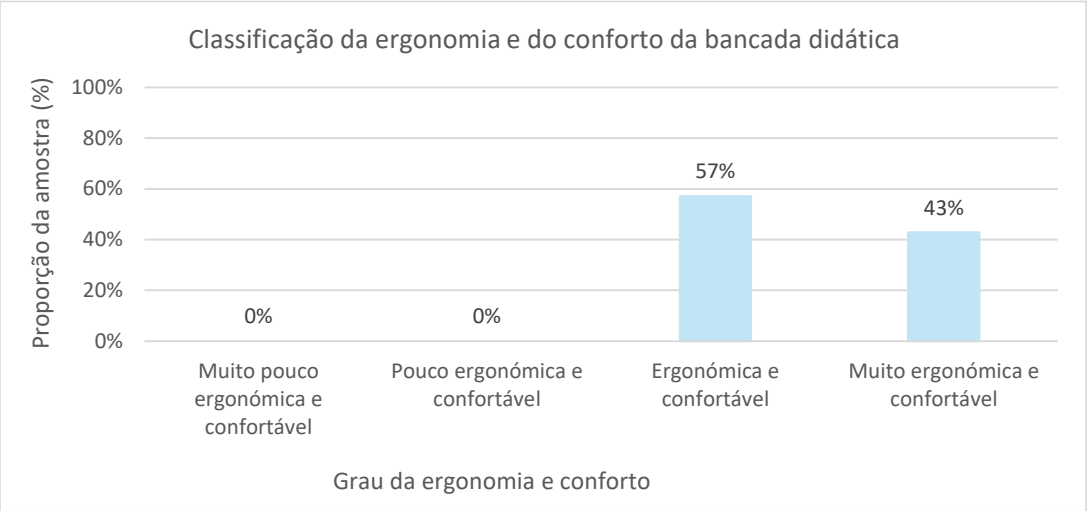


Figura 90 - Distribuição percentual das respostas à questão que avalia a ergonomia e o conforto da bancada didática

A segunda questão procurou avaliar o grau de interesse dos participantes na atividade (Figura 91). Verifica-se que 86% dos inquiridos consideraram a experiência “Muito interessante” e 14% classificaram-na como “Interessante”. Não se registaram respostas nas categorias negativas. Este resultado confirma o elevado envolvimento dos participantes e destaca a relevância da metodologia adotada, capaz de captar a atenção e a motivação dos utilizadores ao longo da atividade.

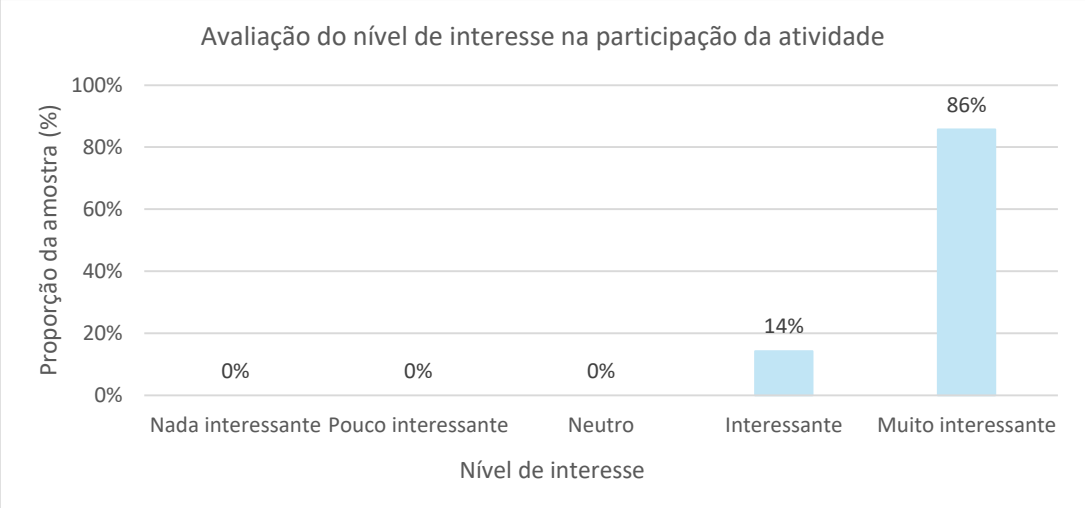


Figura 91 - Distribuição percentual das respostas obtidas à questão que avalia o interesse de participação na atividade

A terceira questão permitiu analisar a intenção de participação em futuras atividades práticas de robótica colaborativa (Figura 92). A totalidade dos participantes respondeu afirmativamente, o que corresponde a 100% da amostra. Este resultado revela uma aceitação da atividade desenvolvida e o potencial de continuidade de expansão de iniciativas semelhantes. A unanimidade das respostas reforça a validade pedagógica do modelo e evidencia o interesse dos participantes em aprofundar os conhecimentos adquiridos.

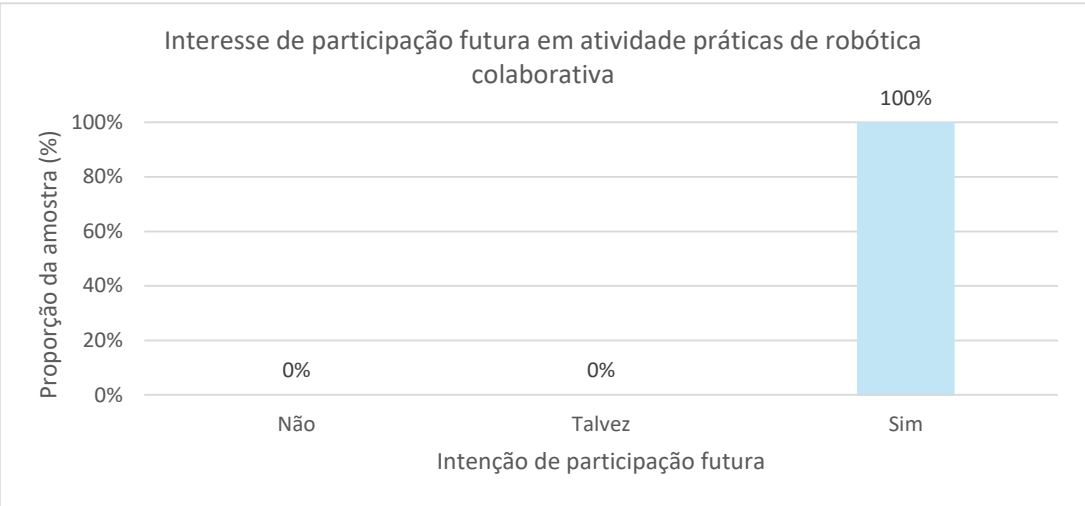


Figura 92 - Distribuição percentual das respostas obtidas à questão que avalia o nível de interesse dos participantes em participar em atividades futuras na área da robótica colaborativa

De forma global, a ergonomia foi avaliada positivamente, a atividade despertou elevado interesse e todos os participantes manifestaram vontade em repetir experiências futuras na área. Estes resultados evidenciam uma avaliação positiva da bancada didática, da atividade e da metodologia adotada. Além disso, estes indicadores sustentam a validação pedagógica do modelo proposto e a sua capacidade de desenvolver um ambiente de aprendizagem estimulante e sustentável.

Confiabilidade do inquérito

De modo a avaliar a confiabilidade do inquérito realizado, procedeu-se à determinação do alfa de Cronbach (α) através da seguinte expressão. *K* corresponde ao número de perguntas, σ_{Yi}^2 corresponde à variância da soma das respostas a cada pergunta e σ_X^2 corresponde à soma das respostas.

$$\alpha = \left(\frac{K}{K-1}\right) * \left(\frac{\sum_{i=1}^{K} \sigma_{Yi}^2}{\sigma_X^2}\right) \quad (3)$$

Obteve-se um valor de 0,93 para o valor do α, o que revela que as respostas obtidas possuem um elevado grau de confiabilidade. Assim, é expectável que para uma maior amostra se obtenham resultados semelhantes.

7.2. Limitações do sistema

Apesar dos resultados positivos alcançados, o sistema desenvolvido apresenta algumas limitações tanto a nível técnico como pedagógico.

Uma das principais limitações é a dependência da interface PolyScope. A programação foi realizada maioritariamente através desta plataforma, que, embora intuitiva e adequada aos objetivos didáticos, não oferece a flexibilidade necessária para explorar cenários mais avançados. A ausência de uma abordagem mais profunda em URScript limita a capacidade de expansão da solução e afasta a experiência do contexto industrial.

Outro constrangimento identificado relaciona-se com o tempo de execução dos exercícios. A carga horária exigida para completar o guião laboratorial revelou-se excessiva, o que contribuiu para a fadiga dos participantes. Esta situação afetou a motivação e o desempenho nas fases finais da atividade. A divisão do guião em duas sessões ou a reestruturação da extensão de determinadas tarefas podem mitigar este impacto.

Finalmente, o sistema não contempla integração com tecnologias complementares, como PLCs, sensores externos ou sistemas de visão artificial. Embora a solução responda aos objetivos pedagógicos de iniciação, a ausência destes elementos compromete o realismo industrial da experiência e limita o seu potencial formativo em contextos mais exigentes.

7.3. Feedback de utilizadores

A análise das respostas ao inquérito permitiu identificar aspetos relevantes sobre a experiência dos participantes durante a utilização da bancada didática. As opiniões recolhidas revelam uma valorização da componente prática, com destaque para a utilidade dos exercícios na consolidação dos conhecimentos e para o desenvolvimento da autonomia na programação de robôs colaborativos.

Os participantes demonstraram um elevado grau de envolvimento emocional, refletido no entusiasmo com que descreveram a atividade e na vontade expressa de participar em iniciativas semelhantes. Esta reação confirma a eficiência da abordagem adotada, não só na componente prática e técnica como também na motivacional.

A estrutura sequencial dos exercícios foi considerada adequada e permitiu uma evolução das competências ao longo da atividade. No entanto algumas observações apontam para a necessidade de incluir desafios adicionais, especialmente para participantes com maior experiência prévia, de forma a garantir um estímulo contínuo e ajustado ao perfil de cada utilizador.

A ergonomia da bancada didática foi reconhecida como um fator que facilitou a aprendizagem, ao proporcionar um ambiente confortável e funcional que favoreceu a concentração e a execução das tarefas. A clareza das instruções e a coerência entre os materiais disponibilizados contribuíram para uma execução fluida das tarefas e para o sucesso da atividade.

As sugestões apresentadas pelos participantes refletem uma atitude crítica e construtiva, o que evidencia o interesse em contribuir para o aperfeiçoamento da proposta desenvolvida. Entre as recomendações mais frequentes surgem propostas para diversificar os cenários de aplicação e reforçar a componente teórica do guião.

Em suma, os resultados obtidos ao longo deste estudo demonstram que a bancada didática desenvolvida cumpriu os objetivos propostas e revelou-se uma ferramenta eficaz no ensino de robótica colaborativa. A realização da atividade prática com a amostra de 14 participantes, foi fundamental para validar o impacto formativo e a aplicabilidade da solução. Observou-se uma melhoria significativa na aprendizagem dos participantes, acompanhada por um aumento do interesse e da motivação para explorar esta área tecnológica. A combinação entre prática orientada, ambiente ergonómico e envolvimento emocional revelou-se determinante para criar uma experiência educativa completa. Este equilíbrio potenciou o desenvolvimento de competências técnicas e pessoais, o que reforçou a confiança dos participantes e promoveu uma aprendizagem mais significativa e duradoura. Assim, conclui-se que a bancada didática não só responde às exigências pedagógicas atuais, como também contribui para a valorização da robótica colaborativa.

8. Conclusões

No presente capítulo apresentam-se as conclusões finais da dissertação, as limitações identificadas no sistema e as propostas de trabalhos futuros.

8.1. Conclusões finais

O presente trabalho teve como principais objetivos desenvolver, implementar e validar uma bancada didática de robótica colaborativa baseada no robô UR3e, orientada para o ensino em contexto académico. A análise global do projeto confirma que os objetivos definidos foram alcançados, com resultados relevantes nas dimensões **científica, técnica, pedagógica e crítica**.

Do ponto de vista **científico**, a revisão da literatura consolidou uma base teórica sólida sobre automação, robótica colaborativa e Indústria 5.0, e permitiu identificar lacunas e justificar a relevância da proposta. No plano **técnico**, a conceção e a construção da bancada materializaram uma estrutura didática modular, ergonómica e segura, capaz de integrar o robô UR3e com periféricos essenciais para recriar cenários de automação. No domínio **pedagógico**, a elaboração do guião laboratorial e a realização de exercícios práticos proporcionaram um ambiente de aprendizagem que favoreceu a aquisição de competências técnicas e o desenvolvimento de competências pessoais, nomeadamente autonomia, confiança e capacidade de resolução de problemas. Finalmente, numa perspetiva **crítica e avaliativa**, os testes laboratoriais com utilizadores e a análise do *feedback* recolhido confirmaram a utilidade da solução, e evidenciaram aspetos a melhorar e oportunidades para trabalhos futuros

A construção da estrutura física, a integração dos sistemas de entradas e saídas e a elaboração de exercícios com níveis de dificuldade progressivos resultaram numa solução que alia competência técnica a relevância pedagógica. A realização deste trabalho permitiu não só atingir os objetivos propostos, como também evidenciar aspetos fundamentais para a eficácia do processo de aprendizagem, nomeadamente a importância da clareza das instruções, da progressividade dos exercícios e da ergonomia da estrutura didática. Verificou-se que apesar de o desempenho técnico da bancada contribuir de forma significativa para o ensino de robótica colaborativa, é o desenvolvimento de material pedagógico (guiões e manual de utilizador) que potencializa a utilidade do equipamento.

Entre as vantagens do projeto destacam-se a flexibilidade que permite a integração de novas tecnologias, a modularidade, a ergonomia da bancada, a integração funcional do robô com os periféricos, a qualidade dos materiais de apoio e a possibilidade de replicação da solução

noutros contextos académicos. Como desvantagens, salientam-se a ausência de acesso remoto, a limitação tecnológica (sem PLCs ou sistemas de visão) e a reduzida dimensão da amostra de validação.

Na Tabela 20 apresenta-se uma síntese da avaliação do cumprimento dos objetivos propostos e ações tomadas no decorrer da dissertação.

Tabela 20 - Cumprimento dos objetivos definidos

Objetivo	Ações tomadas	Avaliação
Projetar uma estrutura modular, ergonómica e segura	Foi desenvolvido o modelo tridimensional da bancada em SolidWorks, com foco na ergonomia, segurança e modularidade do conjunto. A estrutura foi construída em laboratório com materiais adequados e resistentes. Além disso, validou-se a estabilidade e a acessibilidade da solução.	✔
Integrar o robô UR3e com sistemas de I/Os digitais e analógicas	O robô colaborativo foi conectado a um painel de I/Os digitais e analógicas, o que permitiu simular diversos cenários de automação. Desenvolveram-se esquemas elétricos em EPLAN e assegurou-se a correta comunicação entre o controlador e os periféricos. Por fim, realizaram-se testes experimentais que confirmaram a correta instalação e operacionalidade dos equipamentos.	✔
Implementar acesso remoto integrado	Não existiu oportunidade para implementar esta funcionalidade devido a limitações de tempo e à ausência de protocolos de segurança previamente definidos para a comunicação remota segura. O trabalho restringiu-se ao uso presencial da bancada, o que constitui uma limitação da solução.	✘
Integrar sistema de visão	O sistema de visão foi integrado no robô, no entanto não foi utilizado e testado por falta de tempo. Com recurso a *softwares* e algoritmos de *machine learning* é possível desenvolver um sistema de visão artificial.	✔/✘
Desenvolver exercícios práticos progressivos	Foram desenvolvidos exercícios de complexidade crescente, que abordam diferentes cenários desde tarefas básicas de movimentação até operações de paletização e *pick and place*. Cada exercício foi documentado com instruções claras, apoiando a consolidação gradual de competências em programação.	✔

Tabela 20 - Cumprimento dos objetivos definidos (continuação)

Elaborar material de apoio, nomeadamente um guião laboratorial	Foi desenvolvido um guião laboratorial estruturado, onde se incluiu regras de segurança no manuseamento e programação de robôs colaborativos, fundamentos teóricos sobre as instruções de programação utilizadas na resolução dos exercícios, os objetivos de cada exercício e a metodologia passo a passo de resolução. O documento foi testado em contexto real e recebeu *feedback* positivo quanto à clareza e organização, o que reforçou a sua utilidade pedagógica.	✔
Validar a solução em sessões laboratoriais e recolher *feedback* de utilizadores	Realizaram-se sessões experimentais com estudantes, seguidas da recolha de *feedback,* com recurso a inquéritos respondidos pelos utilizadores. Os inquéritos estavam organizados em cinco temas o que permitiu tirar conclusões sobre a organização da atividade, o conteúdo técnico, o desenvolvimento pessoal dos participantes e a ergonomia e conforto da bancada. A análise das respostas confirmou a utilidade e pertinência da solução desenvolvida.	✔

Para além dos objetivos definidos, elaborou-se um manual de utilizador que complementa o guião laboratorial, constituindo uma mais-valia para a utilização autónoma da bancada. O trabalho permitiu ainda desenvolver competências pessoais relevantes, nomeadamente a autonomia na gestão de tarefas e a capacidade de planear e orientar um projeto de investigação. A participação nas sessões laboratoriais exigiu a explicação das atividades e o esclarecimento de dúvidas, o que reforçou a capacidade de comunicação em ambiente académico.

Com base no que foi apresentado, é possível concluir que a presente dissertação contribuiu para o avanço da prática educativa em robótica colaborativa, disponibilizou uma estrutura replicável no ensino de engenharia e forneceu uma base sólida para futuras investigações. Ao articular teoria, prática laboratorial e aproximação à realidade industrial, o trabalho reforçou a relevância da robótica colaborativa na formação de profissionais preparados para responder aos desafios da Indústria 5.0.

8.2. Limitações e trabalhos futuros

O resultado final da bancada desenvolvida evidencia que a mesma apresenta tecnologias como sistema de visão e IA, mas carece de acesso remoto e de tecnologias de deteção de códigos. Por este motivo, considera-se que a bancada é de nível intermédio no que respeita à quantidade de tecnologias integradas, mas apresenta margem para otimização futura.

No presente trabalho foi validada a modularidade, aplicabilidade e pertinência pedagógica da bancada didática desenvolvida, o que verifica a sua utilidade no ensino de robótica colaborativa.

Contudo, surgiram limitações que condicionaram o potencial da solução em cenários mais avançados. Neste sentido, torna-se necessário reconhecer essas limitações e apresentar propostas de evolução capazes de otimizar o trabalho realizado.

As principais limitações e respetivas propostas de melhoria são as seguintes:

- **Ausência de acesso remoto** – A bancada apenas pode ser utilizada em contexto presencial, o que restringe a sua aplicação em modelos de ensino híbrido ou à distância. Como proposta de evolução, recomenda-se a integração de um módulo de conectividade que permita a programação e a monitorização remota do sistema, para aumentar a flexibilidade e a acessibilidade da solução.
- **Dependência do PolyScope** – A programação centrou-se essencialmente na interface gráfica, não abrangendo de forma aprofundada o URScript, o que limitou a aproximação a práticas industriais de maior complexidade. A evolução futura deve incluir o desenvolvimento de exercícios mais complexos em URScript, de modo a proporcionar maior realismo e rigor técnico no processo de ensino.
- **Integração tecnológica restrita** – A solução foi implementada apenas com o robô UR3e e um painel de I/Os, sem recurso a PLCs, sistemas de visão artificial ou outros periféricos comuns em células industriais, o que reduziu a abrangência dos cenários laboratoriais. Como trabalho futuro, recomenda-se a integração destas tecnologias adicionais, para enriquecer os exercícios e aproximar a bancada das condições industriais reais.
- ***Gripper* limitado** – O *gripper* disponível possui uma abertura máxima de 30 mm, o que condiciona o tipo e a dimensão das peças manipuladas. Para ultrapassar esta limitação, propõe-se o desenvolvimento de um *gripper* de atuação pneumática com ventosas, capaz de manipular objetos de diferentes formas e tamanhos.
- **Dimensão reduzida da amostra de participantes** – A fase de validação contou com um número limitado de utilizadores, o que dificultou a generalização dos resultados obtidos. Recomenda-se a realização de novos testes com grupos mais numerosos, de modo a desenvolver uma avaliação do impacto pedagógico da bancada estatisticamente mais sólida.

Em síntese, as limitações identificadas não comprometem a utilidade pedagógica da bancada, mas constituem oportunidades de evolução e melhoria. O seu aproveitamento contribui para aumentar a versatilidade, o realismo industrial e o impacto formativo da solução.

Referências

[1] A. K. Gupta, S. K. Arora, e J. R. Westcott, *Industrial Automation and Robotics*. Dulles: Mercury Learning and Information, 2015.

[2] M. P. Groover, *Automation, Production systems, and Computer-Integrated Manufacturing*, 4ª Edição. Londres, Reino Unido: Pearson Higher Education, 2015.

[3] G. Morel, C. E. Pereira, e S. Y. Nof, «Historical survey and emerging challenges of manufacturing automation modeling and control: A systems architecting perspective», *Annu. Rev. Control*, vol. 47, pp. 21–34, 2019, doi: 10.1016/j.arcontrol.2019.01.002.

[4] F. A. Lievano-Martínez, J. D. Fernández-Ledesma, D. Burgos, J. W. Branch-Bedoya, e J. A. Jimenez-Builes, «Intelligent Process Automation: An Application in Manufacturing Industry», *Sustain.*, vol. 14, n. 14, 2022, doi: 10.3390/su14148804.

[5] C. W. de Silva, «Intelligent robotics-misconceptions, current trends, and opportunities», *Intell. Robot.*, vol. 1, n. 1, pp. 3–17, 2021, doi: 10.20517/ir.2021.01.

[6] «Ultimate, the first industrial robot». Acedido: 24 de Novembro de 2024. Disponível em: https://www.youtube.com/watch?v=hxsWeVtb-JQ&t=22s

[7] «Welding robots in an automotive plant». Acedido: 24 de Novembro de 2024. Disponível em: https://www.youtube.com/watch?v=0L7Xk5_s3QQ

[8] S. Z. Jovanović, J. S. Đurić, e T. V Šibalija, «Robotic Process Automation: Overview and Opportunities», *Int. J. ' 'Advanced Qual.*, vol. 46, n. May, 2019.

[9] M. Bahrin, M. Othman, N. Azli, e M. Talib, «Industry 4.0: A REVIEW ON INDUSTRIAL AUTOMATION AND ROBOTIC», vol. 1, 2015.

[10] «FREEP!K». Acedido: 24 de Novembro de 2024. Disponível em: https://www.freepik.com/free-photo/assembly-line-production-new-car-automated-welding-car-body-production-line-robotic-arm-car-production-line-is-working_26150931.htm

[11] D. B. Bunse, P. H. Kagermann, e P. W. Wahlster, «Industrie 4.0 Smart Manufacturing for the Future», *Berlin Ger. Trade Invest*, vol. 26, n. November, 2014.

[12] J. Amador, «Robôs nas Empresas Portuguesas», vol. X, Abril de 2024.

[13] B. A. Kadir, O. Broberg, e C. Souza Da Conceição, «Designing human-robot collaborations in industry 4.0: Explorative case studies», *Proc. Int. Des. Conf. Des.*, vol. 2, 2018, doi: 10.21278/idc.2018.0319.

[14] R. Bogue, «Europe continues to lead the way in the collaborative robot business», *Ind. Robot Int. J. Robot. Res. Appl.*, vol. 43, n. 1, 2016, doi: 10.1108/IR-10-2015-0195.

[15] S. Patil, V. Vasu, e K. V. S. Srinadh, «Advances and perspectives in collaborative robotics:

a review of key technologies and emerging trends», *Discov. Mech. Eng.*, vol. 2, n. 1, 2023, doi: 10.1007/s44245-023-00021-8.

[16] M. Faccio, M. Bottin, e G. Rosati, «Collaborative and traditional robotic assembly: a comparison model», *Int. J. Adv. Manuf. Technol.*, vol. 102, n. 5–8, pp. 1355–1372, 2019, doi: 10.1007/s00170-018-03247-z.

[17] S. Schillmoeller, «BMW Group Harnesses Potential of Innovative Automation and Flexible Assistance Systems in Production», Munique, 2 de Março de 2017.

[18] ISO, «ISO/TS 15066:2016 Robots and robotic devices — Collaborative robots». Acedido: 10 de Dezembro de 2024. Disponível em: https://www.iso.org/standard/62996.html

[19] Ł. Sobaszek e A. Gola, «Perspective and methods of human-industrial robots cooperation», *Trans Tech Publ. Ltd*, vol. 791, 2015, doi: 10.4028.

[20] S. Electric, «schneider electric implementa robots baxter en su cadena de montaje». Acedido: 11 de Dezembro de 2024. Disponível em: https://www.infoplc.net/historias-exito/item/102267-schneider-electric-implementa-robots-baxter-cadena-montaje

[21] ABB, «YuMi takes over THT assembly». Acedido: 11 de Dezembro de 2024. Disponível em: https://new.abb.com/news/detail/73577/cstmr-yumi-takes-over-tht-assembly

[22] U. Robots, «Universal robots products». Acedido: 11 de Dezembro de 2024. Disponível em: https://www.universal-robots.com/media/442094/ur3_left.png

[23] G. Charalambous, S. Fletcher, e P. Webb, «Development of a Human Factors Roadmap for the Successful Implementation of Industrial Human-Robot Collaboration», vol. 490, 2016, doi: https://doi.org/10.1007/978-3-319-41697-7_18.

[24] Fanuc, «Fanuc». Acedido: 22 de Dezembro de 2024. Disponível em: https://crx-completesme.fanuc.eu/smartguide/?utm_term=collaborative robots&utm_campaign=&utm_source=adwords&utm_medium=ppc&hsa_acc=5417858290&hsa_cam=21941141482&hsa_grp=172091824318&hsa_ad=723246583596&hsa_src=g&hsa_tgt=kwd-534580774957&hsa_kw=collaborat

[25] «Representação do sistema de eixos do cobot». Acedido: 1 de Setembro de 2025. Disponível em: http://m.airkingrobots.com/nd.jsp?mid=5&id=96

[26] I. Staretu, «COLLABORATIVE ROBOTS (COBOTS) – SYSTEMATIZATION AND KINEMATICS», vol. 66, pp. 81–89.

[27] U. Robots, «Robôs colaborativos para a indústria automóvel». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.universal-robots.com/pt/industrias/automóvel-e-fornecedores/

[28] U. Robots, «INDUSTRIAL COLLABORATIVE ROBOTS IN FOOD & Beverage». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.universal-robots.com/fi/toimialat/food-and-beverage/

[29] U. Robots, «REVOLUCIONANDO A INDÚSTRIA FARMACÊUTICA COM COBOTS». Acedido: 30 de Dezembro de 2024. Disponível em: https://br.videos.universal-robots.com/revolucionando-a-industria/join

[30] A. Robotics, «Amazon Robotics usa Amazon SageMaker e AWS Inferentia para habilitar inferências de ML em escala». Acedido: 30 de Dezembro de 2024. Disponível em: https://aws.amazon.com/pt/solutions/case-studies/amazon-robotics-case-study/

[31] ARROW, «Industrial Protocols Comparison: Fieldbus vs Ethernet & More». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.arrow.com/en/research-and-events/articles/industrial-connectivity-protocols

[32] U. Robots, «Os 10 protocolos de comunicação mais usados na automação industrial». Acedido: 27 de Dezembro de 2024. Disponível em: https://www.universal-robots.com/br/blog/os-10-protocolos-de-comunicacão-mais-usados-na-automacão-industrial/

[33] A. Kocamuftuoglu, O. Akbay, e S. Kaba, «A Comparative Study on Industrial Communication Protocols Using IoT Platforms», *Eurasia Proc. Sci. Technol. Eng. Math.*, vol. 14, pp. 57–65, 2021, doi: 10.55549/epstem.1050178.

[34] LGEPC, «Thèmes de recherche». Acedido: 27 de Dezembro de 2024. Disponível em: https://lgepc.dz/services/

[35] F. S. D. B. Pereira, M. C. D. O. Duarte, e W. D. A. Inácio, «Um Estudo sobre Analisadores de Protocolos de Comunicação para a Rede PROFINET», Minas Gerais, Brasil, 2024.

[36] A. K. Guislande e D. A. L. C. Dias, «Estudo e Implementação de uma Aplicação Industrial Utilizando o Protocolo Ethernet / IP», Minas Gerais, Brasil.

[37] M. Breque, L. De Nul, e A. Petridis, *Industry 5.0*. Luxemburgo: Europian Commission, 2021. doi: 10.1002/9781119865216.ch2.

[38] J. Barata e I. Kayser, «Industry 5.0 - past, present, and near future», *Procedia Comput. Sci.*, vol. 219, pp. 778–788, 2023, doi: 10.1016/j.procs.2023.01.351.

[39] S. Proia, R. Carli, G. Cavone, e M. Dotoli, «Control Techniques for Safe, Ergonomic, and Efficient Human-Robot Collaboration in the Digital Industry: A Survey», *IEEE Trans. Autom. Sci. Eng.*, vol. 19, n. 3, pp. 1798–1819, 2022, doi: 10.1109/TASE.2021.3131011.

[40] M. Silveira *et al.*, «3D VISION OBJECT IDENTIFICATION USING YOLOv8», *Int. J. Mechatronics Appl. Mech.*, vol. 2024, n. 17, pp. 7–15, 2024, doi: 10.17683/ijomam/issue17.1.

[41] A. Keshvarparast, D. Battini, O. Battaia, e A. Pirayesh, *Collaborative robots in manufacturing and assembly systems: literature review and future research agenda*, vol. 35, n. 5. Springer US, 2024. doi: 10.1007/s10845-023-02137-w.

[42] A. Agrawal, Yu Sun, J. Barnwell, e R. Raskar, «Vision-guided robot system for picking objects by casting shadows», *Int. J. Rob. Res.*, vol. 29, n. 2–3, pp. 155–173, 2010, doi: 10.1177/0278364909353955.

[43] M. Javaid, A. Haleem, R. P. Singh, S. Rab, e R. Suman, «Significant applications of Cobots in the field of manufacturing», *Cogn. Robot.*, vol. 2, n. October, pp. 222–233, 2022, doi: 10.1016/j.cogr.2022.10.001.

[44] T. Jiang, H. Cui, e X. Cheng, «A calibration strategy for vision-guided robot assembly system of large cabin», *Meas. J. Int. Meas. Confed.*, vol. 163, p. 107991, 2020, doi: 10.1016/j.measurement.2020.107991.

[45] A. A. Khan, A. A. Laghari, e S. A. Awan, «Machine Learning in Computer Vision: A Review», *EAI Endorsed Trans. Scalable Inf. Syst.*, vol. 8, n. 32, pp. 1–11, 2021, doi: 10.4108/eai.21-4-2021.169418.

[46] COGNEX, «Como verificar códigos 1-D». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.cognex.com/pt-br/what-is/barcode-verification/how-to-verify-1d-

codes

[47] KEYENCE, «O que são códigos 2D». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.keyence.com.br/ss/products/auto_id/barcode_lecture/basic_2d/intro/

[48] COGNEX, «Embossed OCR Code Reading». Acedido: 30 de Dezembro de 2024. Disponível em: https://www.cognex.com/pt-br/industries/consumer-products/labeling-marking/embossed-ocr-code-reading

[49] P. Amiri, M. Müller, e M. Richards-brown, «A Statistical Analysis of Commercial Articulated Industrial Robots and Cobots», *Res. Sq.*, 2024.

[50] A. Borboni, K. V. V. Reddy, I. Elamvazuthi, M. S. AL-Quraishi, E. Natarajan, e S. S. Azhar Ali, «The Expanding Role of Artificial Intelligence in Collaborative Robots for Industrial Applications: A Systematic Review of Recent Works», *Machines*, vol. 11, n. 1, 2023, doi: 10.3390/machines11010111.

[51] R. Galin, R. Meshcheryakov, S. Kamesheva, e A. Samoshina, «Cobots and the benefits of their implementation in intelligent manufacturing», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 862, n. 3, 2020, doi: 10.1088/1757-899X/862/3/032075.

[52] F. Pereira e J. Machado, *Sistemas de acesso remoto a máquinas e processos industriais Parte 1*. Porto, Portugal: Quânta Editora - Conteúdos Especializados, Lda, 2024.

[53] M. Intelligence, «Tamanho do mercado de robôs colaborativos e análise de ações - Tendências e previsões de crescimento (2024-2029)». Acedido: 9 de Novembro de 2024. Disponível em: https://www.mordorintelligence.com/pt/industry-reports/collaborative-robot-market

[54] U. Robots, «Sobre a empresa». Acedido: 8 de Novembro de 2024. Disponível em: https://www.universal-robots.com/br/sobre-a-universal-robots/

[55] G. Skills, «Robô colaborativo». Acedido: 9 de Novembro de 2024. Disponível em: https://growskills.pt/robotica/robotica-colaborativa/robos-colaborativos-universal-robots/

[56] T. Robot, «Company Profile». Acedido: 8 de Novembro de 2024. Disponível em: https://www.tm-robot.com/en/company-profile/

[57] T. Robot, «TM ROBOT x OMRON». Acedido: 8 de Novembro de 2024. Disponível em: https://www.tm-robot.com/en/techman-x-omron/

[58] T. Robot, «TM ROBOT». Acedido: 8 de Novembro de 2024. Disponível em: https://www.tm-robot.com/en/

[59] «OMRON TM Collaborative Robot». Acedido: 2 de Agosto de 2025. Disponível em: https://www.omron.com.au/solutions/robotic-solutions/robotics/tm-series/

[60] E. Montini *et al.*, «Collaborative Robotics: A Survey From Literature and Practitioners Perspectives», *J. Intell. Robot. Syst. Theory Appl.*, vol. 110, n. 3, 2024, doi: 10.1007/s10846-024-02141-z.

[61] F. M. Amin, M. Rezayati, e H. W. Van De Venn, «A Mixed-Perception Approach for Safe Human – Robot», *Sensors*, vol. 20, n. 6347, 2020.

[62] M. Peruzzini, E. Prati, e M. Pelicciari, «A framework to design smart manufacturing systems for Industry 5.0 based on the human-automation symbiosis», *Int. J. Comput. Integr. Manuf.*, vol. 37, n. 10–11, pp. 1426–1443, 2023, doi:

10.1080/0951192X.2023.2257634.

[63] A. A. Santos *et al.*, «Integration of Artificial Vision and Image Processing into a Pick and Place Collaborative Robotic System», *J. Intell. Robot. Syst. Theory Appl.*, vol. 110, n. 4, 2024, doi: 10.1007/s10846-024-02195-z.

[64] H. ASAAD, S. ASKAR, A. KAKAMIN, e N. FAIQ, «Exploring the Impact of Artificial Intelligence on Humanrobot Cooperation in the Context of Industry 4.0», *Appl. Comput. Sci.*, vol. 20, n. 2, pp. 138–156, 2024, doi: 10.35784/acs-2024-21.

[65] M. Peterhansl, «Remote Control of a Collaborative Robot with Virtual Reality and Joystick in a 5G Network», *Proc. - Int. Conf. Adv. Comput. Inf. Technol. ACIT*, pp. 473–478, 2023, doi: 10.1109/ACIT58437.2023.10275571.

[66] «RTS-200 : ROBOTICS TRAINING SYSTEM».

[67] «SMC International Training». Acedido: 30 de Outubro de 2024. Disponível em: https://www.smctraining.com/en/webpage/indexpage/55

[68] G. Skills, «Grow Skills». Acedido: 30 de Outubro de 2024. Disponível em: https://growskills.pt/didatics/smc-training/robotica-smc/

[69] S. I. Training, «SMC International Training». Acedido: 30 de Outubro de 2024. Disponível em: https://www.smctraining.com/en/productconfigurator/dynamicconfigurator/48/1

[70] T. Robot, «TM Academy». Acedido: 29 de Novembro de 2024. Disponível em: https://www.academy.tm-robot.com/product

[71] G. Skills, «Grow Skills». Acedido: 9 de Novembro de 2024. Disponível em: https://growskills.pt/

[72] G. Skills, «Bancada Robótica GrowSkills». Acedido: 9 de Novembro de 2024. Disponível em: https://growskills.pt/didatics/kit-didatico-ur/

[73] I. Azevedo, M. Pinto, A. F. Silva, e C. Felgueiras, «Teaching Automation: A Review of Industrial Automation Learning Systems», em *Machado, J., Trojanowska, J., Soares, F., Rea, P., Butdee, S., Gramescu, B. (eds) Innovations in Mechatronics Engineering IV. icieng 2025*, Springer, Cham, 2025. doi: https://doi.org/10.1007/978-3-031-94223-5_39.

[74] A. Gómez-Espinosa *et al.*, «Design and construction of a didactic 3-DOF parallel links robot station with a 1-DOF gripper», *J. Appl. Res. Technol.*, vol. 12, n. 3, pp. 435–443, 2014, doi: 10.1016/S1665-6423(14)71624-4.

[75] L. Kobras, B. Meussen, e M. Soll, «Didactic Design of a Remote Collaborative Robotics Laboratory», *2023 IEEE 2nd Ger. Educ. Conf. GECon 2023*, 2023, doi: 10.1109/GECon58119.2023.10295126.

[76] A. S. Nakashima, «Automated industrial inspection workbench for human machine interface (HMI) consoles», Instituto Politécnico de Bragança, 2018.

[77] A. Filipescu, G. Simion, D. Ionescu, e A. Filipescu, «IoT-Cloud , VPN , and Digital Twin-Based Remote Monitoring and Control of a Multifunctional Robotic Cell in the Context of AI, Industry, and Education 4.0 and 5.0», *MDPI*, 2024, doi: 10.3390/ s24374511.

[78] F. Pereira, R. Lopes, R. Sarmento, e C. Felgueiras, «Didactic bench to support automation learning - Industry 4.0 education», *15th Int. Conf. Technol. Learn. Teach. Electron. TAEE 2022 - Proc.*, pp. 1–5, 2022, doi: 10.1109/TAEE54169.2022.9840744.

[79] N. Lima, C. Viegas, e F. Garcia-Pêalvo, «Didactical use of a remote lab: A qualitative

reflection of a teacher», *ACM Int. Conf. Proceeding Ser.*, pp. 99–108, 2019, doi: 10.1145/3362789.3362891.

[80] E. Mercier, M. H. Goldstein, P. Baligar, e R. Jephthah Rajarathinam, *Collaborative Learning in Engineering Education*. 2023. doi: 10.4324/9781003287483-23.

[81] N. Caetano e M. Felgueiras, «Sustainable development in higher education: Different teaching & learning approaches», *ACM Int. Conf. Proceeding Ser.*, pp. 469–472, 2019, doi: 10.1145/3362789.3362950.

[82] M. Pinto, I. Azevedo, P. J. Silva, e A. M. Lopes, «Collaborative Robotics Learning Kits : A Literature Review», em *Machado, J., Trojanowska, J., Soares, F., Rea, P., Butdee, S., Gramescu, B. (eds) Innovations in Mechatronics Engineering IV. icieng 2025*, Springer, Cham, pp. 246–257. doi: https://doi.org/10.1007/978-3-031-93554-1_22.

[83] «ONE PROFILE». Acedido: 4 de Setembro de 2025. Disponível em: https://www.oneprofile.pt/#

[84] «80/20 BUILD YOUR IDEA». Acedido: 28 de Julho de 2025. Disponível em: https://8020.net/45-4545-lite.html

[85] «EQUINOTEC». Acedido: 28 de Julho de 2025. Disponível em: https://equinotec.com/produto/parafuso-s12x30-t50/

[86] «Manutan». Acedido: 28 de Julho de 2025. Disponível em: https://www.manutan.pt/pt/map/rodizio-giratorio-com-haste-roscada-m10-x-30-mm-capacidade-de-40-a-70-kg-giratorio-a008311?shopping=true&utm_source=google&utm_medium=cpc&utm_campaign=GG-PM-M-PT-PT-Zombie&gad_source=1&gad_campaignid=22283128994&gbraid=0AAAAA

[87] «Manutan». Acedido: 28 de Julho de 2025. Disponível em: https://www.manutan.pt/pt/map/rodizio-giratorio-com-haste-roscada-m10-x-30-mm-capacidade-de-40-a-70-kg-giratorio-com-travao

[88] «thomann». Acedido: 28 de Julho de 2025. Disponível em: https://www.thomann.pt/the_t.akustik_melamine_50_100_100_gray_4pcs.htm?gad_source=1&gad_campaignid=21039691910&gbraid=0AAAADuDMCX_4Ooq7dGVWgJ1ll80Ulc00&gclid=CjwKCAjwv5zEBhBwEiwAOg2YKHWfBGAK9PrvCf9GMw6O3wDIHjK3hM-JXRd67_22aIf5QagtCUIjHBoCo9sQAvD_BwE

[89] «Parafuso-express.pt». Acedido: 28 de Julho de 2025. Disponível em: https://www.parafuso-express.pt/parafuso-spax-cabeca-redonda-pozi/15636-parafuso-spax-cabeca-redonda-pozi-2-4x35-revestimento-wirox-3663072027275.html?utm_campaign=googleads&utm_source=shopping&utm_medium=allemagne&gad_source=1&gad_campaignid=19749390436&

[90] «Universal Robots». Acedido: 28 de Julho de 2025. Disponível em: https://www.universal-robots.com/pt/produtos/ur3-robot/

[91] rolnorte, «Parafuso sextavado». Acedido: 12 de Agosto de 2025. Disponível em: https://rolnorte.pt/produtos/parafuso-sextavado-interior-din912-inox-a2-puci316/.

[92] Zimmer, «Gripper». Acedido: 11 de Agosto de 2025. Disponível em: https://www.zimmer-group.com/en-us/products/components/handling-technology/hrc-gripper/hrc-03/products/hrc-03-118505.

[93] U. Robots, «Teach Pendant». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/marketplace/products/01tP4000001uE4bIAE/

[94] Limora, «Parafuso cabeça hexagonal». Acedido: 12 de Agosto de 2025. Disponível em: https://www.limora.com/pt/parafuso/11793/?gad_source=1&gad_campaignid=22397684336&gbraid=0AAAAA_AFLc3wLQOte6lVQR5Pzrq10pdkQ&gclid=Cj0KCQjwzOvEBhDVARIsADHfJJRthFHjRN5iK-QcUpnJ65np7jshve-IKtDBUqySuLcgyQ-kIJ0xCDcaAhh4EALw_wcB

[95] DigiKey, «Anilha plana». Acedido: 12 de Agosto de 2025. Disponível em: https://www.digikey.pt/pt/products/detail/essentra-components/173000200022/4104130?gclsrc=aw.ds&gad_source=1&gad_campaignid=20195109022&gbraid=0AAAADrbLlgcN0ctzRcD_7IUCorcdeXad&gclid=Cj0KCQjwzOvEBhDVARIsADHfJJTa3QVK7tI6bAPgU1FuyIZID1ZcQPtg9r60qgDmXSIDgkm

[96] rolnorte, «Porca hexagonal». Acedido: 12 de Agosto de 2025. Disponível em: https://rolnorte.pt/produtos/femea-sextavada-aco-inoxidavel-a2-fai18/

[97] norelem, «Porca martelo». Acedido: 12 de Agosto de 2025. Disponível em: https://norelem.es/pt/Visão-geral-de-produtos/Sistema-flexível-de-peças-normalizadas/07000/Porcas-Parafusos-Arruelas-Elementos-de-segurança/Porcas-martelo/p/agid.32846

[98] U. Robots, «Control Boxes». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/developer/hardware-and-motion/electrical-interfaces-control-box/

[99] U. Robots, «Standard cable extension». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/marketplace/products/01tP40000071NjSIAU/

[100] U. Robots, «Universal Robots». Acedido: 28 de Julho de 2025. Disponível em: https://www.universal-robots.com/br/blog/como-os-cobots-abordam-os-principais-desafios-da-producão-de-carros/

[101] U. Robots, «TCP». Acedido: 2 de Agosto de 2025. Disponível em: https://academy.universal-robots.com/modules/e-Series-core-track/English/module3/story_html5.html?courseId=2166&language=English&trainingId=5xCrknuMIKz

Atividade científica e pedagógica

Artigos publicados e submetidos:

- **Publicado:**

  Pinto, M., Azevedo, I., Silva, P.J., Lopes, A.M., Silva, A.R., Santos, A.A. (2025). Collaborative Robotics Learning Kits: A Literature Review. In: Machado, J., Trojanowska, J., Ottaviano, E., Xavior, M.A., Valášek, P., Basova, Y. (eds) Innovations in Mechanical Engineering IV. icieng 2025. Lecture Notes in Mechanical Engineering. Springer, Cham. https://doi.org/10.1007/978-3-031-93554-1_22

- **Publicações aceites:**

  M. Pinto, I. Azevedo, F. Pereira, A.F. Silva, P.J. Silva, C. Felgueiras and A.A. Santos. Teaching Kit for Collaborative Robotics. International Conference on Machine Design 2025, FEUP, Porto, Portugal, 11-12 September 2025.

  Adriano A. Santos, Filipe Pereira, António Ferreira da Silva, Paulo J. Silva1, Nídia S. Caetano, Florinda F. Martins, Margarida Pinto, Inês Azevedo, Carlos Felgueiras. Remote Laboratories - A Sustainable Path to a Multicultural Higher Education Environment. Proceedings of TEEM 2025. Lecture Notes in Educational Technology. Springer, Singapore. Salamanca, Spain, 21-24 October 2025.

Participação em seminário:

Apresentação e exibição das potencialidades do kit de robótica colaborativa no seminário "Robótica i5.0 Services", realizado no ISEP, 16 maio 2025. O seminário contou com a presença da Sra. Presidente do ISEP, Prof. Maria João Viamonte, do Diretor do DEM, Prof. João Bastos, das Eng. Margarida Pinto e Inês Azevedo do MEM bem como dos Eng. Miguel Oliveira da Universal Robots e do Eng. Ruben Almeida da GrowSkills, moderados pelo Prof. Adriano A. Santos.

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

NOME: Margarida Sofia de Sousa Pinto

ISEP, Porto, 13 de setembro de 2025

Apêndice A

Projeto em EPLAN das ligações elétricas.

EPLAN GmbH & Co. KG

An der alten Ziegelei 2
40789 Monheim am Rhein
Tel. +49 (0)2173 - 39 64 - 0

Empresa / Cliente	
Descrição do projeto	Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designaç
Número de projeto	IEC_bas001
Comissão	EPLAN
Fabricante (Empresa)	EPLAN GmbH & Co. KG
Coluna	Exemplo de projeto EPLAN
Nome do projeto	Esquema elétrico da caixa de controlo do UR3e
Realizar	
Tipo	
Localização de instalação	
Responsável pelo projeto	
Características da peça	

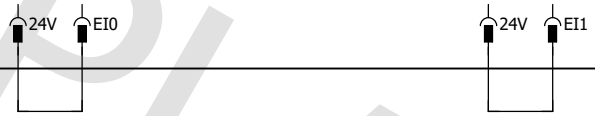
Criado em 12/03/2025
Editado em 05/09/2025 de (abreviatura) Asus Quantidade de páginas 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

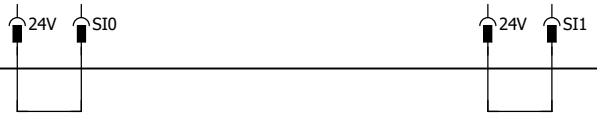
Safety

Emergency Stop



Default connections

Safeguard Stop

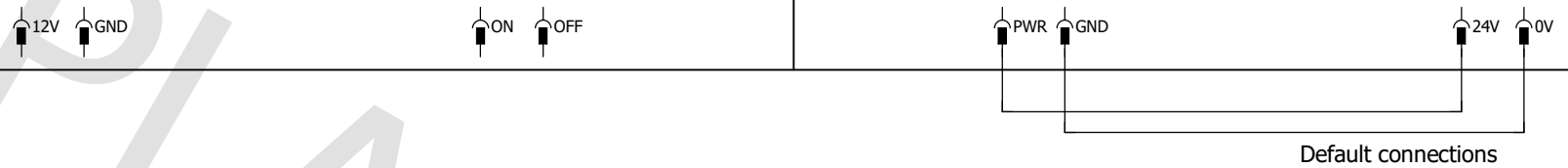


Default connections

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Remote	Power



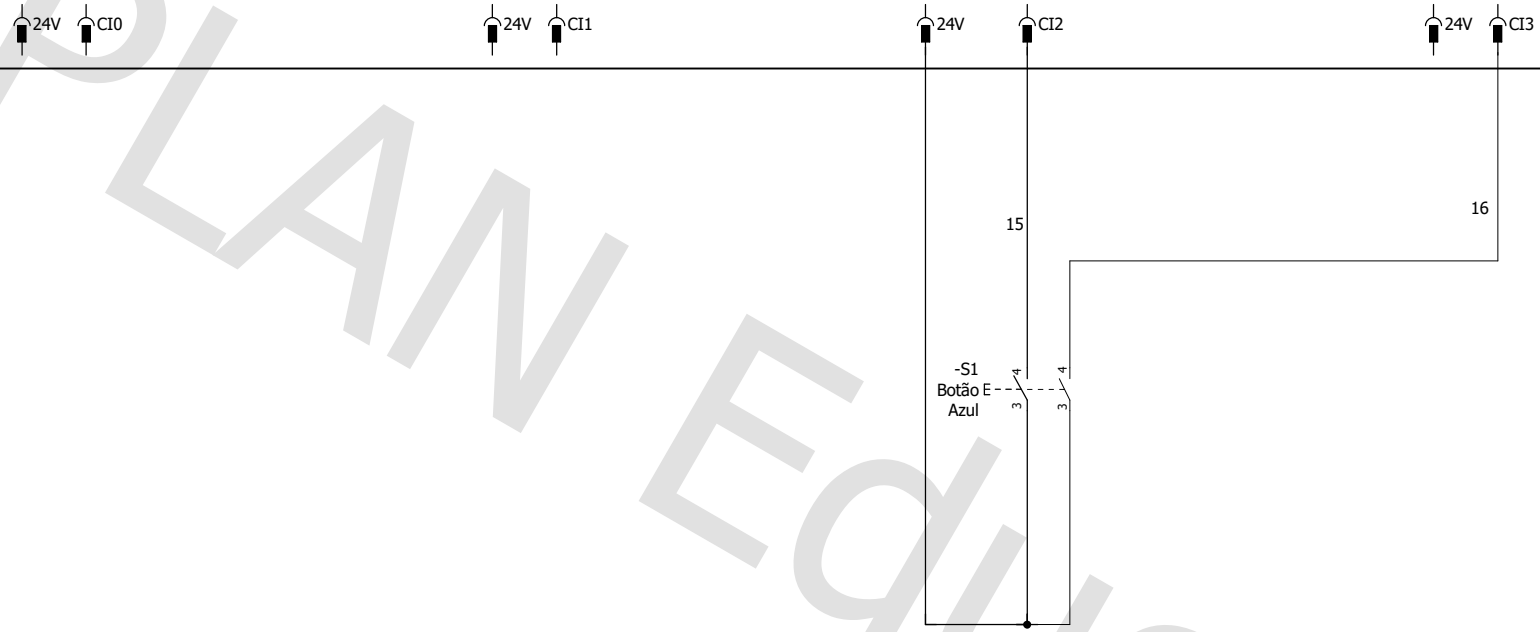
Default connections

16 18

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Configurable Inputs



17 19

			Data	28/06/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	18
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	4 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Configurable Inputs

18 20

			Data	28/06/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	19
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	5 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Configurable Outputs

19 21

			Data	28/06/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3
			Editor	Asus					+
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha 20
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página 6 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Configurable Outputs

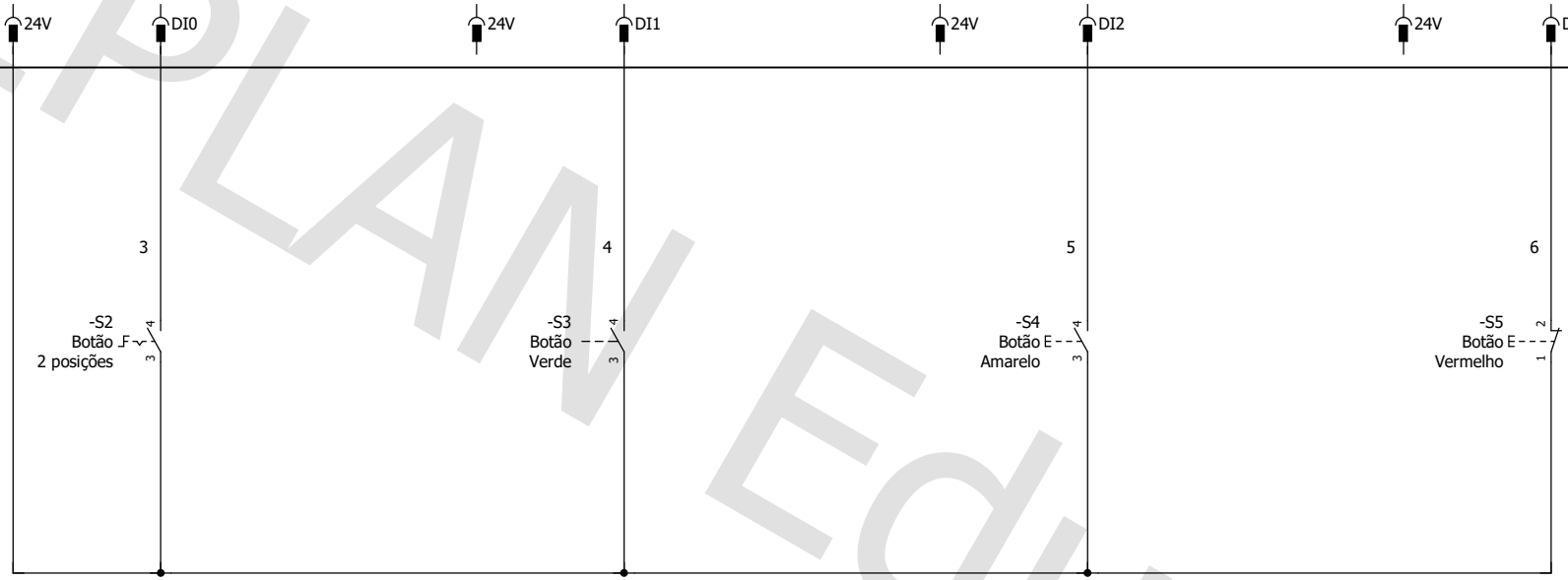
20 22

			Data	28/06/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	21
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	7 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Digital Inputs



21 23

			Data	31/07/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	22
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	8 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Digital Inputs

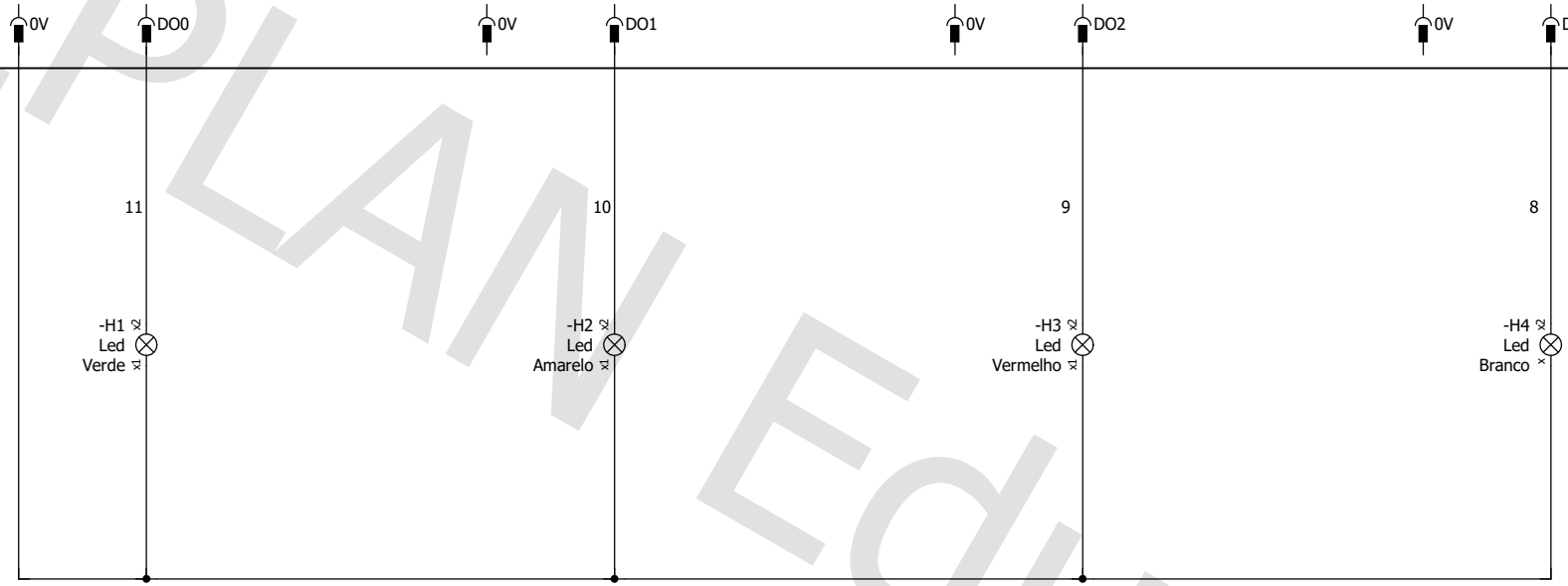
22 24

			Data	28/06/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	23
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	9 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Digital Outputs



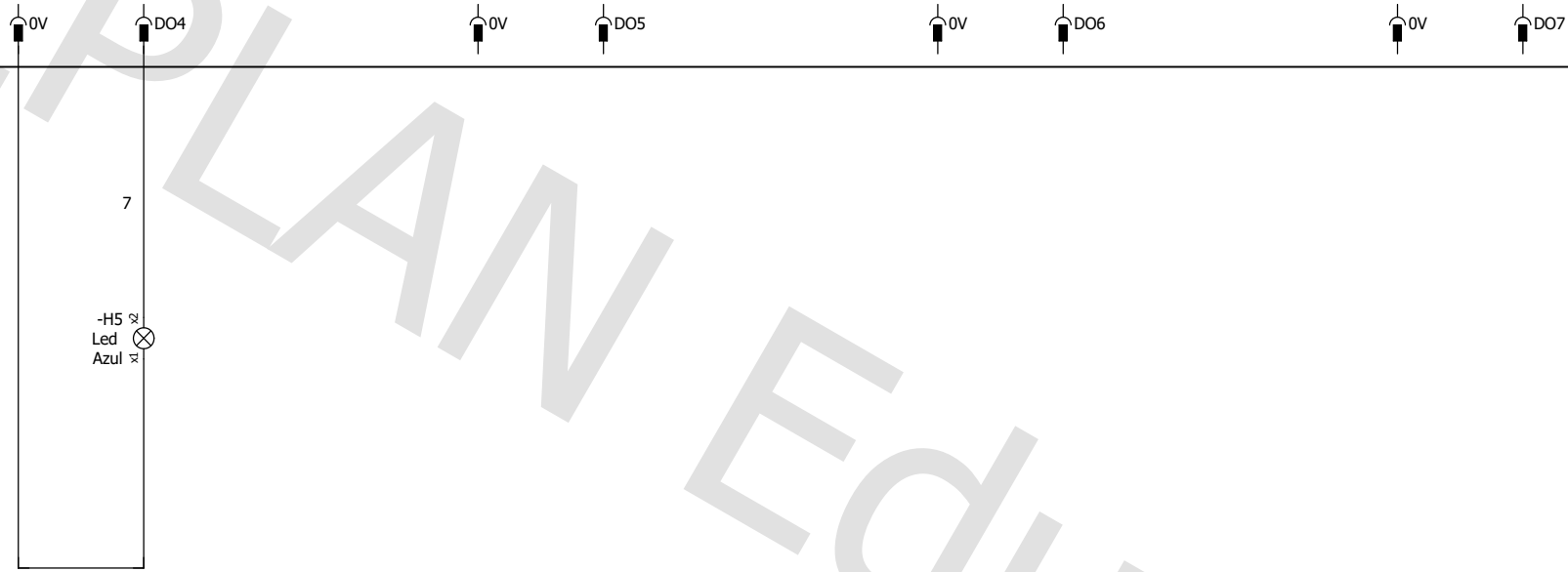
23 25

			Data	05/09/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	24
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	10 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Digital Outputs



			Data	05/09/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3	
			Editor	Asus					+	
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha	25
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página	11 / 12

-UR3

UNIVERSAL ROBOTS CB3.1 UR3

Analog I/O (is not galvanically isolated from Control Box)

Analog Inputs

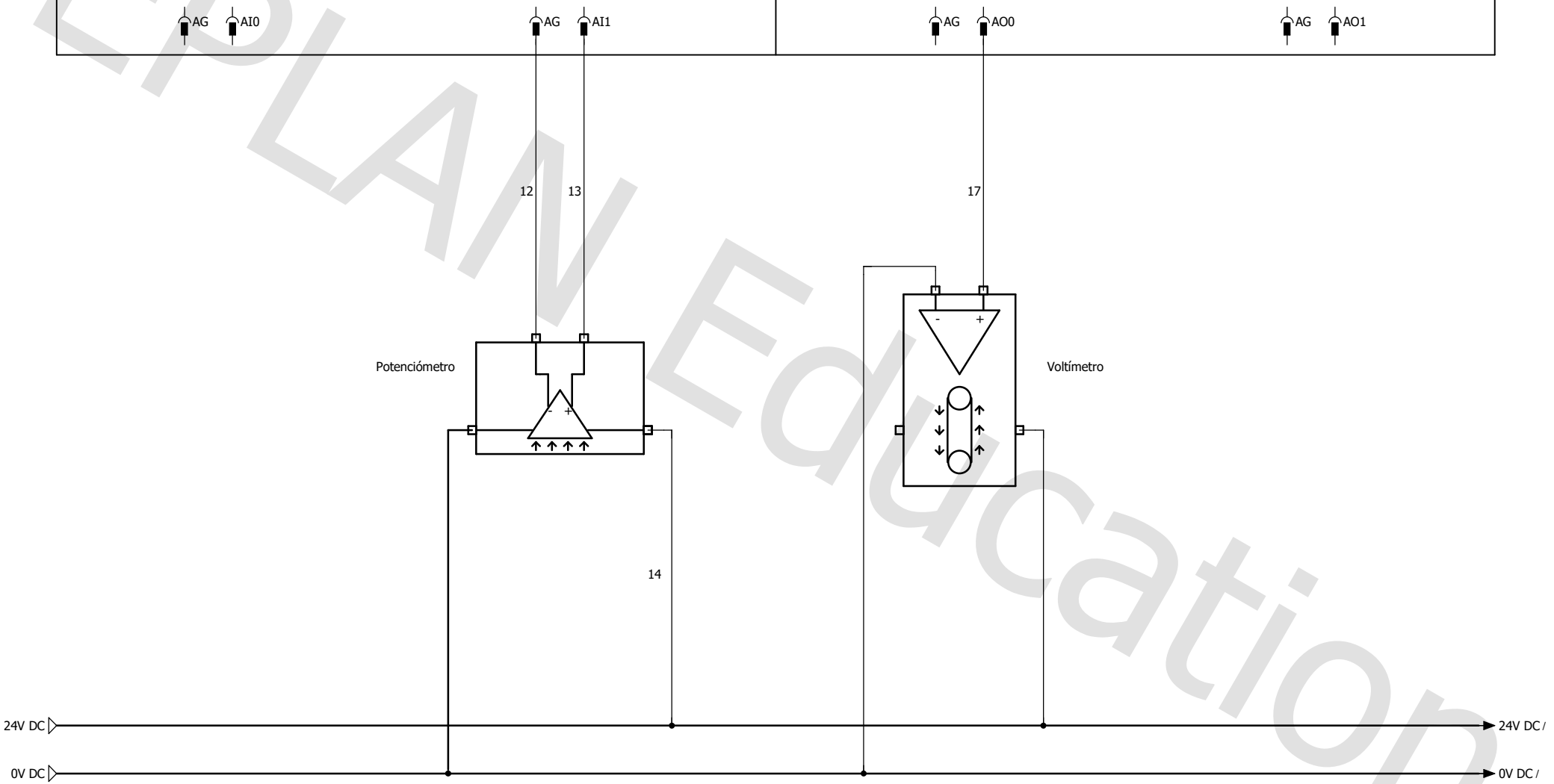
AG AI0

AG AI1

Analog Outputs

AG AO0

AG AO1



25

			Data	29/07/2025	EPLAN	EPLAN GmbH & Co. KG	UR3 Controller I/O		= UR3
			Editor	Asus					+
			Verif.		Projeto de base com estrutura de identificação de acordo com a norma IEC: Estrutura de página com designação da função e designação da localização			IEC_bas001	Folha 26
Alteração	Data	Nome	Orig.		Em substituição de	Substituído por			Página 12 / 12

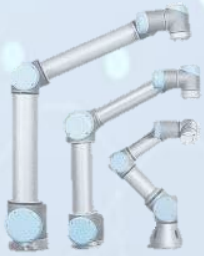
Apêndice B

Manual de utilizador.

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MANUAL DE UTILIZADOR

Robótica colaborativa – UR3e



Margarida Sofia de Sousa Pinto

Porto, setembro 2025

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

1.1. Finalidade do documento

O presente manual tem como objetivo fornecer ao utilizador uma descrição detalhada das instruções de programação do ambiente PolyScope, interface gráfica de controlo e programação dos robôs colaborativos Universal Robots. O documento destina-se especificamente ao modelo UR3e e aborda as instruções básicas, avançadas, *templates* e URCaps, incluindo exemplos práticos de aplicação.

O UR3e é um robô colaborativo de seis eixos, desenvolvido pela Universal Robots, reconhecido pela sua versatilidade, precisão e facilidade de integração em ambientes de produção modernos. Compacto e leve, foi desenvolvido para executar tarefas repetitivas e de precisão em espaços reduzidos, sendo ideal para aplicações como:

- Montagem de componentes eletrónicos e mecânicos;
- Manipulação de peças de pequena dimensão;
- Testes e inspeções em bancada.

1.2. Âmbito de utilização

O conteúdo aqui apresentado aplica-se à criação de programas para o robô UR3e através do PolyScope. O manual não substitui a documentação oficial de instalação, manutenção ou segurança fornecida pelo fabricante. O foco recai exclusivamente na programação de tarefas no ambiente PolyScope, não sendo abrangidos tópicos relacionados com redes industriais ou configuração de sistemas externos.

1.3. Importância das instruções de programação

As instruções disponíveis no PolyScope constituem os elementos fundamentais para a criação de programas de manipulação. Estas permitem:

- Definir trajetórias e pontos de trabalho (*waypoints*);
- Configurar variáveis internas e gerir entradas/saídas digitais e analógicas;
- Implementar lógica condicional e estruturas de repetição;
- Modularizar programas através de sub-rotinas;
- Incorporar funções adicionais com recurso a *templates* e URCaps.

O domínio correto destas instruções é essencial para assegurar a execução precisa das tarefas, minimizar riscos operacionais e garantir a fiabilidade em contextos industriais.

1.4. Normas de segurança aplicáveis

A operação do robô UR3e deve ser sempre realizada em conformidade com as normas internacionais em vigor, nomeadamente:

- **ISO/TS 15066:2016** [1] — Robots and robotic devices - Collaborative robots;
- **ISO 10218-1:2025** [2]— Robotics — Safety requirements - Part 1: Industrial robots;
- **ISO 10218-2:2025** [3]— Robotics - Safety requirements - Part 2: Industrial robot applications and robot cells.

Estas normas estabelecem requisitos técnicos e procedimentais para a utilização segura de robôs industriais e colaborativos, incluindo limites de velocidade, força, carga útil e zonas de operação.

1.5. Responsabilidade do utilizador

O utilizador é responsável por garantir que:

- O robô se encontra instalado em conformidade com as instruções do fabricante;
- O payload e o Tool Center Point (TCP) estão corretamente definidos antes da execução de qualquer programa;
- Os limites de carga, alcance, velocidade e aceleração do UR3e são rigorosamente respeitados;
- São implementadas medidas de segurança adequadas, incluindo delimitação de áreas de trabalho e acessibilidade a botões de paragem de emergência;
- Todos os programas são previamente validados em baixa velocidade e em condições controladas, em conformidade com a análise de risco da aplicação.

2. Regras de segurança

2.1. Considerações gerais

O manuseamento e operação de robôs colaborativos requer a adoção de práticas de segurança rigorosas, com o objetivo de proteger a integridade física dos operadores, evitar danos no equipamento e assegurar a continuidade do processo produtivo. Embora o UR3e seja classificado como robô colaborativo, esta característica não dispensa a implementação de medidas preventivas adequadas.

Todas as operações devem ser realizadas em conformidade com as normas internacionais:

- **ISO 10218-1 e ISO 10218-2** — Requisitos de segurança para robôs industriais e sistemas de robótica;
- **ISO/TS 15066** — Requisitos específicos para robôs colaborativos, incluindo limites de força e contacto físico.

A responsabilidade pela segurança recai sobre o utilizador, que deve garantir que os programas, as condições de operação e o ambiente de trabalho cumprem as normas em vigor e a análise de risco aplicável.

2.2. Procedimentos antes da operação

Antes de iniciar qualquer atividade com o UR3e, é obrigatório cumprir as seguintes verificações:

- **Leitura do manual do fabricante**: consultar a documentação oficial disponibilizada pela Universal Robots;
- **Área de trabalho**: assegurar que o espaço em torno do robô se encontra livre de obstáculos, ferramentas soltas ou objetos suscetíveis de interferir com os movimentos;
- **Inspeção do robô**: confirmar a integridade dos cabos, o correto estado mecânico dos componentes e a operacionalidade de atuadores ou ferramentas externas;
- **Ligações elétricas**: validar a conformidade das conexões de alimentação, entradas e saídas digitais e analógicas;
- **Equipamento individual**: utilizar calçado adequado e vestuário ajustado, evitar roupas largas, joias ou acessórios que possam ser apanhados acidentalmente.

2.3. Procedimentos durante a operação

Durante a execução de programas, devem ser observadas as seguintes práticas:

- **Zonas de exclusão**: nunca introduzir mãos ou qualquer parte do corpo na área de alcance do robô enquanto este estiver em movimento;

- **Botões de emergência**: conhecer a localização e funcionamento de todos os dispositivos de paragem de emergência. Sempre que necessário utilizar o botão de emergência (*E-Stop*) presente no *Teach Pendant* (Figura 1). Para reativar o robô siga as instruções apresentadas no *PolyScope*;

Figura 1 - Botão de emergência (*E-Stop*) presente no *Teach Pendant*

- **Supervisão contínua**: assegurar vigilância constante, em especial durante testes ou em modo de programação manual;
- **Monitorização de anomalias**: interromper imediatamente a operação em caso de movimentos inesperados, ruídos anormais ou falhas em sensores.

2.4. Programação segura

Na fase de desenvolvimento e teste de programas, devem ser seguidas as boas práticas abaixo:

- **Simulação prévia**: sempre que possível, validar trajetórias em simulação ou em modo de velocidade reduzida;
- **Limites de velocidade e aceleração**: aplicar parâmetros reduzidos em cenários de teste e configurar limites de força/torque adequados;
- **Prevenção de colisões**: Nunca programar movimentos bruscos e rápidos que possam comprometer a segurança dos operados, provocar colisões com o meio envolvente ou danificar o braço robótico;
- **Modo *Freedrive* seguro**: Utilizar o modo *Freedrive* só quando pretende programar manualmente os pontos.

2.5. Procedimentos após a operação

No final de cada sessão de trabalho, devem ser cumpridos os seguintes passos:

- **Encerramento controlado**: terminar os programas em execução antes de desligar o sistema de controlo;
- **Organização do espaço de trabalho**: Manter o espaço circundante ao robô livre, sem objetos que possam interferir no movimento do braço robótico;
- **Registo de anomalias**: documentar e comunicar qualquer irregularidade ao responsável técnico, abstendo-se de reutilizar o robô até que o problema seja resolvido.

2.6. Formação e responsabilidade

- **Formação obrigatória**: apenas operadores devidamente formados e autorizados devem programar ou operar o UR3e;
- **Cumprimento normativo**: todas as operações devem respeitar as normas ISO aplicáveis, com particular destaque para a **ISO/TS 15066**, que define critérios de contacto físico seguro em robôs colaborativos;
- **Supervisão em contexto educativo**: em ambiente académico, é obrigatória a presença de um docente ou técnico qualificado durante os ensaios.

Estas medidas são fundamentais para prevenir acidentes e garantir um ambiente de trabalho colaborativo, seguro e eficiente.

3. Especificações

3.1. Especificações técnicas

A bancada apresenta um conjunto de especificações técnicas que asseguram a segurança, resistência, funcionalidade e adaptabilidade ao contexto de ensino e aprendizagem em robótica colaborativa. Na Tabela 1 expõe-se as especificações técnicas da estrutura da bancada e do robô colaborativo UR3e.

Tabela 1 - Especificações técnicas do sistema

Estrutura da bancada	• **Dimensões gerais da estrutura:** 610 x 605 x 765 mm^3; • **Peso:** aproximadamente 23 kg; • **Tipo de perfil estrutural:** perfil técnico de alumínio 45 x 45 mm^2; • **Superfície de trabalho:** madeira e alumínio; • **Capacidade de carga:** adequada para suportar o robô UR3e e periféricos (aproximadamente 15-20 kg); • **Mobilidade:** quatro rodízios, dois com travão para transporte e fixação segura.
Robô colaborativo UR3e	• **Carga útil:** 3 kg; • **Alcance:** 500 mm; • **Precisão:** ± 0,1 mm; • **Número de juntas (eixos):** 6; • **Interface de Programação:** PolyScope (interface gráfica) ou URScript.

3.2. Especificações funcionais

O sistema desenvolvido pretende suportar o ensino, mais precisamente a automação e robótica colaborativa. As especificações funcionais do sistema são:

- **Sistema de controlo:** controlador CB3.1 com interface PolyScope;
- **Expansão e elementos de controlo:** Entradas e saídas digitais e analógicas (5 botões, 5 lâmpadas/*Leds*, 1 potenciómetro e 1 voltímetro) para integração com sensores e atuadores externos;
- **Segurança:** cablagem organizada, painel de controlo acessível e estrutura estável;
- **Aplicações:** ensino e investigação em automação e robótica colaborativa.

Estas especificações garantem uma solução equilibrada no que respeita funcionalidade, segurança e ensino, o que promove um ambiente de aprendizagem prático e eficiente.

4. Arquitetura do sistema

O sistema didático é constituído pela estrutura móvel, o robô colaborativo UR3e [4], o controlador do robô [5], o Teach Pendant [6] com integração do software PolyScope 5 [7], conjunto com entradas e saídas digitais e analógicas (I/O Digital e Analog) e gripper HCR-03-118505 Zimmer [8]. A Figura 2 apresenta o esquema da bancada projetada evidenciando e representando todos os elementos que a constituem.

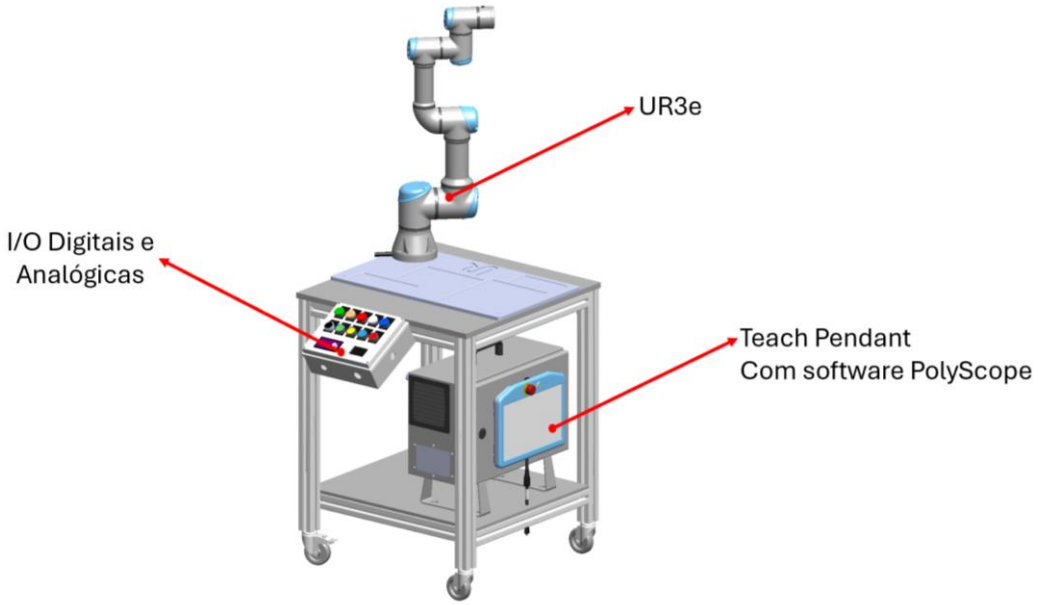


Figura 2 - Sistema UR3e + *Teach Pendant* + I/O

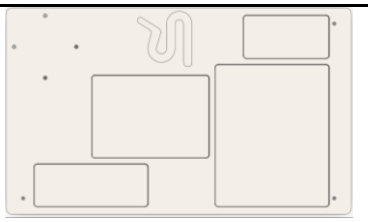
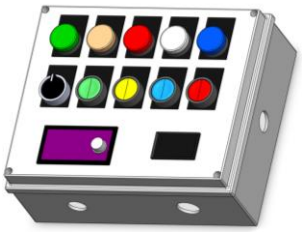
4.1. Equipamentos

Os equipamentos foram organizados de modo a otimizar o espaço disponível e garantir a segurança e o conforto dos utilizadores. O robô UR3e foi instalado na parte central da superfície superior, sobre a base de alumínio (área de trabalho do robô), garantindo uma área de utilização livre de obstáculos. Na zona frontal da estrutura encontra-se o painel de I/O digitais e analógicas, inclinado ergonomicamente para facilitar a operação dos botões e do potenciómetro, bem como a visualização dos *LEDs* e do voltímetro. O controlador do robô e o Teach Pendant (equipado com o software PolyScope) localizam-se na prateleira inferior da bancada, com fácil acesso para manutenção, configuração e manuseamento

Acoplado ao UR3e encontra-se o *gripper* HCR-03-118505 Zimmer. Este *gripper* é de atuação elétrica e é o componente responsável pela manipulação de objetos.

A Tabela 2 apresenta uma descrição dos equipamentos integrados na bancada, incluindo o robô colaborativo UR3e, os seus periféricos e a caixa de I/Os digitais e analógicas.

Tabela 2 – Cobot UR3e, periféricos e modos de fixação

Designação	Descrição	Representação
Robô colaborativo UR3e	Robô colaborativo fácil de utilizar e programar, rápido de implementar e seguro para trabalhar lado a lado com o operador.	 [9]
Base de trabalho (em alumínio) 610 x 365 x 10 mm³	Base de trabalho do robô colaborativo, situado entre o robô e o tampo da mesa. Contribui para a fixação do robô. Este componente é caracterizado por apresentar diferentes zonas de trabalho do robô colaborativo.	
2-JAW PARALLEL GRIPPERS HRC-03-118505	O *gripper* permite agarrar peças com precisão e firmeza, garantindo um posicionamento estável durante as operações. A sua conceção compacta e robusta assegura versatilidade e fiabilidade na execução de tarefas de montagem, movimentação ou inspeção.	 [8]
Controlador CB3.1	O controlador é um componente essencial do sistema global. É o centro nevrálgico para gerir os movimentos do robô, as interações com dispositivos externos e a integração de sistemas de automação mais amplos.	
3PE Teach Pendant e-Series	Interface de comunicação equipada com o *software* PolyScope. A partir do PolyScope é possível desenvolver a programação e execução dos exercícios/programas pretendidos. Integra-se na estrutura através do encaixe nos dois suportes circulares presentes na parte frontal do controlador.	 [6]
Caixa de I/O digitais e analógicas	Caixa que integra todos os componentes de entradas e saídas digitais e analógicas.	

4.1.1. Caixa de I/O digitais e analógicas

A integração de I/O digitais e analógicas confere à bancada um elevado grau de versatilidade e expansão. Esta capacidade permite a conexão com sensores, atuadores e outros dispositivos externos, o que torna possível a simulação de sistemas industriais mais complexos. Assim, os utilizadores podem aprofundar os seus conhecimentos não só em robótica colaborativa, como também explorar conceitos associados à automação. Esta abordagem multidisciplinar contribui para o desenvolvimento de competências técnicas em diferentes áreas. Além disso, estes elementos permitem uma interação intuitiva com o sistema e facilitam o controlo de parâmetros como a velocidade e estado da operação. A disposição e organização dos componentes na caixa (Figura 3) pretende proporcionar uma utilização simples e eficiente para o utilizador.

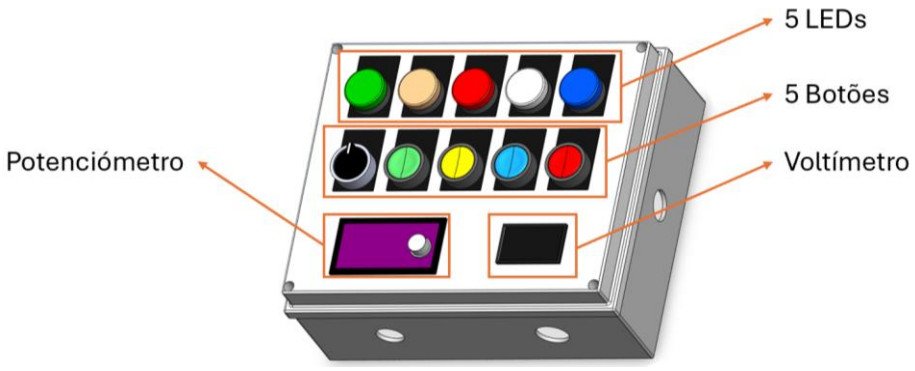


Figura 3 - Caixa de I/O com representação da disposição dos componentes

A Tabela 3 apresenta os elementos que compõem a caixa de I/O digitais e analógicas, incluindo botões de pressão e sinalizadores luminosos (LEDs). Na coluna de representação encontra-se apenas um exemplo de botão e um exemplo de LED, uma vez que os restantes modelos diferem apenas na cor da superfície visível do atuador ou do difusor luminoso. Todos os botões e *Leds* foram adquiridos ao fornecedor Soflight, pelo que se apresenta a designação utilizada pelo mesmo.

Tabela 3 – Elementos inseridos na caixa de I/O digitais e analógicas

Designação	Descrição	Representação
SL-CPB01-AA31	Botão de pressão verde	
SL-CPB01-AA42	Botão de pressão vermelho	
SL-CPB01-AA51	Botão de pressão amarelo	
SL-CPB01-AA61	Botão de pressão azul	
SL-CPB01-AD21	Seletor de 2 posições fixas	

Tabela 3 – Elementos inseridos na caixa de I/O digitais e analógicas (continuação)

SL-CPB01-22DS-GR-024	Sinalizador compacto Ø22mm *Led* cor verde 24V	
SL-CPB01-22DS-RD-024	Sinalizador compacto Ø22mm *Led* cor vermelho 24V	
SL-CPB01-22DS-YE-024	Sinalizador compacto Ø22mm *Led* cor amarelo 24V	
SL-CPB01-22DS-BL-024	Sinalizador compacto Ø22mm *Led* cor azul 24V	
SL-CPB01-22DS-WH-024	Sinalizador compacto Ø22mm *Led* cor branco 24V	
JOY-IT VM433	Voltímetro de painel digital 0..33VDC (4 dígitos) - vermelho	
0-10V Signal Generator DM-HOD-1AO-010V	Potenciómetro analógico com geração de sinal 0-10V	

4.2. Softwares

O sistema utiliza o *software* PolyScope, incorporado no Teach Pendant do robô UR3e, como interface principal para a programação e controlo do cobot. Este ambiente gráfico permite a criação de sequências de movimento e lógica sem necessidade de programação avançada, sendo ideal para contextos didáticos.

5. Conexão ao controlador

A integração dos componentes com o controlador foi realizada de forma a assegurar a compatibilidade elétrica e lógica entre os dispositivos. Para melhor compreensão, a descrição da integração dos componentes é apresentada em dois subconjuntos distintos:

- Robô colaborativo UR3e, controlador e Teach Pendant;
- Caixa de I/Os digitais e analógicas e respetiva ligação ao controlador.

5.1. Conexão entre o cobot e periféricos

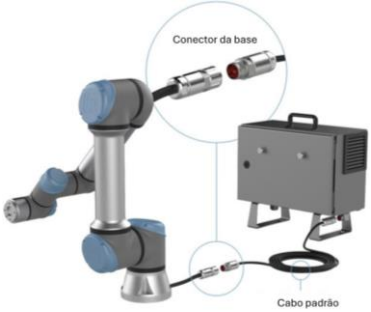
O robô colaborativo UR3e está ligado diretamente ao controlador Universal Robots, que integra a fonte de alimentação, os terminais de comunicação e os módulos de I/O (Figura 4). A comunicação entre o robô e o controlador é assegurada pelo cabo padrão 165100.

Figura 4 - Controlador com representação dos terminais [10]

O Teach Pendant, responsável pela interface homem-máquina, liga-se ao controlador através de um conector. Este elemento permite o desenvolvimento, simulação e execução de programas no ambiente PolyScope.

A Tabela 4 apresenta a designação, descrição e representação gráfica dos elementos, incluindo ainda uma explicação da sua respetiva função no sistema.

Tabela 4 - Designação, descrição e representação dos elementos de ligação do subconjunto 1

Designação	Descrição	Representação
Cabo padrão 165100	Cabo que realiza a ligação do robô colaborativo UR3e ao controlador.	 [11]  [12]
3PE Teach Pendant e-Series	A ligação do Teach Pendant é feita ao controlador através de um cabo padrão de 4,5 m de comprimento que está acoplado no componente.	 [6]

5.2. Conexão da caixa de I/O digitais e analógicas

O controlador disponibiliza entradas e saídas digitais (DI/DO), bem como entradas e saídas analógicas (AI/AO), que permitem a ligação de sensores e atuadores externos. Foi através destes terminais que se estabeleceu a comunicação com a caixa de I/O digitais e analógicas. A Figura 5 apresenta a disposição e categorização dos blocos de entradas e saídas que o controlador do robô UR3e dispõe.

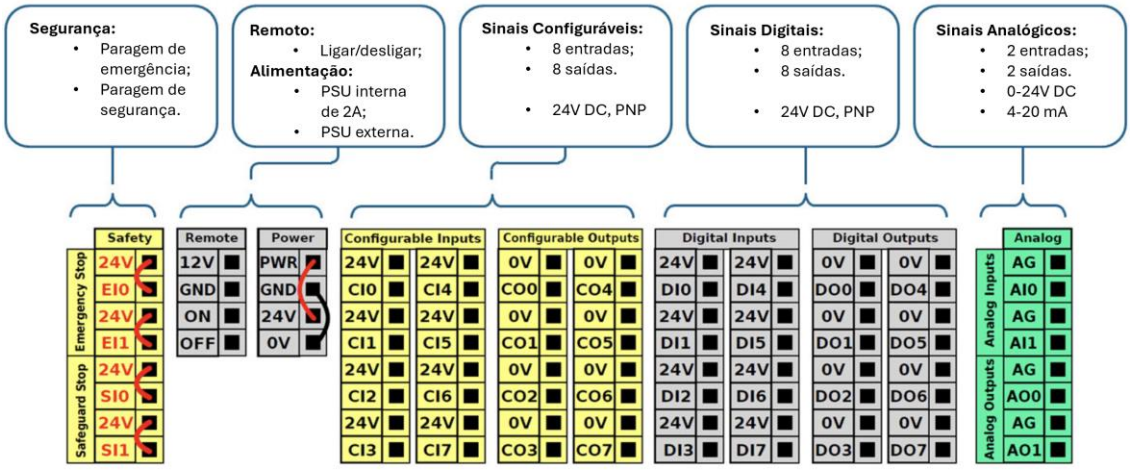


Figura 5 - Representação dos blocos do controlador do UR3e

Após a ligação dos componentes, foi estabelecida a correspondência entre cada dispositivo e a respetiva entrada ou saída digital no controlador, conforme apresentado nas Tabelas 5-8. Estas tabelas organizam de forma clara quais os dispositivos conectados às entradas digitais (botões) e analógicas (potenciómetro) e às saídas digitais (LEDs) e analógicas (voltímetro), permitindo identificar rapidamente a função de cada canal.

Tabela 5 - Correspondência entre as entradas digitais e respetiva ligação no controlador

Entradas digitais	
Botão de 2 posições	DI0
Botão Verde	DI1
Botão Amarelo	DI2
Botão Azul	DI3
Botão Vermelho	DI4

Tabela 6 - Correspondência entre as saídas digitais e respetiva ligação no controlador

Saídas digitais	
LED Verde	DO0
LED Amarelo	DO1
LED Vermelho	DO2
LED Branco	DO3
LED Azul	DO4

Tabela 7 - Correspondência entre as entradas analógicas e respetiva ligação no controlador

Entradas analógicas	
Potenciómetro	DAI1

Tabela 8 - Correspondência entre as saídas analógicas e respetiva ligação no controlador

Saídas analógicas	
Voltímetro	DAO0

6. Protocolos de comunicação

O sistema desenvolvido está preparado para operar com diferentes protocolos de comunicação industrial, o que permite a ligação do robô colaborativo UR3e a controladores externos, como PLCs, módulos de expansão e dispositivos periféricos habitualmente utilizados em ambientes de automação. Esta abordagem assegura a interoperabilidade entre os diferentes elementos do sistema e permite a construção de células robóticas didáticas com elevado grau de complexidade.

O sistema desenvolvido está preparado para operar com os protocolos MODBUS TCP e EtherNet/IP, dispositivos IIoT ou plataformas SCADA (referentes a sistemas de supervisão, controlo e aquisição de dados que monitorizam e gerem processos industriais).

- **MODBUS TCP/IP**

O protocolo MODBUS TCP/IP está disponível no controlador do cobot UR3e e permite a integração direta com controladores lógicos programáveis (PLCs) e outros equipamentos. Trata-se de um protocolo aberto, amplamente utilizado na indústria, que possibilita a troca de dados através de uma rede Ethernet de forma estruturada e fiável.

- **EtherNet/IP**

O protocolo EtherNet/IP é um protocolo determinístico (os dados são enviados em intervalos de tempo previsíveis e consistentes) e orientado a objetos, ideal para comunicação em tempo real com módulos de I/O, sensores ou controladores de segurança.

Módulos e Dispositivos Compatíveis

A utilização destes protocolos assegura uma comunicação fiável, o que possibilita a ligação a uma vasta gama de dispositivos industriais. Entre os principais módulos e equipamentos que podem ser integrados no sistema destacam-se:

- **PLCs** (Controladores Lógicos Programáveis) – Lógica de controlo, comunicação com sensores/atuadores, e sincronização de tarefas com o robô;
- **HMIs** (Interfaces Homem-Máquina) – Visualização em tempo real do estado do sistema e interface de operação;
- **Variadores de Velocidade** – Controlo de motores;
- **Servomotores e Servo Drives** – Operações de posicionamento com elevada precisão;
- **Cilindros pneumáticos e válvulas eletropneumáticas** – Atuação mecânica de objetos, acionados por sinais digitais provenientes do PLC ou do robô.
- **Módulos de I/O digitais e analógicos** – Expansões para sensores e atuadores diversos;
- **Sensores industriais** (óticos, proximidade, pressão, temperatura) – *Feedback* sobre condições do processo;

- **Relés, contactores e botoneiras** – Elementos auxiliares de comando e proteção;
- **Módulos de segurança** – Dispositivos que asseguram o cumprimento das normas de segurança funcionais (ISO 13849, IEC 62061);
- ***Gateways* industriais** (ex.: eWON Flexy 205) – Acesso remoto seguro ao sistema, com funcionalidades de recolha de dados, monitorização remota e diagnóstico em tempo real.

Alguns dos módulos e dispositivos anteriormente descritos não estabelecem comunicação direta com o controlador do robô colaborativo UR3e. Nestes casos, é necessária a presença de um dispositivo intermédio, geralmente um PLC, que atua como elemento de controlo e interface. É o que acontece, por exemplo, com variadores de velocidade, cilindros pneumáticos e *servo drivers*, que são comandados diretamente pelo PLC, que por sua vez comunica com o robô através dos protocolos industriais MODBUS TCP ou EtherNet/IP.

Acesso remoto

Um dos principais benefícios da adoção de protocolos industriais abertos e de arquitetura modular é a capacidade de aceder remotamente ao sistema. Através da utilização de *gateways* como o eWON Flexy 205, torna-se possível a ligação à *cloud*, o acesso remoto seguro ao controlador do robô, a recolha contínua de dados e a execução de comandos à distância. Estas funcionalidades permitem trabalhar remotamente no robô e, desta forma, alinhar o sistema com os princípios da Indústria 4.0, tornando-o altamente relevante para contextos didáticos e industriais onde a conectividade e flexibilidade são fatores diferenciadores.

Deste modo, a capacidade de comunicação com múltiplos dispositivos torna a bancada um sistema altamente didático, o que permite simular arquiteturas industriais completas com vários níveis de complexidade.

7. Interface do PolyScope

O PolyScope é a interface de comunicação utilizada pela Universal Robotics (UR) e serve para programar, monitorizar e controlar o funcionamento do braço robótico.

7.1. Ambiente inicial da interface

A compreensão do ambiente inicial do PolyScope é fundamental para a utilização eficiente do robô, uma vez que esta interface constitui o ponto de partida para as operações de configuração, programação e controlo. Na Figura 6 apresenta-se a página inicial do *software*, com destaque para as diferentes áreas e funcionalidades disponíveis.

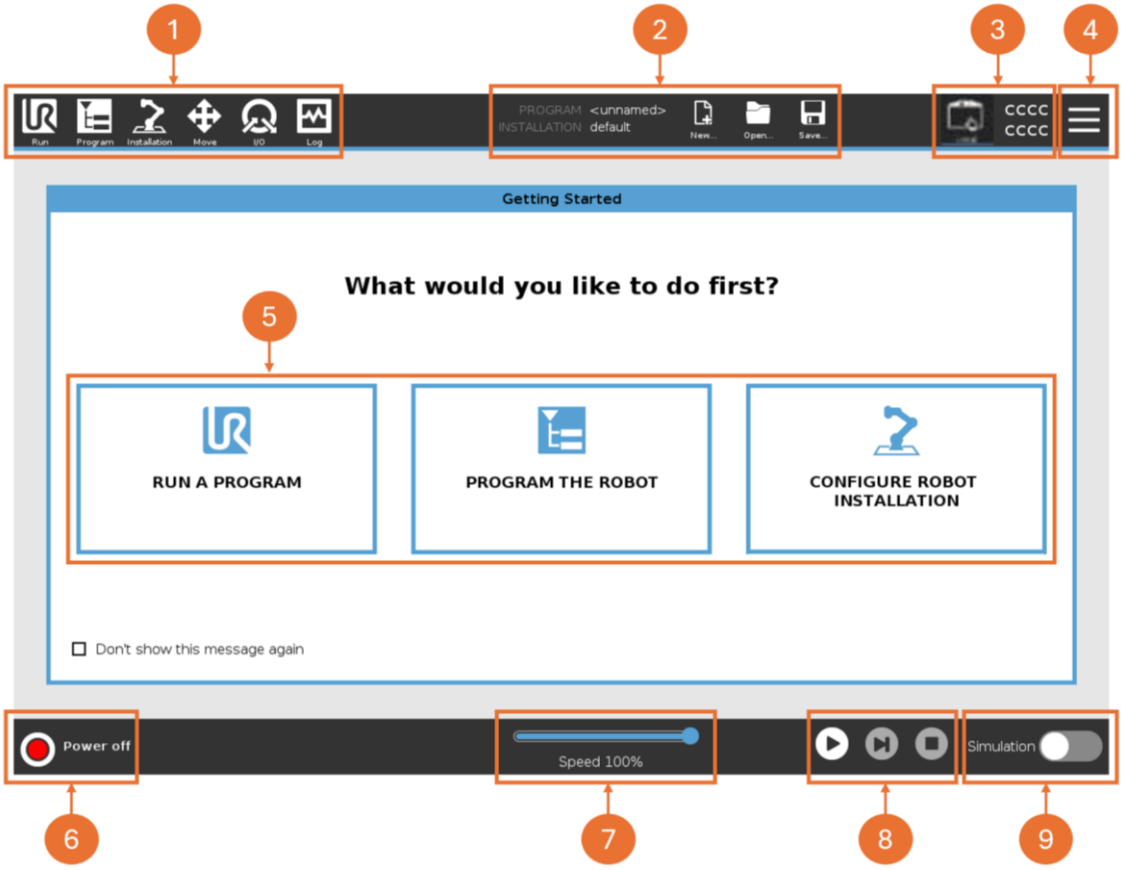


Figura 6 - Ambiente inicial da interface

As zonas representadas na figura anterior apresentam as seguintes funcionalidades:

1. Conjunto de ícones relacionados com a configuração, programação, movimento e configuração do robô;
2. Conjunto de ícones de gestão de arquivos. Permitem criar, guardar e gerir programas;
3. Ícones relacionados com o controlo local ou remoto do controlador, verificação de segurança, modos de operação do sistema (manual/automático) e ferramentas URCaps;

4. Ícone do menu. Dá acesso a várias configurações do sistema operacional como definição de senha, atribuição de endereço IP, escolha do idioma da interface, entre outros;
5. Conjunto de ícones de acessibilidade rápida para executar um programa já existente, criar um novo programa e configurar parâmetros de instalação;
6. Botão de ligar e desligar. Elemento que permite ativar ou desativar o robô, com indicação do estado atual do sistema;
7. Controlador de velocidade. Permite ajustar a velocidade de movimento do robô;
8. Ícones de controlo dos programas. Permitem iniciar, pausar e parar um programa;
9. Botão deslizante que ativa a simulação no sistema operacional e impede os movimentos reais do robô.

7.2. Ativação do robô

O braço robótico, mesmo após ativação do seu controlador, permanece no estado desativo. Para que o robô possa ser configurado, movido e programado, é necessário ativá-lo. Para tal, deve ser acionado o botão que se encontra no canto inferior esquerdo da tela inicial do Polyscope (assinalado com o número 6 na Figura 6). De seguida, é necessário seguir uma sequência de três passos até que o botão associado ao estado do robô passe para o estado **Normal**. Segue-se a referida sequência:

1. Selecionar o botão de ativação que demonstra o estado do robô ***Power off***;
2. Selecionar o botão ***ON***, assinalado com o número 2 na Figura 7;
3. Selecionar o botão ***START***, assinalado com o número 3 na Figura 7;

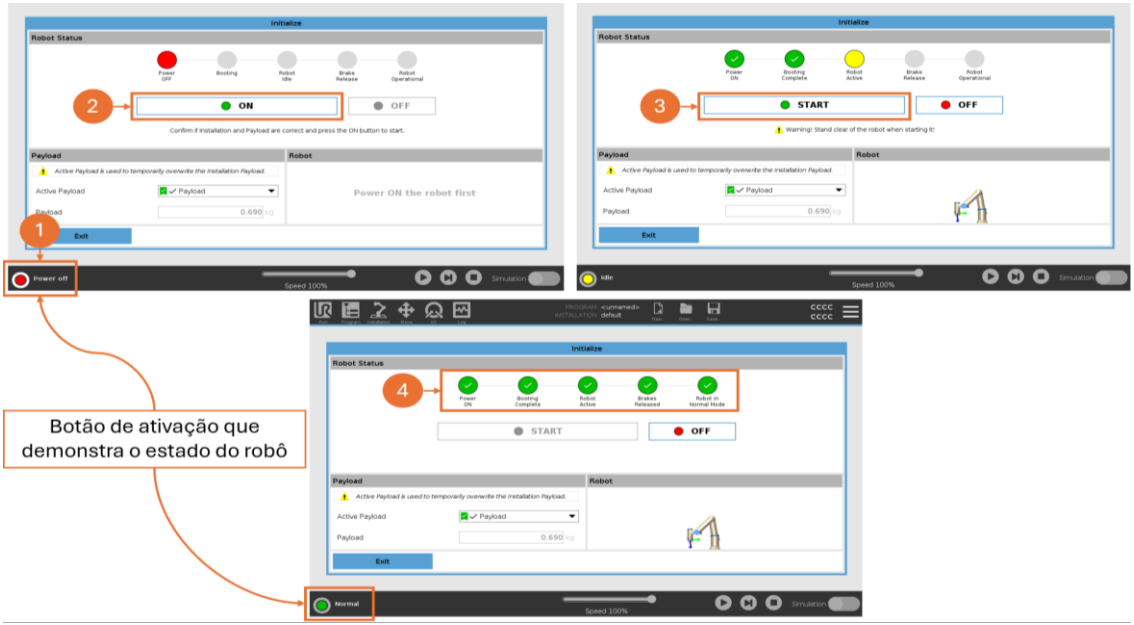


Figura 7 - Ativação do robô

Após esta sequência, o botão que demonstra o estado do robô passa a **Normal** e os cinco estados do robô devem aparecer a verde (ver secção assinalada com o número 4 na Figura 7).

7.3. Instalação básica

Numa frase prévia à programação do cobot, é necessário configurar os parâmetros essenciais do *Tool Center Point* (TCP), correspondente ao ponto central entre garras do *gripper* (Figura 8). É ainda necessário configurar a carga útil em quilogramas (kg).

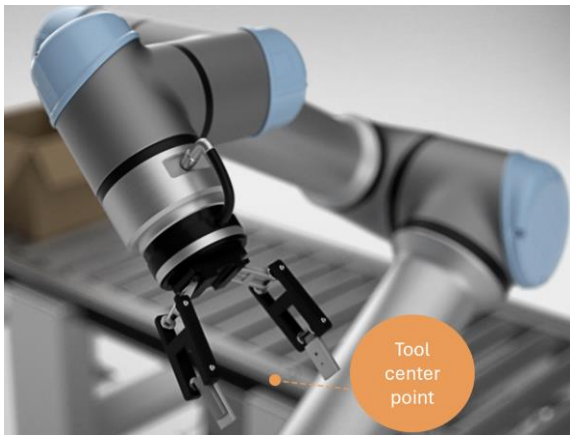


Figura 8 - Localização do TCP, adaptado de [13]

De modo a facilitar a compreensão da configuração do TCP e da carga útil, expõem-se um caso prático da parametrização resolvido o passo a passo. De referir que cada ferramenta tem o seu TCP, pelo que é necessário considerar diferentes aspetos.

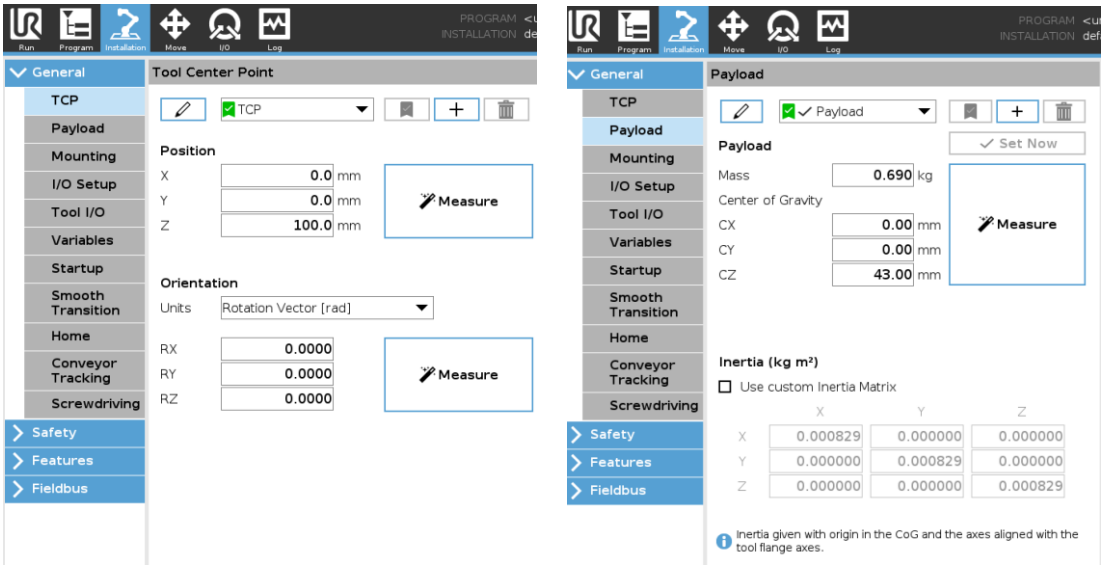
Caso prático resolvido

Realizar a configuração de uma ferramenta com os seguintes dados:

- TCP: X=0 mm; Y=0 mm; Z=100 mm.
- Carga: 0,690 kg; centro de gravidade CX=0, CY=0, CZ=43 mm.

Resolução:

Inicialmente, para garantir uma correta parametrização do robô, é necessário atribuir um nome à configuração desejada, preencher os parâmetros relativos à posição (no TCP) e à *payload* (para a carga útil). De seguida, é necessário proceder à ativação dos referidos parâmetros através da opção “Set Now”, presente na interface do Teach Pendant. A presença de um ícone verde junto ao nome da configuração indica que a definição do TCP e da carga útil foi aplicada com sucesso, como se verifica na Figura 9.



(a) (b)

Figura 9 - Parametrização do robô (a) Definição do TCP, (b) Definição da carga útil

Após a configuração, o Teach Pendant apresenta uma visualização tridimensional (3D) do robô, refletindo os dados introduzidos. Após analise da Figura 10 verifica-se que o sistema de coordenadas TCP é representado graficamente, com o eixo Z orientado segundo o eixo da flange do robô, o que facilita a interpretação espacial da carga. A carga útil (*payload*) é ilustrada por um cubo 3D, cujo volume é proporcional ao valor atribuído durante a configuração. O centro de gravidade da carga é evidenciado pela posição relativa do cubo face à flange do robô, permitindo uma análise visual da distribuição de massa.

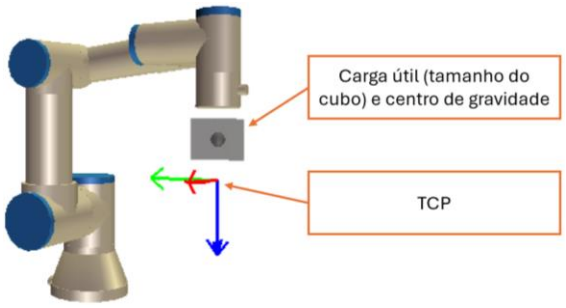


Figura 10 - Representação dos dados configurados

7.4. Movimento do robô

O movimento do robô constitui um elemento central na programação e operação do UR3e, sendo a base para a execução de qualquer tarefa. A compreensão dos diferentes modos de movimentação e das ferramentas de controlo disponíveis no PolyScope é essencial para garantir precisão, repetibilidade e segurança nas operações. Nesta secção são descritos os tipos de movimentos disponíveis na aba superior *Move* da interface.

7.4.1. Movimento em modo Freedrive

O movimento em modo *Freedrive* só pode ser realizado fisicamente (modo manual). Na Figura 11 ilustra-se o processo de utilização deste modo. O processo inicia com a seleção do botão ***Freedrive*** (1) localizado no painel inferior da interface. Após esta seleção, é apresentado no lado direito do ecrã um menu (2) no qual o botão ***Freedrive - Press & Hold*** (3) deve ser mantido pressionado enquanto se movem manualmente as juntas do robô. Para facilitar o seu posicionamento, é possível limitar o movimento a determinados eixos cartesianos ou à rotação.

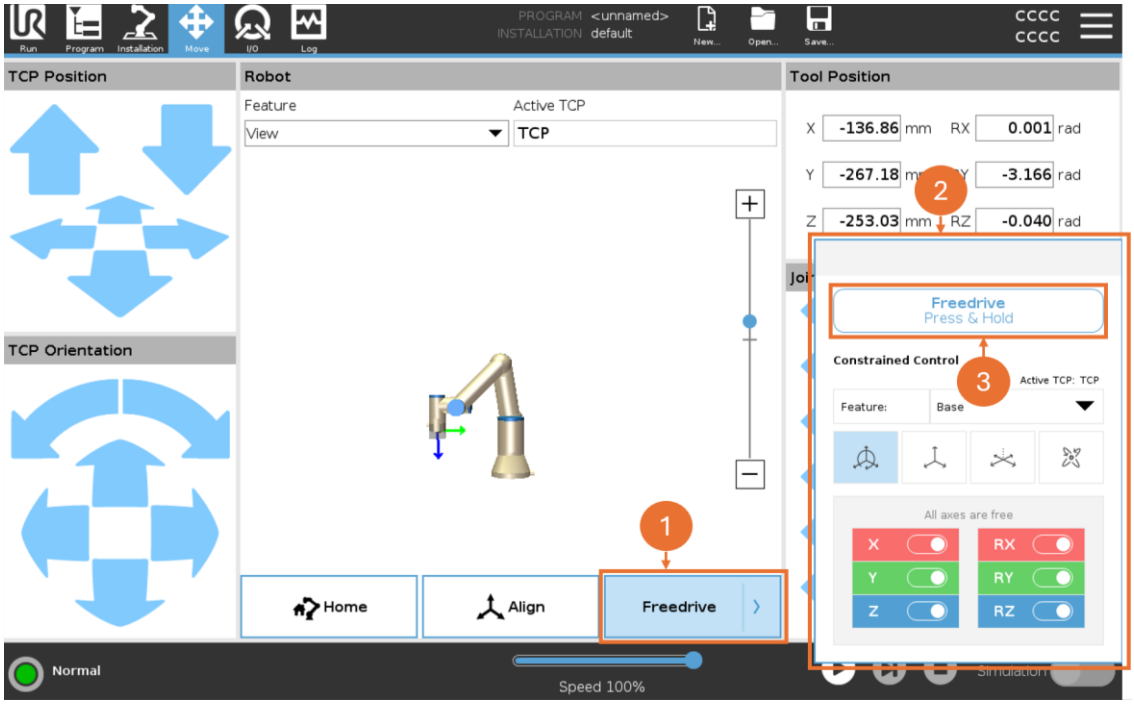


Figura 11 - Movimento do robô em modo Freedrive

7.4.2. Movimento a partir do Teach Pendant

Através do Teach Pendant o movimento pode ser feito de duas maneiras:

1. Janela de movimento

A janela de movimento permite ao utilizador realizar os diferentes tipos de movimentação do robô, nomeadamente movimentos das juntas, lineares e de reorientação. Como ilustrado na Figura 12, esta janela inclui um conjunto de botões gráficos que facilitam a execução de movimentos lineares do TCP ao longo dos eixos cartesianos, bem como movimentos de reorientação baseados na rotação em torno desses mesmos eixos. Adicionalmente, é disponibilizada uma área dedicada ao controlo direto das juntas do robô, onde é possível ajustar individualmente a posição de cada articulação. Esta interface permite ainda a seleção do sistema de coordenadas de referência (por exemplo, base ou ferramenta), em função do tipo de movimento realizado, o que garante maior precisão e controlo nas operações de manipulação.

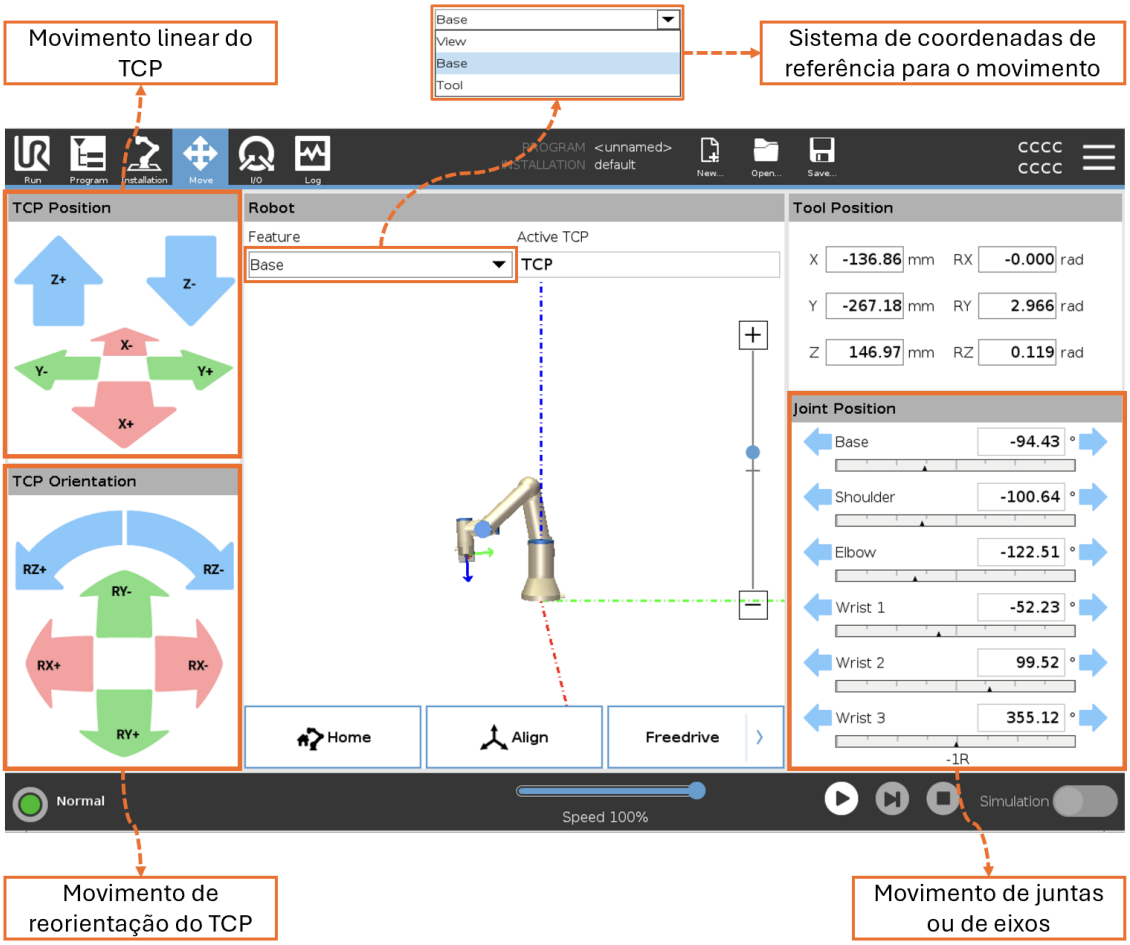


Figura 12 - Janela de movimento

2. Janela de ajuste numérico de posição

Na Figura 13 é possível observar a **janela de ajuste numérico de posição**, que permite a inserção precisa dos dados de posição do robô. Esta funcionalidade encontra-se disponível ao selecionar um dos campos numéricos correspondentes ao TCP. Após aceder à janela e pressionar sobre um dos campos numéricos de posição, é exibido um teclado flutuante, que possibilita a introdução direta dos valores pretendidos. Adicionalmente, encontram-se disponíveis botões identificados com os símbolos “+” e “–” junto de cada coordenada, permitindo o aumento ou diminuição incremental dos valores apresentados. A partir desta janela, é também possível escolher o sistema de coordenadas onde os dados são modificados.

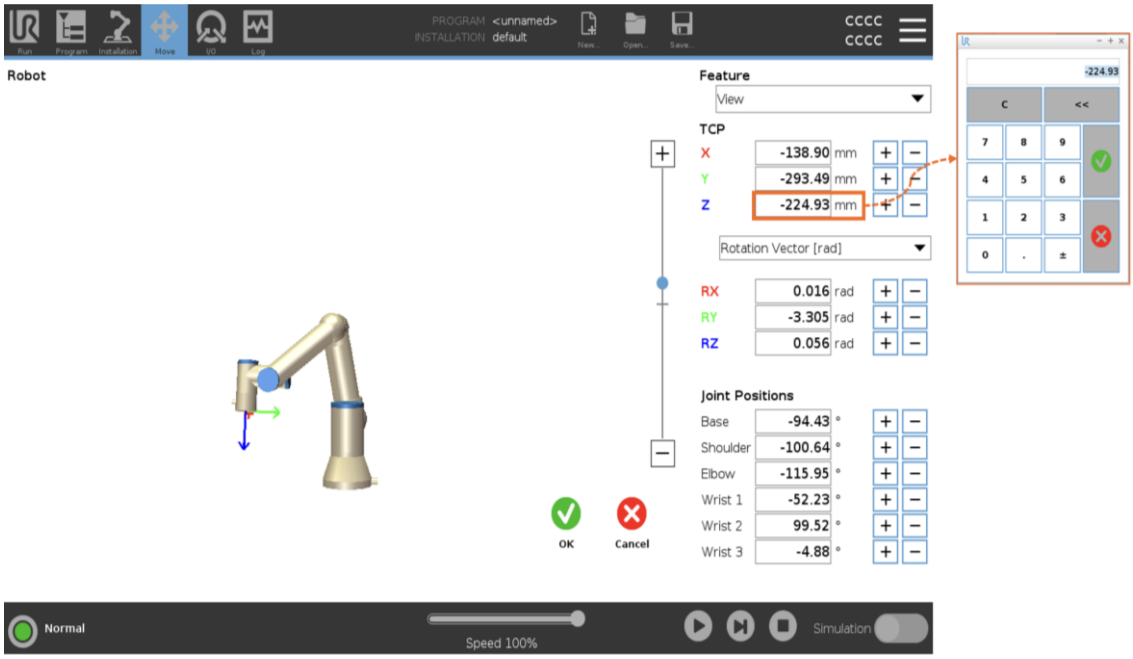


Figura 13 - Janela de ajuste numérico de posição

8. Ambiente de programação

O ambiente de programação do PolyScope apresenta uma interface gráfica intuitiva e funcional. Este ambiente dispõe de diferentes áreas estruturadas que facilitam a criação e edição de programas, destinados ao controlo do cobot UR3e.

A Figura 14 ilustra a página inicial do ambiente de programação, organizada em várias secções numeradas, cada uma com uma função específica:

1. **Menu de comandos de instruções**: Situado no lado esquerdo da interface. Esta área contém os blocos de programação disponíveis, organizados por categorias, tais como *Basic*, *Advanced*, *Templates* e URCaps.
2. **Árvore do programa**: Esta secção corresponde à zona de elaboração do programa. Nesta são introduzidas todas as secções, instruções e subprogramas que constituem o programa.
3. **Barra de ferramentas do programa**: Localizada na parte inferior da árvore do programa, esta barra disponibiliza funcionalidades como mover, eliminar, cortar, copiar e colar linhas de código do programa. Estas ferramentas permitem gerir eficientemente a estrutura do programa.
4. **Área de comando, parametrização e informação**: Situada no lado direito da árvore do programa. Esta secção corresponde à zona principal de edição, onde se visualiza e edita as secções/instruções da árvore principal do programa. Esta área inclui separadores como *Command*, *Graphics* e *Variables*, que permitem aceder a diferentes aspetos como comandos específicos, configuração de variáveis e visualização gráfica.

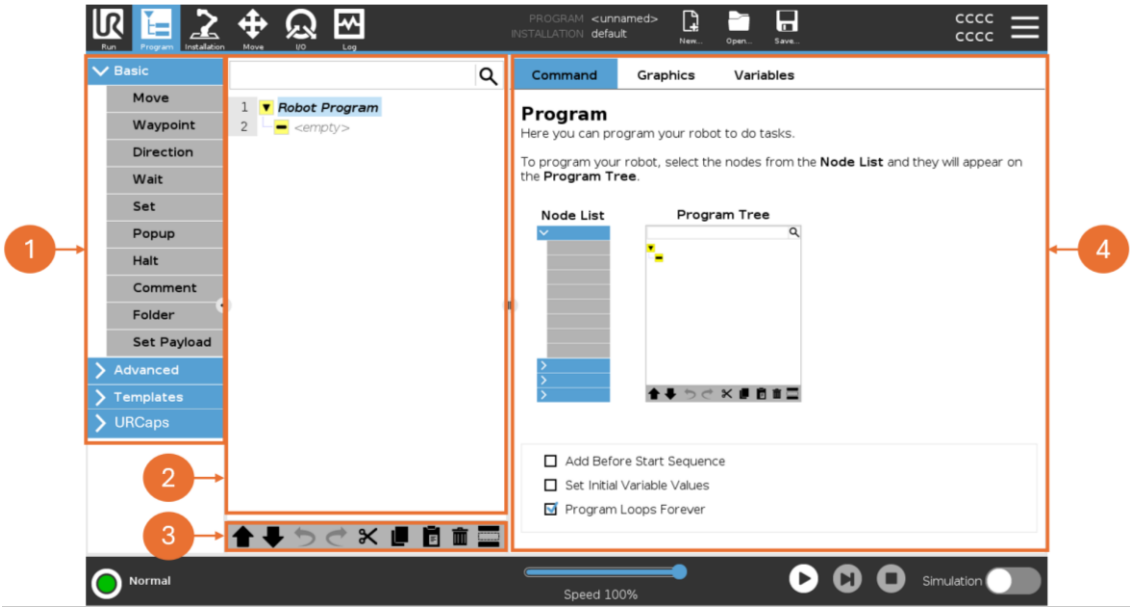


Figura 14 - Página inicial do ambiente de programação

Na Figura 15 apresenta-se de forma detalhada a barra de ferramentas do programa.

Figura 15 - Barra de ferramentas do programa

Através da análise da Figura 15, seguindo a ordem da esquerda para a direita, verificam-se as seguintes funções:

1. Mover linha para cima
2. Mover linha para baixo
3. Desfazer
4. Refazer
5. Cortar
6. Copiar
7. Colar
8. Eliminar
9. Ocultar

8.1. Programação dos robôs UR

A programação dos robôs UR realiza-se a partir do Teach Pendant, através de dois modos diferentes:

- **Interface gráfica PolyScope**: Baseia-se na criação de uma árvore de programação, na qual o ajuste de todos os parâmetros é feito por comandos interativos, semelhantes a formulários. Esta forma de programação destaca-se pela sua intuitividade e acessibilidade, o que simplifica a criação, edição e teste de programas.
- **URScript**: Consiste em escrever código em texto simples baseado em *scripts*. Esta escrita pode ser feita diretamente na consola ou num editor externo. Os *scripts* devem ser chamados a partir da interface gráfica.

8.2. Estrutura do programa principal

O programa principal do robô executa-se linha a linha, com sentido descendente, e de forma cíclica. Para restringir a ciclicidade de execução, o utilizador deve inserir restrições. Na estrutura do programa podem ser inseridas várias secções, tais como:

- **Programa principal** (***Robot Program***): Corresponde ao corpo principal do programa, no qual as instruções são executadas sequencialmente, linha a linha.
- **Antes de começar** (***Before Start***): Secção dedicada à execução de instruções iniciais, antes do início do ciclo principal do programa.

- **Variáveis iniciais** (***Initial Variables Values***): Área destinada à definição dos valores iniciais das variáveis utilizadas no programa.
- **Pastas** (***Folder***): Utilizam-se para armazenar e organizar partes do programa. Não têm efeito lógico no funcionamento da sequência.
- **Subprogramas** (***SubProg***): Porções de código que são executadas quando são invocados a partir do programa principal. Podem ser guardados como arquivos para reutilização noutros programas.
- **Subtarefas** (***Threads***): Programas paralelos que são executados em simultâneo com o programa principal.
- **Eventos** (***Event***): Programa que é executado de forma semelhante a uma instrução, com a diferença de que não interrompe a execução do programa principal.

Na Figura 16 é apresentada a estrutura em árvore das referidas secções no ambiente de programação do robô.

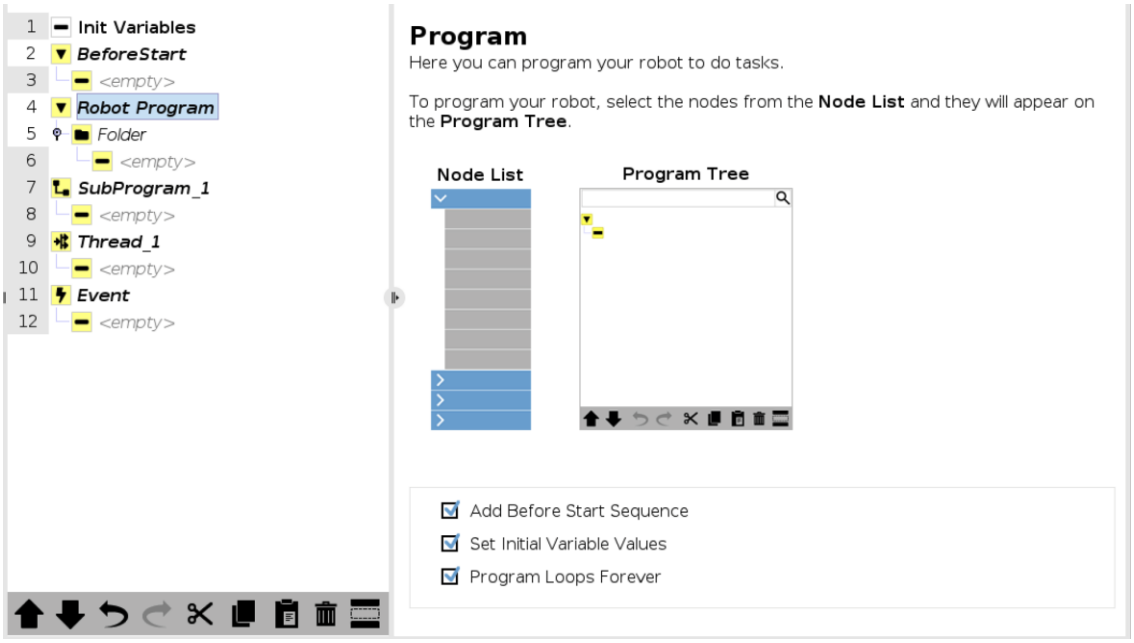


Figura 16 - Estruturação da árvore de um programa

8.3. Instruções de programação

A interface gráfica estrutura as instruções de programação em três grupos base:

- ***Basic***: Contém as instruções básicas de programação. Com elas é possível criar, por exemplo, uma sequência de movimento.
- ***Advanced***: Contém as instruções avançadas de programação. Permite introduzir instruções de controlo de fluxo como condicionais ou ciclos, e blocos lógicos para a estruturação do programa como subprogramas, subtarefas, eventos, entre outros.

- ***Templates***: Corresponde a uma coleção de modelos pré definidos que facilitam a implementação e programação de um determinado tipo de operação, como paletização, aparafusamento, empilhamento e desempilhamento de peças, entre outros.

Na Figura 17 é apresentada a estrutura completa das instruções disponíveis nos três grupos de programação do sistema.

(a) (b) (c)

Figura 17 - Instruções de programação (a) *Basic*, (b) *Advanced* e (c) *Templates*

Na interface de programação dos robôs da Universal Robots, para além dos grupos de instruções base disponibilizados pelo sistema, podem ser adicionadas novas secções através das designadas **URCaps**, que correspondem a extensões de *software* desenvolvidas por fabricantes terceiros. Estas extensões permitem o controlo direto de periféricos instalados no robô como câmaras ou *grippers*. Na Figura 18 encontram-se dois módulos URCap instalados no robô UR3e: **Image Execution**, associado ao funcionamento da câmara, e **Zimmer**, correspondente aos comandos do Gripper. De referir que apesar de ambos os módulos URCpas se encontrarem instalados no robô, apenas se utilizaram os comandos associados ao *gripper*.

Figura 18 - Instruções de programação dos periféricos (câmara e *gripper*) instalados

8.4. Secção Basic

Neste subcapítulo apresentam-se as dez instruções Basic disponíveis no ambiente de programação do UR3e, disponíveis na Figura 19.

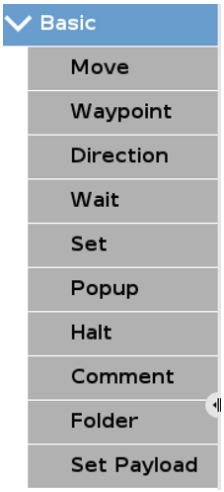


Figura 19 - Secção Basic

8.4.1. Instrução *Move*

A instruções ***Move*** corresponde a uma instrução de movimento e está disponível na secção *Basic*. Esta instrução permite a definição de trajetórias que o manipulador deve seguir para mover o ponto TCP entre diferentes posições. Esta função comporta três tipos distintos de movimento:

- ***MoveJ* (Movimento em arco):** O *MoveJ* executa a trajetória do TCP através da interpolação entre as posições articulares (juntas) do robô, o que culmina num movimento em arco. Este tipo de movimento é caracterizado por ser mais rápido e energeticamente eficiente, mas menos preciso em relação à trajetória cartesiana percorrida pelo TCP.

  Indicação: *MoveJ* é utilizado sempre que não seja necessário seguir uma trajetória linear precisa, como em deslocações entre tarefas ou aproximações iniciais.

- ***MoveL* (Movimento Linear):** O *MoveL* garante o deslocamento do TCP ao longo de uma linha reta no espaço cartesiano entre dois pontos. Este tipo de movimento é fundamental para tarefas que exigem precisão na trajetória como soldadura, colagem ou inserção de componentes.

  Indicação: *MoveL* é utilizado em operações que exigem contacto contínuo ou movimento controlado em linha reta.

- ***MoveP* (Movimento Planeado):** O *MoveP* permite a definição de sequências de *Waypoints* que o robô percorre de forma suave e contínua, minimizando paragens entre os pontos intermédios. Através da mistura de trajetórias, garante-se uma transição fluída e ideal para aplicações que requerem velocidade e fluidez como pintura ou polimento.

  Indicação: *MoveP* é utilizado quando é necessário um movimento fluido entre múltiplos pontos sem paragens intermédias.

Os parâmetros ajustáveis associados às instruções de movimento encontram-se disponíveis no comando da instrução, localizado na zona direita do ambiente de programação. Nesta área é

possível definir o tipo de movimento a executar (MoveJ, MoveL ou MoveP), a velocidade e aceleração das juntas, o TCP a utilizar e o sistema de coordenadas de referência. Na Figura 20 apresenta-se a janela de configuração destes parâmetros.

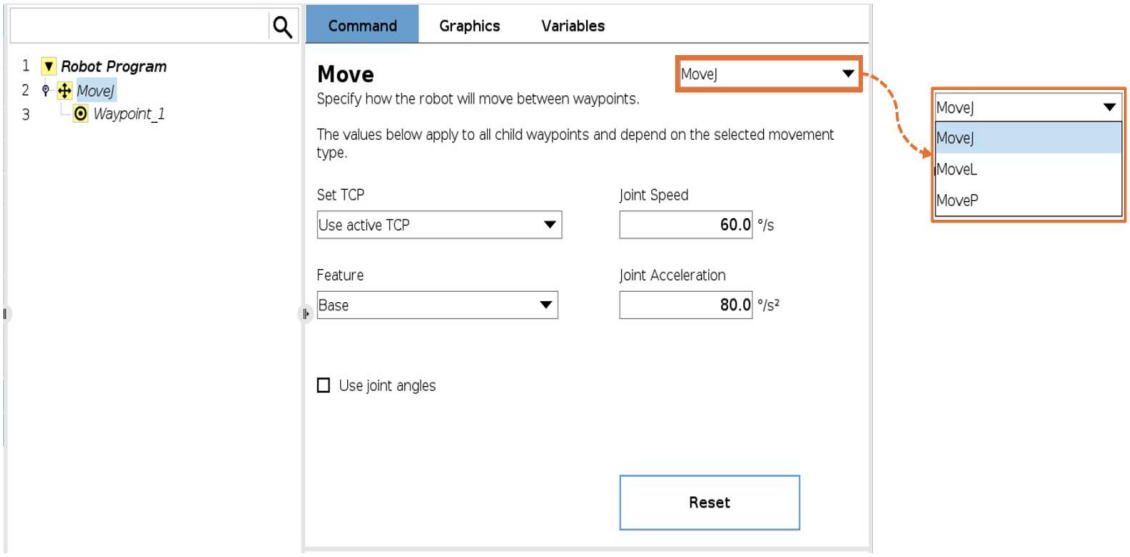


Figura 20 - Janela de ajuste dos parâmetros das instruções de movimento

8.4.2. Instrução Waypoint

A instrução *Waypoint* define uma posição no espaço que o robô deve atingir, podendo incluir orientação, velocidade e aceleração. A forma como essa instrução é usada varia ligeiramente conforme o modo de programação selecionado. De referir que uma instrução deste tipo vem associada a uma instrução *Move*.

A instrução permite definir posições que o robô deve atingir durante a execução do programa. A Figura 21 mostra três formas distintas de configurar essa posição: ***Fixed***, ***Relative*** e ***Variable***.

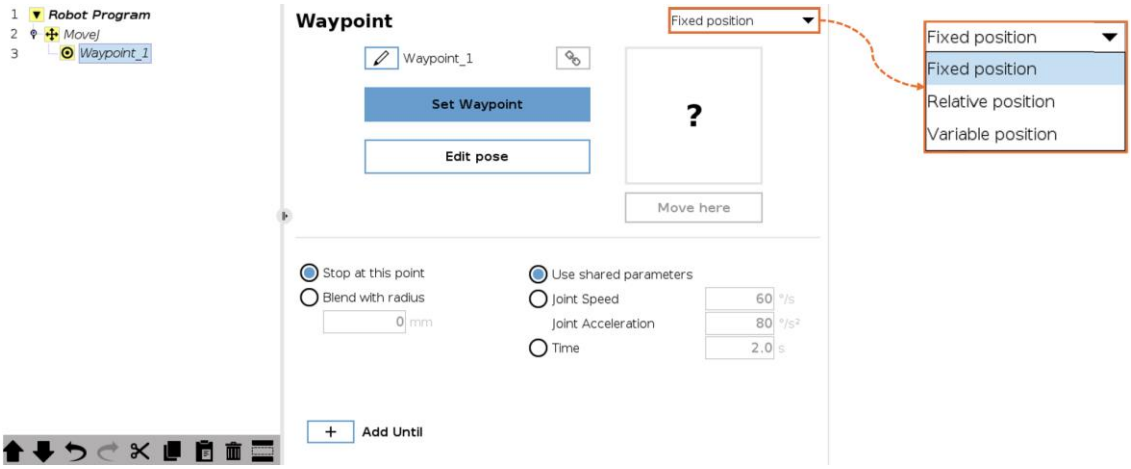


Figura 21 - Apresentação da instrução *Waypoint*

1. *Fixed Position* (Posição Fixa):

- Define uma posição absoluta no espaço cartesiano;

- O robô desloca-se para a posição definida, independentemente da sua localização inicial;
- Ideal para tarefas repetitivas com alta precisão.

2. *Relative Position* (Posição Relativa)

- Define a posição em relação a outro ponto de referência (por exemplo, o ponto anterior);
- Indicada para quando o robô precisa de fazer movimentos em sequência com deslocamentos constantes.

3. *Variable Position* (Posição Variável)

- A posição é definida por uma variável, que pode mudar durante o programa;
- Permite lógica dinâmica e adaptação a diferentes cenários.

8.4.3. Instrução *Direction*

A instrução *Direction* permite realizar movimentos lineares do TCP da ferramenta ao longo de um dos eixos cartesianos, com uma distância definida em milímetros. Este movimento é executado a partir da posição atual do TCP, assim que o programa atinge a linha onde a função está inserida. Para garantir que o movimento é interrompido no momento apropriado, é necessário configurar uma condição de paragem. O sistema disponibiliza quatro opções distintas para definir essa condição: uma expressão lógica, uma distância específica, a deteção de contacto da ferramenta ou a ativação de uma entrada digital ou analógica.

A Figura 22 apresenta a janela de configuração da instrução *Direction,* na qual é possível ajustar os parâmetros de movimento e definir a condição de paragem na secção *Until* da instrução.

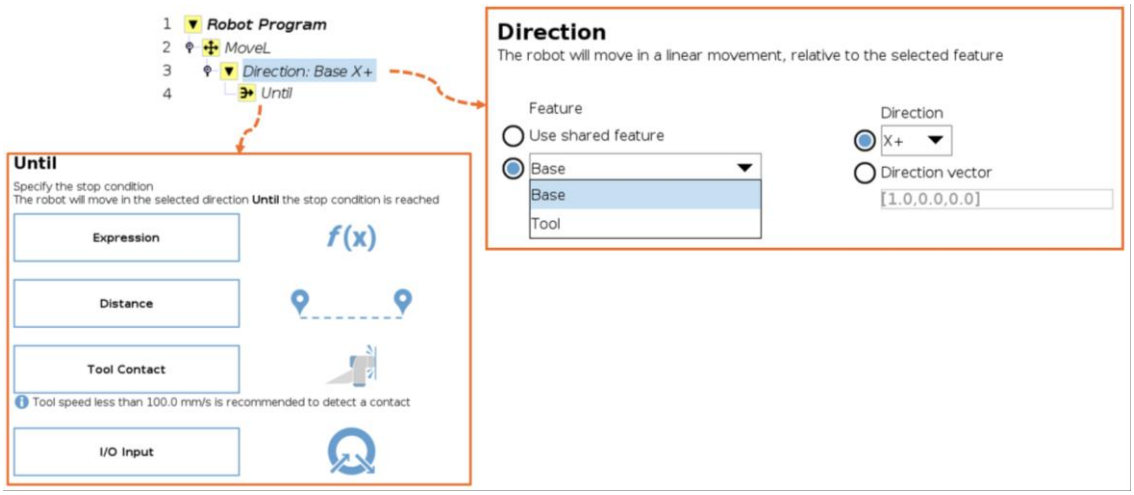


Figura 22 - Janela de ajuste de parâmetros da instrução *Direction*

8.4.4. Instrução *Wait*

A instrução *Wait* é amplamente utilizada na programação de robôs, pois permite introduzir pausas controladas na execução do programa. Esta funcionalidade é essencial para realizar compassos de espera prévios à ocorrência de determinados eventos ou condições, como a receção de sinais externos provenientes de entradas digitais ou analógicas ou pelo decurso de um intervalo de tempo definido. A opção de aguardar um número específico de segundos revela-se particularmente útil em operações que envolvem o funcionamento simultâneo com atuadores externos, como *grippers*. Nestes casos, para garantir que o robô segura a peça antes de prosseguir para a próxima instrução, é comum inserir uma instrução *Wait* de, por exemplo, 1 segundo. Sem essa pausa, a execução seria tão rápida que o robô avançaria para a instrução seguinte antes que o movimento de fecho da garra estivesse concluído. A Figura 23 apresenta o ecrã de configuração desta instrução no ambiente de programação, onde é possível definir o tempo de espera, bem como condições associadas a sinais digitais, analógicos ou expressões lógicas personalizadas.

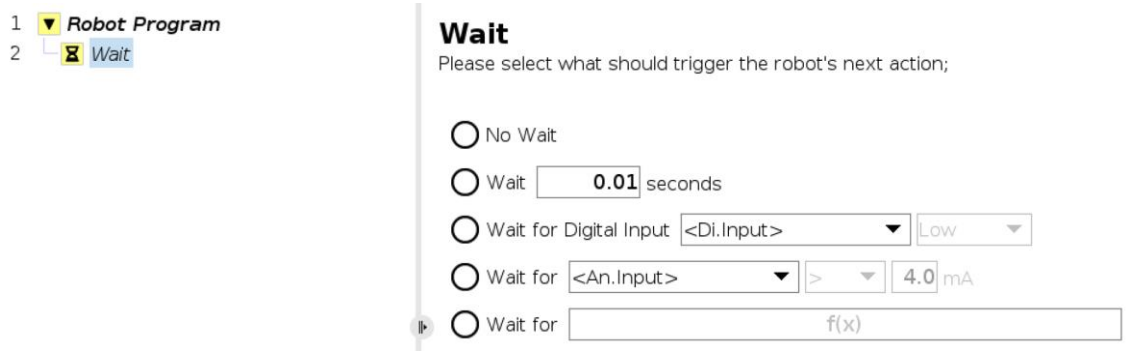


Figura 23 - Instrução Wait

8.4.5. Instrução Set

A instrução Set permite o controlo direto de **saídas digitais ou analógicas**, variáveis de sistema ou flags de execução. Esta instrução utiliza-se para ativar e desativar equipamentos externos, tais como lâmpadas/leds e voltímetro.

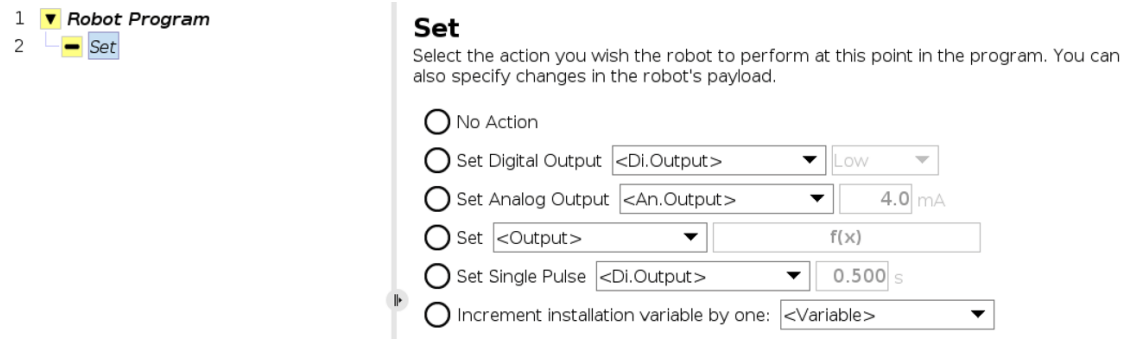


Figura 24 - Leitura de saídas digitais com instrução *Set*

8.4.6. Instrução Popup

A instrução *Popup* permite apresentar mensagens ao operador durante a execução do programa, funcionando como uma ferramenta de comunicação e controlo. Na Figura 25, observa-se a interface de configuração desta instrução, onde o utilizador pode escolher entre dois modos de entrada: **Texto** ou **Variável**. Ao optar por Texto, insere diretamente a mensagem que se pretende exibir. Se selecionar Variável, utiliza uma variável previamente definida para gerar o conteúdo da mensagem, o que permite maior flexibilidade e adaptação a diferentes contextos operacionais.

A interface oferece três tipos de popup: **Mensagem**, **Aviso** e **Erro**. A **Mensagem** serve para informar, o **Aviso** para alertar o operador de uma situação que requer atenção e o **Erro** indica uma condição crítica que pode exigir intervenção imediata. Além disso, esta instrução permite interromper a execução do programa ao apresentar o *popup* "*Halt Program execution at this popup*".

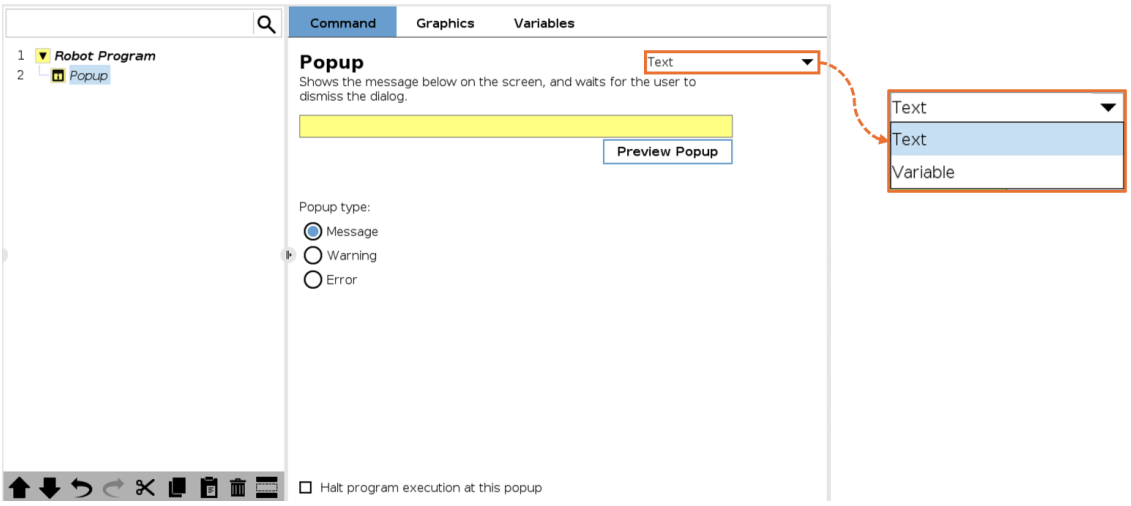


Figura 25 - Apresentação da instrução *Popup*

8.4.7. Instrução Halt

A instrução Halt tem como função interromper a execução de um programa. Para a utilizar basta inseri-la na linha de código pretendida (Figura 26).

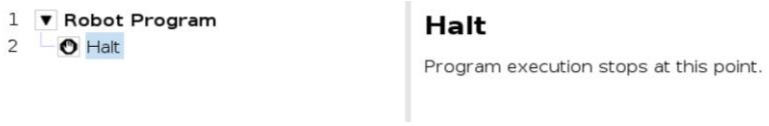


Figura 26 - Apresentação da instrução Halt

8.4.8. Instrução Comment

A instrução Comment permite inserir anotações/comenários não executáveis no programa (Figura 27). Embora não influencie a lógica, é essencial para a documentação e compreensão do código, especialmente em ambientes industriais colaborativos.

Figura 27 - Apresentação da instrução Comment

8.4.9. Instrução Folder

A instrução Folder permite agrupar logicamente instruções relacionadas, facilitando a organização estrutural e leitura do programa, nomeadamente em sistemas complexos (Figura 28).

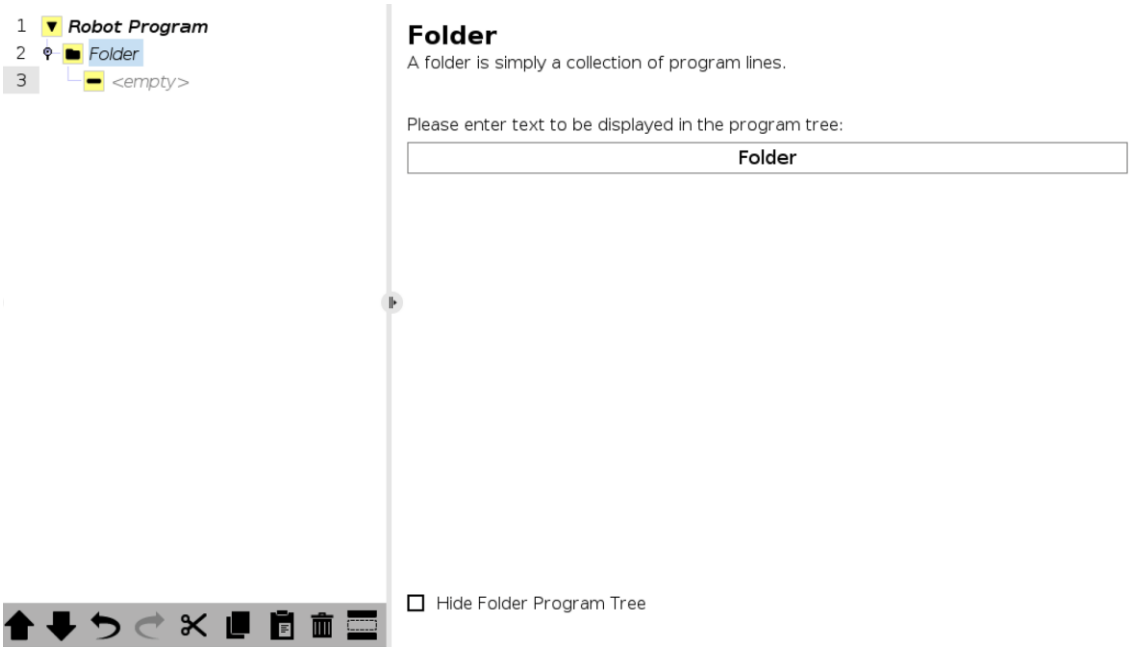


Figura 28 - Apresentação da instrução Folder

8.4.10. Instrução Set Payload

A instrução Set Payload define a carga útil que o robô manipula. A Figura 29 mostra duas opções:

- **Payload**: aplica o valor definido na instalação do robô (exemplo: 0,690 kg com centro de gravidade em (0,0,0) mm).
- **Custom Payload**: permite inserir manualmente a massa e o centro de gravidade, adaptando-os à ferramenta ou peça utilizada.

Desta forma, o programador escolhe entre um valor pré-configurado ou um valor personalizado, assegurando precisão e segurança na execução dos movimentos.

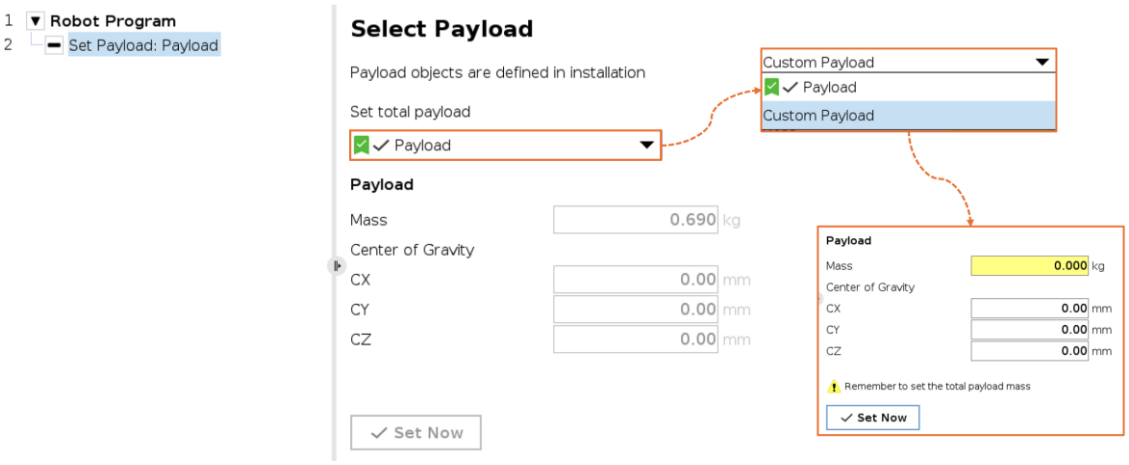


Figura 29 - Apresentação da instrução Set Payload

8.5. Secção Advanced

Neste subcapítulo apresentam-se as dez instruções Advanced disponíveis no ambiente de programação do UR3e e disponíveis na Figura 30.

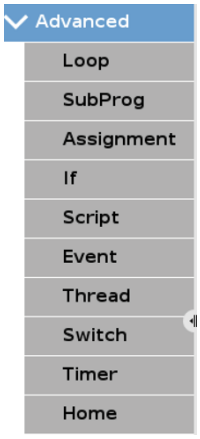


Figura 30 - Secção Advanced

8.5.1. Instrução Loop

Permite a repetição contínua ou controlada de um conjunto de instruções. A repetição pode ser contínua, restringida a ***X*** vezes ou enquanto uma condição for verdadeira (Figura 31).

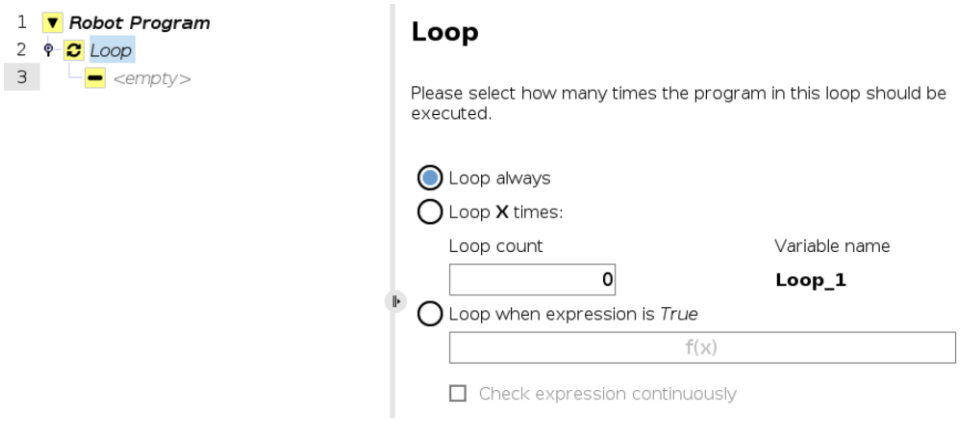


Figura 31 - Apresentação da instrução Loop

8.5.2. Instrução Subprog

A instrução Subprog permite desenvolver porções de código que são executados quando se invocam os subprogramas a partir do programa principal. Podem ser guardados como arquivos para reutilização noutros programas. Tal como se observa na Figura 32, a instrução apresenta opções que permitem carregar ficheiros já existentes, guardar novos subprogramas, limpar conteúdos ou atribuir um novo nome ao subprograma, o que garante maior flexibilidade na organização e reaproveitamento do código.

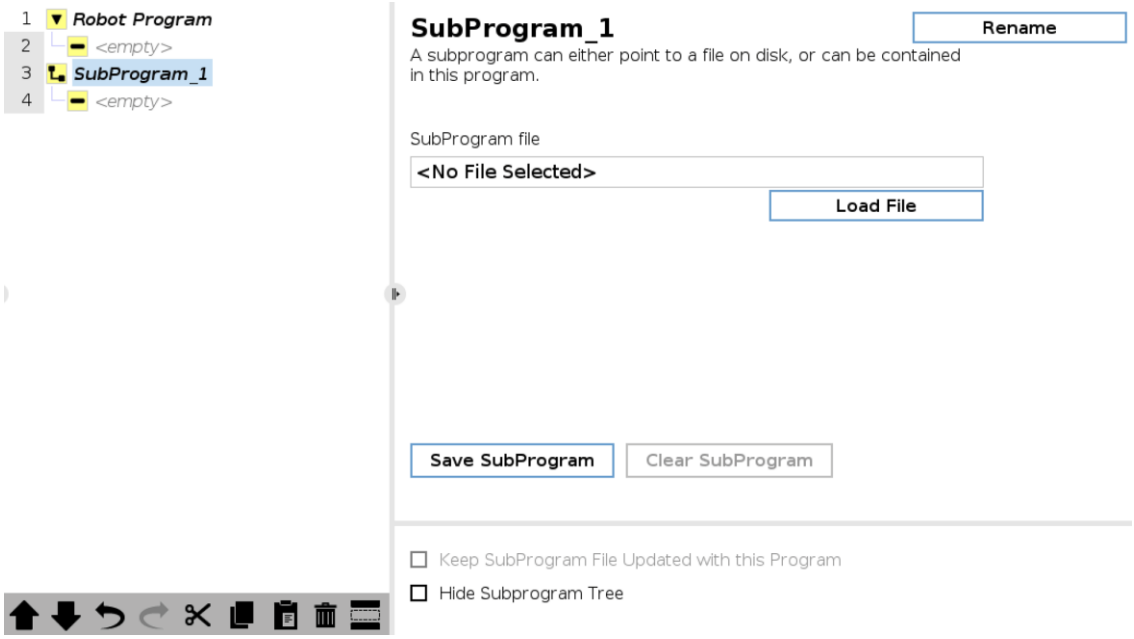


Figura 32 - Apresentação da instrução Subprog

8.5.3. Instrução Assignment

A instrução Assignment permite definir e criar variáveis, atribuindo-lhes valores resultantes de expressões ou introduzidos diretamente pelo operador. Conforme ilustrado na Figura 33, é possível selecionar a variável pretendida e indicar a sua origem: através de uma expressão

matemática ou lógica, ou através de um valor fornecido pelo operador. Neste último caso, podem solicitar-se diferentes tipos de dados, como uma resposta binária (Sim ou Não), um número inteiro, um número decimal ou uma cadeia de texto. Esta flexibilidade torna a instrução essencial para o controlo dinâmico do programa no ambiente PolyScope. No subcapítulo 8.8.1 apresenta-se a definição de variáveis, bem como os diferentes tipos disponíveis no ambiente de programação PolyScope.

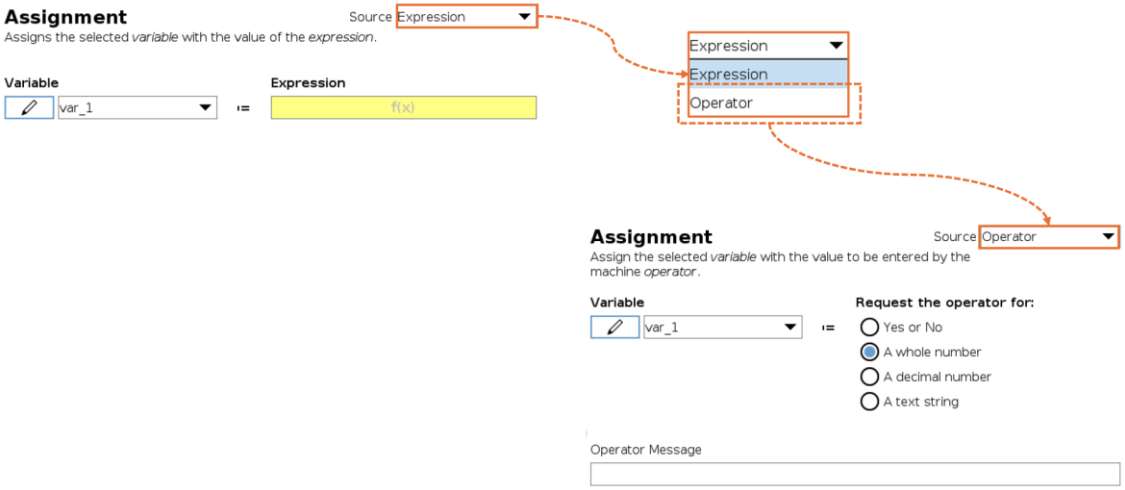


Figura 33 - Apresentação da instrução Assignment

8.5.4. Instruções If

As instruções If são instruções de controlo e fluxo de um programa são utilizadas para tomar decisões e direcionar a execução para zonas específicas do código, conforme os sinais recebidos ou valores de variáveis. Na Figura 34 apresenta-se a interface desta instrução. De notar que a adição das instruções ElseIf e Else realiza-se no comando da instrução If.

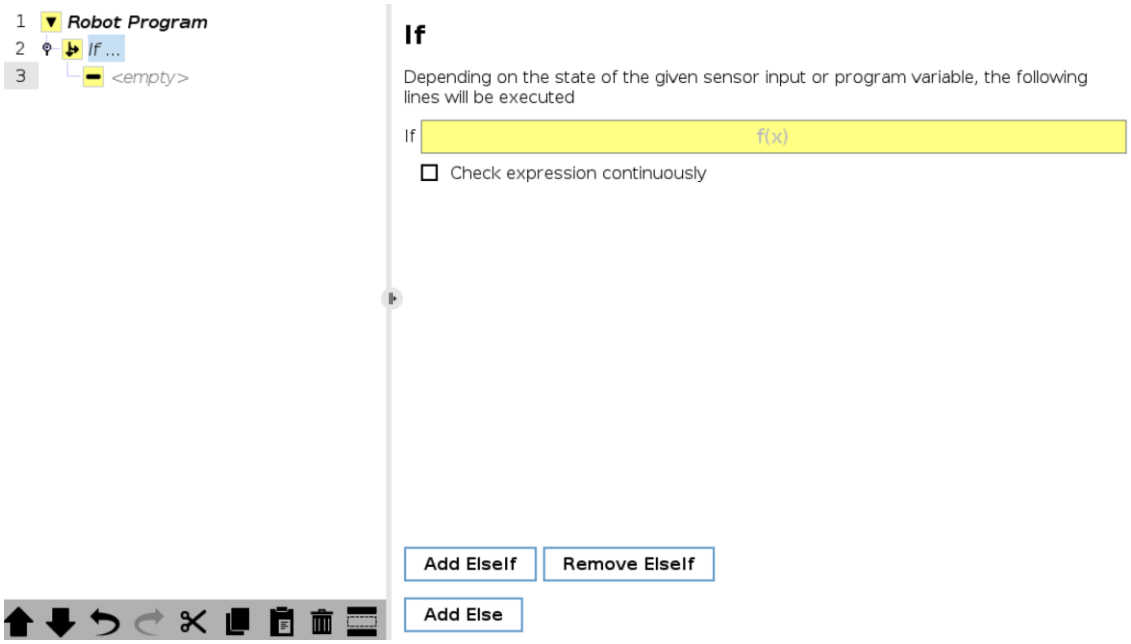


Figura 34 - Apresentação da instrução If

As instruções condicionais *If*, *ElseIf* e *Else* são geralmente utilizadas na maioria das linguagens de programação. Neste caso, na linguagem da interface PolyScope, são utilizadas da mesma forma. De seguida é descrito o funcionamento de cada uma das instruções:

- ***If***: Avalia o valor de um sinal ou variável. Se esse valor for verdadeiro, as linhas de código associadas são executadas.
- ***ElseIf***: Permite executar determinadas linhas de código se a condição for satisfeita. Junto com a instrução If e/ou outras instruções ElseIf, é possível criar um programa de seleção com base no valor de uma variável.
- ***Else***: Executa as linhas de código associadas se nenhuma das condições anteriores (If ou ElseIf) for satisfeita.

Na Figura 35 apresenta-se um exemplo de aplicação deste tipo de instruções de controlo. Nestas, em função do valor de uma variável denominada contador, executa-se o primeiro, segundo ou terceiro subprograma. Se o número não estiver entre 1 e 3, a interface exibe um aviso que indica “Contagem concluída”.

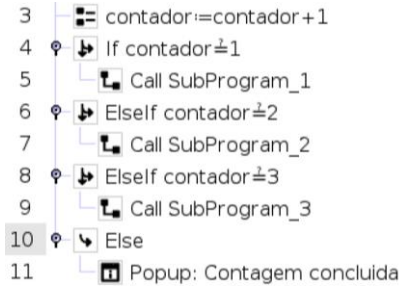


Figura 35 - Exemplo do uso da condicional *If*

8.5.5. Instrução Script

A instrução Script permite introduzir código adicional que o controlador UR executa. Esta funcionalidade amplia as possibilidades do programa para além dos blocos standard disponíveis no PolyScope. Tal como mostra a Figura 36, o utilizador pode escrever o código diretamente numa linha (Line) ou importar o conteúdo a partir de um ficheiro externo (File). Esta opção possibilita a integração de funções personalizadas, comandos específicos ou rotinas desenvolvidas previamente, o que assegura maior capacidade de adaptação às necessidades da aplicação.

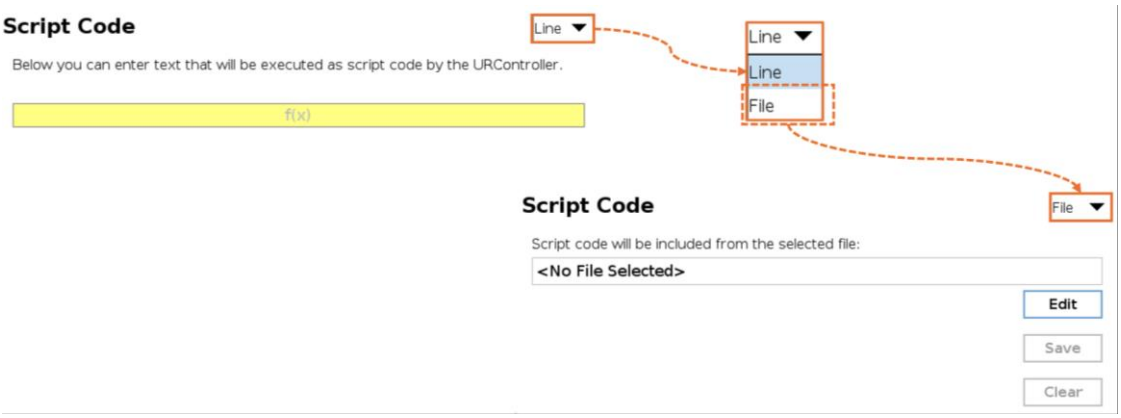


Figura 36 - Apresentação da instrução Script

8.5.6. Instrução Event

A instrução Event permite definir a execução de um conjunto de ações quando ocorre uma determinada condição, como a alteração do estado de uma variável ou de uma entrada de sensor. Ao contrário da instrução Interrupt, a execução do programa principal prossegue em simultâneo com o código do evento (Figura 37). Durante a execução de um evento, não é possível iniciar novos eventos. Esta instrução possibilita uma resposta imediata a situações específicas, aumentando a flexibilidade e a capacidade de adaptação do programa.

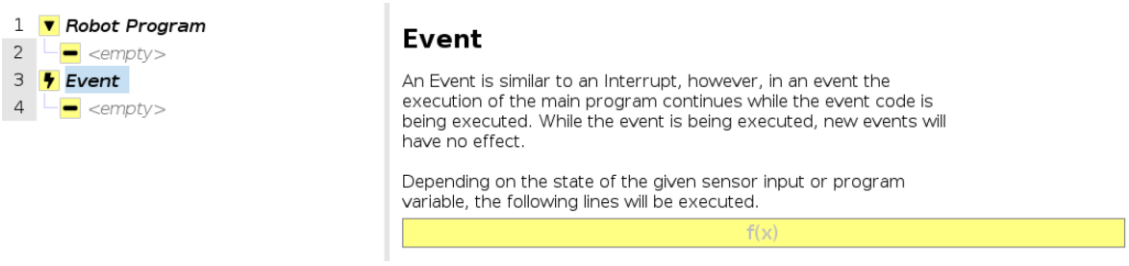


Figura 37 - Apresentação da instrução Event

8.5.7. Instrução Thread

A instrução **Thread** permite criar um programa paralelo que é executado em simultâneo com o programa principal. Com esta instrução é possível realizar operações de entrada/saída,

aguardar sinais e modificar variáveis de forma independente à lógica do programa principal do robô. A configuração da Thread inclui a opção *Loops Forever*, que assegura a repetição contínua das ações e a opção *Track program execution*, que possibilita o acompanhamento da sua execução (Figura 38). Esta funcionalidade é especialmente útil para controlar outros equipamentos ou supervisionar processos enquanto o robô executa a sua tarefa principal.

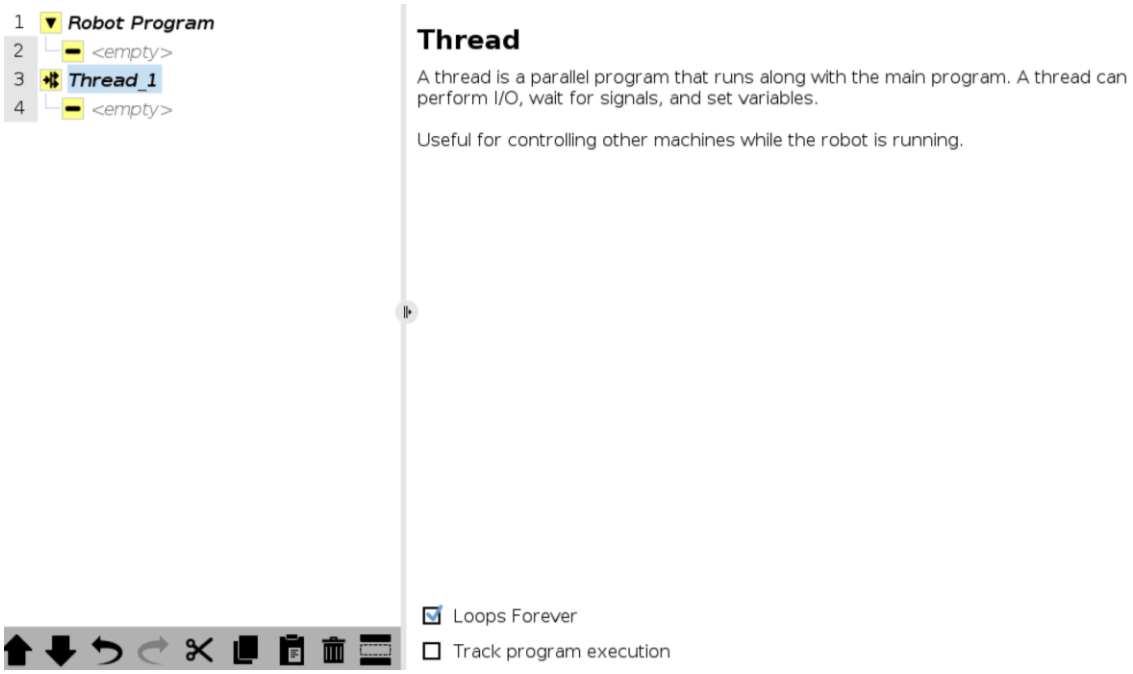


Figura 38 - Apresentação da instrução Thread

8.5.8. Instrução Switch

A instrução condicional ***Switch*** permite tomar decisões de controlo de fluxo com base no valor de uma variável. O seu modo de utilização é semelhante ao da condicional *If* em conjunto com *ElseIf*, no entanto de forma mais simplificada. A instrução lê o valor de uma variável e, com base nesse valor, executa cada um dos casos atribuídos. Se o valor da variável não corresponder a nenhum caso, é possível executar o *Default Case* que corresponde a um caso padrão. A Figura 39 apresenta o comando da instrução Switch, é a partir deste comando que se parametrizam as ações pretendidas.

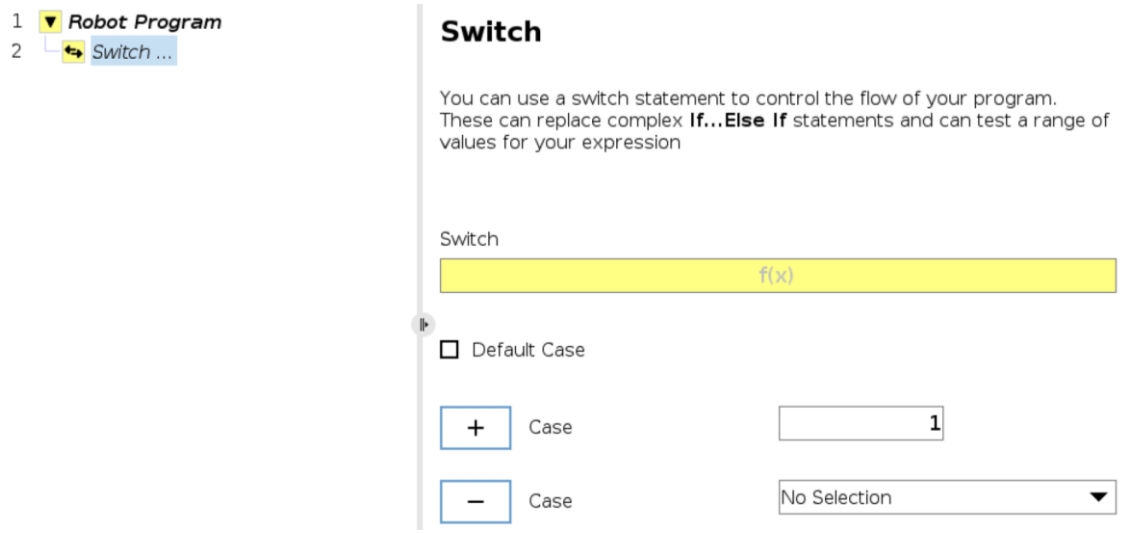


Figura 39 - Apresentação da instrução Switch

A Figura 40 apresenta o exemplo anteriormente resolvido com a estrutura condicional *If*, porém, neste caso, foi utilizada a instrução *Switch*.

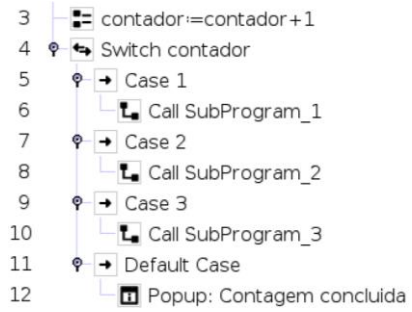


Figura 40 – Exemplo do uso do *Switch*

8.5.9. Instrução Timer

A instrução Timer permite medir intervalos de tempo durante a execução do programa. O valor medido é automaticamente armazenado numa variável associada ao temporizador, que pode ser utilizada em cálculos ou em condições lógicas. Esta instrução disponibiliza três ações: *Start*, que inicia a contagem; *Stop*, que interrompe a medição mantendo o valor registado; e *Reset*, que reinicia o temporizador para zero. O Timer é útil para controlar tempos de ciclo, monitorizar atrasos ou garantir que determinadas operações decorrem dentro de limites temporais definidos. Na Figura 41 apresenta-se a interface desta instrução.

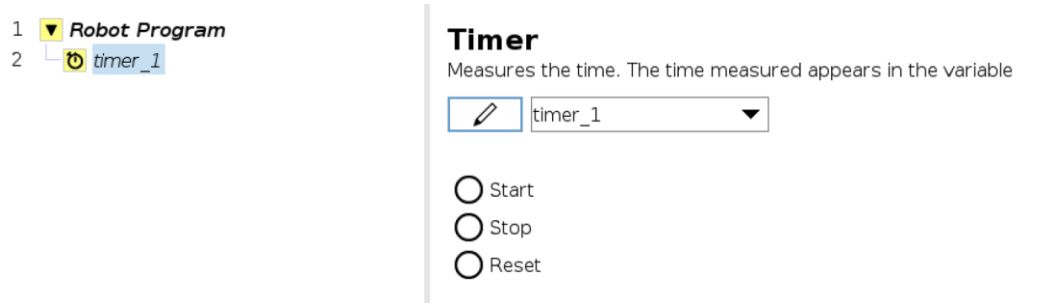


Figura 41 - Apresentação da instrução Timer

8.5.10. Instrução Home

A instrução Home permite mover o robô para a posição inicial definida durante a instalação. O movimento pode ser configurado em função da velocidade e aceleração das juntas (*Joint Speed* e *Joint Acceleration*) ou pelo tempo total de execução. Esta instrução assegura que o robô regressa a uma posição de referência segura, utilizada normalmente como ponto de partida ou de retorno no ciclo de trabalho. A função pode ainda ser associada à configuração de segurança (*Safe Home*), permitindo ligar a posição inicial a uma saída de segurança. A Figura 42 apresenta o ambiente de programação desta instrução.

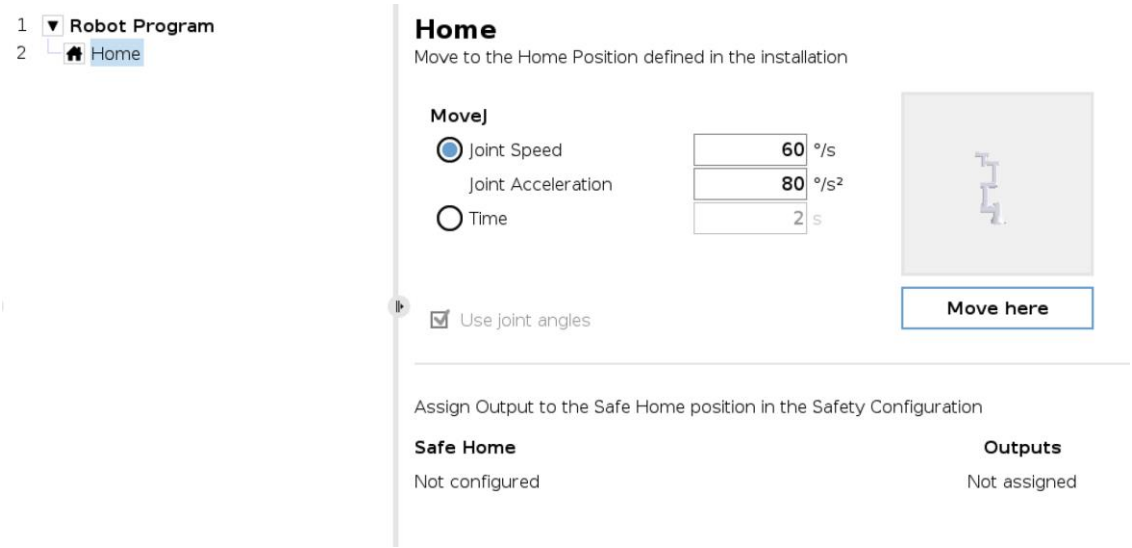


Figura 42 - Apresentação da instrução Home

8.6. Secção Templates

Neste subcapítulo apresentam-se as cinco instruções presentes na secção Templates do ambiente de programação do UR3e, disponíveis na Figura 43.

Figura 43 - Secção Templates

8.6.1. Instrução *Seek*

A instrução *Seek* permite realizar operações automáticas de empilhamento e desempilhamento de peças, através de parâmetros previamente definidos como direção, distância máxima e sequência de movimentos, considerando alguns graus de liberdade como força e localização, por exemplo. Na Figura 44 apresenta-se a interface da instrução. O utilizador deve definir inicialmente a finalidade da função: empilhamento ou desempilhamento. Cada opção implica um processo de parametrização distinto. O subcapítulo 0 descreve passo a passo a utilização e parametrização no modo de desempilhamento (*Destacking*).

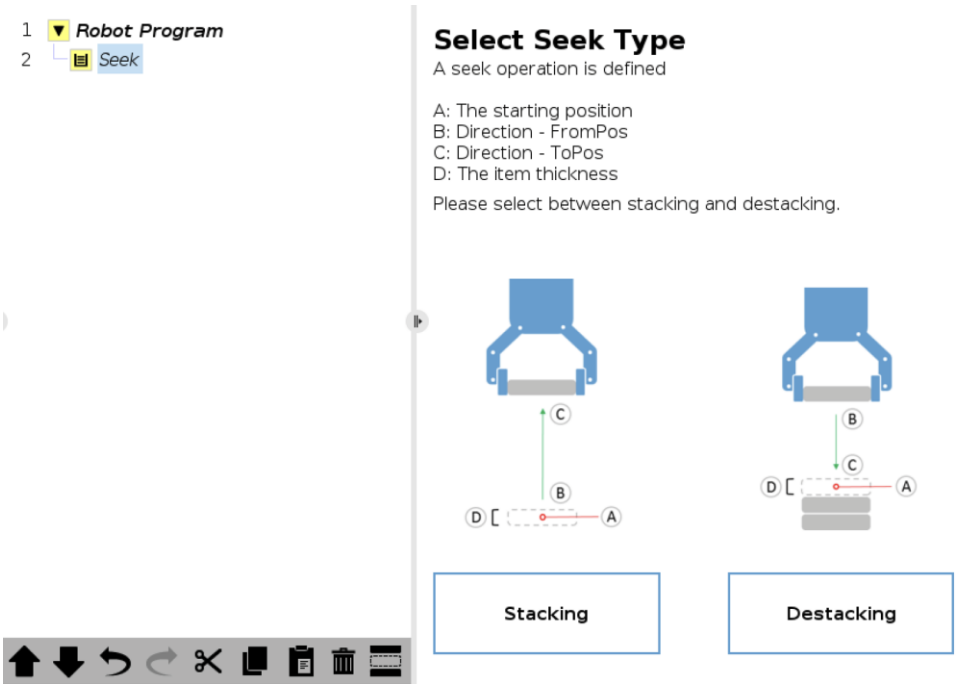


Figura 44 - Apresentação da instrução Seek

8.6.2. Instrução Force

A instrução Force permite executar uma parte do programa em modo de força, no qual o robô mantém liberdade de movimento numa direção específica para aplicar a força definida. Esta funcionalidade é útil em tarefas que exigem contacto controlado com o ambiente, como inserções (operações em que o robô coloca uma peça dentro de outra), polimento ou encaixes.

Tal como se observa na Figura 45, é possível selecionar diferentes modos de funcionamento (Simple, Frame, Point, Motion) e escolher a referência da força em relação à *Base* ou à *Tool*. Além disso, o utilizador define numericamente a intensidade da força a aplicar, medida em Newtons (N), e pode testar a configuração através do botão Test em combinação com a função em modo *freedrive* do Teach Pendant.

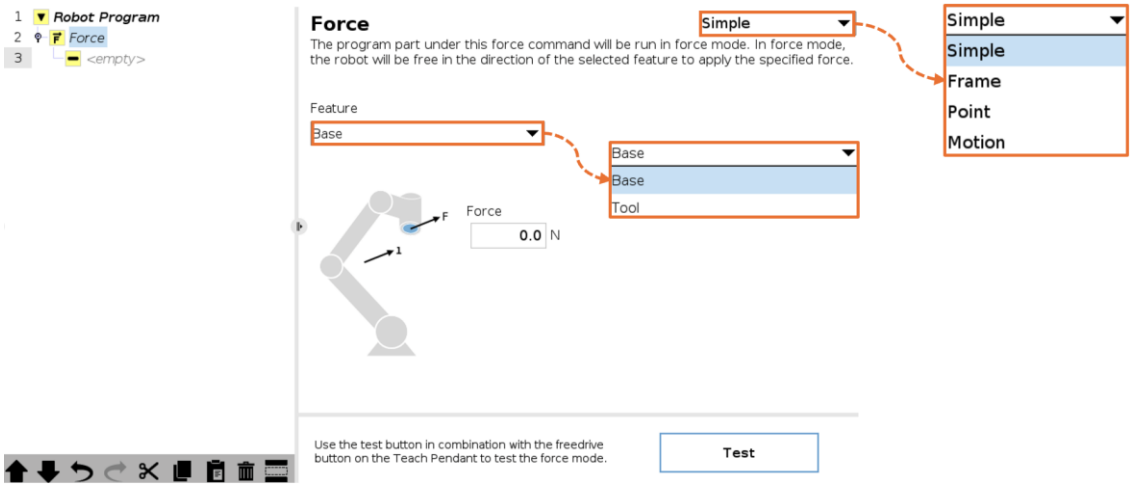


Figura 45 - Apresentação da instrução Force

Na Tabela 9 apresentam-se os diferentes modos de funcionamento da instrução Force, uma descrição sobre o seu funcionamento e aplicação típica.

Tabela 9 – Apresentação dos diferentes modos de funcionamento da instrução Force

Modo	Funcionamento	Aplicação típica
Simple	Aplica uma força linear constante numa direção definida (ex.: eixo X, Y ou Z).	Empurrar uma peça, pressionar um botão, apoiar contra uma superfície.
Frame	Aplica a força em relação a um sistema de coordenadas definido pelo utilizador.	Seguir a orientação de um objeto ou superfície inclinada.
Point	Direciona a força para um ponto específico no espaço, independentemente da posição da ferramenta.	Empurrar ou ajustar peças em direção a um alvo fixo.
Motion	Combina movimento com aplicação de força, adaptando-se ao contacto.	Operações de polimento, lixagem, encaixe ou ajuste fino.

8.6.3. Instrução Palletizing (paletização)

A instrução *Palletizing* permite realizar a paletização e despalatização de peças em três padrões diferentes, designadamente linha, grelha e irregular (diferentes orientações). A instrução é constituída por uma estrutura de programação pré-definida que deve ser devidamente parametrizada pelo utilizador (Figura 46). No *software* PolyScope do UR3e é possível programar diferentes estratégias de paletização para otimizar o espaço e a eficiência do processo. No subcapítulo 0 apresenta-se o passo a passo do modo de utilização e parametrização desta instrução (no modo de paletização).

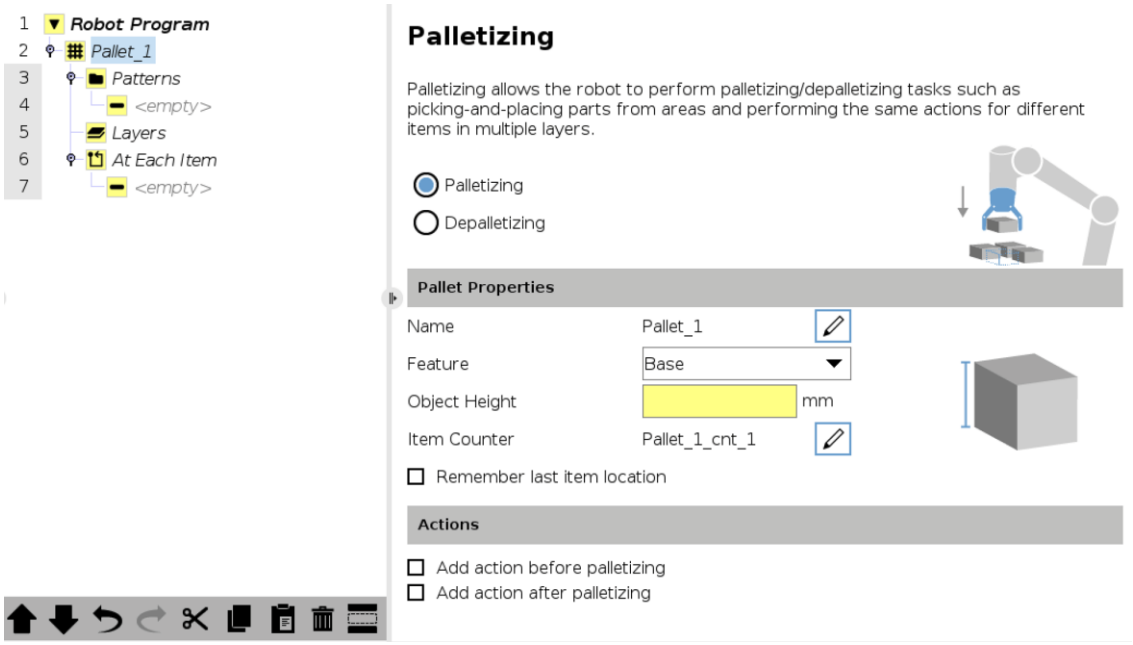


Figura 46 - Apresentação da instrução Palletizing

A Figura 47 ilustra quatro exemplos comuns de configurações. Estas configurações podem ser implementadas diretamente no PolyScope, através da opção de paletização, definindo o número de camadas, filas, colunas, e deslocamentos nos eixos X, Y e Z.

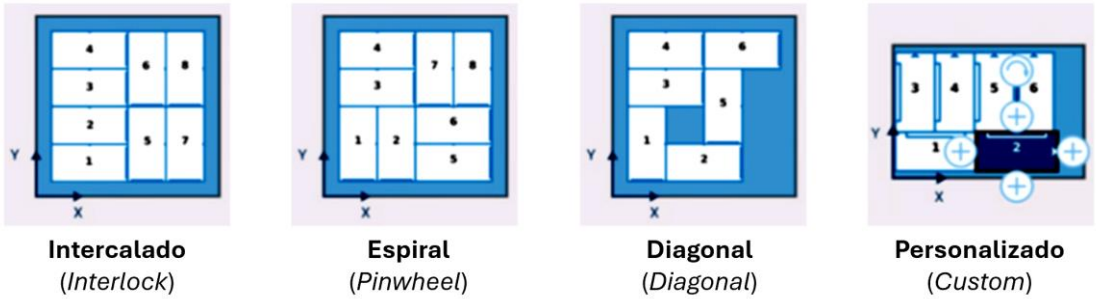


Figura 47 - Exemplos de padrões de paletização

8.6.4. Instrução Conveyor Tracking

A instrução *Conveyor Tracking* permite que o robô sincronize os seus movimentos com o deslocamento de uma ou mais transportadoras. Desta forma, as operações de manipulação ocorrem em movimento contínuo sem necessidade de parar o tapete. Quando a função de *Conveyor Tracking* está corretamente configurada na instalação, o robô ajusta os seus movimentos de acordo com o avanço da correia transportadora. Tal como se observa na Figura 48, é possível selecionar qual a transportadora ativa (*Conveyor 1* ou *Conveyor 2*) e estruturar o programa sob este nó, assegurando que todos os movimentos definidos permanecem relativos ao movimento da correia.

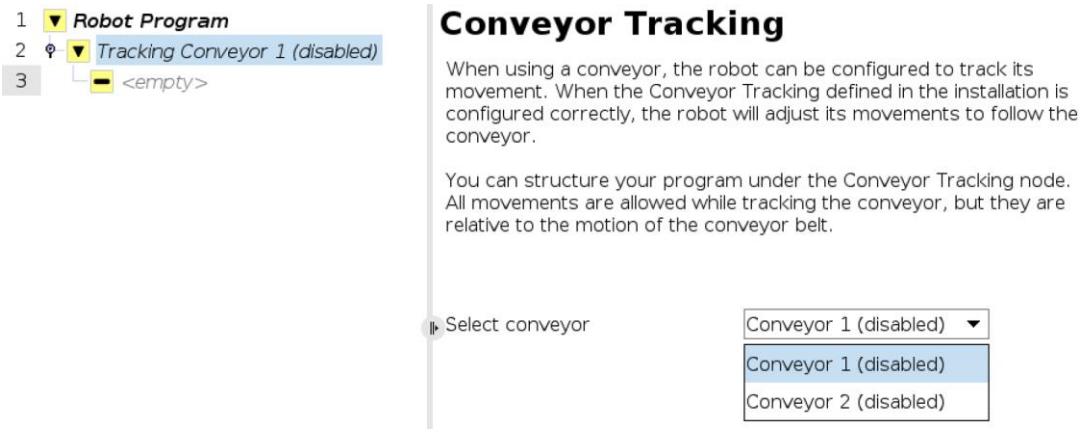


Figura 48 - Apresentação da instrução Conveyor Trackig

8.6.5. Instrução Screwdriving

A instrução Screwdriving permite controlar parafusadoras elétricas ligadas ao robô e executar operações de aparafusamento ou desaparafusamento no programa. O utilizador define o sentido de rotação, a velocidade, o binário aplicado e a duração da operação. Além disso, permite associar sinais de entrada e saída para sincronizar a operação da ferramenta com o ciclo do robô. Esta funcionalidade assegura precisão, repetibilidade e eficiência em processos de montagem automatizada.

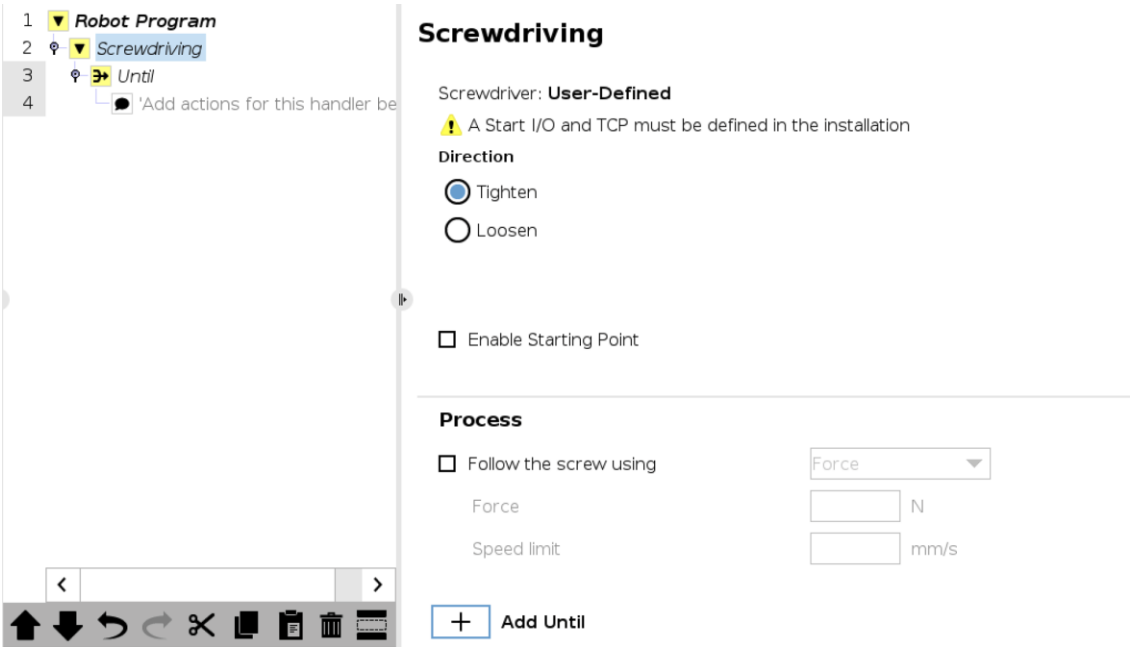


Figura 49 - Apresentação da instrução Screwdriving

8.7. Secção URCaps

Neste subcapítulo apresentam-se as instruções URCaps disponíveis no ambiente de programação do UR3e, com foco na instrução Zimmer.

Figura 50 - Secção URCaps

8.7.1. Instrução Zimmer

As instruções Zimmer são responsáveis pelos comandos do do *gripper*. Na Figura 51 encontra-se o ambiente de programação da instrução Zimmer. Esta instrução dispõe de diversas opções que o utilizar pode selecionar conforme a aplicação desejada.

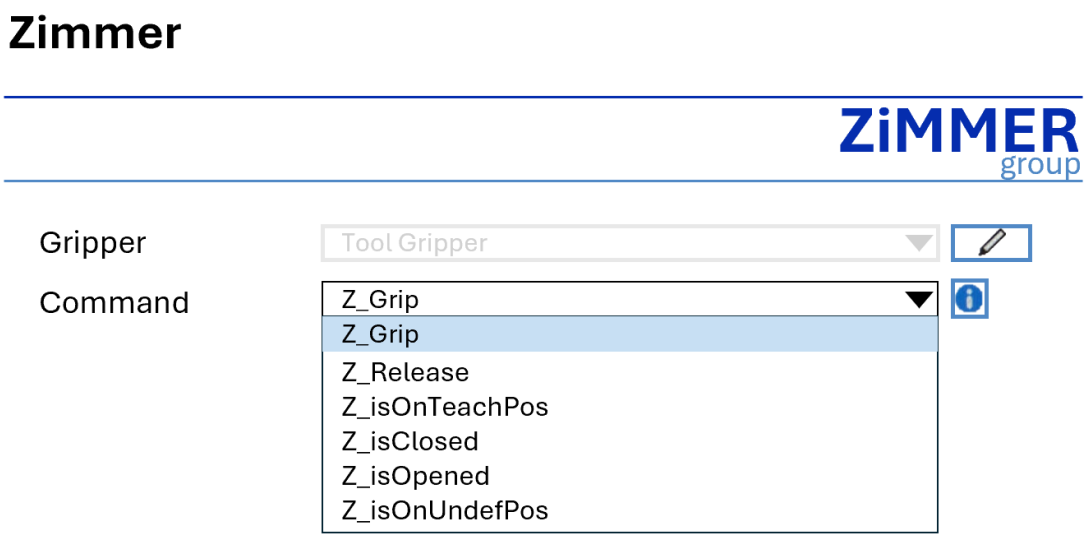


Figura 51 - Apresentação da instrução Zimmer

As duas operações mais utilizadas correspondem à abertura e ao fecho das garras do *gripper*. Segue-se a apresentação destas instruções:

- **Abrir:**
  - Z_Release: Gripper 1.
- **Fechar:**
  - Z_Grip: Gripper 1.

8.8. Conceitos e instruções de programação complementares

8.8.1. Variáveis

Uma variável é um elemento de programação que ocupa uma zona de memória no controlador, utilizada para armazenar valores em função do tipo de dados, que posteriormente são processados e modificados no modo de execução.

As variáveis podem apresentar diferentes tipos de dados como booleana, número inteiro, número real, *string* e posição. Na Tabela 10 são apresentados os diferentes tipos de variáveis, o seu valor e um exemplo.

Tabela 10 - Tipos de dados das variáveis

Tipo	Valor	Exemplo
Booleana	*True* ou *False*	Botão_verde = *True*
Número inteiro	Entre -2147483648 e 2147483647	Contador=0
Número real	Números de pontos flutuantes de 32 bits	Posição Y = 0.15
String	---	"Olá, mundo"
Posição	p[X,Y,Z,rX,rY,rZ]	p[0.2,0.3,0,0,0,0]

No ambiente de programação PolyScope e UR, as variáveis podem ser de três tipos:

- **Variáveis de programa:**
    - Criadas diretamente na árvore do programa;
    - Usadas para controlar a lógica e o comportamento do robô durante a execução;
    - Específicas para o programa em que foram criadas.

- **Variáveis de instalação:**
    - Definidas na aba de instalação do PolyScope;
    - Permitem configurar elementos como posições, características de ferramentas, *payloads*, entre outros;
    - Podem ser reutilizadas em diferentes programas dentro da mesma instalação.

- **Variáveis de *Script* (URScript):**
    - Utilizadas em *scripts* personalizados e escritas em URScript;
    - Permitem maior controlo e flexibilidade como *loops*, condições e chamadas de funções;
    - Úteis para programadores mais experientes que pretendem ir além da interface gráfica.

8.8.2. Leitura de entradas digitais

A leitura de entradas digitais na estrutura de um programa pode ser feita de várias formas, das quais se destacam as seguintes:

- **Leitura do sinal em instruções de controlo de fluxo**: Neste caso, se o valor do sinal for verdadeiro (True), o bloco de programa dependente da condição é executado, como

por exemplo numa instrução *If*. No exemplo presente na Figura 52, se a entrada digital 1 for ativa, o programa excuta a trajetória 1. Por outro lado, se a entrada digital 2 for ativa, o programa executa a trajetória 2.

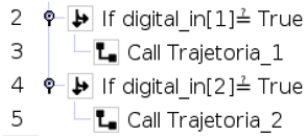


Figura 52 - Leitura de entradas digitais na instrução *If*

- **Espera por um valor na entrada digital:** A execução é interrompida numa linha do programa até que o valor do sinal seja avaliado como LO (Baixo) ou HI (Alto). Neste caso, a execução do programa principal fica a aguardar até que essa linha de espera seja verificada. Este comportamento pode ser observado na Figura 53, que apresenta o modo de utilização da instrução *Wait*, configurada para aguardar um sinal *High* (alto) na entrada digital *digital_in[1].*

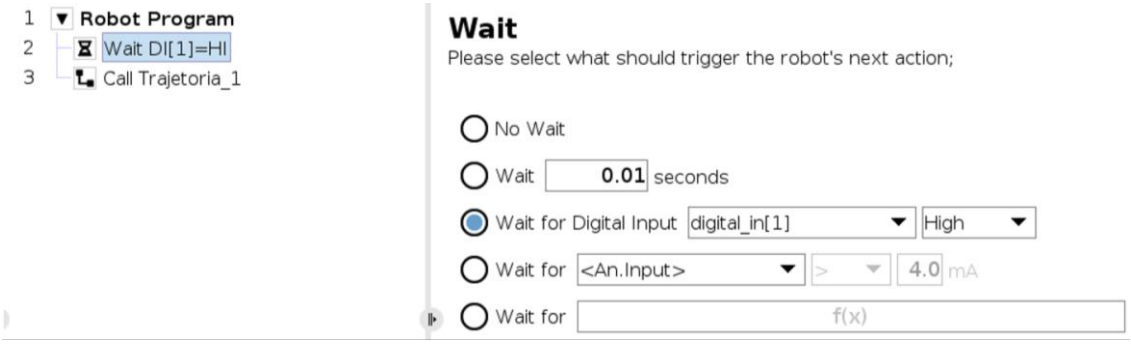


Figura 53 - Leitura de entradas digitais na instrução *Wait*

8.8.3. Leitura de saídas digitais

As saídas digitais podem ser lidas e escritas de forma semelhante às entradas digitais. A sua inserção no programa é realizada através da instrução *Set,* que se encontra no menu básico de programação (Basic). Na Figura 54 é possível visualizar as diferentes opções disponíveis para a configuração desta instrução.

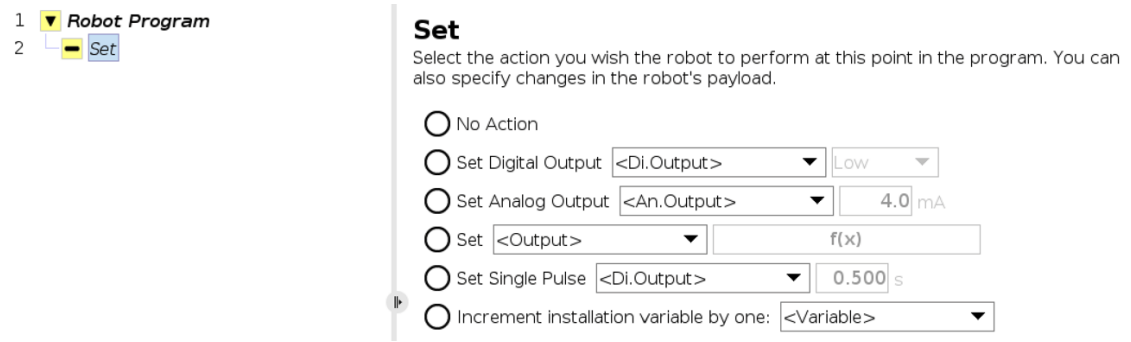


Figura 54 - Leitura de saídas digitais com instrução *Set*

Existem três formas principais de utilização das saídas digitais, designadamente:

- **Set Digital Output** (ajustar saída digital): Neste caso, ativa-se ou desativa-se a saída digital pretendida, utilizando os sinais *High* para ativar e *Low* para desativar.
- **Set** (ajustar saída com função lógica): Controla uma saída com base numa função lógica, por exemplo, ativar uma saída digital somente quando uma determinada entrada digital estiver ativa.
- **Set Single Pulse** (ajustar pulso único): Ativa uma saída com um pulso de flanco positivo, cujo tempo é ajustável.

8.8.4. Leitura de entradas analógicas

A leitura de entradas analógicas no ambiente PolyScope é realizada de forma direta e contínua. Na bancada desenvolvida há um potenciómetro conectado à entrada analógica 1, o que significa que qualquer variação no valor do potenciómetro, seja por ajuste manual ou alteração na resistência, é imediatamente refletida na aba de I/O do sistema.

A leitura em tempo real da entrada analógica permite monitorizar com precisão os sinais analógicos recebidos, facilitando o controlo de processos que dependem de valores variáveis como intensidade de luz, temperatura, posição ou velocidade de movimento do robô (exemplo de exercício apresentado no capítulo 10.

8.8.5. Leitura de saídas analógicas

No ambiente PolyScope, as saídas analógicas podem ser utilizadas para enviar sinais contínuos ou variáveis para dispositivos externos, como voltímetros. Ao contrário das entradas analógicas, que são lidas diretamente, as saídas analógicas são definidas ou ajustadas por comandos no programa. No entanto, é possível monitorizar o valor atual da saída através da aba de I/O, o que permite verificar se o sinal enviado corresponde ao valor programado. As saídas analógicas são configuradas através da instrução *Set*. O capítulo 10 apresenta um exercício no qual o valor de um contador é vinculado à saída analógica zero, que está conectada ao voltímetro do sistema.

8.8.6. Posicionamento com variáveis

A linguagem de programação da UR permite utilizar pontos de passagem, ou destinos, com base nos valores de variáveis. Para isso, é necessário criar uma variável de posição (*pose*) e, posteriormente, atribuir valores a cada um dos elementos dessa variável.

Variável *Pose* (posição)

Uma variável do tipo *Pose* é representada por uma matriz 1x6, na qual as primeiras três colunas são de posição (X, Y, Z) e as três restantes de rotação (rX, rY, rZ). Os valores de posição são dados em metro e os de rotação em radianos. Cada elemento deve manter o formato de número real e pode ser atribuído com variáveis. O capítulo 10 apresentam-se dois exemplos de exercícios resolvidos que utilizam este tipo de configuração para a movimentação do robô.

8.8.7. Uso de variáveis como um contador

Para utilizar uma variável como se fosse um contador, utiliza-se uma sintaxe semelhante à das linguagens de programação usuais. Para que a variável seja incrementada (ou decrementada) numa quantidade relativamente ao valor que ela possui, deve-se aplicar a seguinte expressão:

Contador = Contador + 1

Para criar uma variável de programa, é necessário introduzir uma instrução *Assignment* (Figura 55). A variável pode ser criada na secção *Before Start* ou na árvore do programa principal.

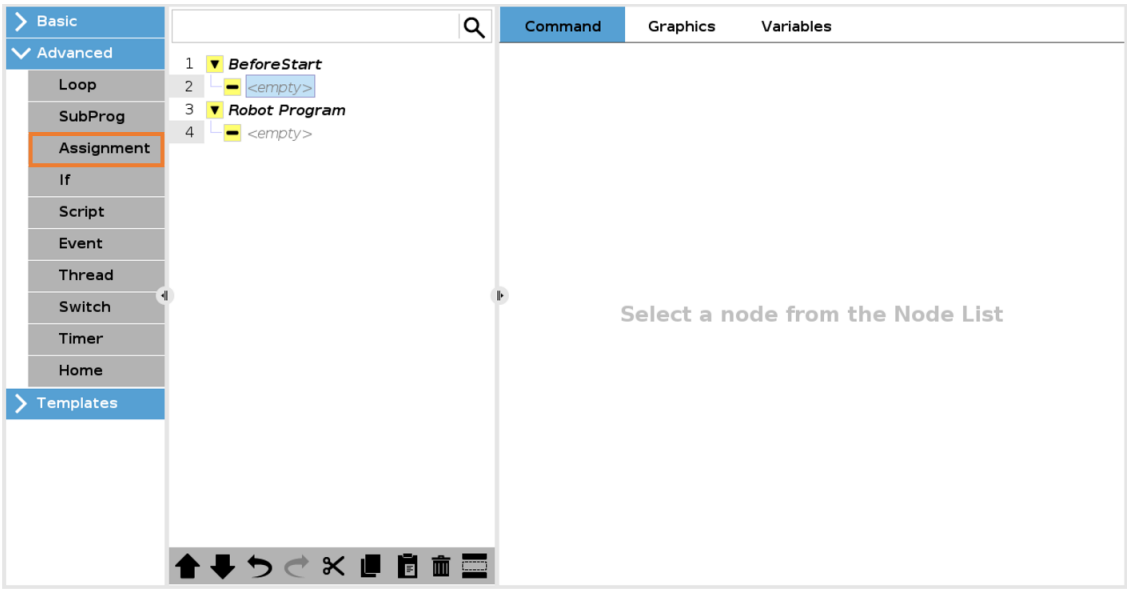


Figura 55 - Inserção da instrução *Assignment*

De seguida, deve ser definido o nome da variável. Para tal segue-se a sequência de três passos presente na Figura 56. Neste caso, atribuiu-se o nome *count* para a variável contador.

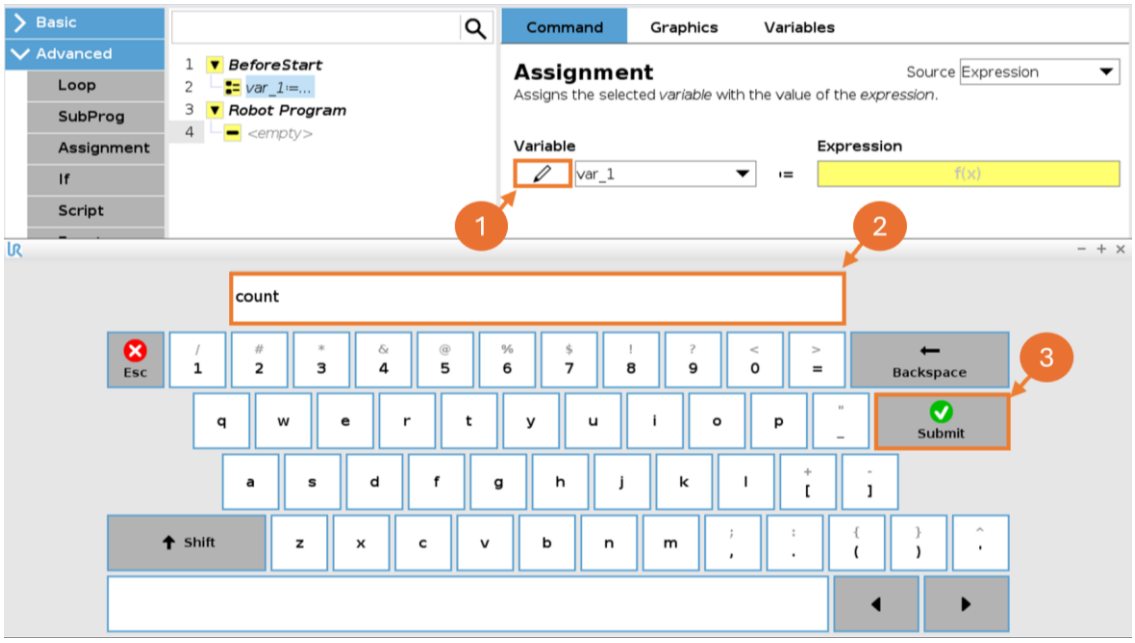


Figura 56 - Atribuição do nome à variável

Após a atribuição do nome à variável, procede-se com a atribuição do valor inicial da variável, que neste caso é zero (Figura 57).

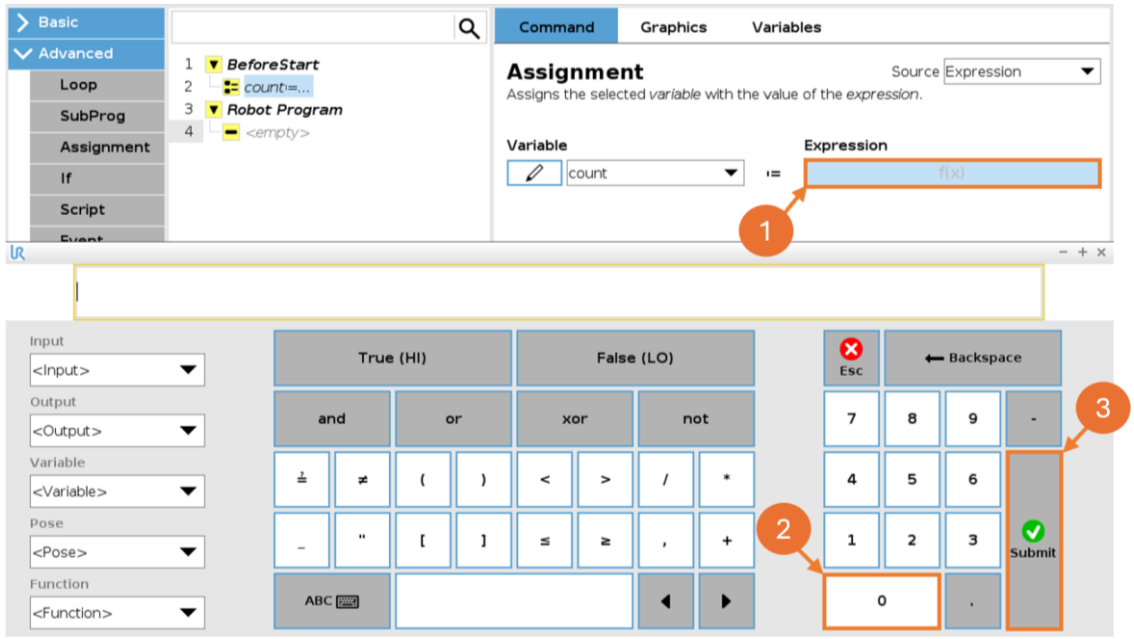


Figura 57 - Atribuição do valor inicial à variável

Nesta fase, a variável contador encontra-se criada. Assim, torna-se possível "chamar" a variável e incrementar o seu valor. Para tal, deve-se inserir uma instrução *Assignment* e, no campo de expressão, colocar a expressão pretendida. Na Figura 58 está presente uma representação do editor de expressões. Independentemente da instrução, o campo de expressões funciona da mesma forma e é composto pela zona de seleção de sinais e variáveis, a zona da expressão e os teclados que permitem editar a expressão.

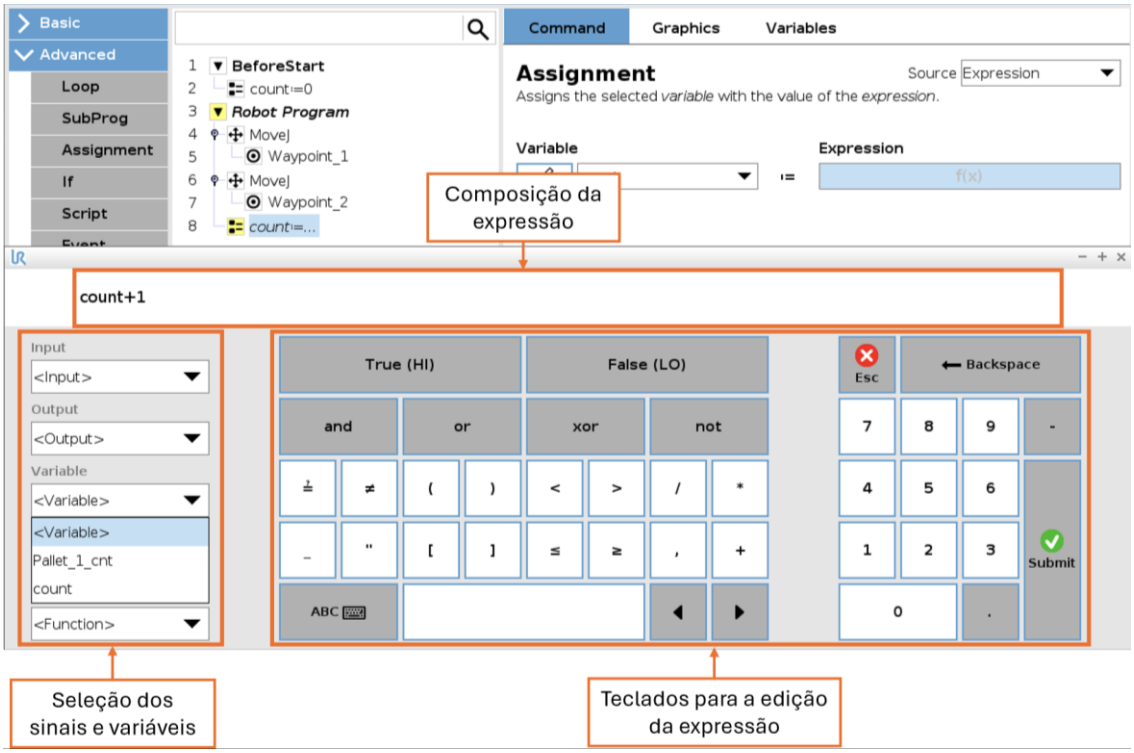


Figura 58 - Editor de expressões

8.8.8. Posição de início ou de origem

A posição de origem é uma posição que o utilizador pode utilizar sempre que necessário, seja através de um programa ou diretamente a partir da janela de movimento. A posição inicial, configurada de fábrica, é designada por *posição zero* e corresponde à configuração ilustrada na Figura 59. Como esta posição não é muito útil do ponto de vista prático, é aconselhável alterá-la de acordo com as necessidades do utilizador.

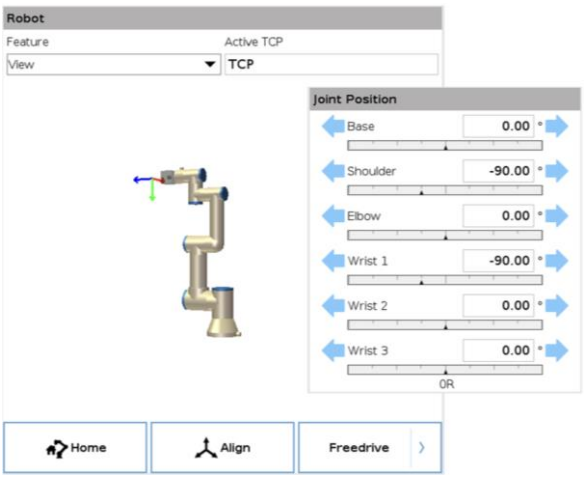


Figura 59 - Posição zero de um robô UR

9. Competências

9.1. Competências técnicas e operacionais

Com o objetivo de demonstrar as funcionalidades do sistema, apresentam-se diversos exercícios práticos que abrangem desde movimentos básicos do robô até a interação com entradas e saídas digitais e analógicas. Estes exercícios permitem explorar diferentes instruções de programação, testar ciclos condicionais e integrar o funcionamento de periféricos, como o *gripper*.

Na Tabela 11 estão descritas as competências técnicas e operacionais associadas a cada um dos exercícios propostos.

Tabela 11 - Competências técnicas e operacionais associadas a cada exercício

	Competências Técnicas	Competências Operacionais
Exercício 1	• Compreender e configurar diferentes tipos de movimento (*MoveJ*, *MoveL*, *MoveP*); • Definir *Waypoints*.	• Interpretar de forma clara as diferentes trajetórias.
Exercício 2	• Programar condições operacionais com instruções lógicas (*If/ElseIf*); • Configurar e testar entradas digitais (DI).	• Reconhecer as diferenças entre respostas a sinais externos.
Exercício 3	• Utilizar instruções de *Popup* para interação com o utilizador; • Gerar mensagens na interface.	• Comunicar de forma clara com o operador durante a execução do programa.
Exercício 4	• Configurar e testar saídas digitais (DO); • Programar ações de resposta complementares (ex: ativar *LED*).	• Representar o estado do sistema com sinais visuais.
Exercício 5	• Utilizar contadores e variáveis lógicas; • Implementar ciclos de *Pick and Place*.	• Executar sequências automáticas simples.
Exercício 6	• Utilizar contadores e variáveis lógicas; • Implementar ciclos de *Pick and Place;* • Programar subprogramas com *Palletizing* e *Seek;* • Utilizar parâmetros de desempilhamento e paletização.	• Automatizar processos com repetição estruturada.

Tabela 11 - Competências técnicas e operacionais associadas a cada exercício (continuação)

Exercício 7	• Modificar a lógica de funcionamento de programas previamente desenvolvidos; • Configurar e testar saídas analógicas; • Relacionar variáveis internas do programa (contador) com sinais analógicos (voltímetro).	• Efetuar leituras no voltímetro e interpretar os valores medidos; • Confirmar correspondência entre o comportamento do programa e o resultado obtido no instrumento de medida; • Desenvolver autonomia na execução de ajustes e testes ao programa.
Exercício 8	• Criar e controlar ciclos de repetição através de contadores; • Relacionar variáveis internas (contador de peças) com sinais analógicos externos (voltímetro); • Garantir a sequência lógica de operações, assegurando a execução ordenada do processo de paletização.	• Efetuar a paletização de peças de forma ordenada e repetitiva, validando a correta transferência entre zonas de recolha e paletização; • Realizar medições no voltímetro e interpretar os valores em função do número de peças já paletizadas;
Exercício 9	• Definir e configurar variáveis para leitura e manipulação de sinais analógicos; • Implementar funções de conversão de valores contínuos em parâmetros de velocidade; • Programar restrições e limites de segurança em variáveis numéricas; • Utilizar comunicação via *socket* para atualização dinâmica de parâmetros de controlo; • Configurar e testar entradas analógicas; • Executar movimentos repetitivos entre pontos predefinidos (Waypoints) com velocidade variável.	• Ajustar a velocidade do robô em tempo real através da interação com o potenciómetro; • Monitorizar o comportamento do sistema para não ultrapassar os limites de segurança; • Desenvolver sensibilidade para interpretar a resposta do robô face às variações de velocidade; • Demonstrar capacidade de intervenção manual para interromper o processo de forma segura; • Gerir a execução contínua do programa garantindo estabilidade e suavidade no movimento.
Exercício 10	• Programar trajetórias baseadas em funções trigonométricas (sin() e cos()); • Definir pontos sucessivos através de incrementos angulares para percorrer curvas aproximadas; • Implementar variáveis e parâmetros de controlo (ângulo θ, raio, incrementos); • Desenvolver sub-rotinas que permitam desenhar formas geométricas no plano de trabalho	• Interpretar e validar trajetórias geométricas representadas por pontos discretos; • Controlar a qualidade do percurso realizado, para assegurar correspondência entre o trajeto programado e a geometria prevista; • Avaliar os efeitos da variação do incremento angular e da velocidade de movimento na suavidade e precisão da trajetória.

Tabela 11 - Competências técnicas e operacionais associadas a cada exercício (continuação)

Exercício 11	• Programar trajetórias complexas em PolyScope, aproximando curvas através de pontos sucessivos gerados com funções trigonométricas (cos() e sin()); • Configurar limites de velocidade no programa, garantindo a correção automática de valores fora do intervalo permitido. • Estabelecer comunicação via socket para transmitir comandos de ajuste de velocidade ao controlador. • Integrar lógica de programação que assegure a atualização em tempo real da velocidade de execução durante o movimento.	• Ajustar a velocidade do robô em tempo real através de um potenciómetro, interpretando a correspondência entre a rotação do potenciômetro e a alteração da velocidade do robô; • Verificar a execução correta do percurso circular; • Avaliar a suavidade e a continuidade do movimento em função da variação dinâmica da velocidade.
Exercício 12	• Implementar lógica condicional baseada em variáveis binárias, permitindo a tomada de decisão em tempo real (aceitação/rejeição); • Sincronizar a execução do programa com ações externas introduzidas pelo operador, como a decisão na fase de inspeção; • Utilizar e atualizar variáveis internas (contador de peças transportadas) para gerir a sequência do processo; • Implementar estratégias de sinalização luminosa através de LEDs, associando estados específicos do programa às diferentes fases do processo (início, inspeção, aceitação, rejeição).	• Confirmar o correto funcionamento do sistema de I/O digital, incluindo a receção de sinais de entrada (botão verde, variável binária do operador) e a emissão de sinais de saída (LEDs); • Realizar inspeção de peças e tomar decisões fundamentadas, assegurando que a classificação como “aceite” ou “rejeitada” corresponde ao destino atribuído pelo robô; • Desenvolver autonomia na identificação de falhas e na realização de ajustes ao programa para corrigir eventuais problemas.
Exercício 13	• Programar ciclos de empilhamento e desempilhamento com controlo de variáveis; • Desenvolver subprogramas para modularizar operações de recolha e posicionamento de peças; • Implementar rotinas de *Pick and Place* otimizadas para múltiplas zonas de trabalho.	• Organizar fluxos de trabalho estruturados para manipulação de peças; • Gerir diferentes zonas de recolha e posicionamento de forma clara e sistemática; • Demonstrar capacidade de planear tarefas industriais aplicáveis a contextos reais de logística e produção.

10. Exercícios

Este capítulo apresenta exercícios práticos desenvolvidos para demonstrar a aplicabilidade da bancada didática no contexto do ensino da robótica colaborativa e da automação. O objetivo principal é permitir aos utilizadores compreender e aplicar os conceitos de entradas e saídas digitais e analógicas através da utilização do robô colaborativo UR3e e do sistema de controlo integrado na bancada. Para além disso, é esperado que os utilizadores se familiarizem com a linguagem de programação do PolyScope.

Exercício 1 - Tipos de Movimento: *MoveJ*, *MoveL* e *MoveP*

Este exercício tem como objetivo analisar e comparar os três tipos de movimento disponíveis na instrução ***Move***: ***MoveJ***, ***MoveL*** e ***MoveP***, sendo que cada um corresponde a um método distinto de interpolação dos eixos do robô. A implementação e execução de cada tipo de movimento permite observar o comportamento prático do robô.

Requisitos do exercício:

- Criar um programa e denominá-lo como *exercício_1*;
- Inserir 3 instruções *Move*;
- Associar os movimentos *MoveJ, MoveL* e *MoveP* às 3 instruções *Move*;
- Definir a posição dos *Waypoints*;
- Renomear os *Waypoints* como *ponto_1, ponto_2* e *ponto_3*, respetivamente.

Metodologia de resolução

Passo 1 – Criação do Programa

Comece por criar um programa com o nome *exercicio_1*. Para isso, selecionar a opção ***Program***, localizada no segundo módulo da aba superior esquerda da interface (Figura 60).

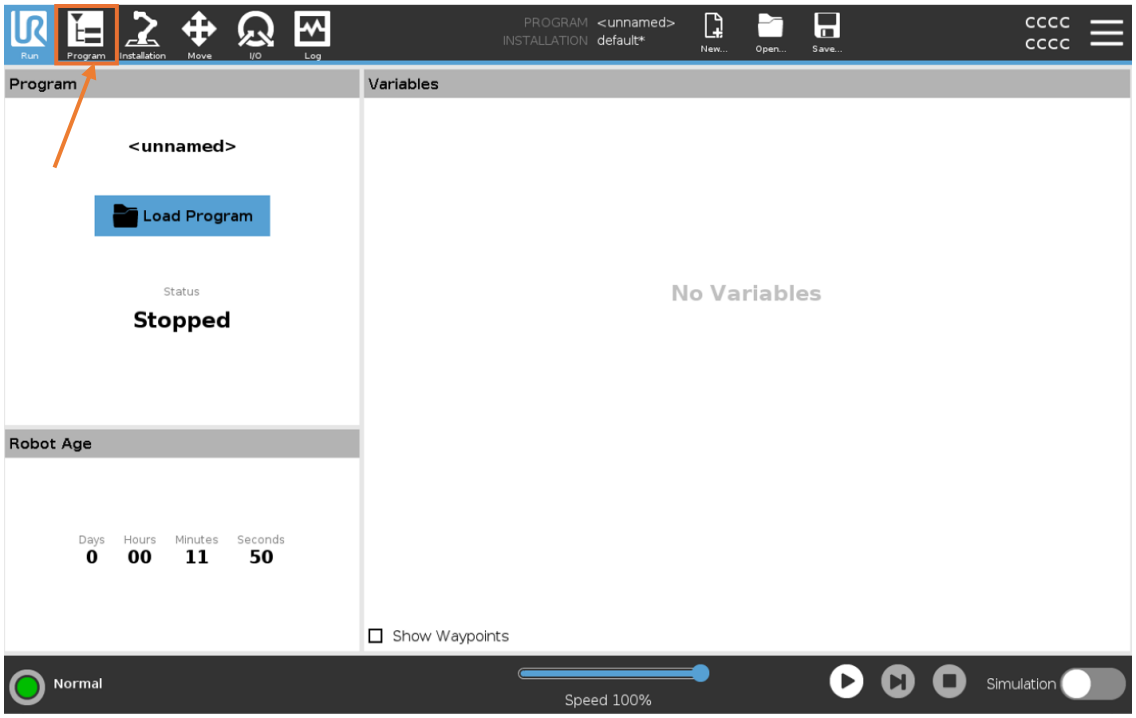


Figura 60 - Criação um programa

Em seguida, guardar o programa através da sequência ***Save > Save Program As > Filename > exercicio_1 > Submit*** (Figura 61, 6 e 7).

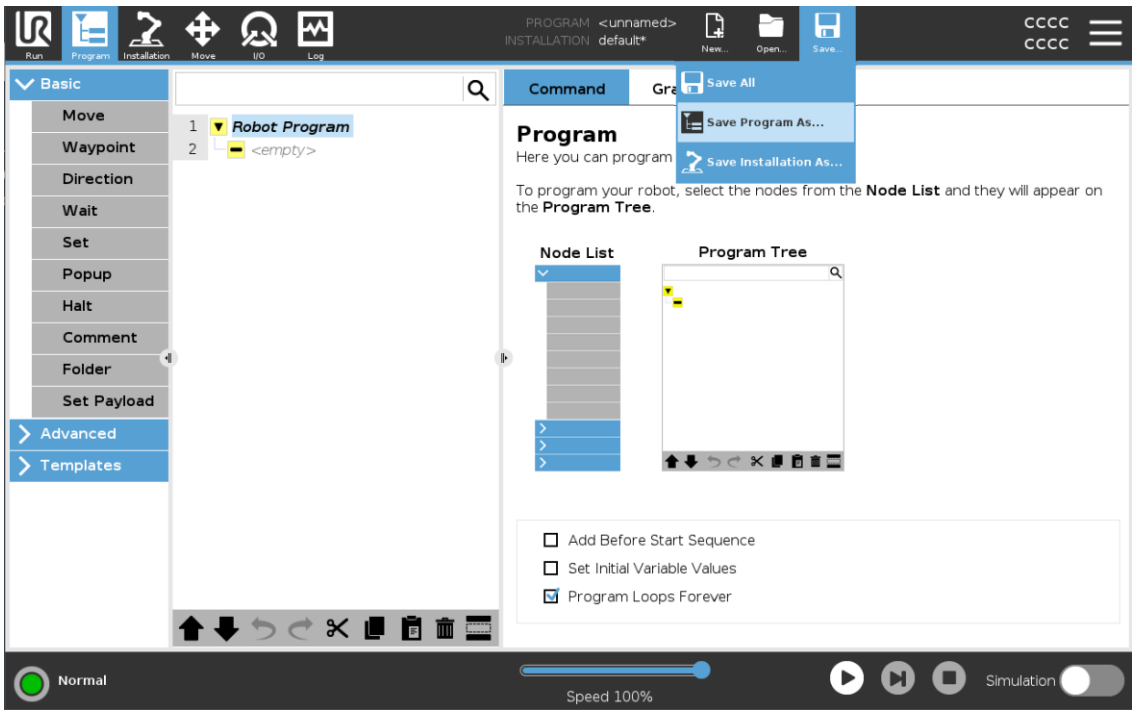


Figura 61 - Guardar um programa

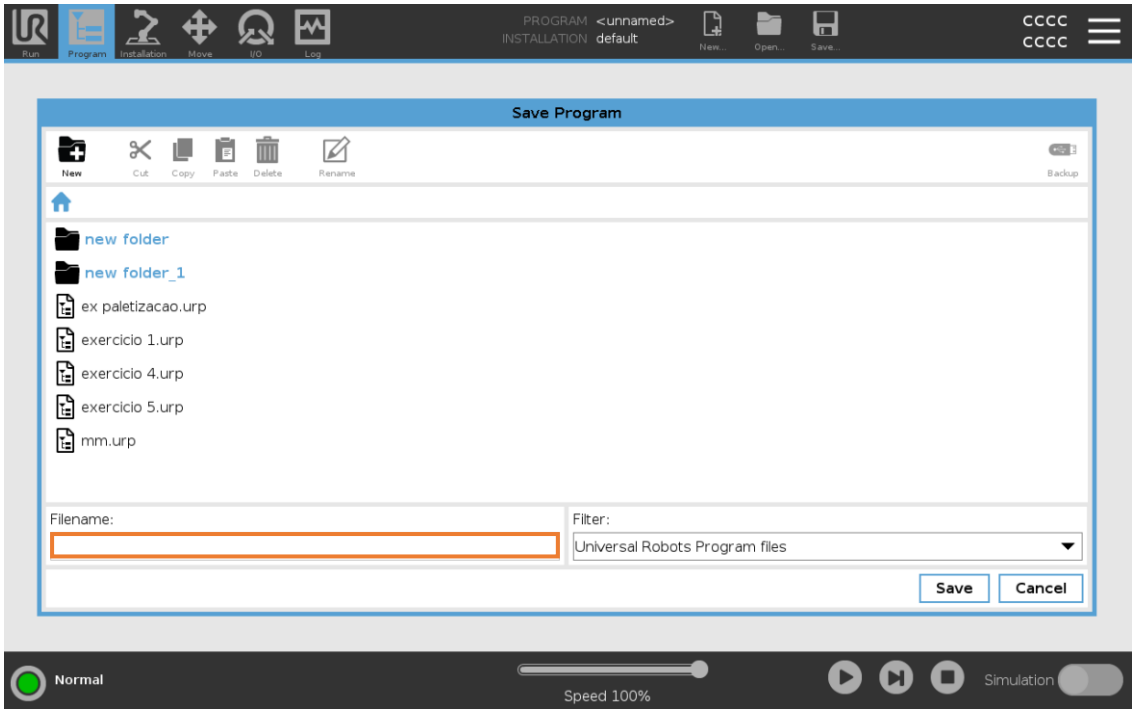


Figura 62 - Atribuir o nome a um programa (1/2)

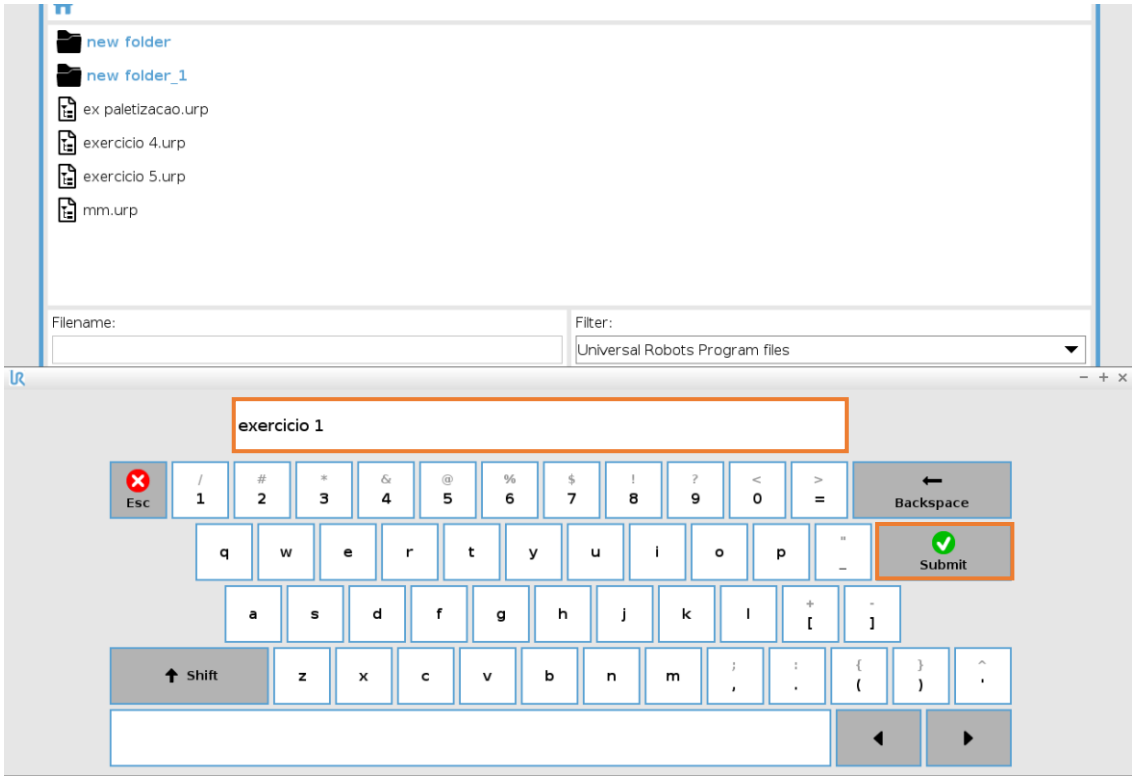


Figura 63 - Atribuir o nome a um programa (2/2)

Passo 2 – Inserção das Instruções de Movimento

Com a ramificação *empty* selecionada, inserir 3 instruções *Move* (Figura 64 e Figura 65).

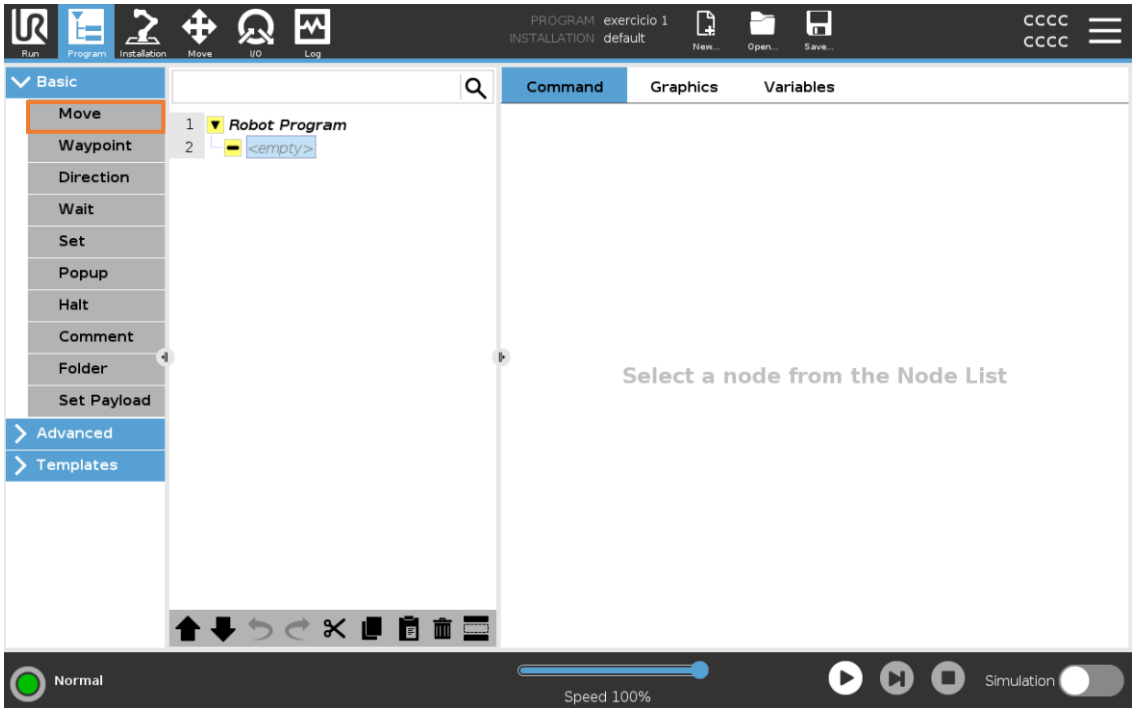


Figura 64 – Inserção da instrução *Move*

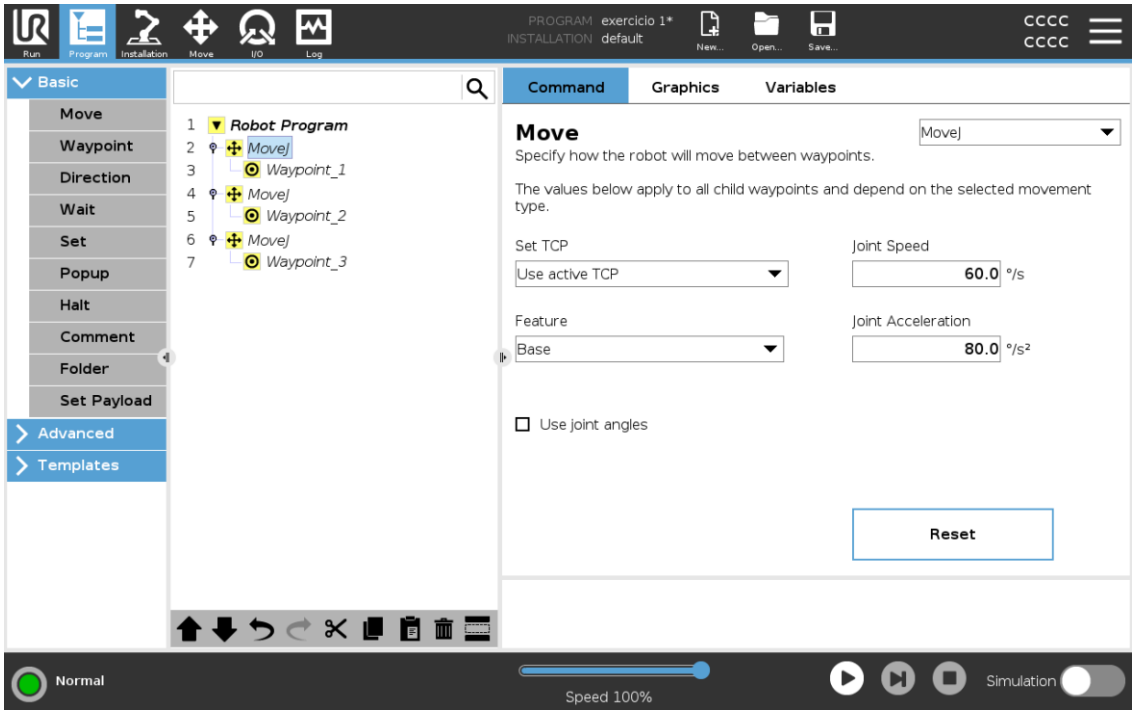


Figura 65 - Instruções *Move* inseridas no programa

Por padrão, a instrução *Move* assume o tipo ***MoveJ***, que pode ser alterado para ***MoveL*** ou ***MoveP*** no menu lateral, pasta comando (Figura 66).

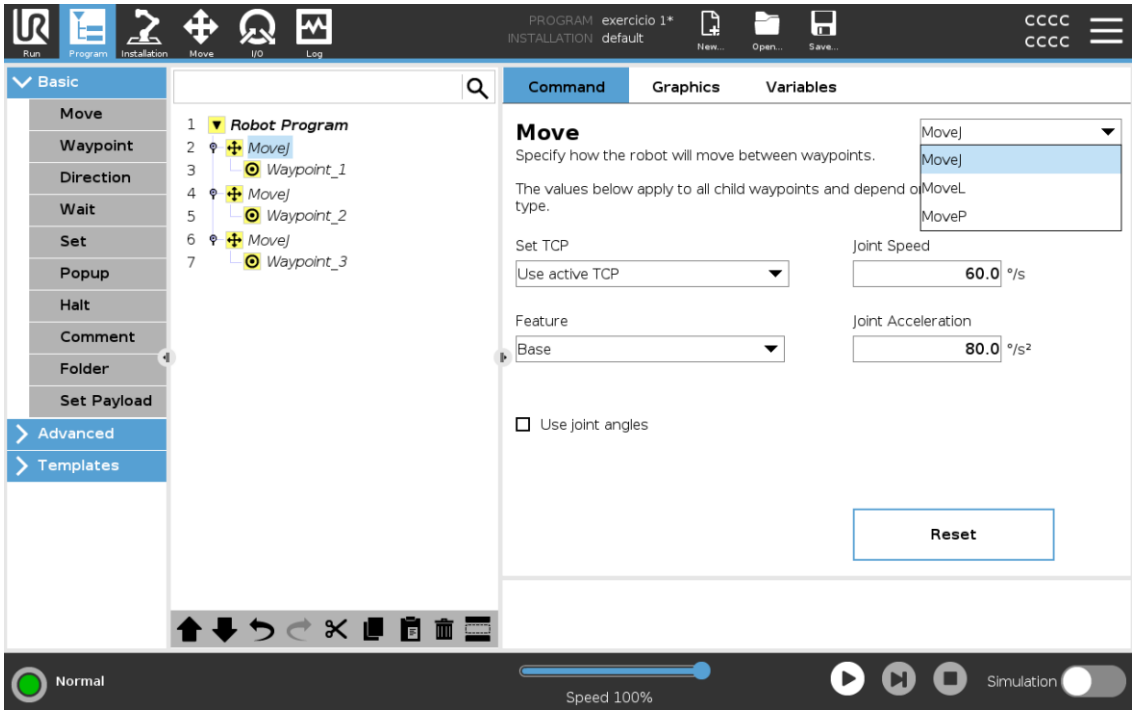


Figura 66 - Alterar o tipo de movimento (*MoveJ, MoveL* e *MoveP*)

De seguida, definir os movimentos como ***MoveJ, MoveL*** e ***MoveP***, respetivamente (Figura 67).

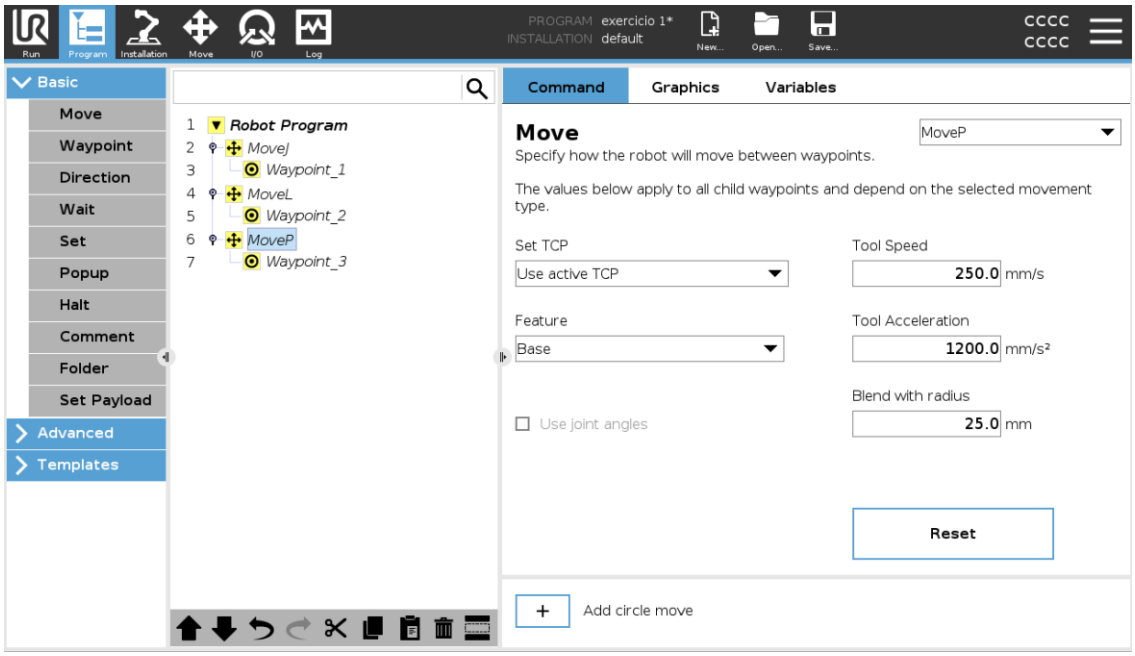


Figura 67 - Ramificação pretendida

Passo 3 – Configuração dos *Waypoints*

Cada instrução *Move* deve conter um *Waypoint* distinto. Para configurar o ***Waypoint_1***, seleciona-se a opção ***Set Waypoint*** (Figura 68).

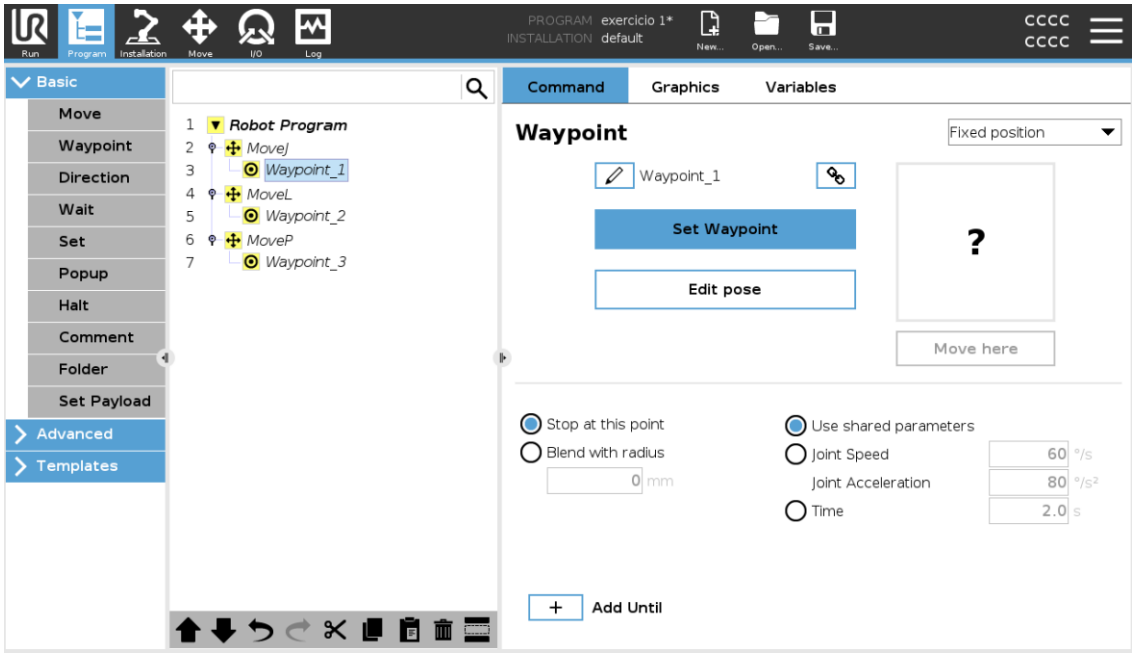


Figura 68 – Configuração do *Waypoint*

Com o modo ***Freedrive*** ativo (pressionar e manter pressionado), posicionar o robô no local pretendido e selecionar ***Ok*** para guardar as coordenadas do ponto (Figura 69). Repetir o procedimento para os *Waypoints 2* e *3*. O movimento do robô pode também ser feito pelas setas existentes no ecrã da interface.

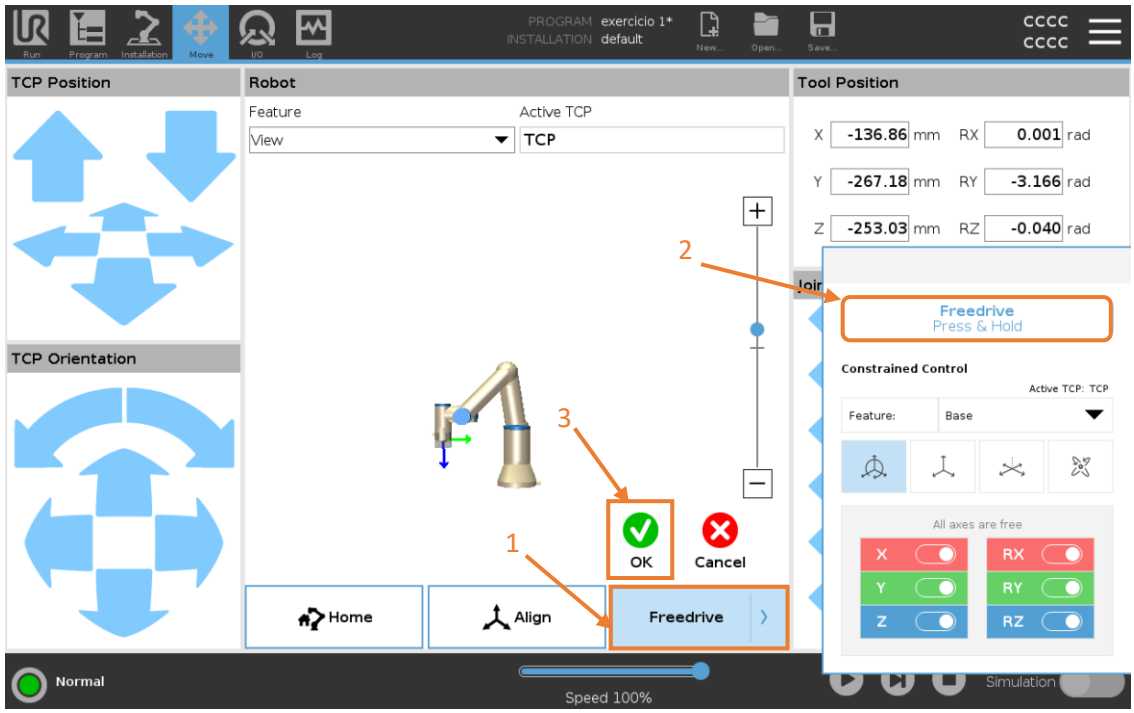


Figura 69 - Posicionar o robô para configurar o *Waypoint*

Passo 4 – Renomeação dos *Waypoints*

Através do ícone de edição que se encontra rodeado a laranja na Figura 70, alterar os nomes dos *Waypoints* para *ponto_1*, *ponto_2* e *ponto_3*, respetivamente.

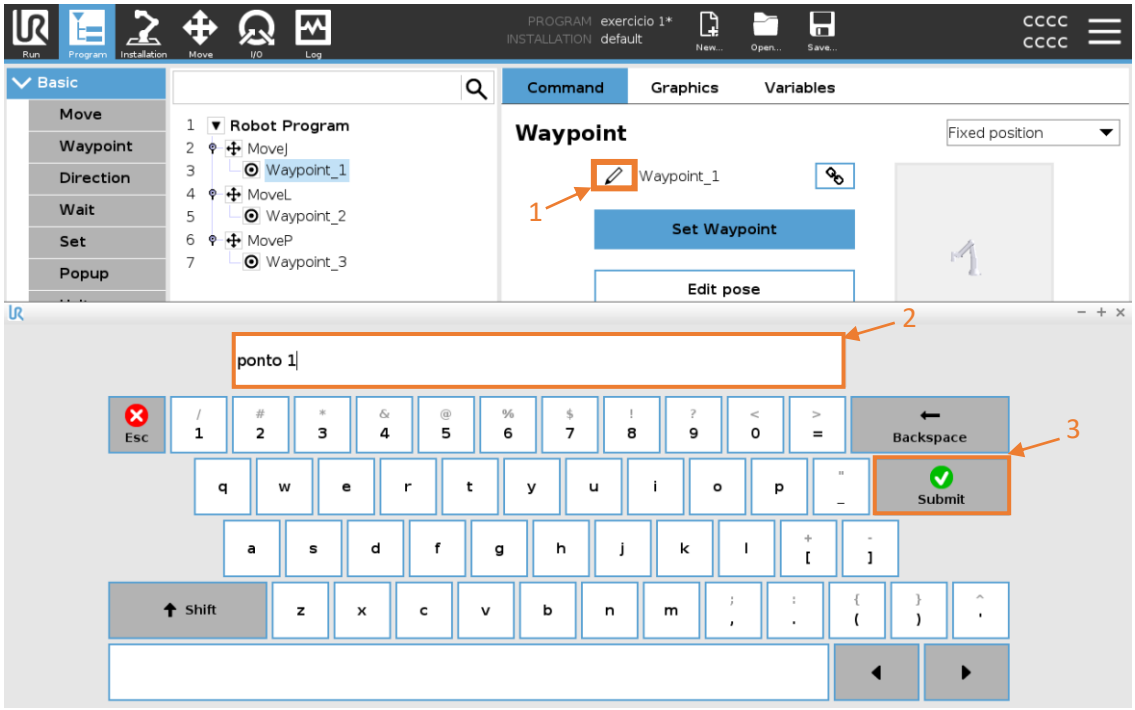


Figura 70 - Renomear o *Waypoint*

Passo 5 – Teste e Validação

Por fim, testar o programa utilizando os botões indicados na interface (Figura 71, 16 e 17).

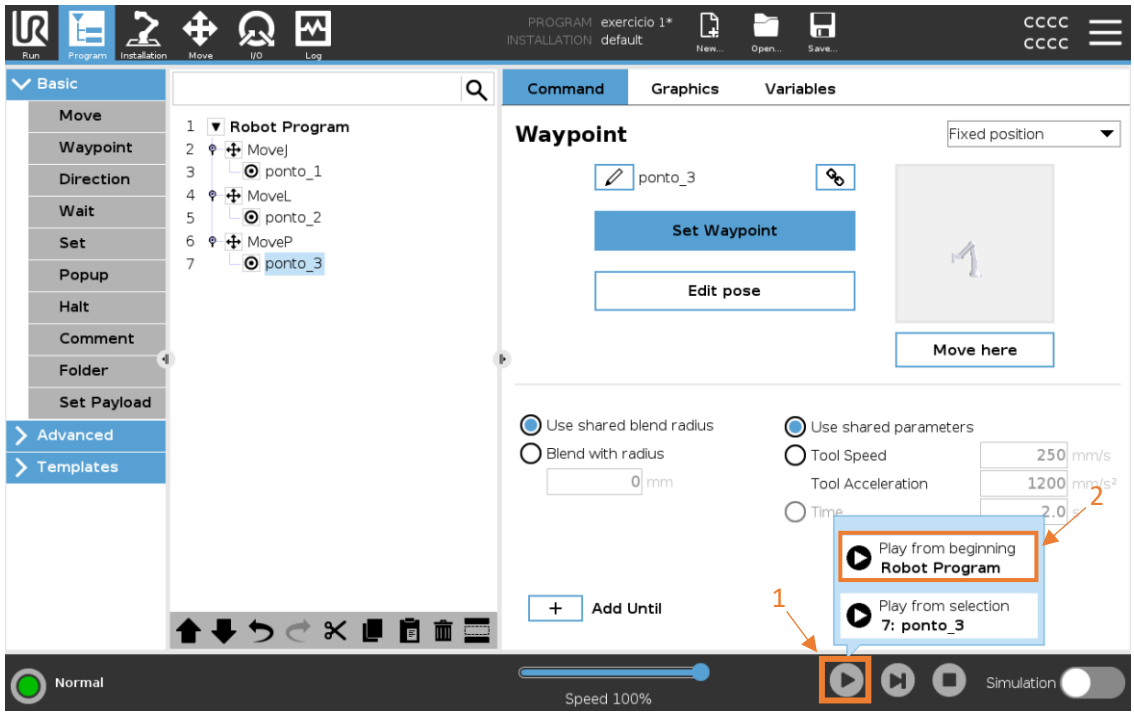


Figura 71 - Solução final e validação do exercício (1/3)

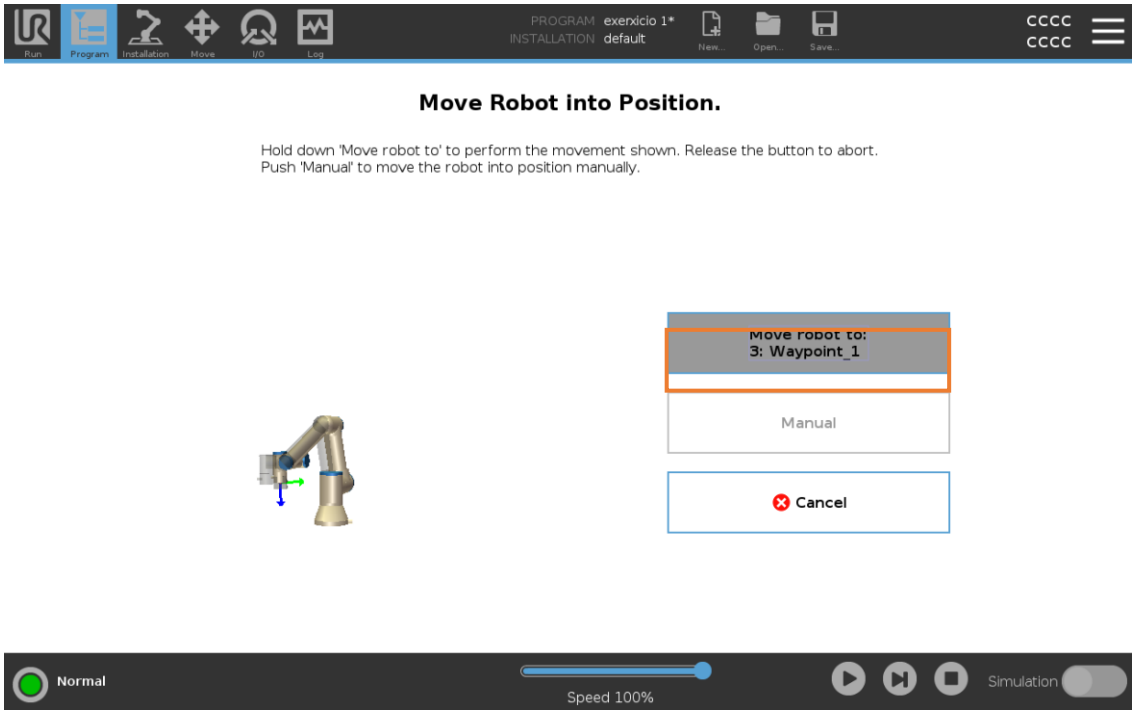


Figura 72 - Solução final e validação do exercício (2/3)

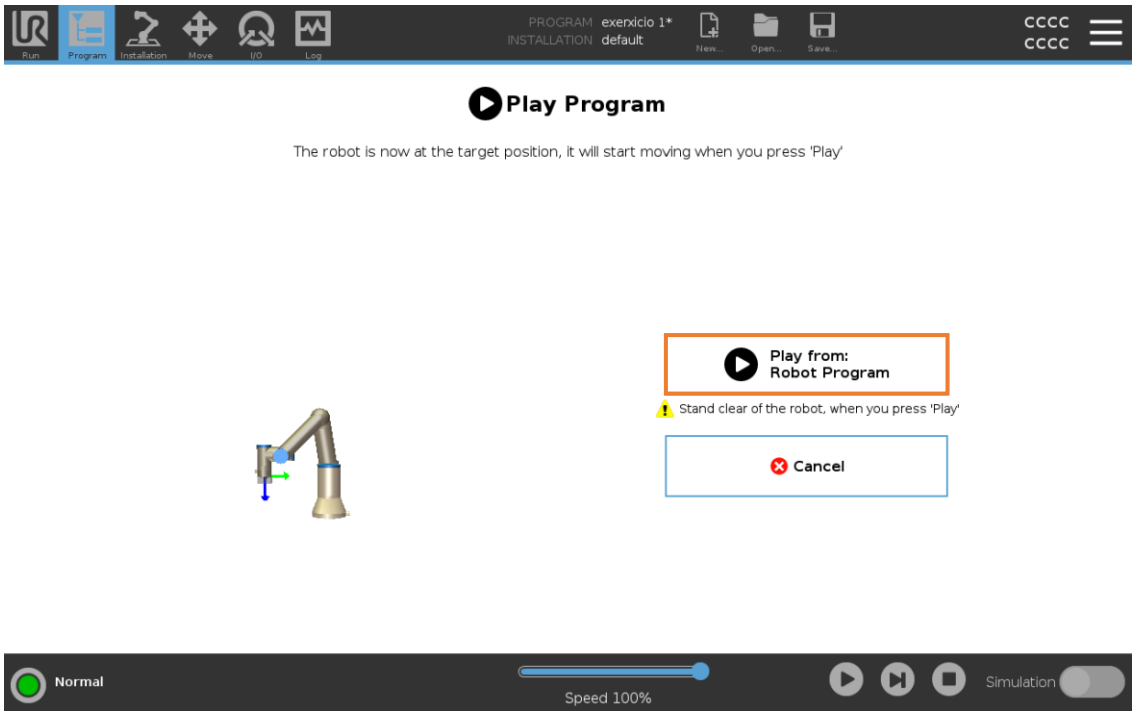


Figura 73 - Solução final e validação do exercício (3/3)

Para terminar o programa deve clicar no botão de parar, representado por um quadrado e assinalado na Figura 74.

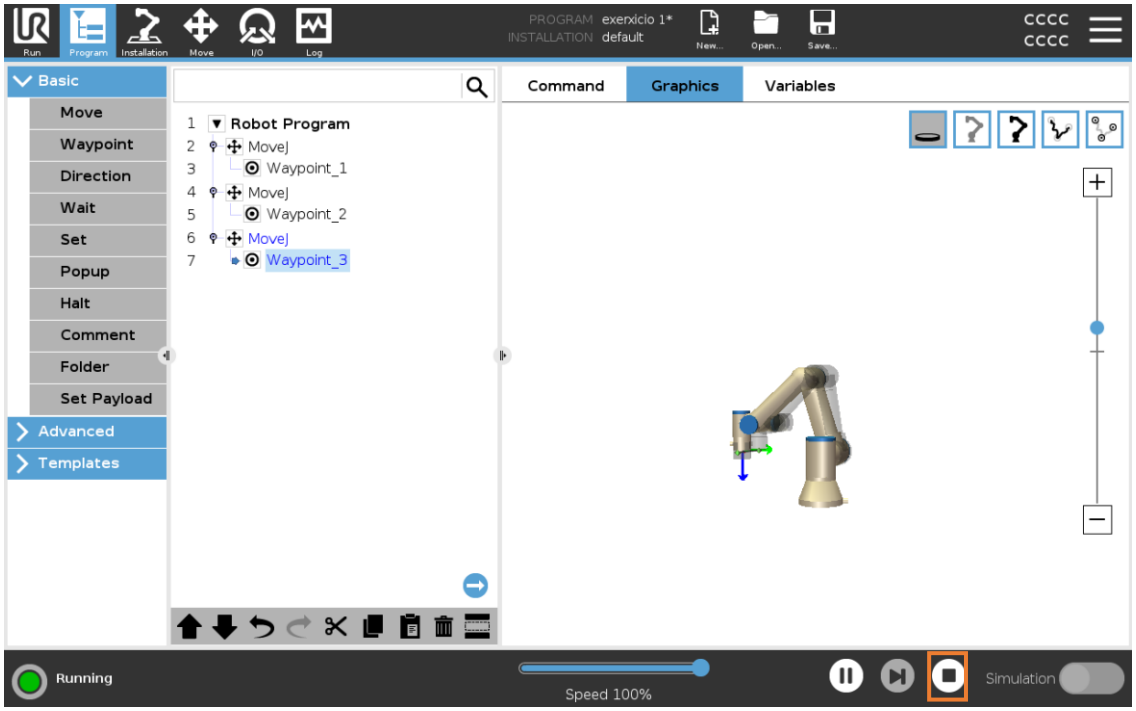


Figura 74 - Botão de parar

Exercício 2 - Integração de I/O com ciclo condicional e movimentos

Este exercício introduz o conceito de interação com o sistema de I/O (entradas/saídas digitais). Pretende-se que o robô reaja a uma entrada digital externa (DI1), ou seja, só pode executar o movimento quando o botão verde (DI1) é acionado. Caso o botão verde não seja acionado, o robô deve manter a posição.

Requisitos do exercício:

- Inserir uma secção de lógica condicional que verifique o estado da entrada digital:
  - Se *DI1 = True*, o programa realiza os movimentos definidos pelo utilizador;
  - Se *DI1 = False*, o programa espera até ser verdadeiro.

Metodologia de resolução

Passo 1 – Criação de um novo programa

Para criar um novo programa deve-se selecionar o ícone *New* rodeado a laranja na Figura 75. De seguida, selecionar *Program* (ver Figura 76).

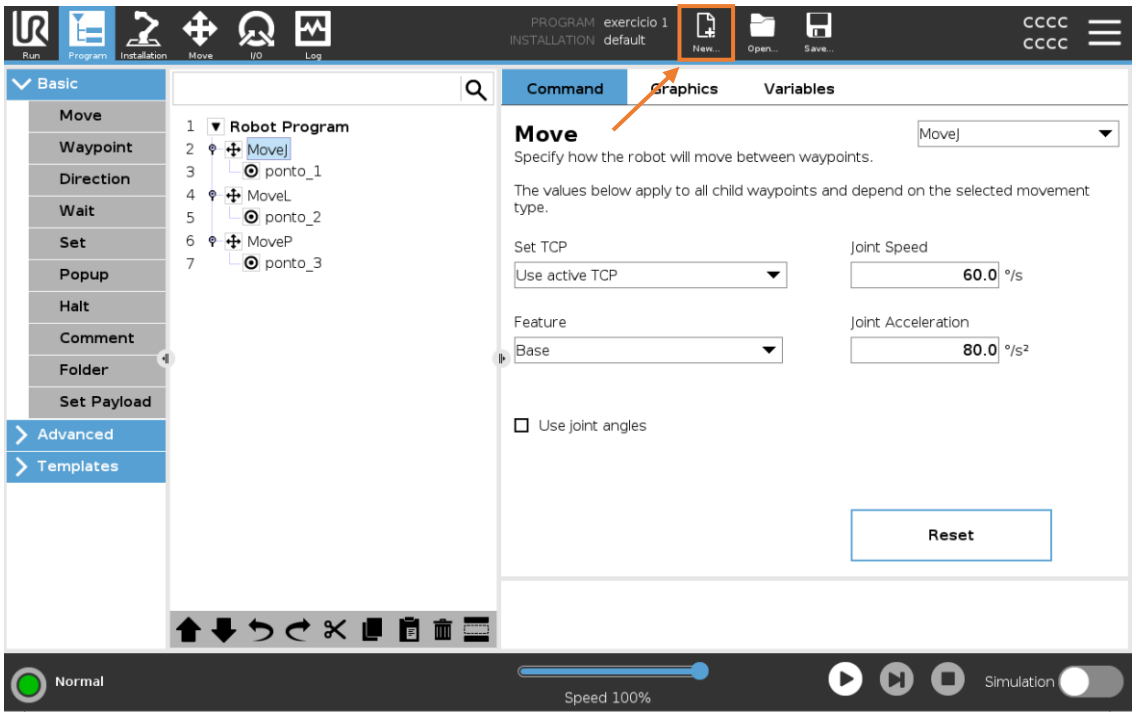


Figura 75 - Criação de um novo programa (1/2)

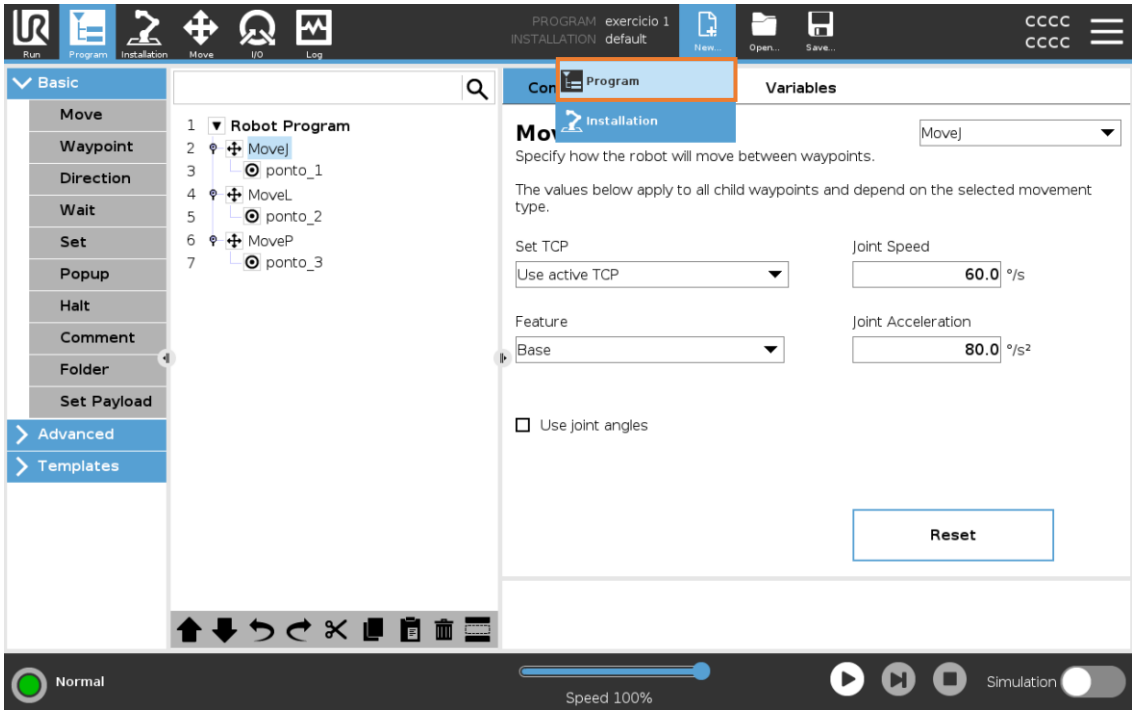


Figura 76 - Criação de um novo programa (2/2)

Por fim, guardar o programa com o nome "*exercicio 2*".

Passo 2 – Criação da Estrutura Condicional

Com a ramificação *empty* selecionada, aceder às instruções avançadas (*Advanced*) do *PolyScope* e inserir uma instrução *If*, conforme ilustrado na Figura 77.

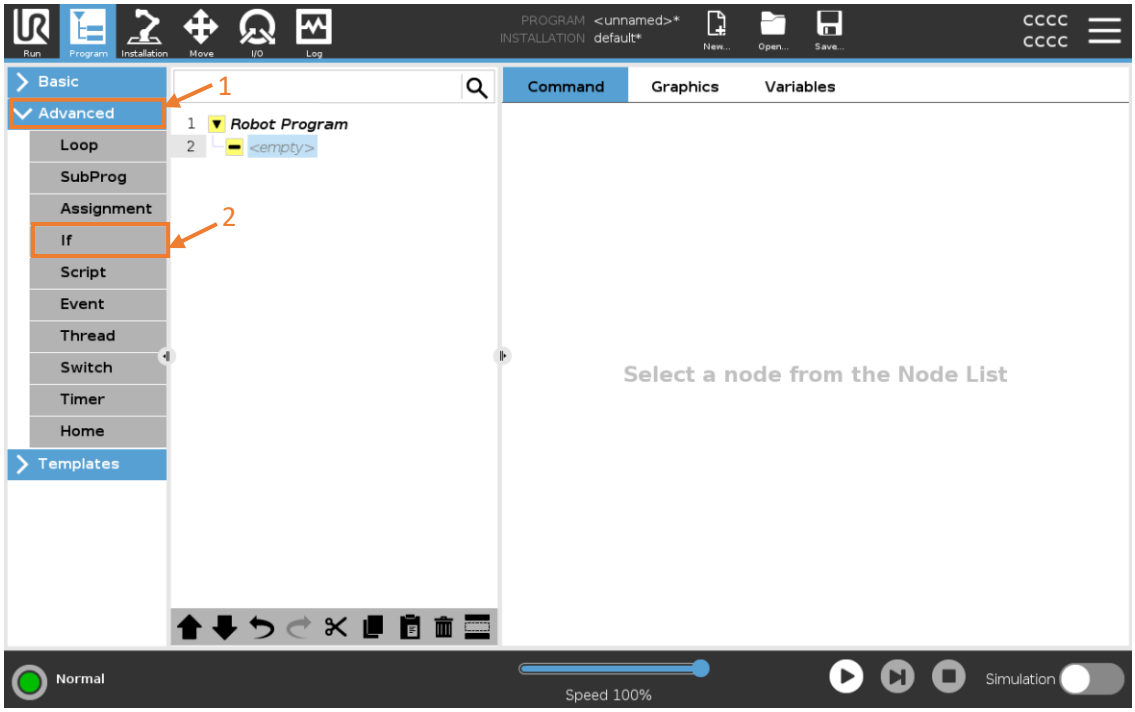


Figura 77 - Adicionar a instrução *If*

De seguida, para configurar a condição lógica seleciona-se o campo amarelo como indicado na Figura 78.

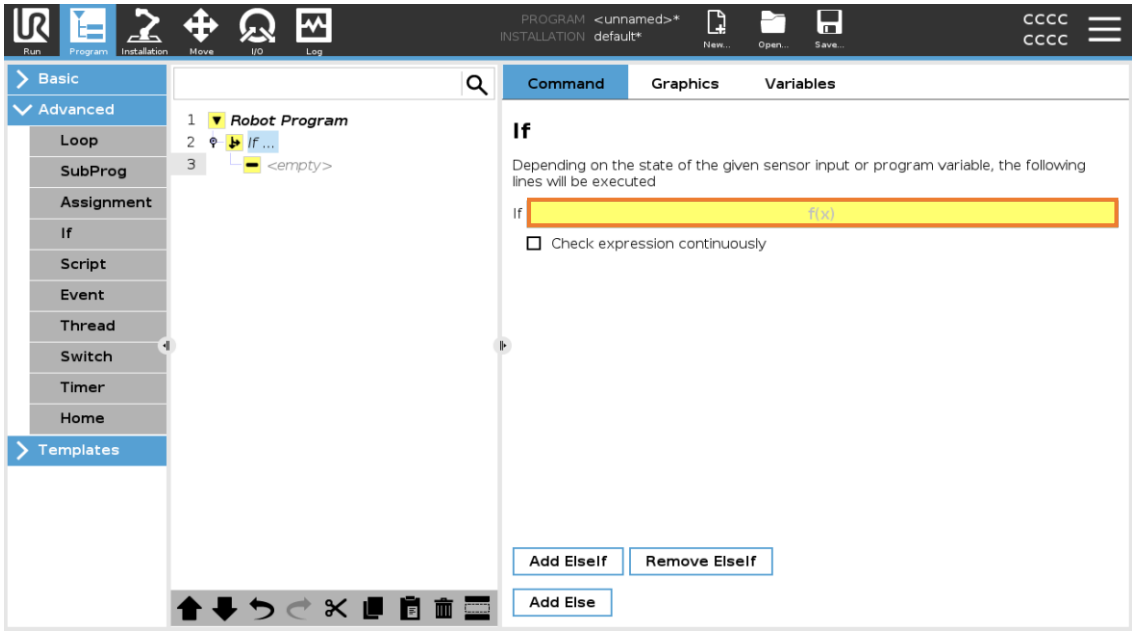


Figura 78 - Local de configuração da expressão *If*

Passo 2 – Definição da Condição *If / ElseIf*

A condição que se pretende programar visa verificar o estado da entrada digital 1 (***digital_in[1]***), associada ao botão verde. A expressão da condição deve verificar se essa entrada (*input*) está em estado ***High*** (ativa ou verdadeira). Para isso, deve-se preencher o campo com a expressão *digital_in[1] = True(HI)* e confirmar com ***Submit*** (Figura 79).

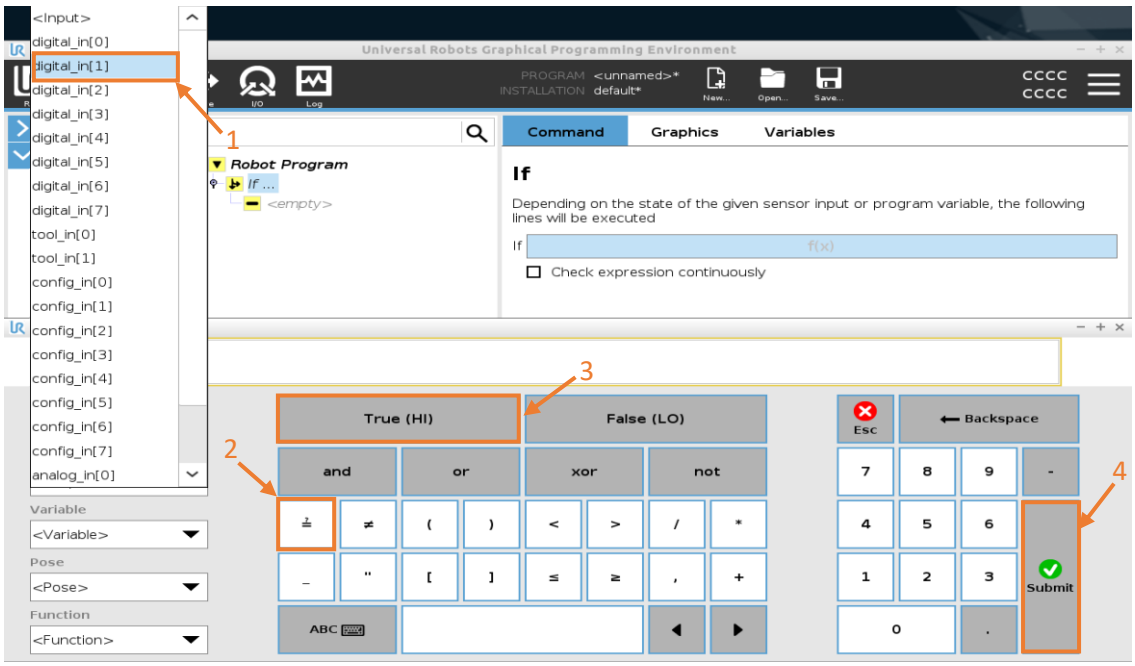


Figura 79 - Configuração da condição *If*

Existe ainda outra condição que tem de ser verificada. Enquanto o botão verde não é ativado, o robô espera. Neste sentido, é necessário adicionar um *ElseIf*. Para realizar esta ação, selecionar *Add ElseIf* conforme mostra na Figura 80.

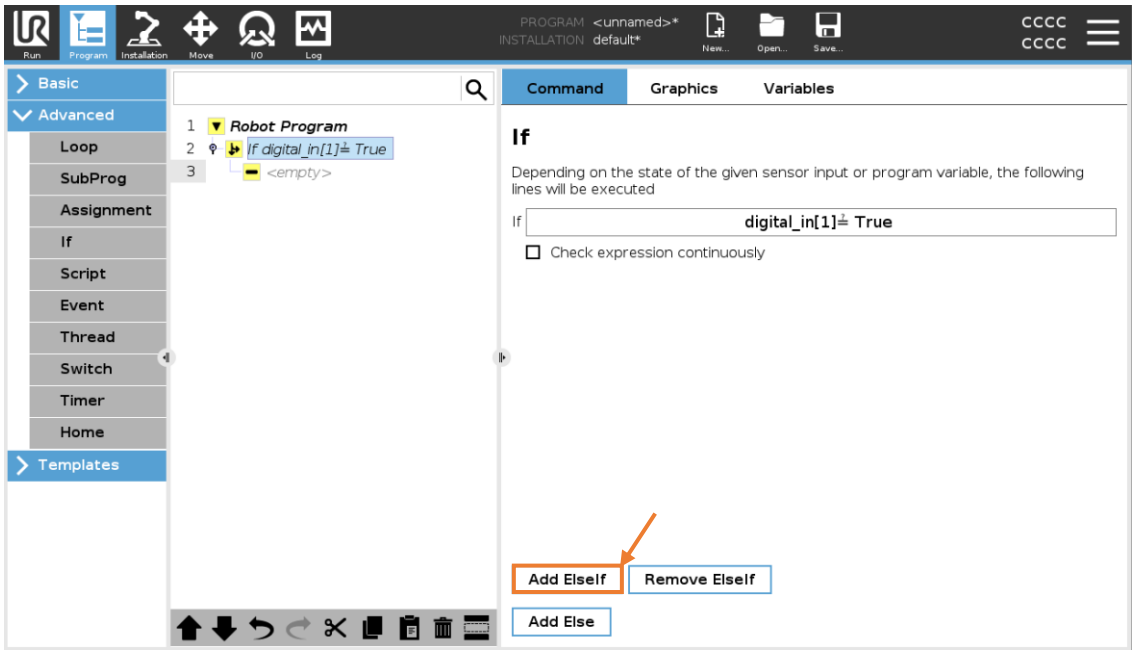


Figura 80 – Inserção da condição *ElseIf*

Neste caso, o campo de expressão é preenchido exatamente da mesma forma que o do *If*, com a exceção de que é necessário verificar se a entrada digital 1 (botão verde) não é acionada. Inserir *digital_in[1] = False (LO)* (Figura 81).

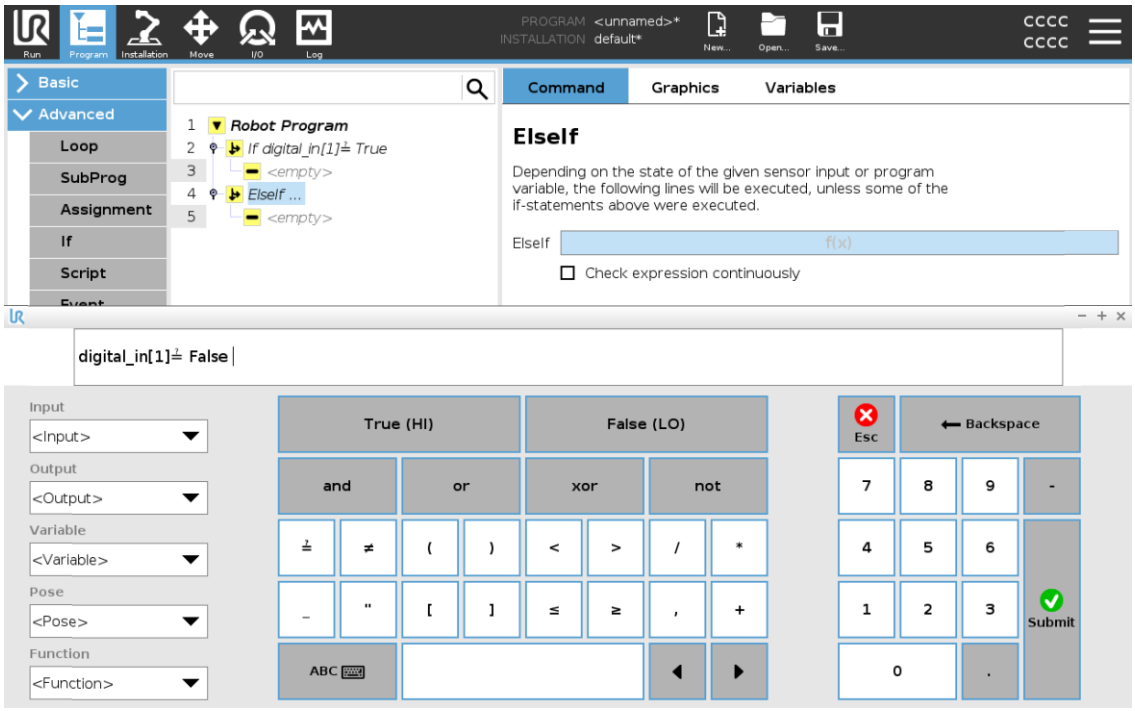


Figura 81 – Expressão para o estado inativo

Passo 3 – Comportamento Condicional do Robô

Consoante o estado da entrada digital, definem-se os seguintes comportamentos:

- **Se DI1 estiver ativa (*True*):** O robô executa duas ou mais instruções de movimento previamente definidas. Devem ser inseridas as instruções ***Move*** com os respetivos *Waypoints*, na secção *empty* da condição *if*.
- **Se DI1 estiver inativa (*False*):** O robô permanece em espera. Para isso, insere-se uma instrução ***Wait*** (localizada em *Basic*), configurando-a para aguardar pela entrada digital 1 em estado *High*. Para configurar a instrução, é necessário selecionar a opção ***Wait for Digital Input***, seguida da escolha da entrada ***digital_in[1]*** e da condição ***High*** (Figura 82).

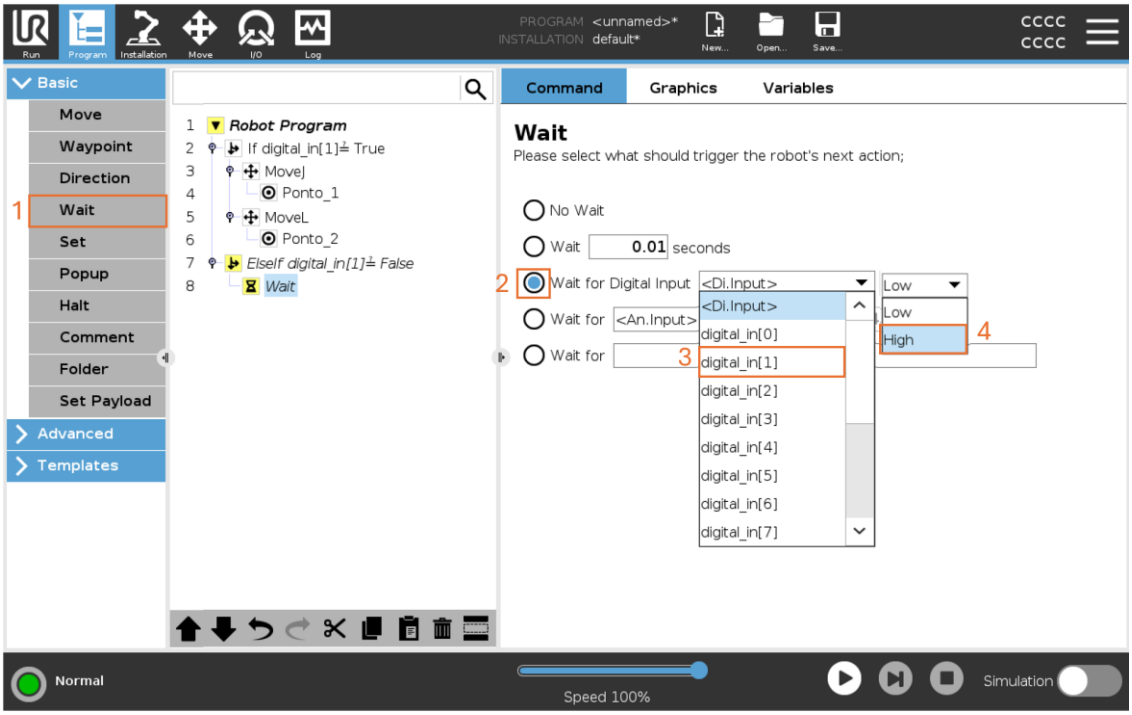


Figura 82 - Configuração da instrução *Wait*

Passo 4 – Teste e Validação

Após a definição da lógica condicional e inserção dos movimentos e tempos de espera, o programa encontra-se pronto para teste (Figura 83).

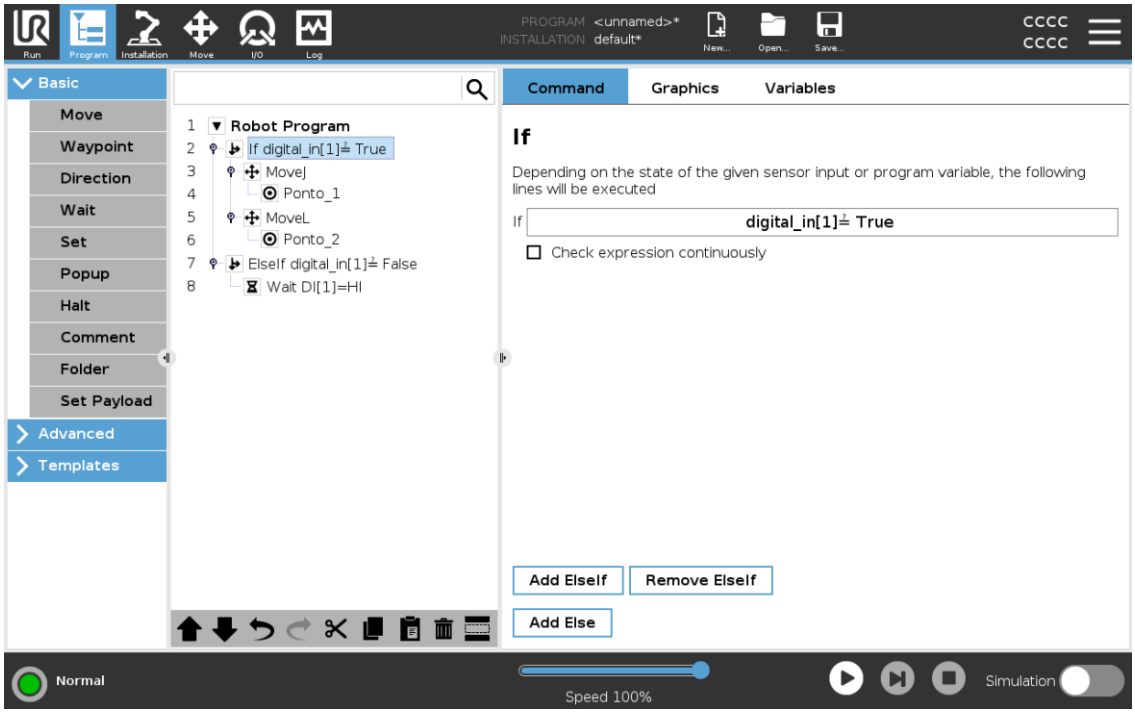


Figura 83 - Solução final e teste do exercício 2

Exercício 3 - Instrução de Mensagem

Este exercício tem como objetivo adicionar uma mensagem de alerta ao programa desenvolvido no exercício anterior (Exercício 3). Enquanto o robô aguarda que o botão verde seja pressionado, deve surgir uma mensagem no ecrã do *Teach Pendant* a alertar o utilizador. A interação deve ser feita através de uma janela de mensagem com o texto: "Pressione o botão verde". Deve ser utilizado o programa desenvolvido no exercício anterior. Proceda com as alterações necessárias.

Requisitos do exercício:

- Inserir uma instrução de mensagem quando *DI1 = False*:
  - Inserir a mensagem "Pressione o botão verde", para alertar o utilizador;
  - O programa continua à espera até que *DI1 = True*.

Metodologia de resolução

Passo 1 – Inserção da instrução *Popup*

Mantendo a lógica do exercício anterior, acrescenta-se uma instrução ***Popup*** na secção *ElseIf* (Figura 84).

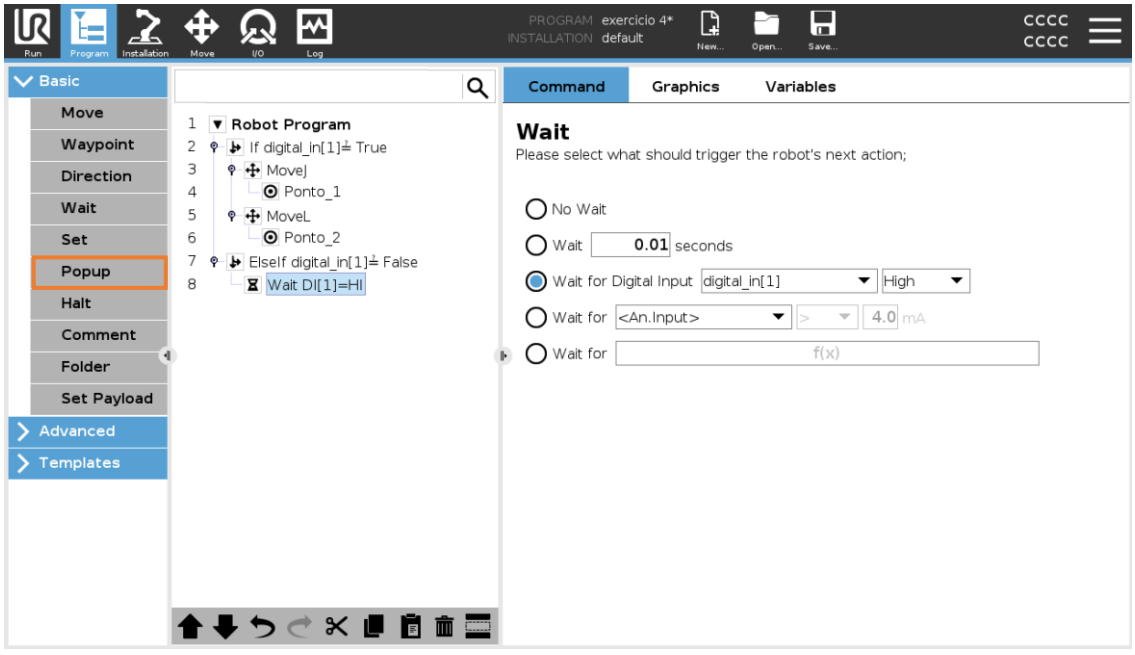


Figura 84 – Inserção da instrução *Popup*

Passo 2 – Reposicionamento da instrução

Como se pode verificar a instrução *Popup* ficou inserida depois da instrução *Wait*. Pode-se alterar o seu posicionamento clicando nas setas assinaladas a laranja na Figura 85. Ao clicar na seta que tem sentido ascendente, a instrução sobe. Ao clicar na seta que tem sentido

descendente a instrução desce. Neste caso pretende-se que a **primeira ação** seja ativar a mensagem enquanto o botão verde não é pressionado.

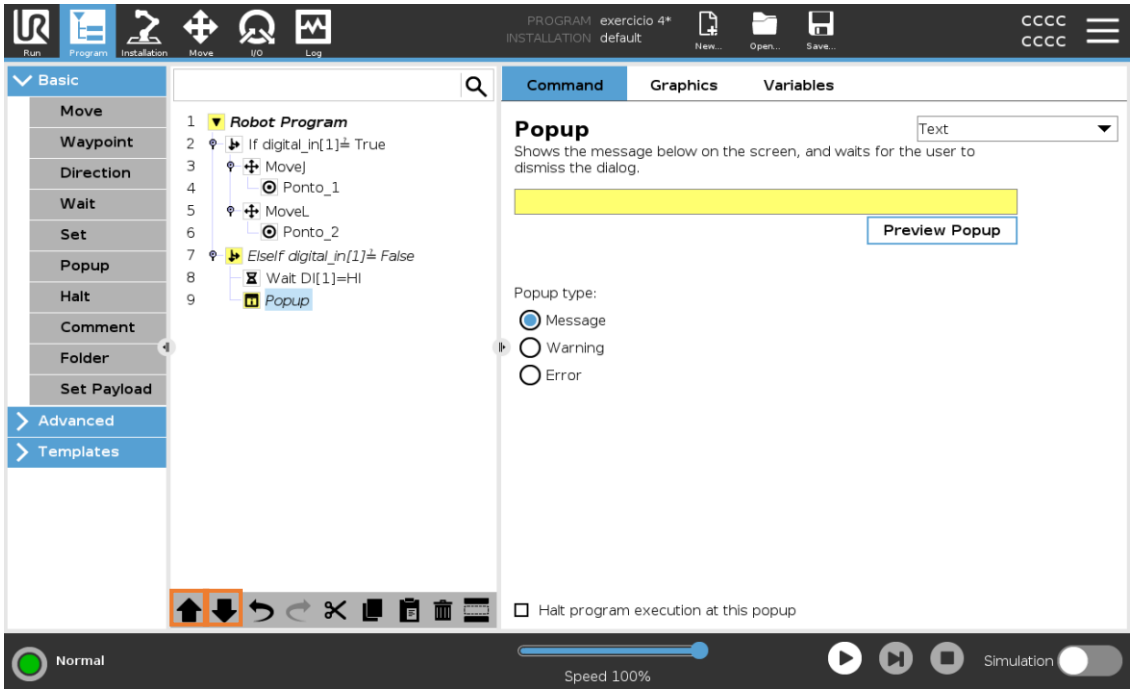


Figura 85 - Alteração do posicionamento de instruções

Passo 3 – Definição da mensagem

No comando da instrução verificar se o tipo de *Popup* selecionado é o *Message* (Figura 86).

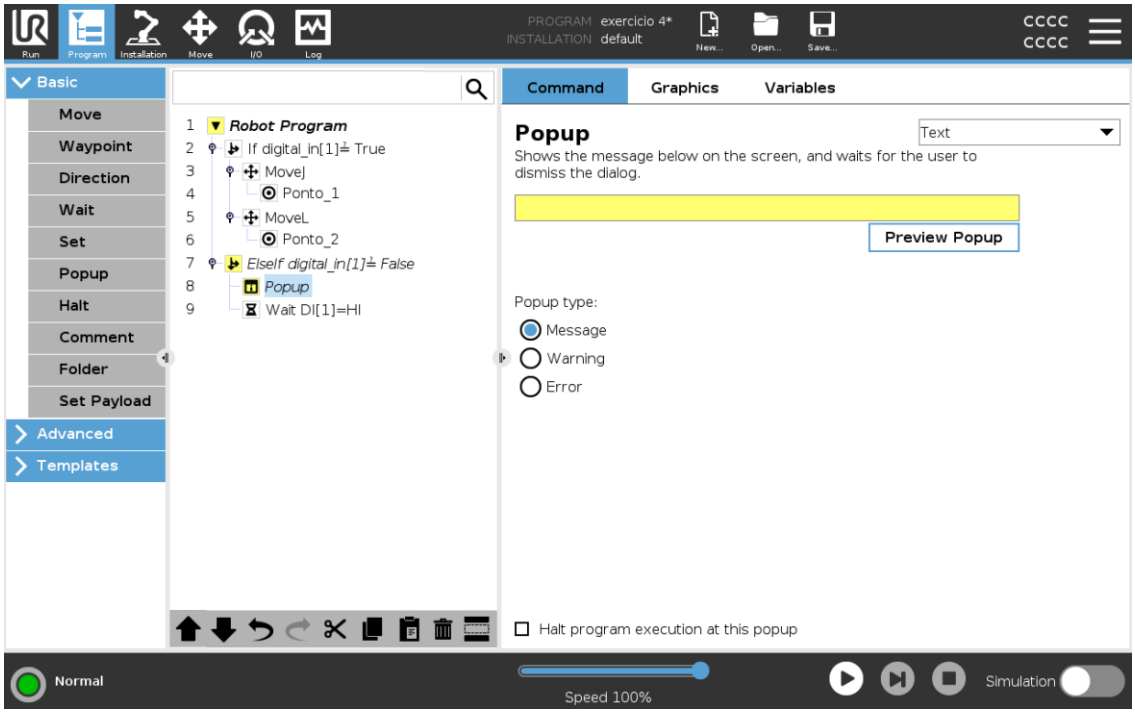


Figura 86 - Instrução *Popup*

De seguida, inserir a mensagem “Pressione o botão verde” no campo de texto correspondente (Figura 87) e validar com *Submit*.

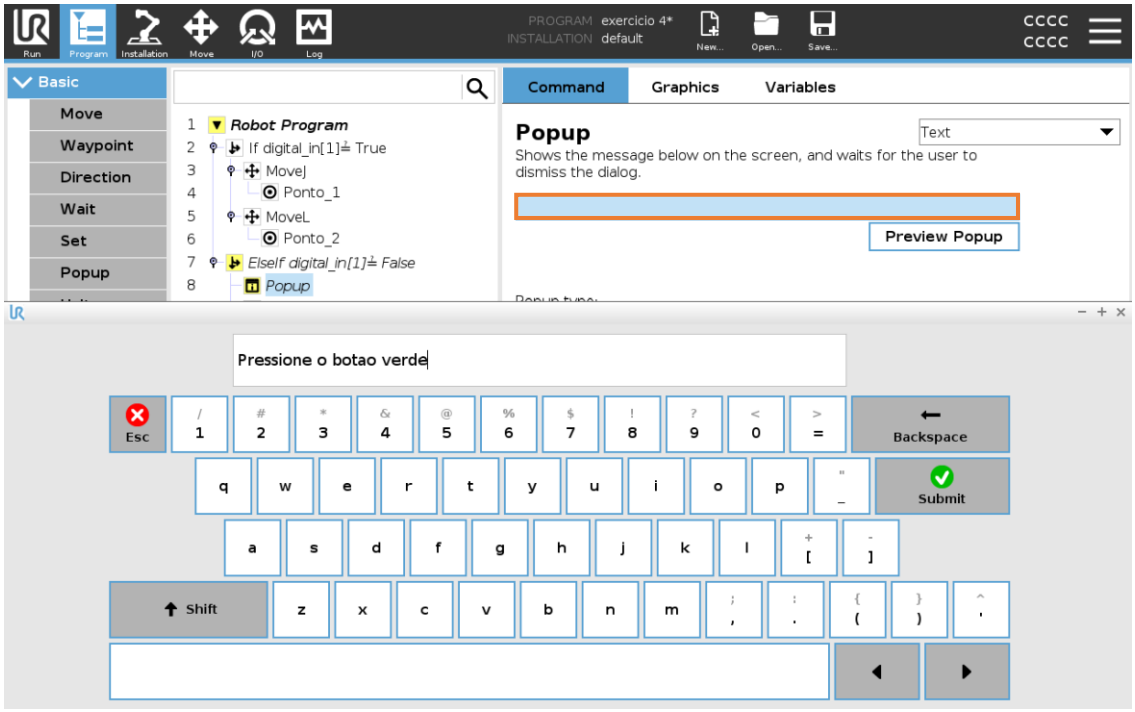


Figura 87 - Configuração da mensagem no *Popup*

Passo 4 – Teste e Validação

Testar o programa e verificar se a mensagem exibida está conforme a Figura 88.

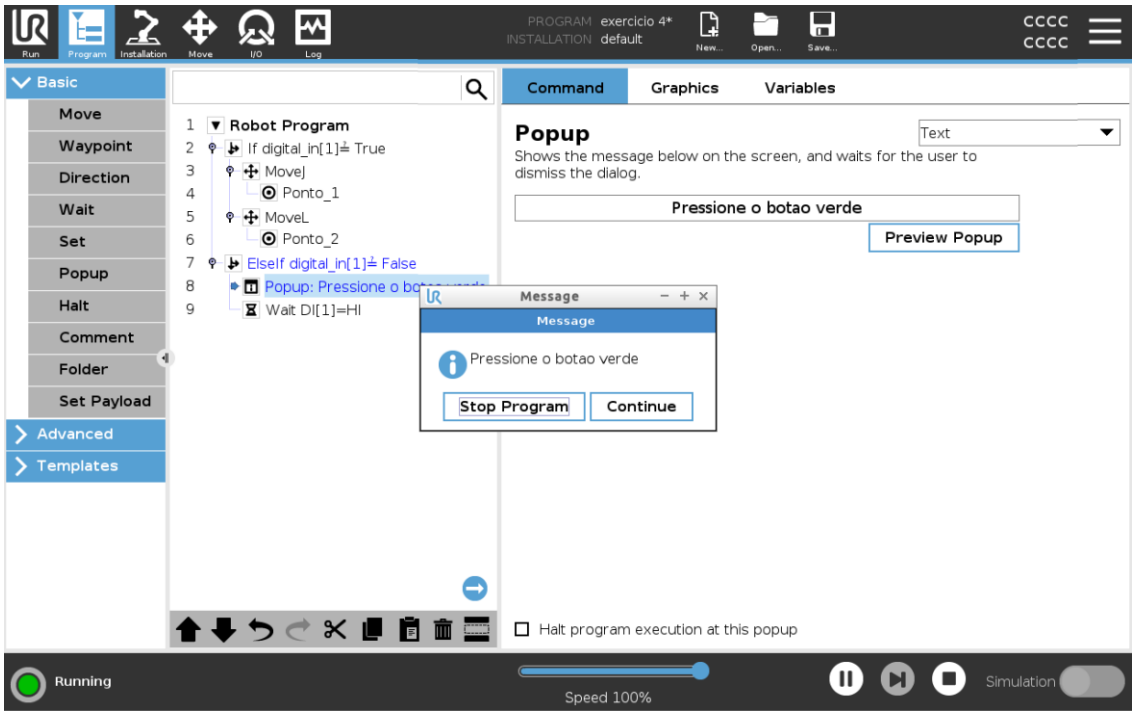


Figura 88 - Solução final e validação do exercício 3

Exercício 4 – Interação com saídas digitais

Pretende-se acrescentar a saída digital (DO0) – *LED* verde ao exercício 3. Utilize o programa anterior e ative o *LED* verde quando o botão verde for pressionado. Proceda com as alterações necessárias.

Requisitos do exercício:

- Se *DI1 = True*, ativar o *LED* verde (DO0) como primeira ação.

Metodologia de resolução

Passo 1 – Inserção da Instrução Set

No interior do bloco ***If*** e com a primeira instrução ***Move*** selecionada, inserir uma instrução ***Set*** para ativar a saída digital (Figura 89).

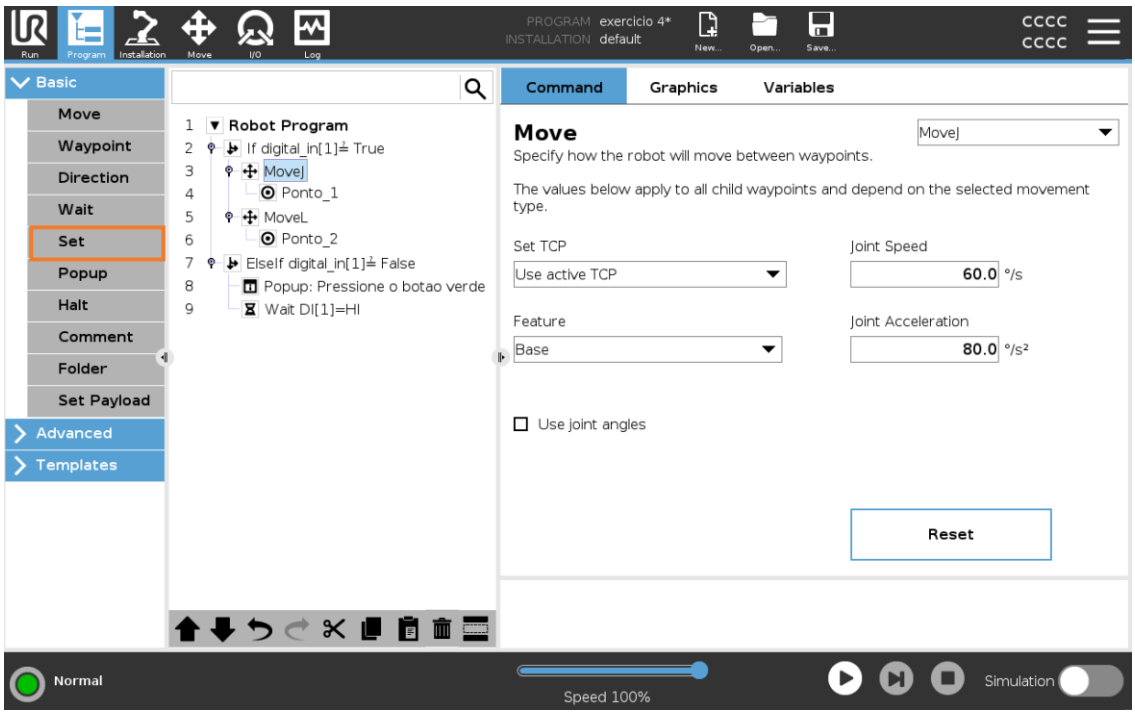


Figura 89 – Inserção da instrução *Set*

Pretende-se que a **primeira ação** a realizar seja o acionamento do Led verde. Para isso deve posicionar corretamente a instrução *Set* (Figura 90).

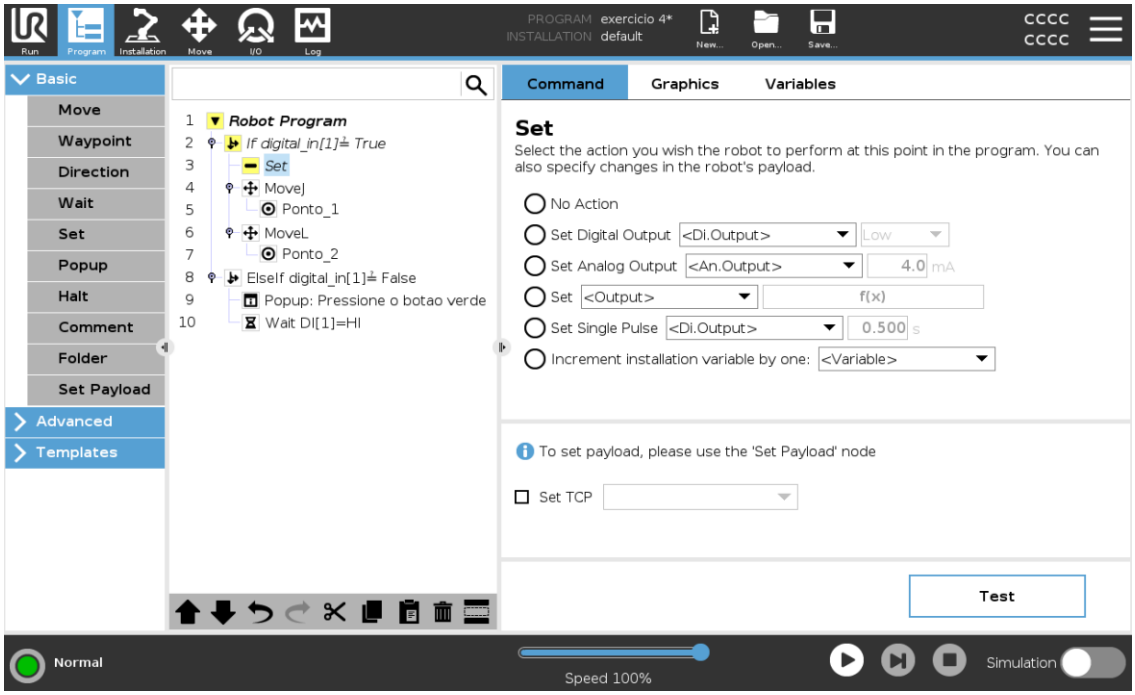


Figura 90 – Posicionamento da instrução *Set*

Passo 2 – Configuração da saída digital

O objetivo é ativar o *LED* verde, definir *Set Digital Output* > *digital_out[0]* > *High* (Figura 91).

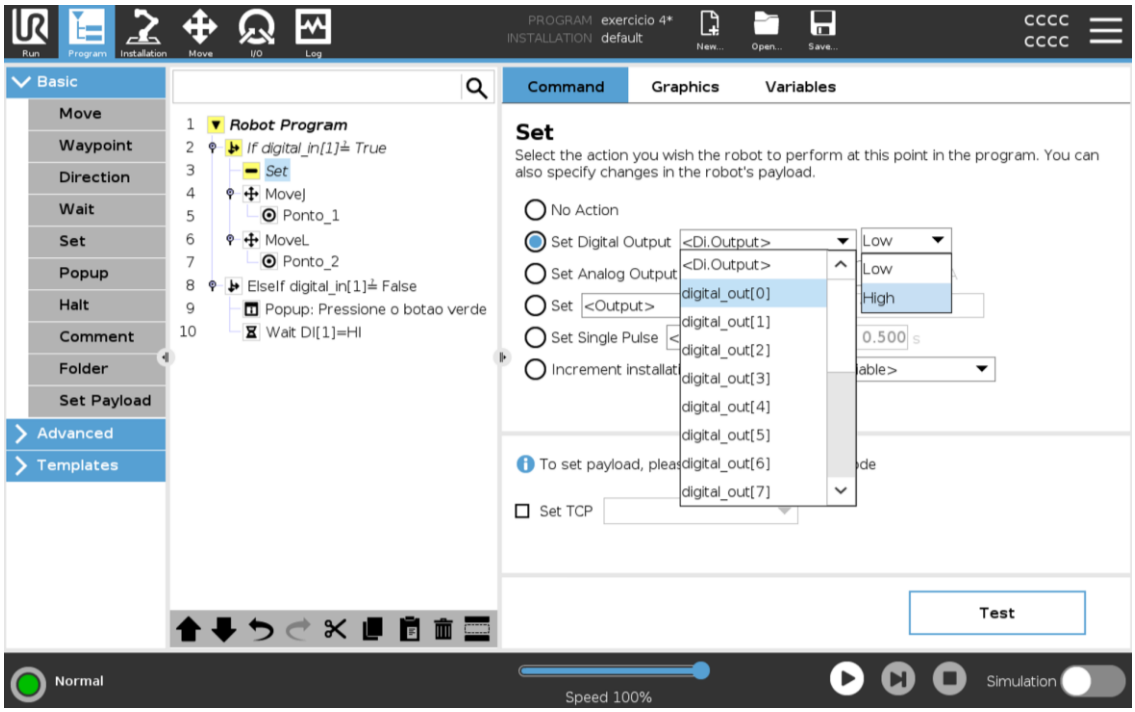


Figura 91 - Configuração da instrução *Set* para ativar o *LED*

Passo 3 – Desativação da saída digital (*Before Start*)

Para evitar que a saída permaneça ativa após o fim do programa, deve-se adicionar uma instrução ***Set***, com o estado da saída definido como ***Low***. Com a segunda instrução *Move*

selecionada, inserir a instrução *Set* (Figura 92) e parametrizá-la (Figura 93). Esta ação garante que o *LED* verde é **desligado** no fim do programa.

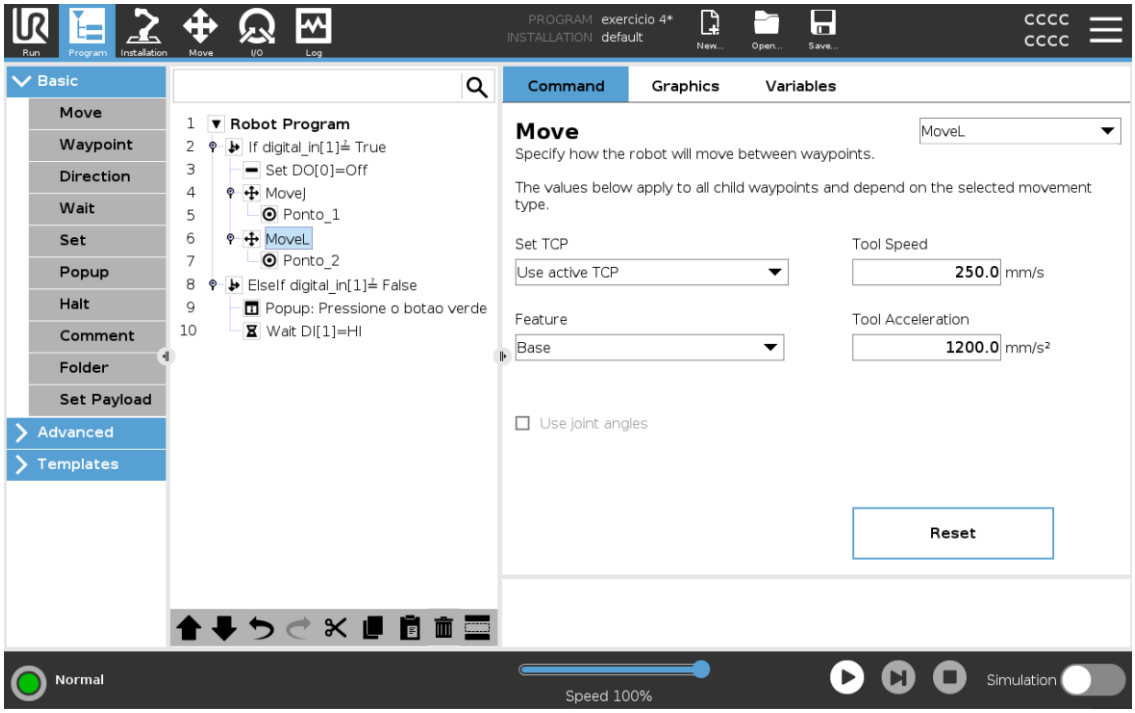


Figura 92 – Inserção da instrução *Set*

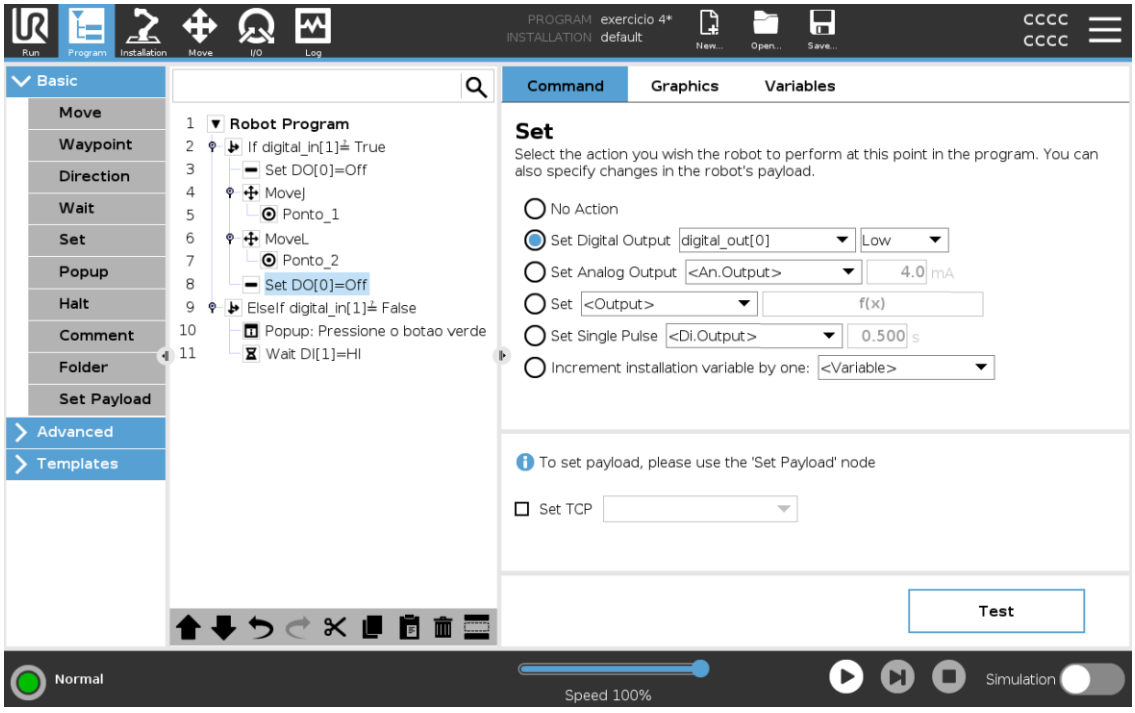


Figura 93 - Parametrização da instrução *Set*

Passo 4 – Teste e Validação

Testar o programa.

Exercício 5 - Ciclo *Pick and Place* com Contador

O objetivo do exercício consiste em desenvolver um programa no *PolyScope* que permita ao robô UR3e executar uma sequência de *Pick and Place*. Na Figura 94, está presente a área de trabalho do robô. Na posição inicial (zona A), encontram-se empilhadas cinco peças, mas apenas a primeira peça deve ser transportada até à posição 1 que se situa na zona B.

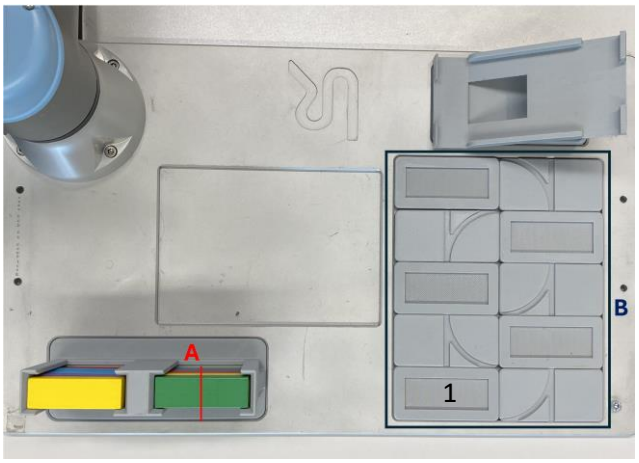


Figura 94 - Área de trabalho do robô

Requisitos do exercício:

1. O robô deve deslocar-se para a posição acima da zona A, que deverá ser definida como *waypoint* *before_start*;
2. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher (definir como *waypoint* *start_point*);
3. Acionar o *gripper* para agarrar a peça;
4. Aguardar 1 segundo;
5. Efetuar o movimento linear inverso até à posição *before_start*;
6. Transportar a peça até um ponto acima da zona B e defini-lo como *before_exit*;
7. Descer em movimento linear até à posição 1 e definir o ponto como *exit_point*;
8. Desativar o *gripper* para largar a peça;
9. Aguardar 1 segundo;
10. Incrementar o contador de peças paletizadas;
11. Após a colocação da 1ª peça, o programa deve terminar automaticamente
12. Durante todo o processo, manter o LED verde (DO0) ligado.

Metodologia de resolução

Passo 1 – Configuração Inicial

Criar um programa denominado "*exercício 5*". Inserir uma secção ***Before Start***.

Para inserir a secção *Before Start*, deve-se ativar a opção *Add Before Start Sequence* situada no comando da secção *Robot Program* (ver Figura 95 e Figura 96). Esta secção permite definir todas as variáveis, condições de segurança e instruções, antes do programa começar.

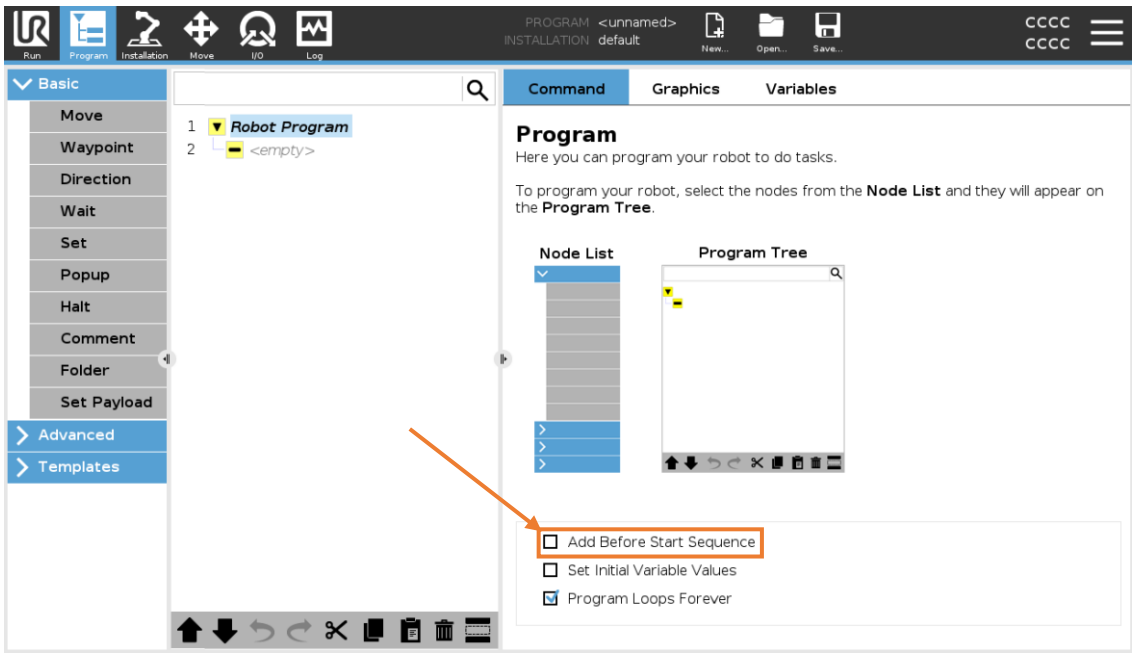


Figura 95 - Inserção da secção *Before Start*

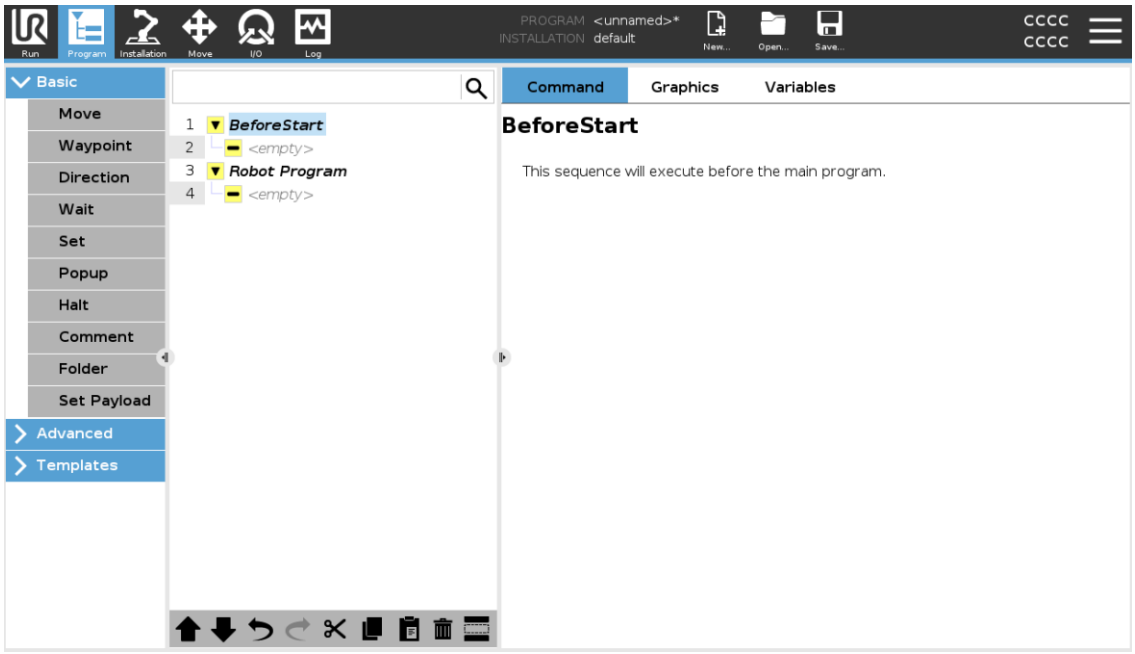


Figura 96 - Secção *Before Start* inserida

Definir as condições iniciais:

- Variável *count* com valor inicial = 0 (Figura 97-43), utilizar a instrução *Assignment*.

- Abertura das garras do *gripper*, através do comando *URCaps > Zimmer > Z_Release* (Figura 100).

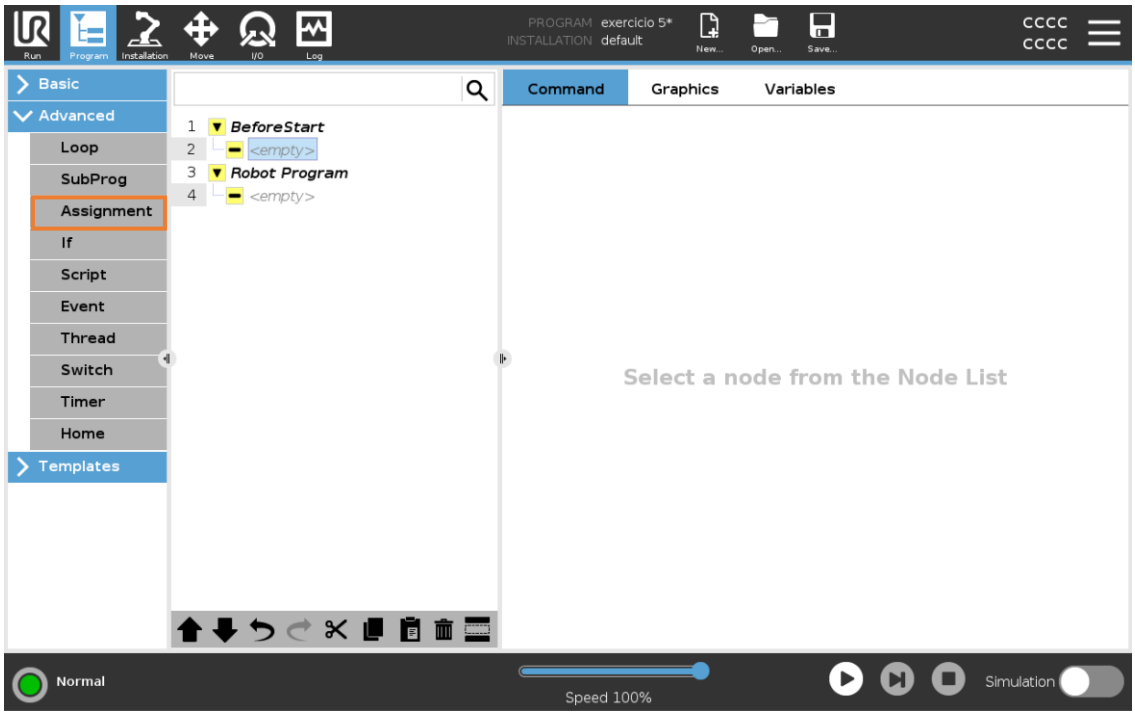


Figura 97 – Inserção da instrução *Assignment*

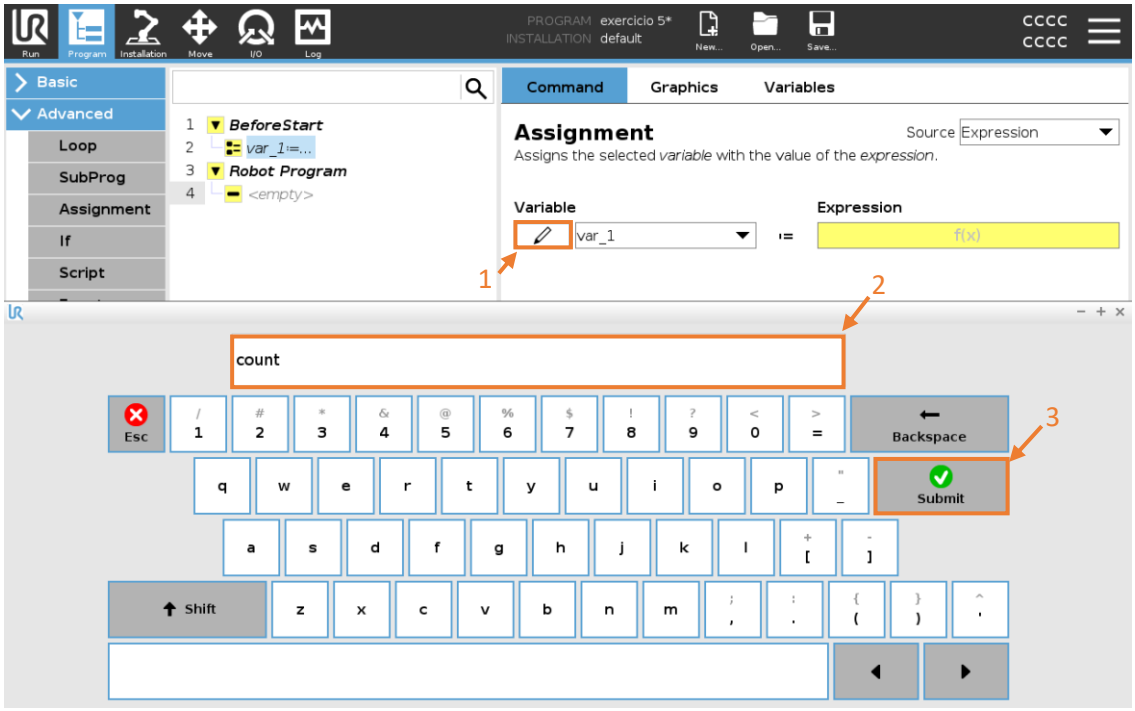


Figura 98 - Configurar a instrução *Assignment*

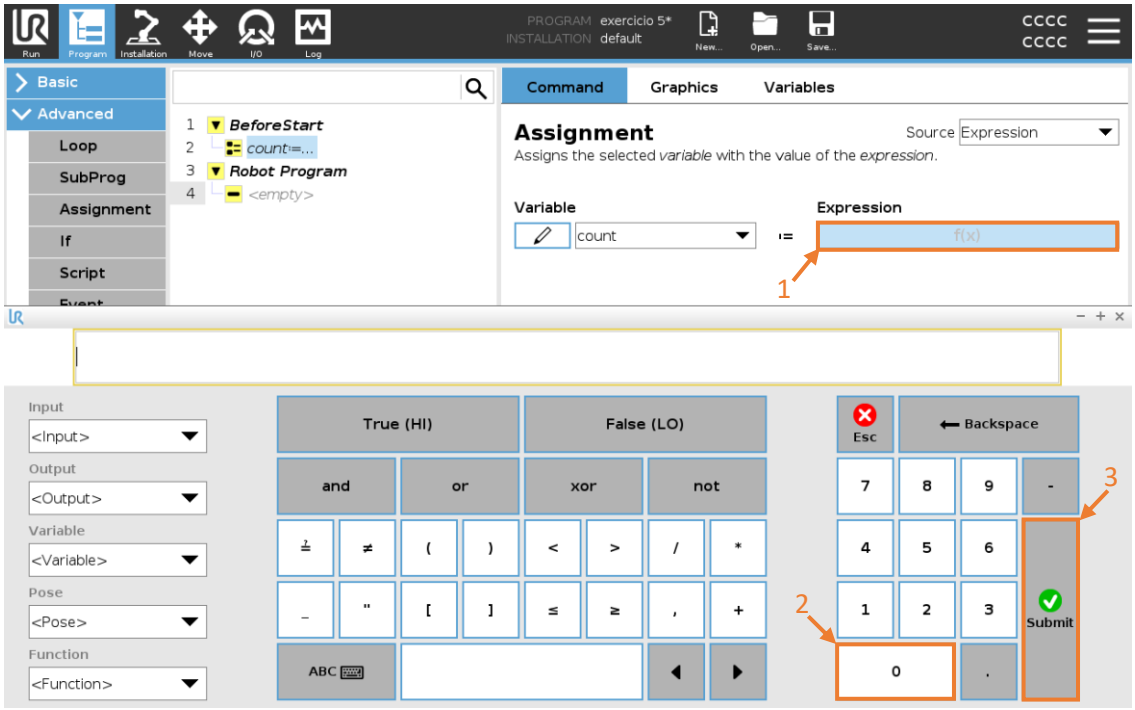


Figura 99 - Definir o valor inicial do contador

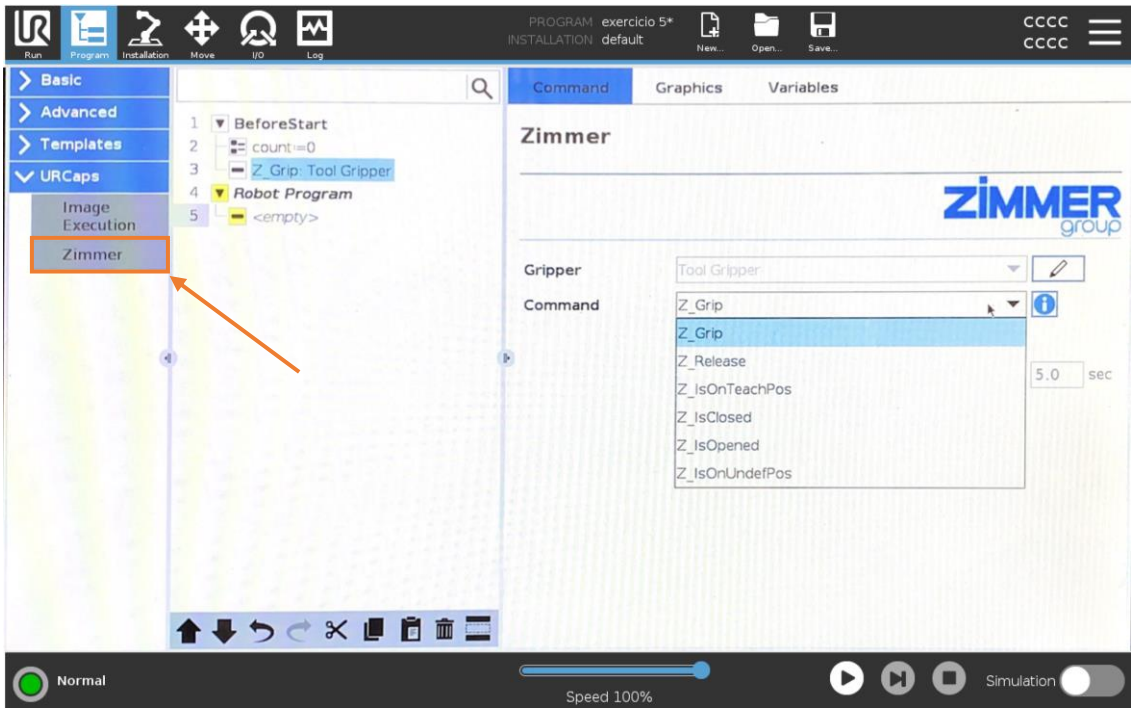


Figura 100 - Comandos do *Gripper*

Passo 2 – Programação do Ciclo *Pick and Place*

O programa principal inicia com uma instrução ***If***, verificando se ***count* < 1**. Numa fase inicial definir a sequência de recolha da peça:

1. Ativar o *LED* verde;

2. Mover o robô para uma posição acima da peça (zona A), posição *before_start*;
3. Mover o robô linearmente até à peça, posição *start_point*;
4. Acionar o Gripper para agarrar a peça, através do comando *URCaps > Zimmer > Z_Grip*.
5. Aguardar 1 segundo.

O objetivo é obter uma ramificação igual à que está presente na Figura 101.

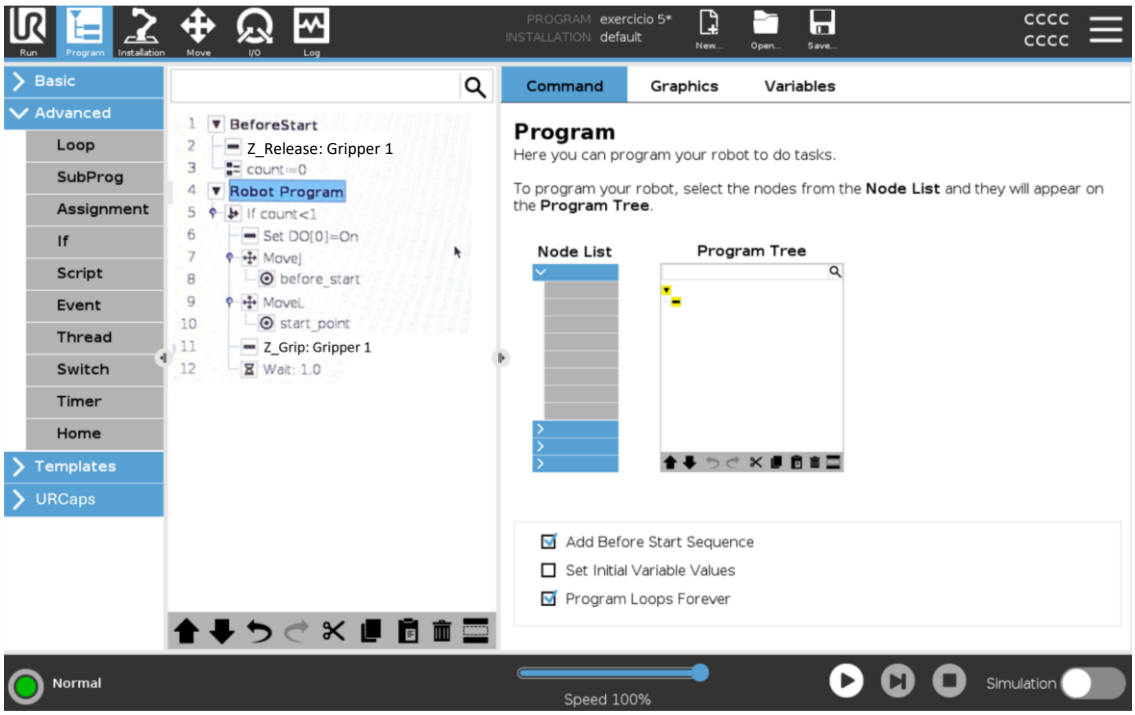


Figura 101 - Sequência de programação 1

De seguida, é necessário definir a sequência de colocação (*Place*) da peça:

1. Mover o robô em movimento linear até a posição *before_start;*
2. Mover o robô até à zona B para depósito, posição *before_exit*;
3. Descer o robô em movimento linear até à posição 1 (*exit_point*);
4. Libertar a peça;
5. Aguardar 1 segundo;
6. Incrementar o contador.

No final obtém-se uma sequência de programação semelhante à que está presente na Figura 102.

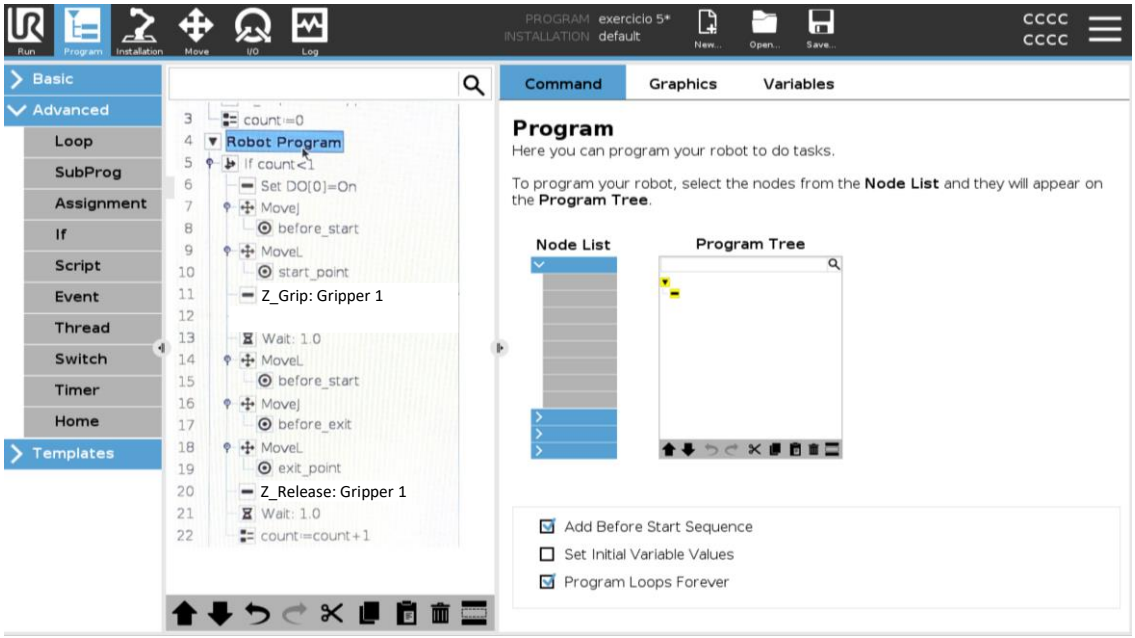


Figura 102 – Conclusão da lógica de movimentação

Passo 3 – Instrução *ElseIf* e Paragem do Programa

Adicionar uma instrução ***ElseIf*** para verificar se ***count = 1*** (Figura 103). Neste caso:

- Mover o robô para a posição *before_start*;
- Desativar o *LED* verde;
- Utilizar a instrução ***Halt*** para parar o programa (a instrução situa-se na secção *Basic*).

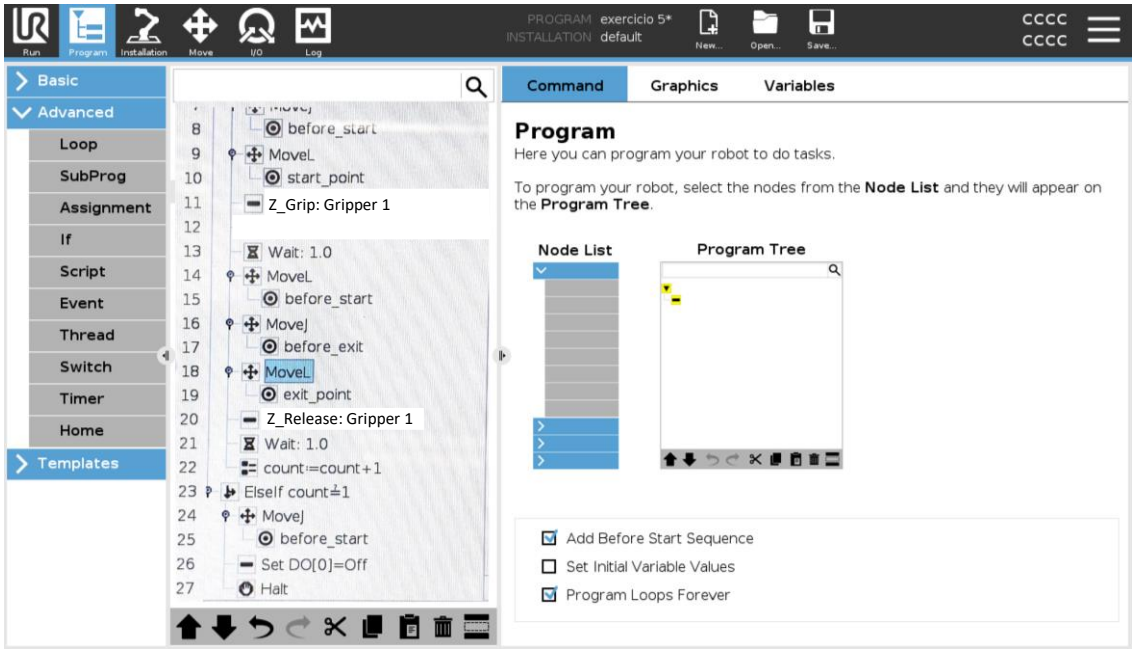


Figura 103 - Instrução *ElseIf* para terminar o programa

Passo 4 – Teste e Validação

Exercício 6 - Paletização de 5 peças

O objetivo deste exercício é desenvolver uma aplicação de paletização automática utilizando o robô colaborativo UR3e. O robô deverá realizar operações de *Pick and Place* para transferir peças da zona de recolha (zona A) para a zona de paletização (zona B), de forma ordenada e repetitiva. A tarefa deverá ser executada até que cinco peças sejam colocadas na zona B (Figura 104), parando automaticamente após a última colocação.

Na Figura 104 encontra-se a representação da superfície de trabalho com as zonas A e B assinaladas. A zona A contém 5 peças empilhadas de diferentes cores. A zona B está subdividida em cinco compartimentos onde as peças deverão ser colocadas.

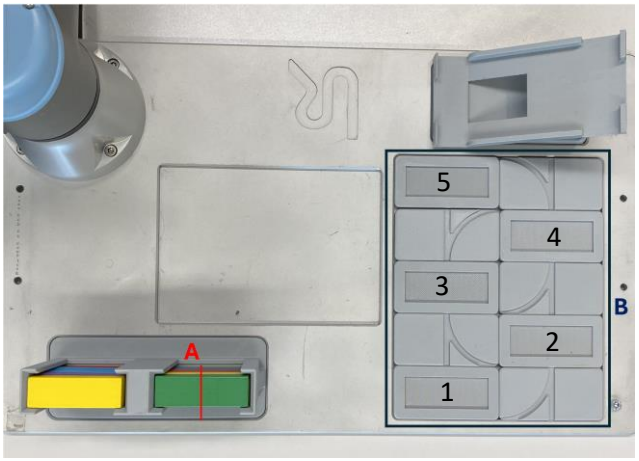


Figura 104 - Sinalização da superfície de trabalho

Requisitos do exercício:

1. A execução do programa deve iniciar-se apenas quando o botão verde (DI1) for pressionado;
2. Após o sinal de início, o robô deve deslocar-se para a posição acima da zona A, que deverá ser definida como *waypoint* *before_start*;
3. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher (definir como *waypoint* *start_point*);
4. Acionar o *gripper* para agarrar a peça;
5. Aguardar 1 segundo;
6. Efetuar o movimento linear inverso até à posição *before_start*;
7. Transportar a peça até à zona B e colocá-la na próxima posição livre (de 1 a 5);
8. Desativar o *gripper* para largar a peça;
9. Aguardar 1 segundo;
10. Incrementar o contador de peças paletizadas;
11. Repetir o ciclo até que cinco peças tenham sido colocadas;
12. Durante todo o processo, manter o LED verde (DO0) ligado;

13. Após a colocação da 5ª peça, o programa deve terminar automaticamente.

Materiais e equipamentos necessários:

- Robô colaborativo UR3e e controlador;
- Interface de programação *PolyScope* para desenvolvimento, execução e monitorização do código;
- *LED* verde – DO0;
- Botões: de 2 posições (DI0) e verde (DI1);
- *Gripper Zimmer*.
- Estrutura física com:
    - zona A: local de recolha de peças;
    - zona B: local de paletização de peças;

Instruções adicionais:

- A espessura/altura das peças a considerar é de **15 mm**.
- A paletização deve seguir a **ordem crescente das posições** na zona B (1 → 5)

Metodologia de resolução 1

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e com o nome *ex_paletização*.

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, definir:

- A variável count (contador) com valor inicial = 0;
- O *gripper* com as garras abertas;
- O *LED* verde em estado desligado;
- O *Popup* para aviso enquanto o robô espera que o botão verde seja acionado.

Passo 3 – Criação de subprogramas

Criar dois subprogramas:

- *Pick*;
- *Place*.

Para criar subprogramas é necessário inserir a instrução *SubProg* com a secção *Robot Program* ativa (Figura 105 e Figura 106).

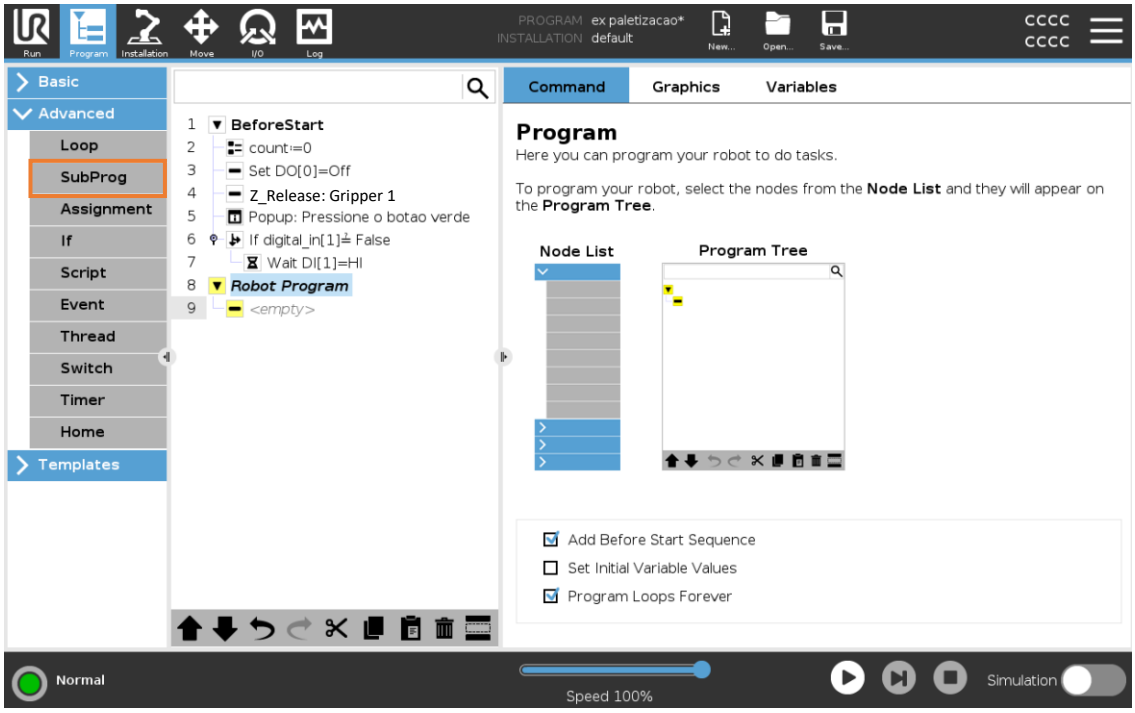


Figura 105 - Inserção da instrução *SubProg*

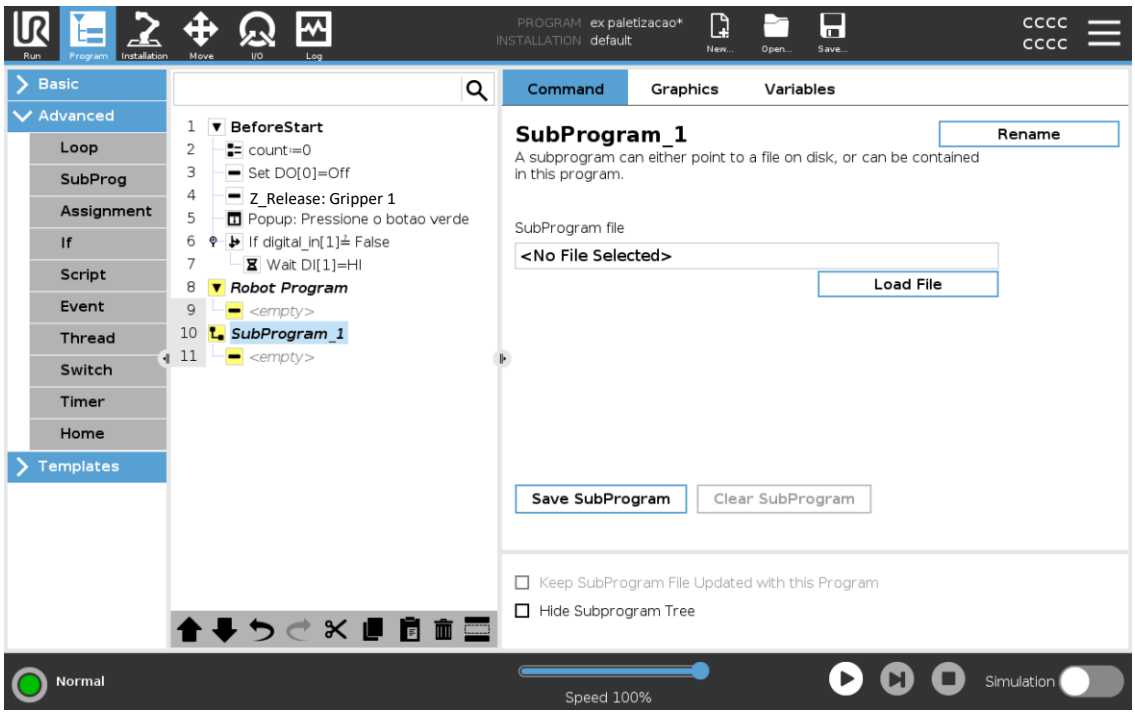


Figura 106 - Instrução *SubProg*

De seguida, atribuir o nome *Pick* ao subprograma 1 (Figura 107) e *Place* ao subprograma 2 (Figura 108).

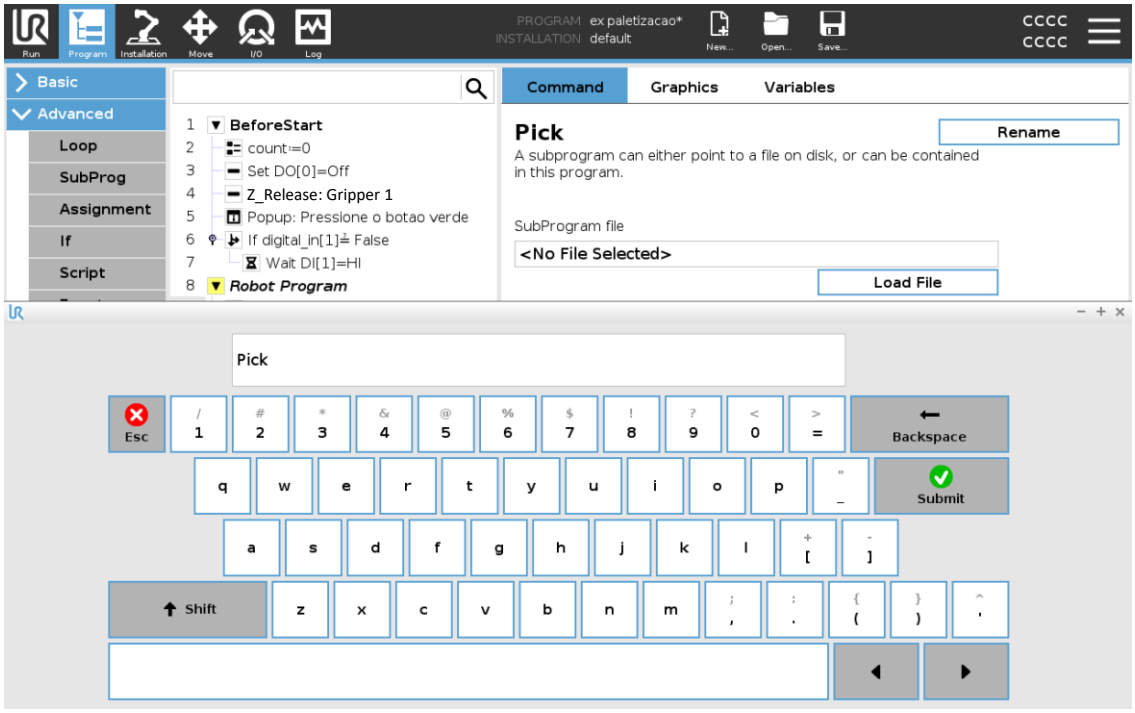


Figura 107 - Denominação da instrução *SubProg*

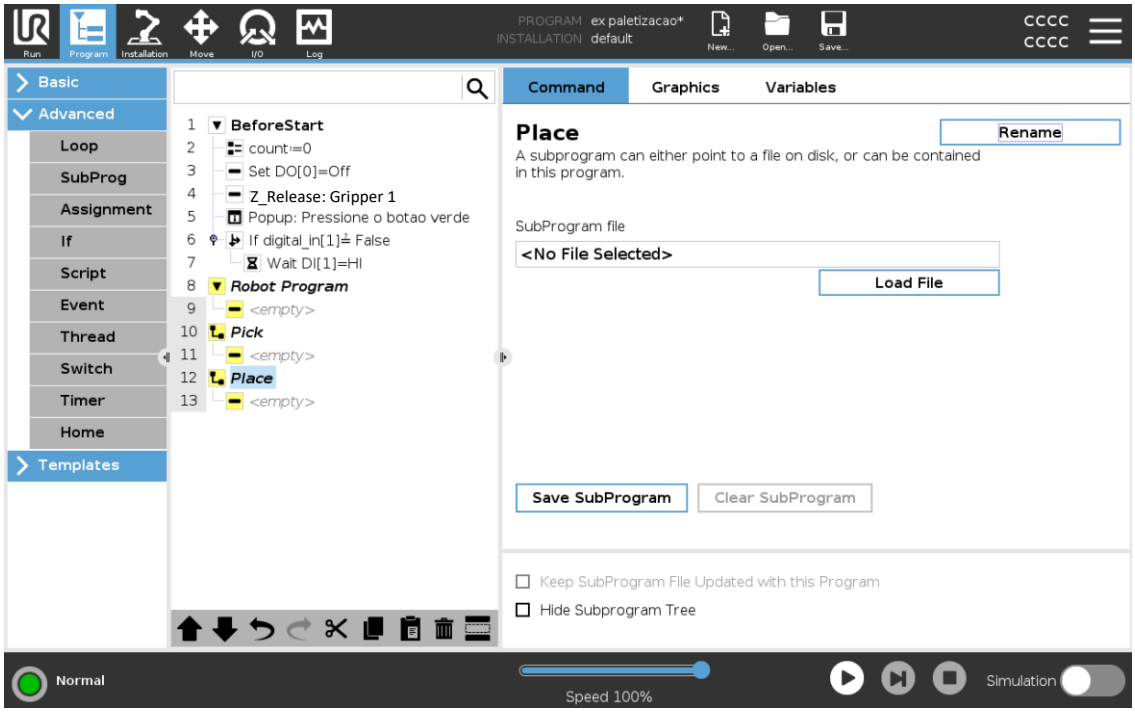


Figura 108 - Ramificação obtida com os dois subprogramas criados

Passo 4 – Configuração do subprograma "*Pick*"

A instrução *Seek* permite realizar o processo de desempilhamento de peças de forma automática, sendo apenas necessário definir as secções da instrução. Como na zona A estão 5 peças empilhadas pode-se utilizar esta instrução. Para a inserir aceda à área de programação *Templates*, selecionando a opção *Seek* (Figura 109).

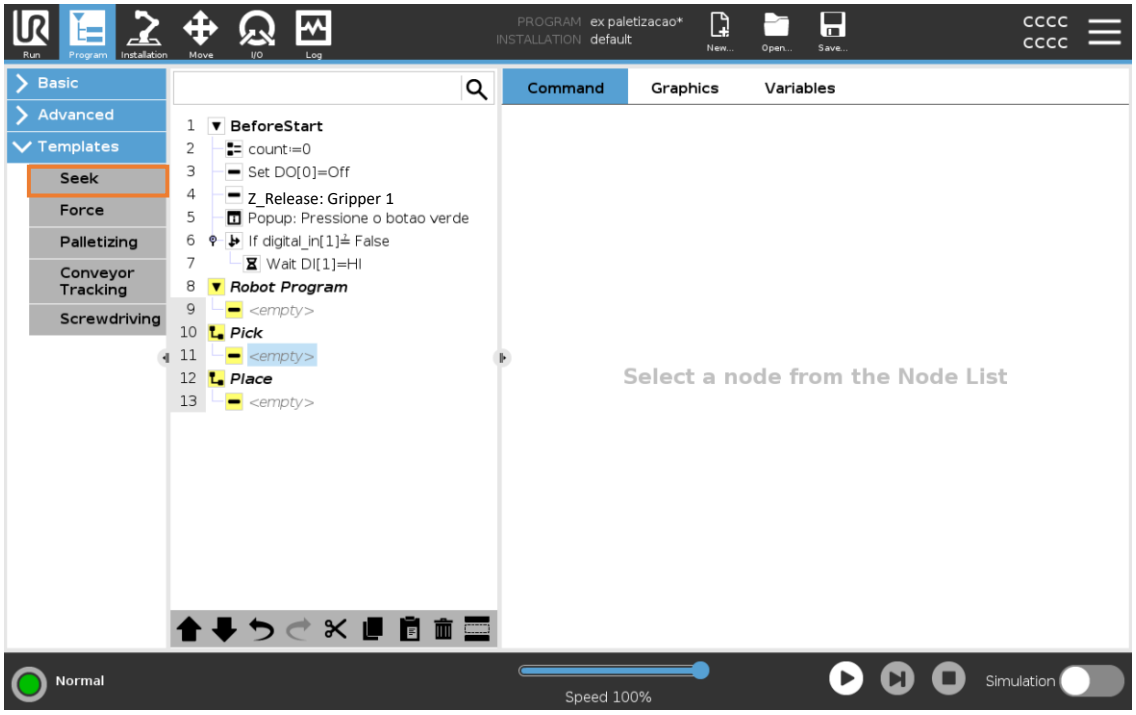


Figura 109 - Inserção da instrução *Seek*

De seguida, selecionar o tipo *Destacking* que corresponde à opção de desempilhamento (Figura 110).

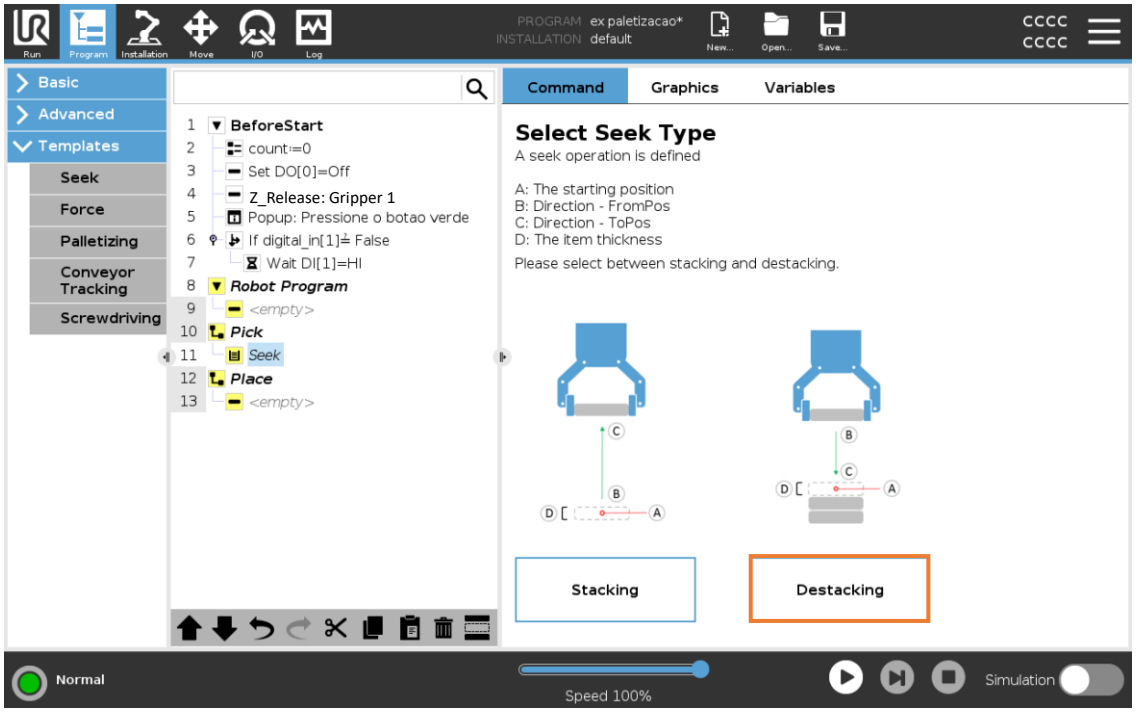


Figura 110 - Seleção do tipo da instrução *Seek*

Para ser possível visualizar todos os parâmetros da instrução é necessário abrir a ramificação presente na secção *Direction* e *PickSequence* (Figura 111). Deve-se obter uma ramificação igual à que está representada na Figura 112.

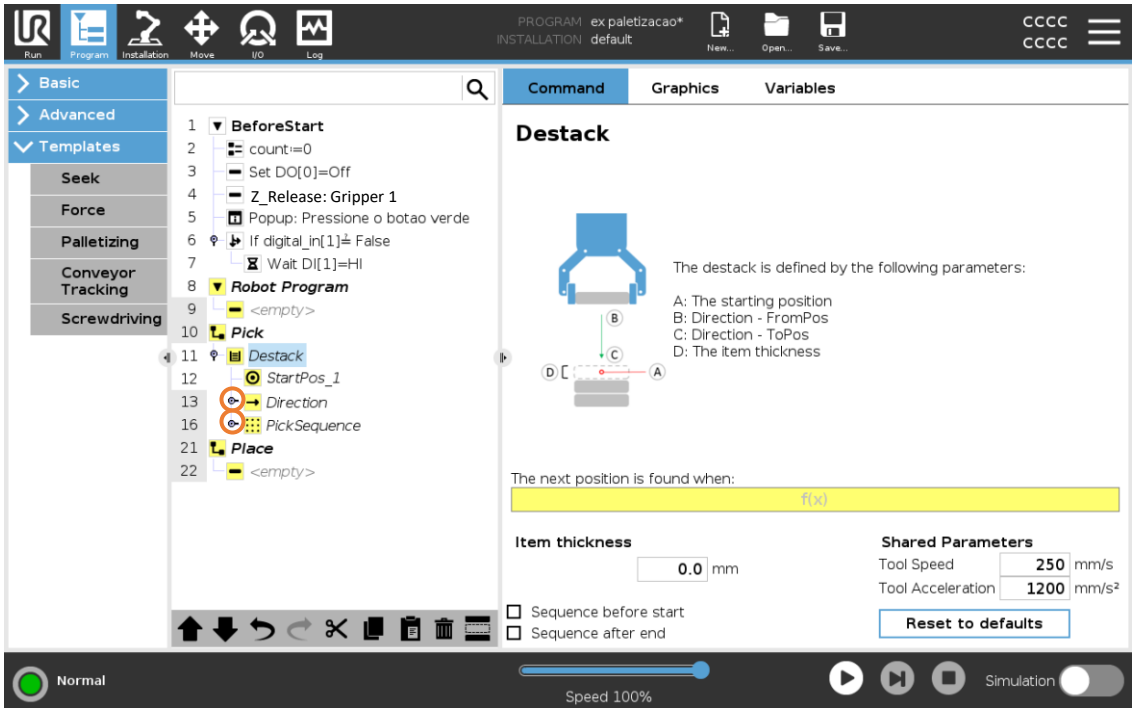


Figura 111 - Tipo *Destack*

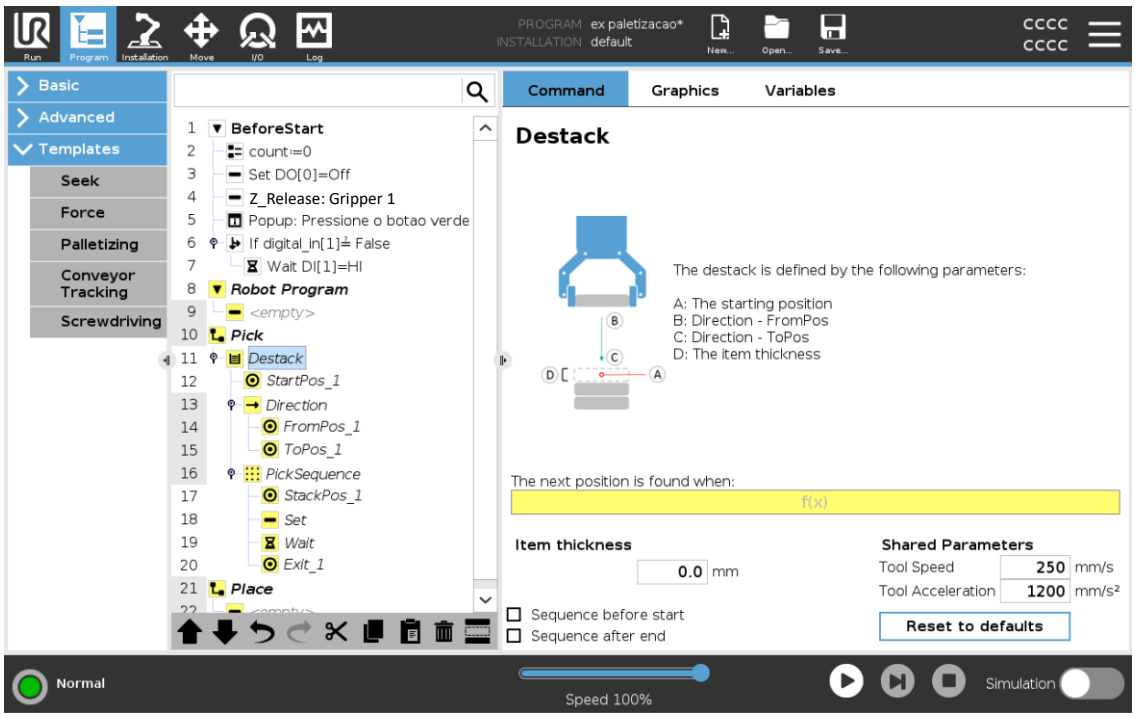


Figura 112 - Parâmetros a configurar na instrução *Seek*

Inicialmente é necessário definir qual é a condição para encontrar a próxima peça e a espessura da peça (ver Figura 113). Neste caso, pode utilizar o botão DI0 de 2 posições como condição. Se o botão estiver ativo serve de sinal de "peça disponível". Desta forma, o robô desce **15 mm** sempre que volta à posição inicial (correspondente à espessura da peça) e pega na peça seguinte.

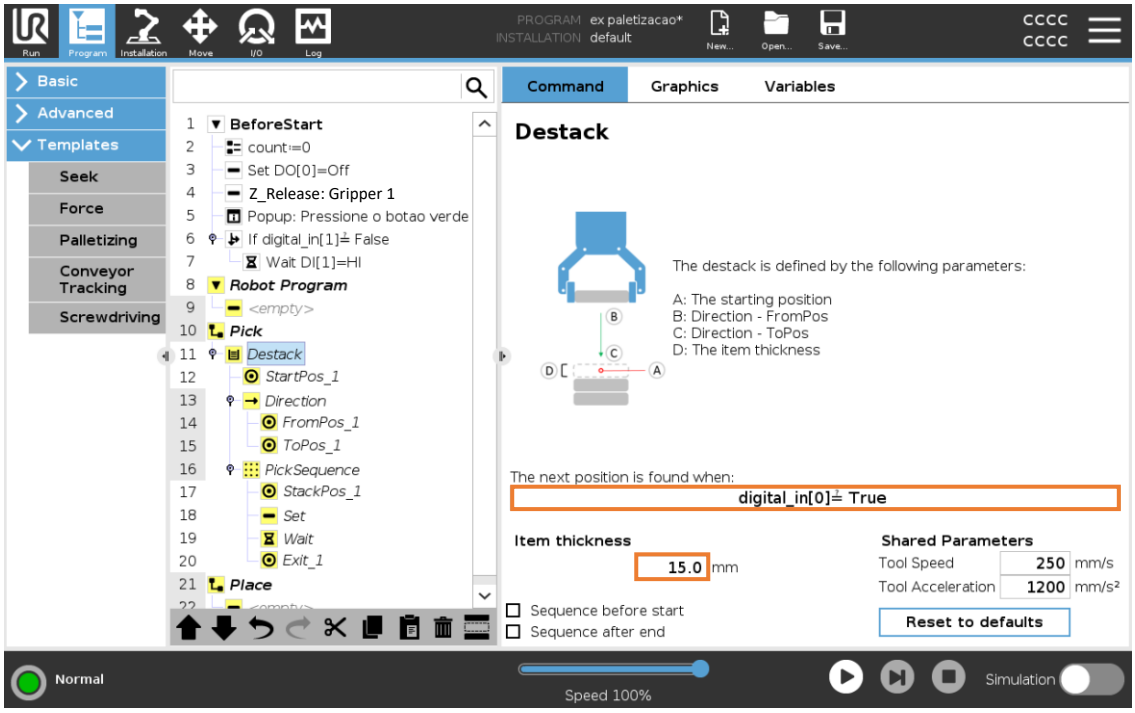


Figura 113 - Parâmetros iniciais da instrução *Seek*

De seguida, define-se o ponto inicial do ciclo *StartPos_1* (Figura 114). Este ponto corresponde ao local de recolha da primeira peça na zona A (aproximadamente localizado no centro da peça).

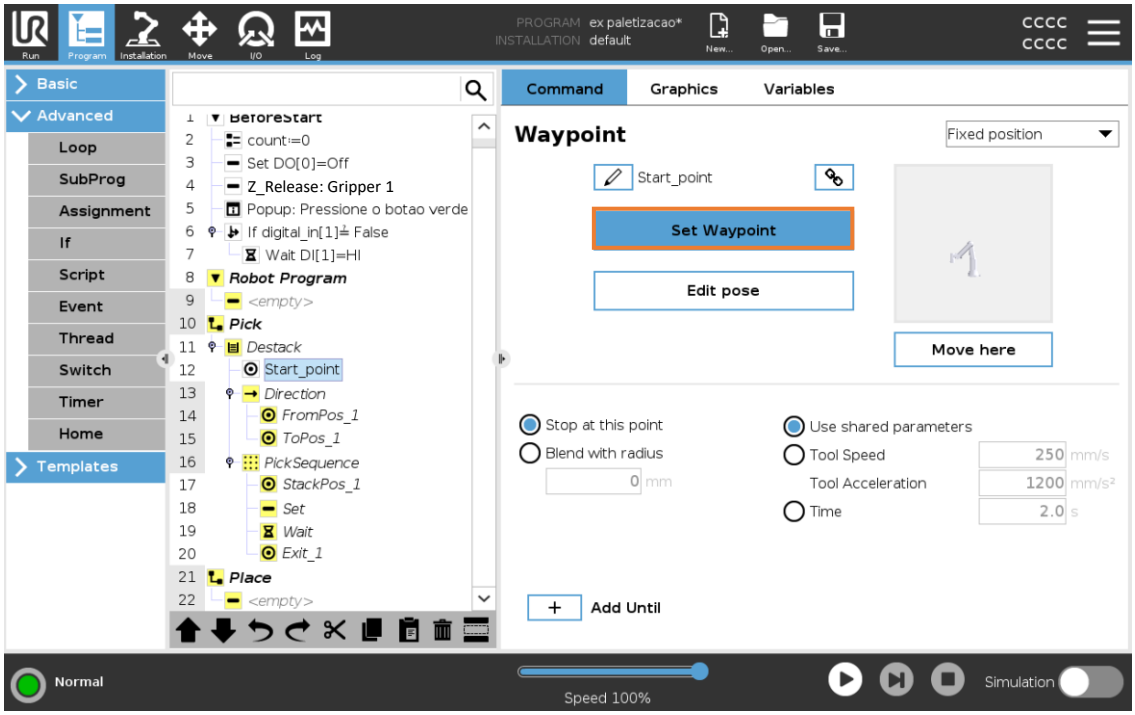


Figura 114 - Definição do ponto de início do movimento

Posteriormente, define-se a direção cartesiana do movimento (descendente no eixo z). O primeiro ponto (*FromPos_1*) corresponde ao ponto linearmente acima da peça a ser recolhida

(*before_start*) (Figura 115) e o segundo ponto (*ToPos_1*) corresponde ao local de recolha (*start_point*) (Figura 116).

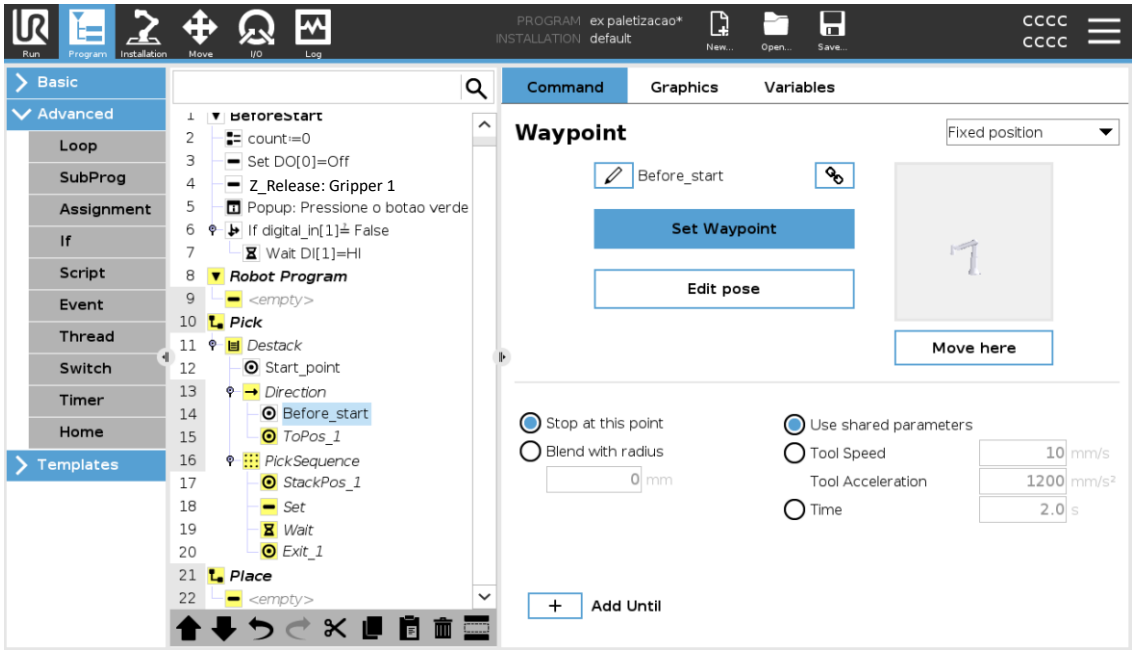


Figura 115 - Definição da direção do movimento_1

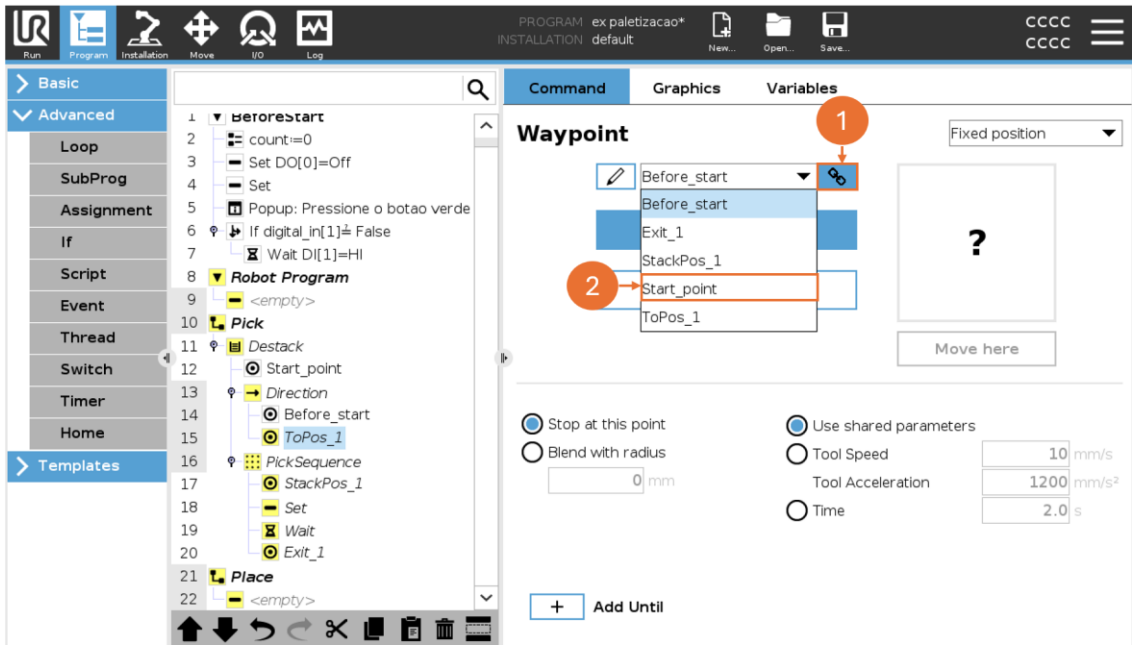


Figura 116 - Definição da direção do movimento_2

Para finalizar a secção da direção, basta definir que o robô para quando atingir 130 mm (Figura 117).

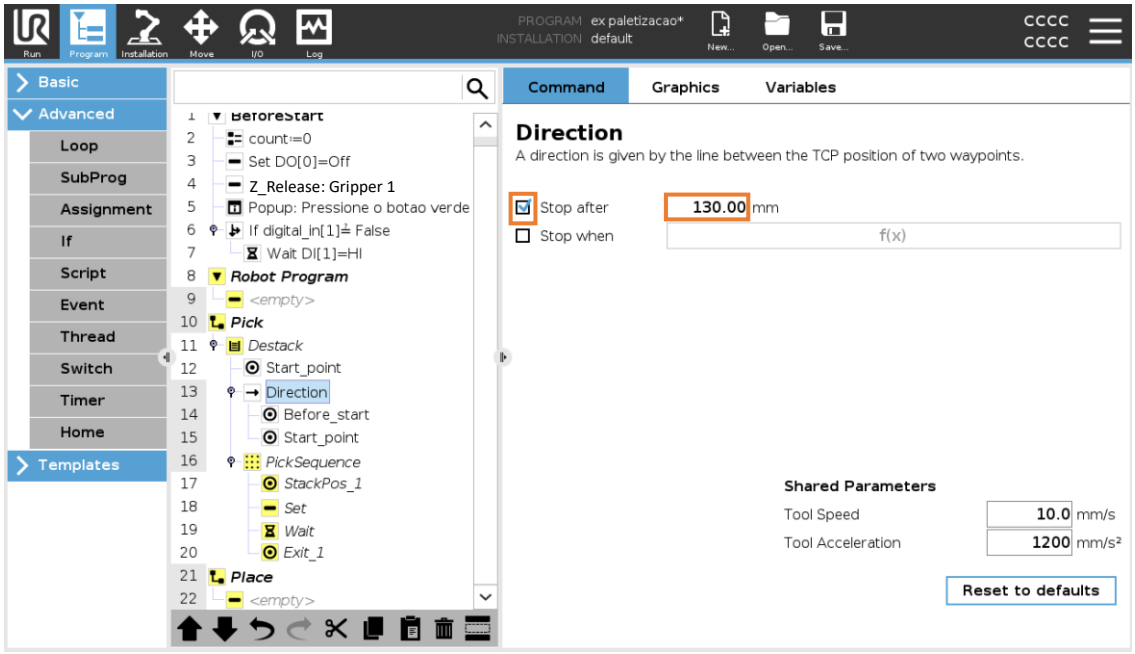


Figura 117 - Definição da direção do movimento_3

A próxima secção corresponde à sequência de movimentos (*PickSequence*) que se pretendem realizar no movimento de recolha. Deve-se seguir a seguinte lógica:

- Definir o *StackPos_1*, que corresponde ao ponto inicial de recolha *start_point*;
- Acionar o *gripper* para agarrar a peça;
- Esperar um segundo;
- Subir em movimento linear até ao ponto *before_start*;
- Incrementar 1 ao contador.

Após a parametrização das secções, finaliza-se o subprograma de recolha das peças (*Pick*) (Figura 118).

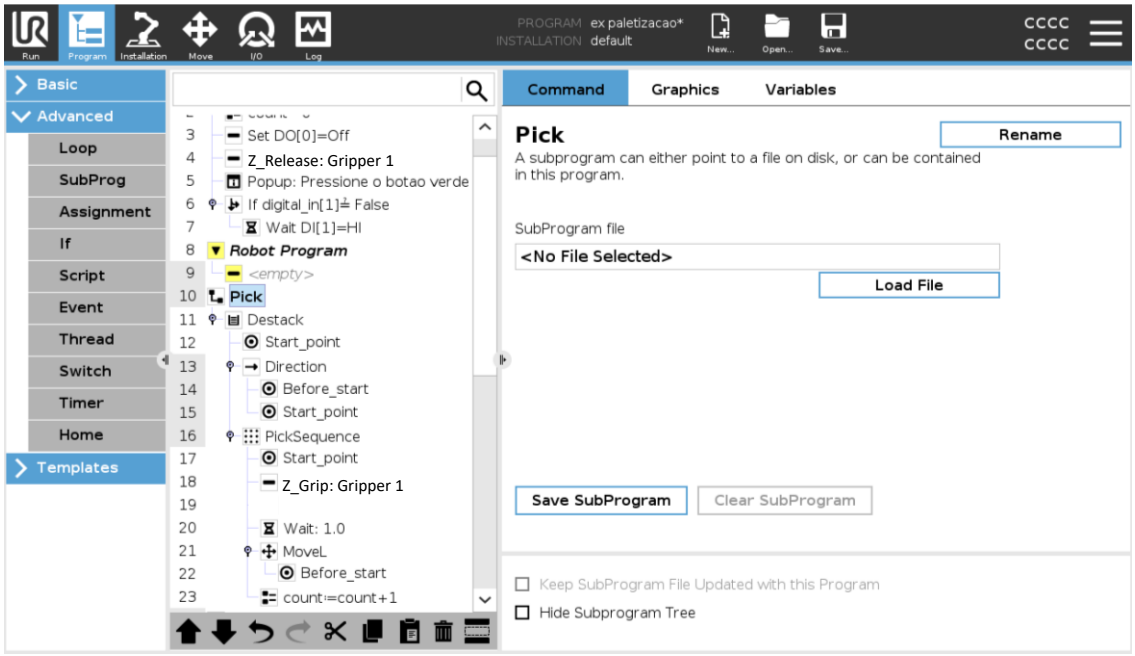


Figura 118 - Subprograma *Pick* finalizado

Passo 5 – Configuração do subprograma "*Place*"

A instrução *Palletizing* permite realizar a paletização de peças de forma mais simplificada. Para começar, insere-se a instrução no subprograma 2 (*Place*) (Figura 119).

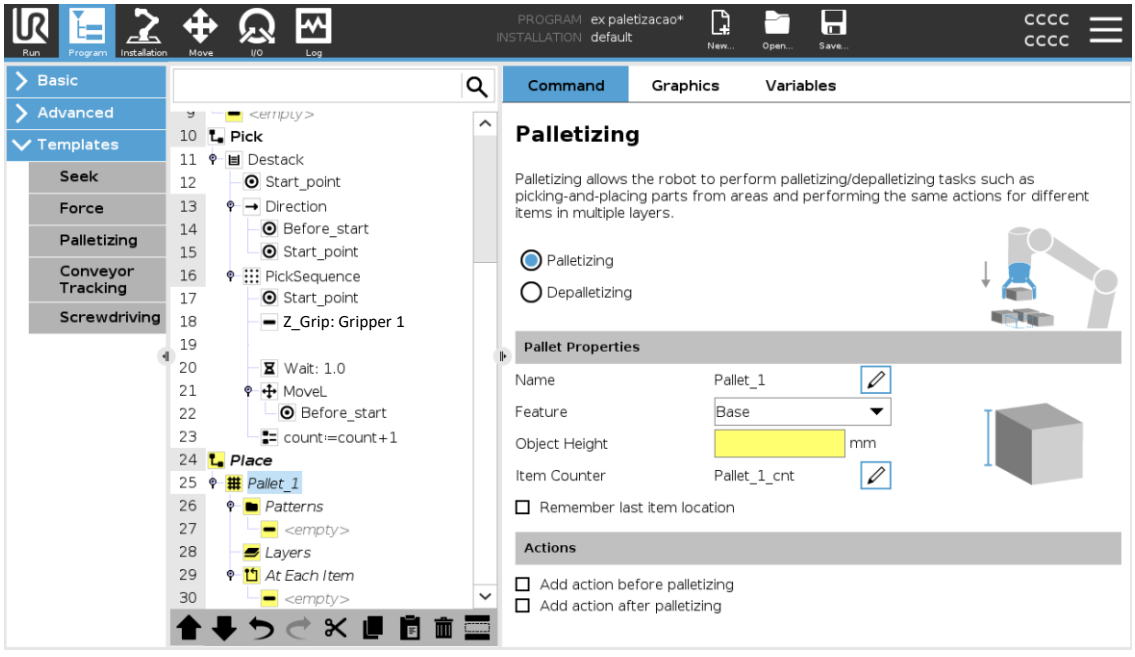


Figura 119 - Instrução *Palletizing*

Numa primeira fase, definir a altura do objeto **15 mm** e ativar a opção *Remember last item location* na secção *Pallet_1* (Figura 120).

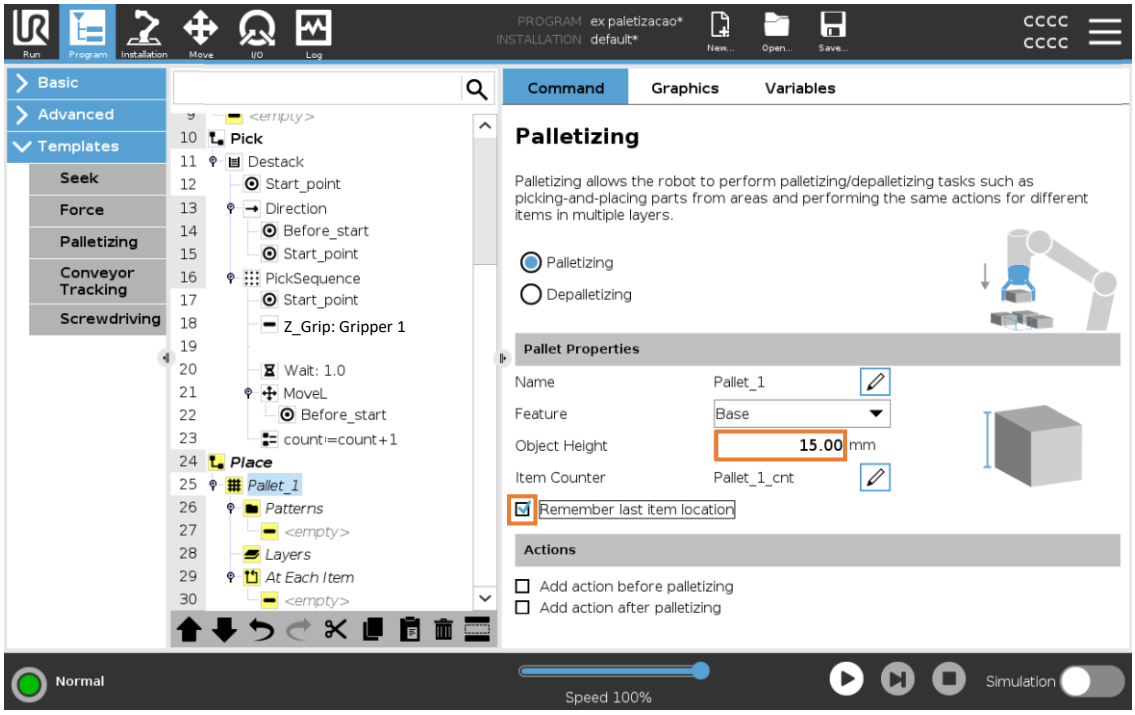


Figura 120 - Parametrização da secção Pallet_1

Numa segunda fase, é necessário definir o tipo de paletização que se pretende realizar. A instrução *Palletizing* permite realizar a paletização de peças em três padrões diferentes, designadamente linha, grelha e irregular. Abaixo segue uma breve explicação das mesmas:

1. **Em Linha (*Line*):**
    - As peças são depositadas sequencialmente ao longo de uma única linha (horizontal ou vertical);
    - Útil quando há apenas uma fila de posições disponíveis.
2. **Em Grelha (*Grid*):**
    - As peças são distribuídas numa matriz com várias linhas e colunas, seguindo um padrão uniforme;
    - Ideal para paletização com organização regular e geométrica.
3. **Irregular (*Irregular*):**
    - As posições de depósito são definidas manualmente ou com base num percurso não sistemático.
    - Indicado para casos onde o layout das paletes ou o posicionamento dos objetos não segue uma simetria.

Neste caso, o tipo de paletização é irregular (Figura 121).

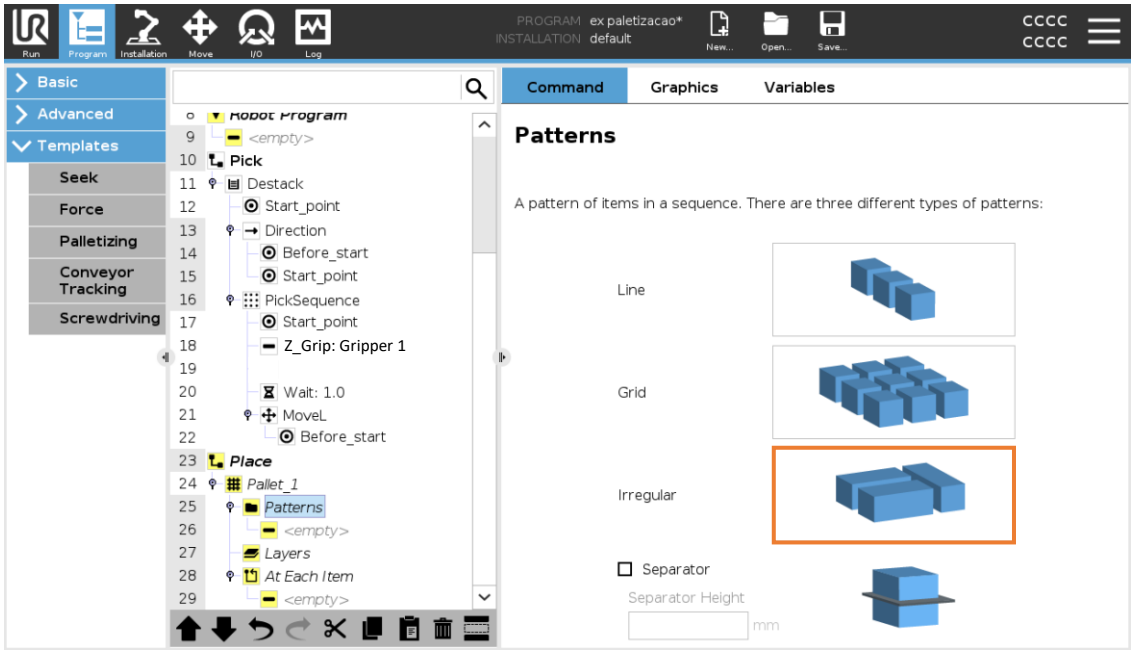


Figura 121 - Opções de Paletização

De seguida, parametriza-se a secção *Irreg_Pattern_1*. Por padrão, a sequência irregular acrescenta de forma automática 3 itens (*Item_1, Item_2* e *Item_3*). Cada Item corresponde a uma posição de deposição. Como o objetivo do exercício é realizar a paletização de 5 peças, é necessário definir 5 itens na sequência. Para inserir os itens, é necessário clicar na opção *Add Item,* conforme indicado na Figura 122.

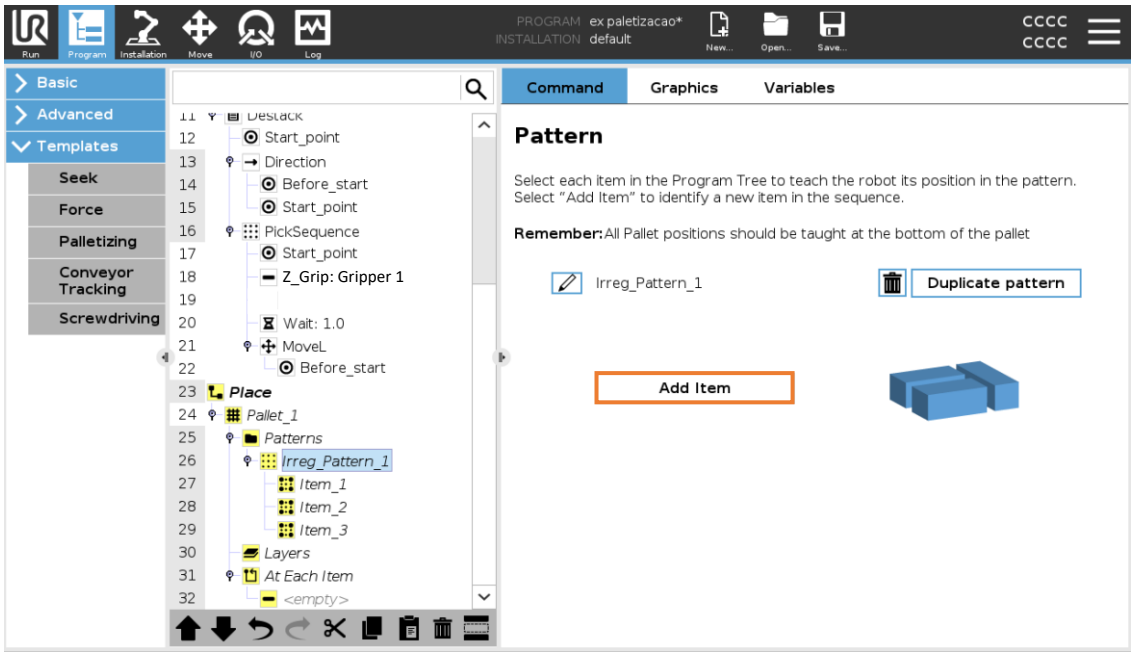


Figura 122 - Adição de itens

Posteriormente, define-se a localização de cada item. A posição de origem de paletização (*Item_1*) deve coincidir com a célula que está assinalada com o número 1 e assim sucessivamente. Na Figura 123 estão presentes a distância entre posições de paletização em

milímetros e a numeração das mesmas. Após a definição do *Item_1* pode definir os restantes itens alterando apenas as coordenadas x e y dos pontos.

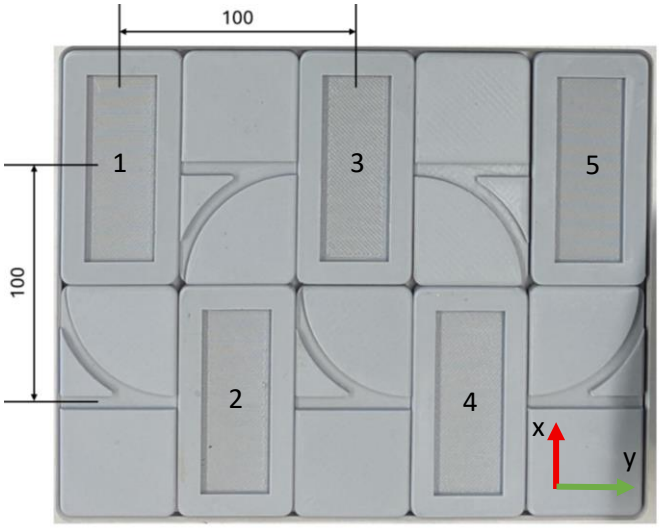


Figura 123 - Distância entre paletes e numeração das mesmas

Numa terceira fase seleciona-se o tipo de camada. Definir o *Irreg_Pattern_1* para a camada 1 (*Layer_1*) (Figura 124).

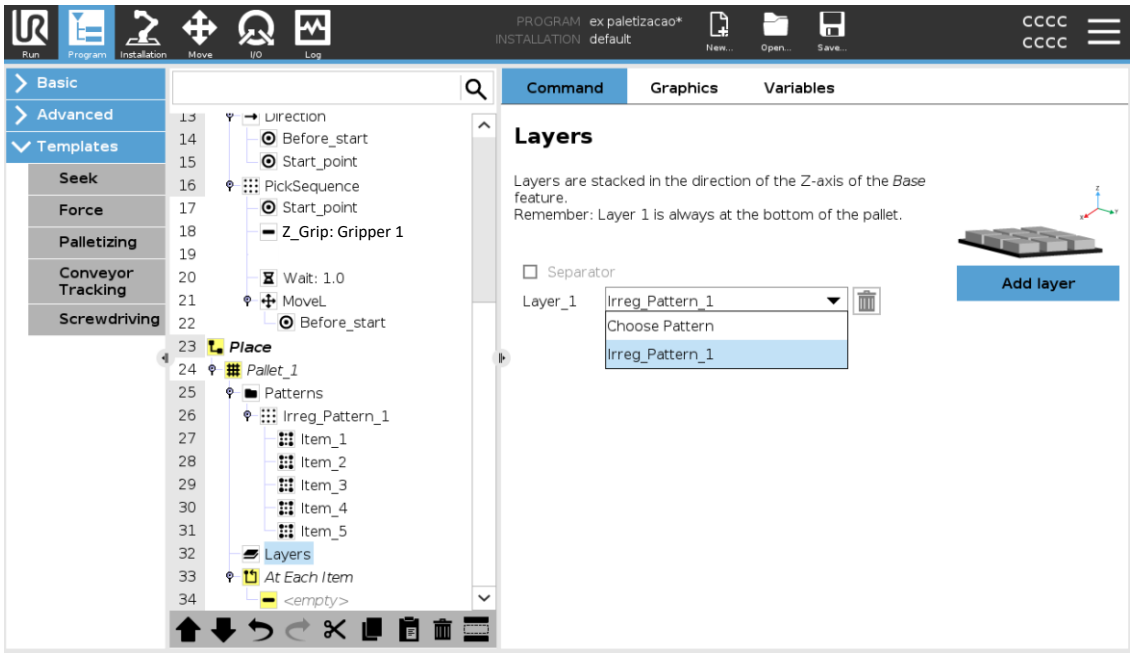


Figura 124 - Parametrização da *Layer*

Numa quarta fase, é definida a sequência de movimentos que vai ser realizada em cada item. Nesta secção segue-se um procedimento inicial de 6 passos:

- Passo 1: Clicar em *Next* (Figura 125);
- Passo 2: Mover o robô para o ponto de referência (Figura 126);

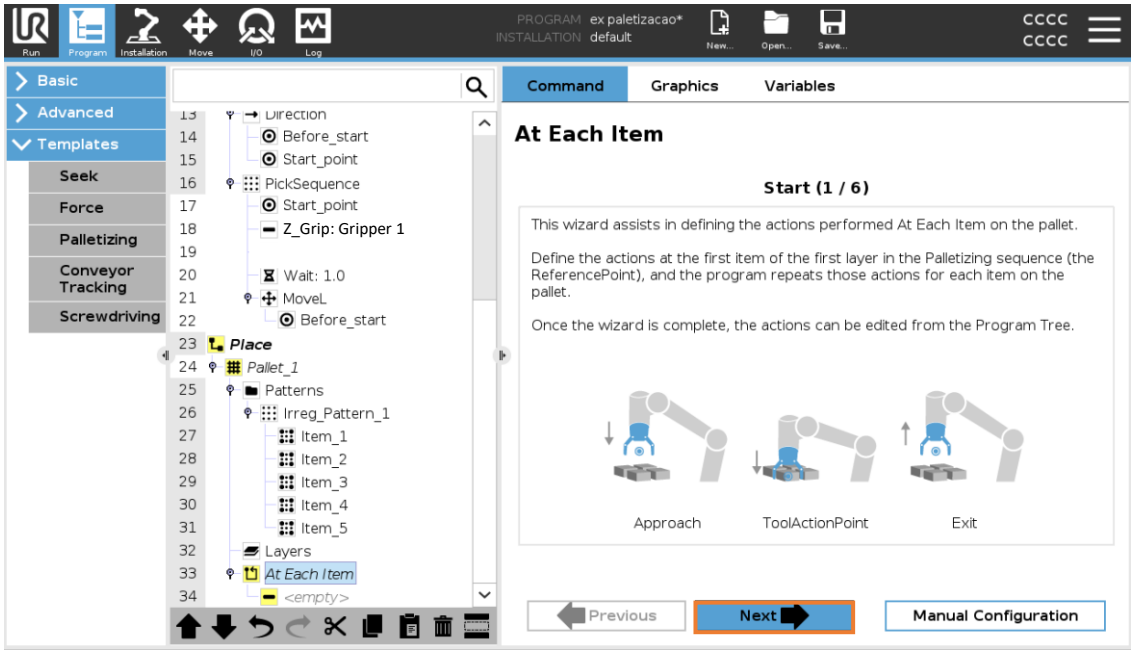


Figura 125 - Primeiro passo da sequência

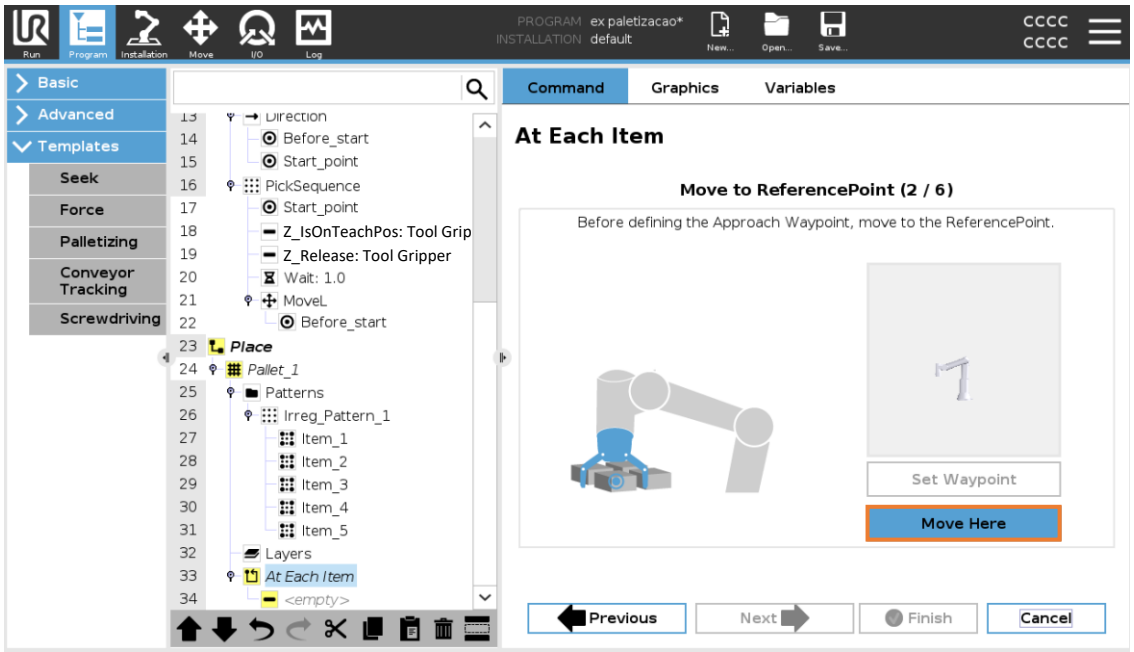


Figura 126 - Segundo passo da sequência

- Passo 3: Definir um ponto de segurança quando o robô se dirige para a palete. Pode-se definir um ponto linearmente acima da posição 1 (cota z superior) (Figura 127);
- Passo 4: Mover o robô para o ponto de referência (Figura 128);

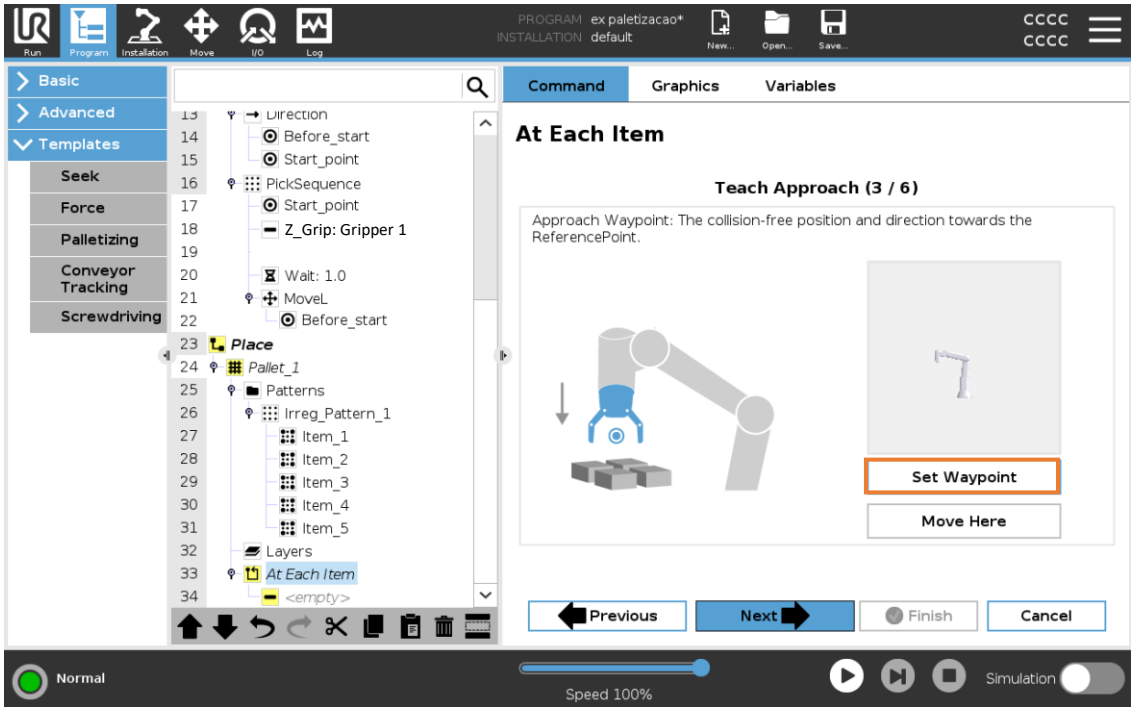


Figura 127 - Terceiro passo da sequência

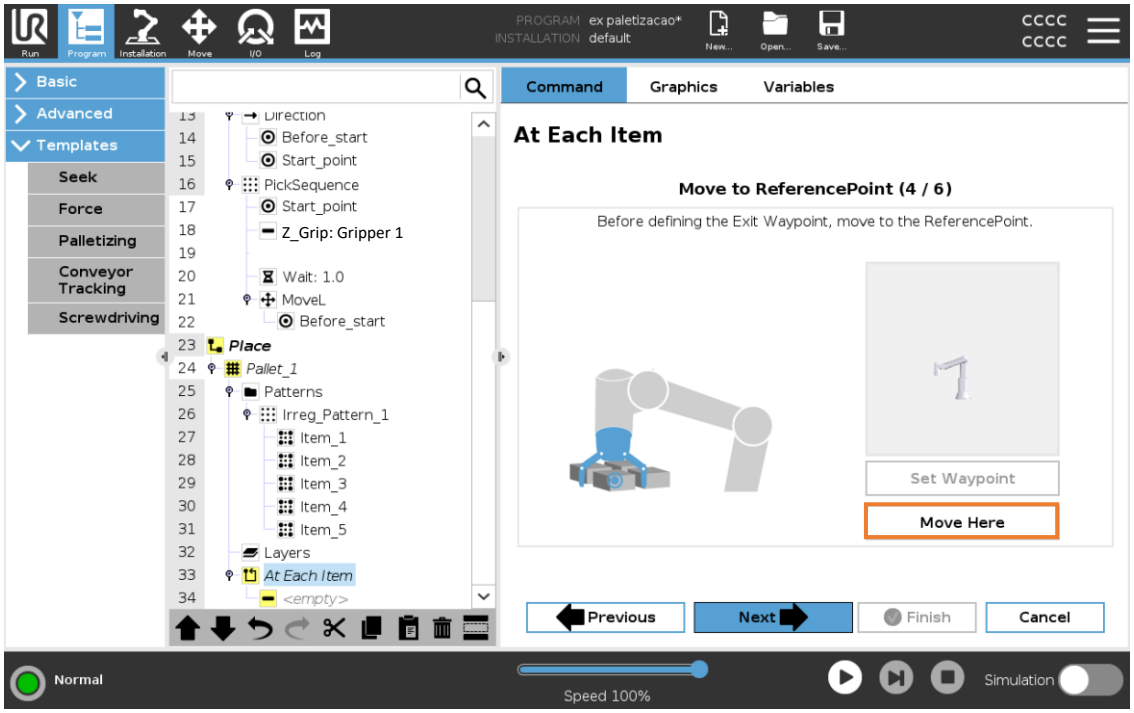


Figura 128 - Quarto passo da sequência

- Passo 5: Definir o ponto de saída. Seguir a mesma lógica do passo 3 (Figura 129);
- Passo 6: Clicar em *Finish* (Figura 130).

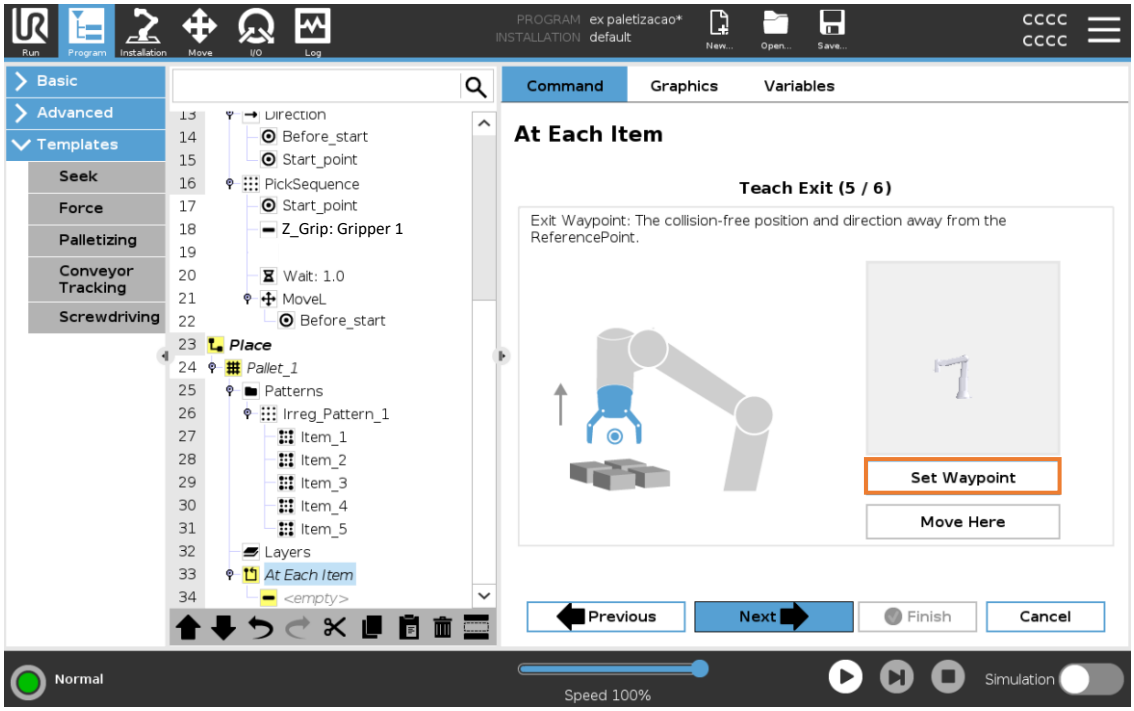


Figura 129 - Quinto passo da sequência

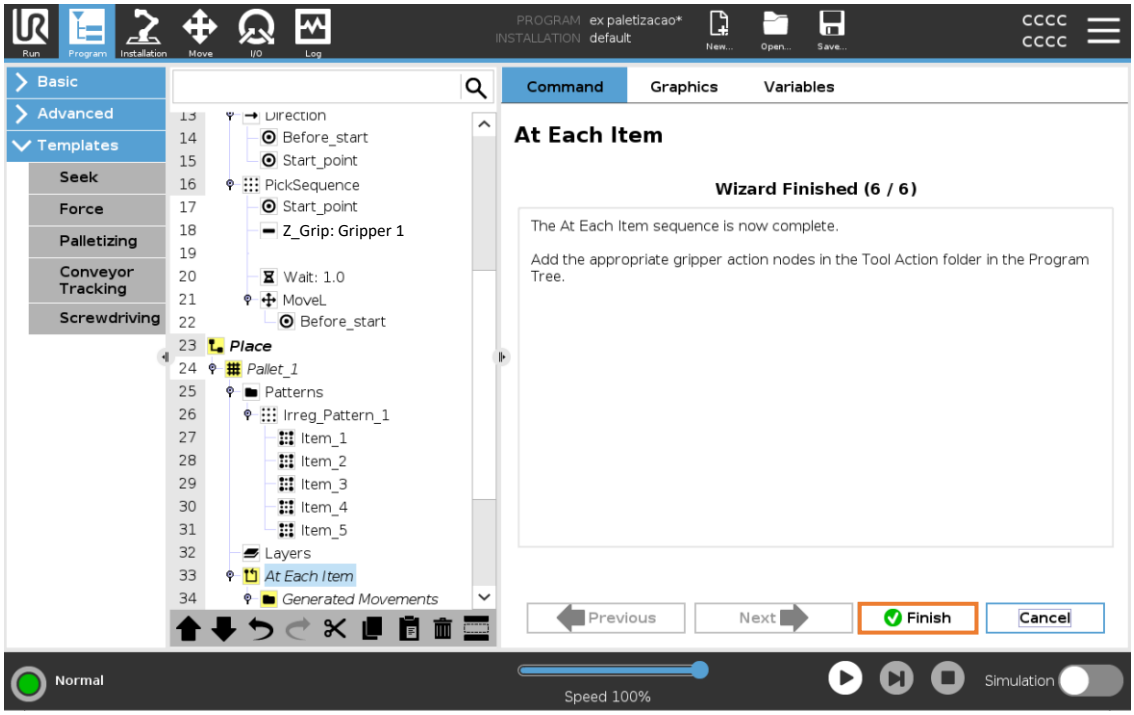


Figura 130 - Sexto passo da sequência

Para completar a secção de movimentos deve inserir a ação que a ferramenta (*gripper*) vai realizar. Quando se atinge a posição de deposição a peça é largada, ou seja, o *gripper* abre as garras (Figura 131).

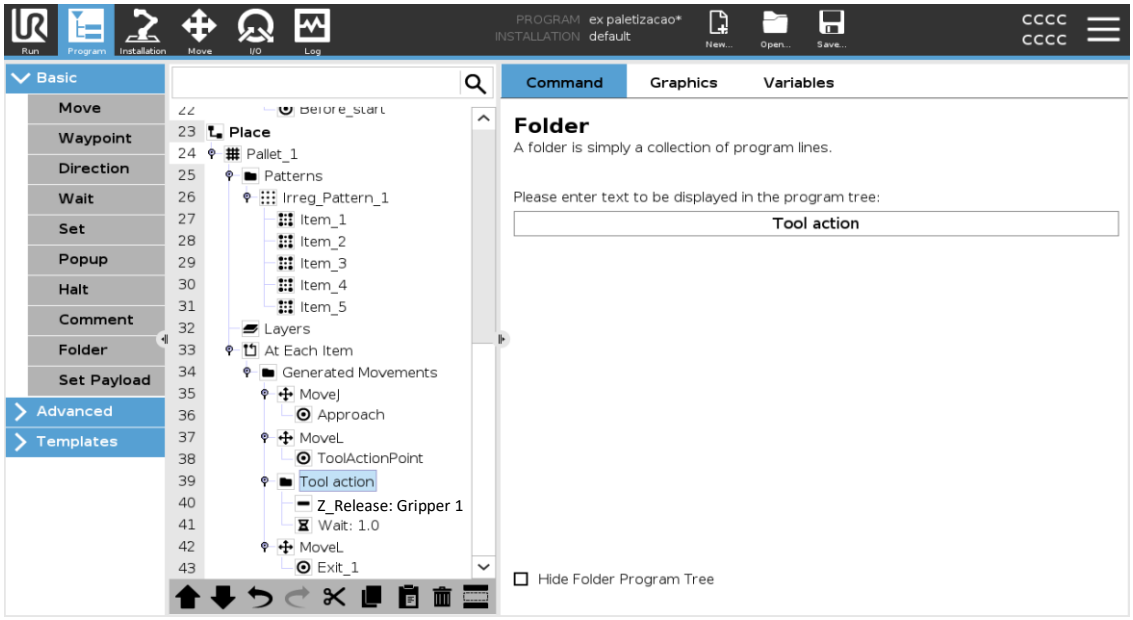


Figura 131 - Parametrização da secção *Tool Action*

Passo 6 – Estruturação do programa principal

Para a estruturação do programa principal é necessário definir as seguintes condições:

- Garantir que o robô começa e termina o programa na posição *Before_start*;
- Inserir uma instrução *If* para *count < 5* e uma instrução *ElseIf* para *count = 5*;
- Na instrução *If* deve-se:
  - Ativar o *LED* verde (DO0);
  - Mover o robô em movimento linear para a posição inicial (*Start_point*);
  - Inserir o subprograma *Pick* (conforme Figura 132) e *Place* (Figura 133).

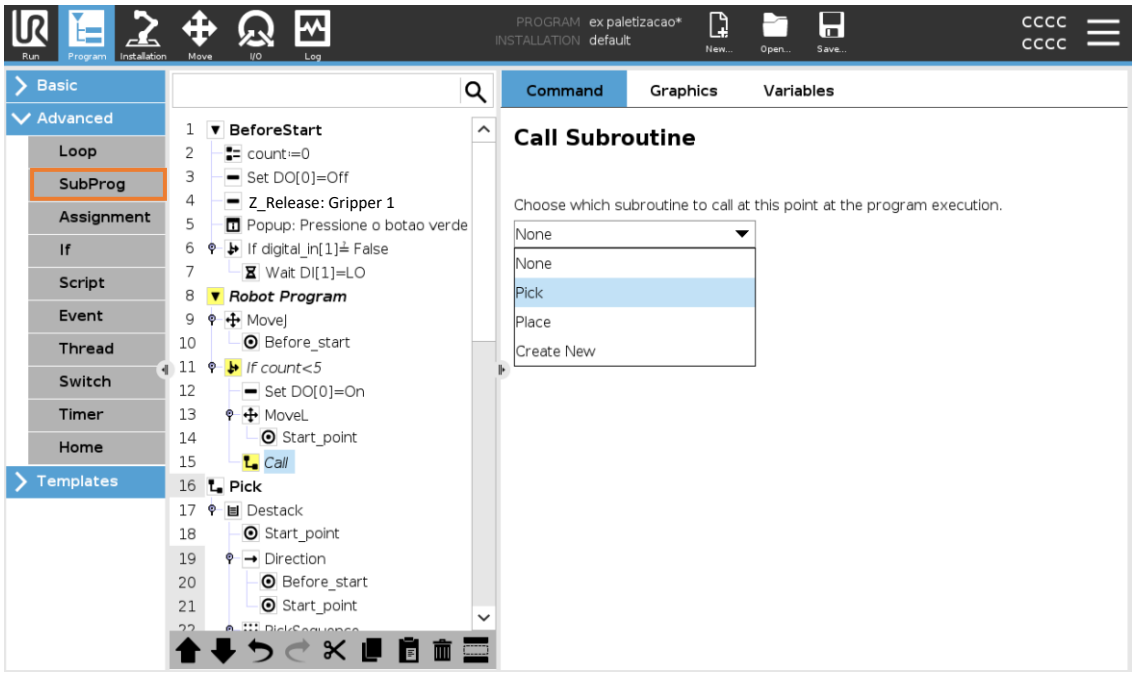


Figura 132 - Inserção do subprograma *Pick* no programa principal

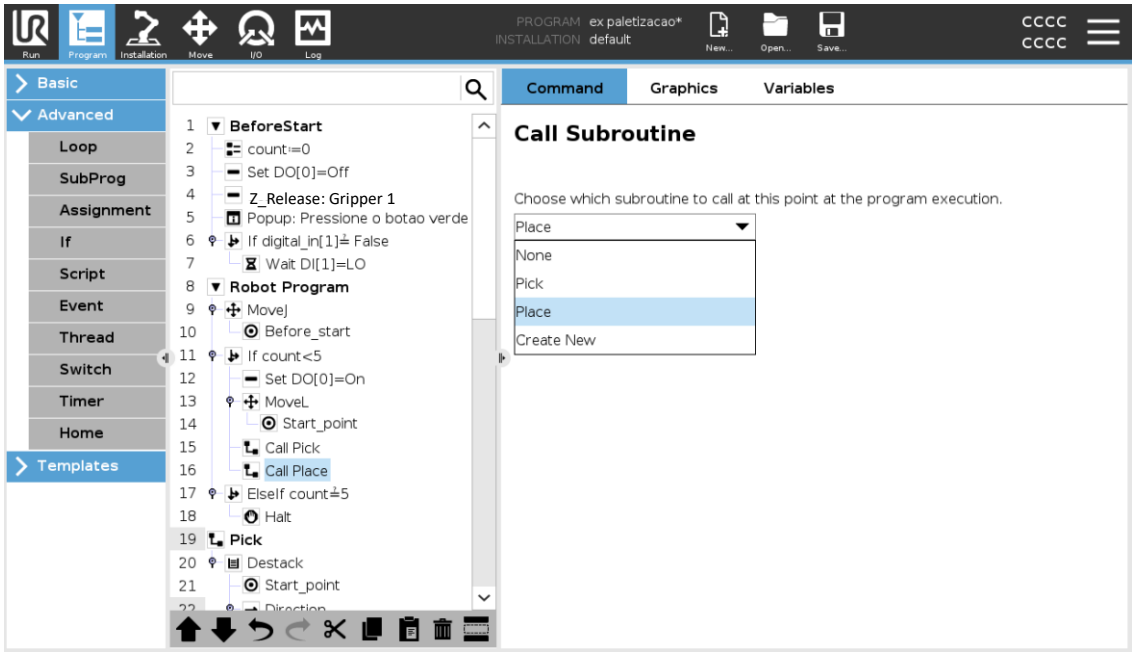


Figura 133 - Inserção do subprograma *Place* no programa principal

- Na instrução *ElseIf* deve-se:
  - Terminar o programa com a instrução *Halt* (Figura 134).

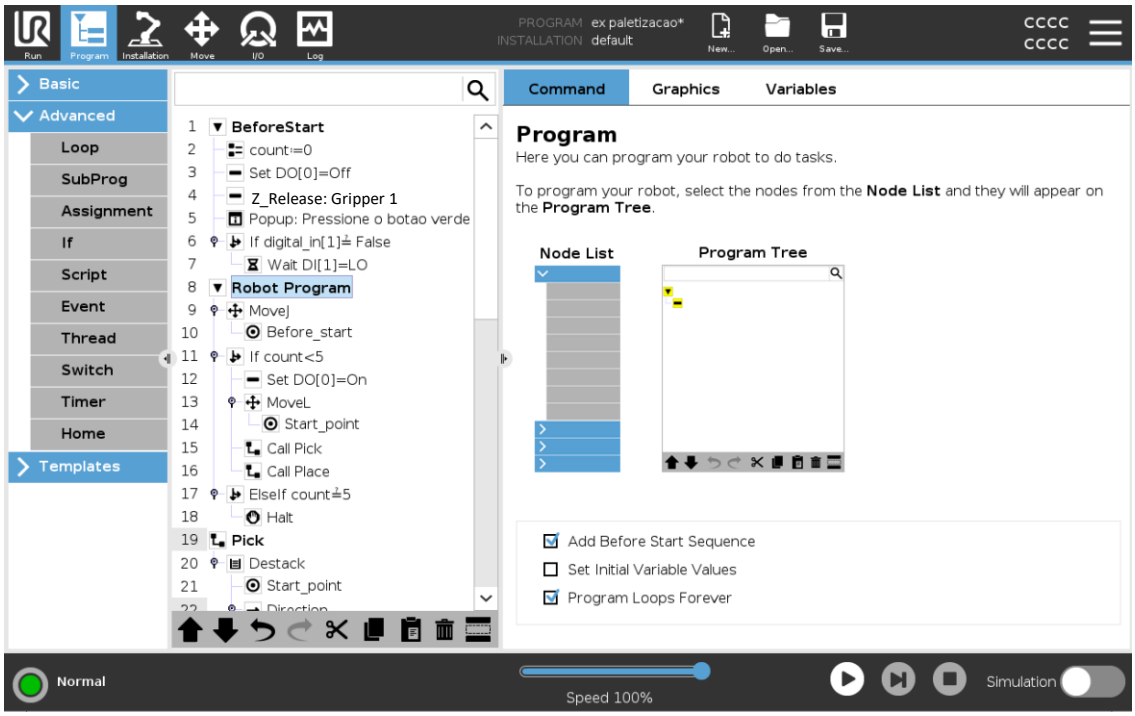


Figura 134 - Estruturação final do programa principal

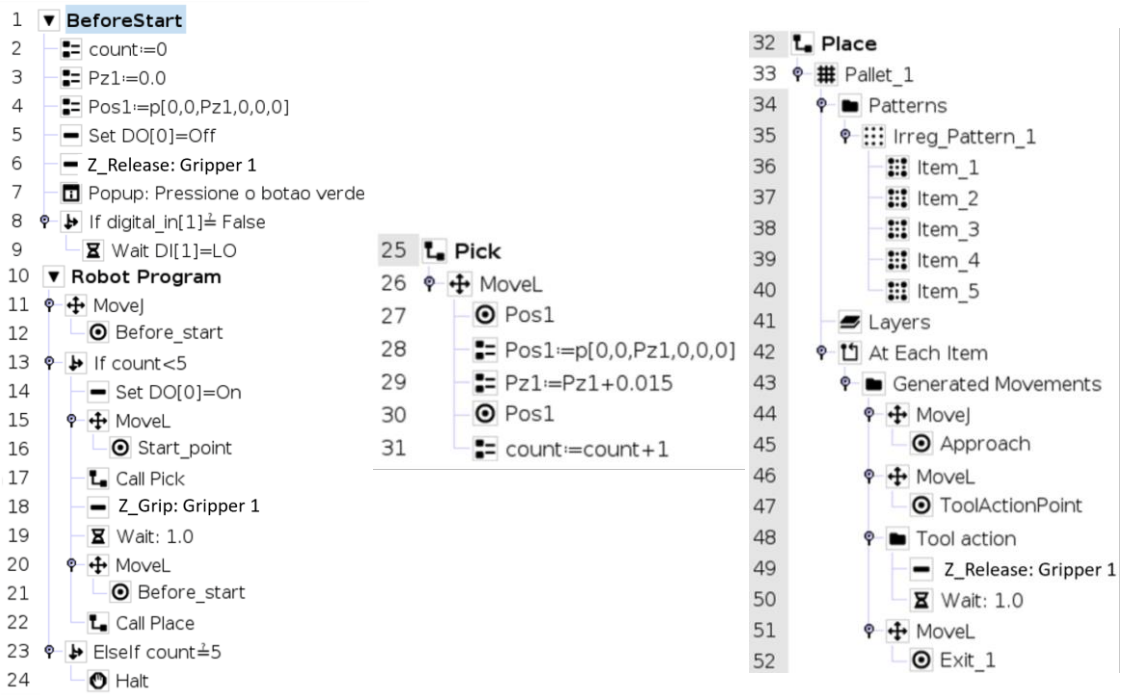
Passo 4 – Teste e Validação

Testar o programa.

Metodologia de resolução 2

Nesta solução, manteve-se a estrutura geral do programa apresentada na solução anterior, alterando-se apenas a configuração do subprograma *Pick*. Nesta abordagem, a despaletização das peças deixa de recorrer à instrução *Seek* e passa a ser realizada a partir de uma posição de recolha previamente definida (Pos1). Esta posição corresponde a uma variável do tipo *pose*, à qual se acrescentam 0,015 m na coordenada Z para cada peça recolhida. Esta alteração implica um controlo mais explícito da trajetória vertical no processo de recolha.

Na Figura 135 apresenta-se a codificação final desenvolvida para o exercício com a nova abordagem na zona de recolha de peças.



(a) (b) (c)

Figura 135 - Solução 2 do exercício 3: Secção *BeforeStart* e estrutura principal do programa (a); Subprograma de recolha de peças *Pick* editado (b); Subprograma de posicionamento de peças *Place* (c)

Passo 1 – Definição das novas variáveis

Para se proceder com a alteração do subprograma Pick deve-se, numa primeira fase, eliminar todas as instruções presentes neste subprograma. De seguida devem ser criadas as variáveis Pz1 = 0 e Pos1 = p[0, 0, Pz1, 0, 0, 0].

Passo 2 – Atribuição do valor de uma variável a um *waypoint*

Numa segunda fase é necessário inserir uma instrução do tipo *MoveL* e configurar o *waypoint* como *Variable Position* e utilizar como referência a variável Pos1. Para realizar esta alteração deve-se seguir a metodologia presente na Figura 136.

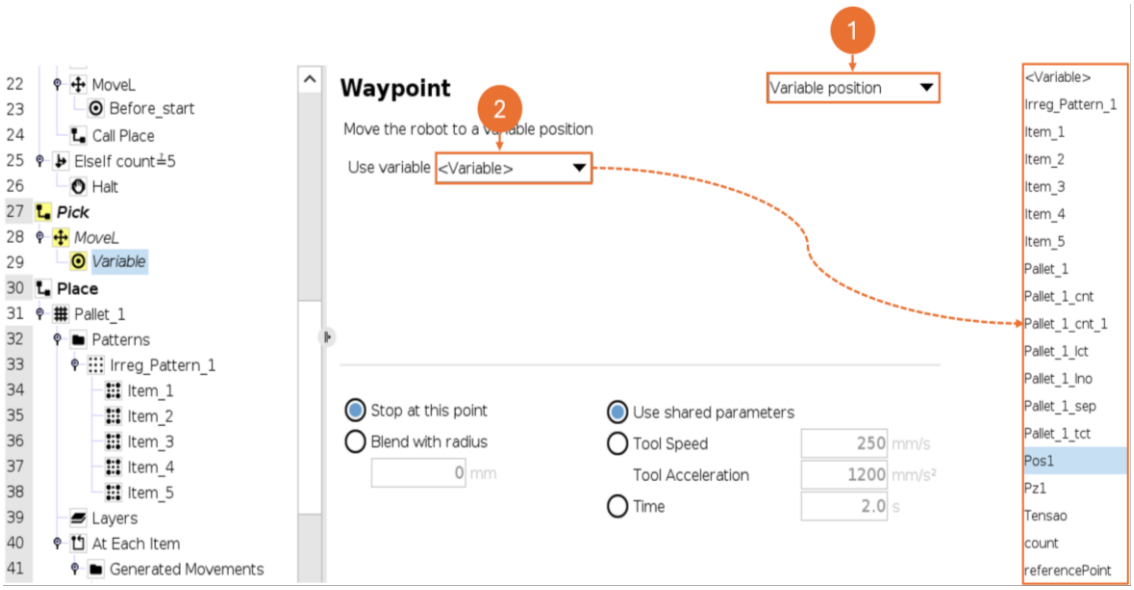


Figura 136 – Associação de uma variável a um ponto

Passo 3 – Conclusão da programação do subprograma

Reatribuir o valor da nova posição à variável Pos1 e incrementar o contador. No final o subprograma Pick deve apresentar a codificação exposta na Figura 137.

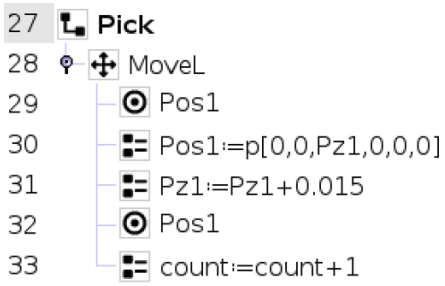


Figura 137 - Codificação final do subprograma Pick

Exercício 7 – Interação com saídas analógicas

O objetivo deste exercício passa por estabelecer uma relação entre o valor do contador utilizado no exercício 6 e a leitura do voltímetro. Ou seja, à medida que o contador é incrementado, o valor lido no voltímetro deve acompanhar essa variação (ex: contador = 1 → voltímetro = 1.0). Deve ser utilizado o programa desenvolvido no exercício anterior e realizadas as alterações necessárias.

Requisito do exercício:

1. Relacionar o valor do voltímetro com o valor do contador.

Metodologia de resolução

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e (a partir da solução 2 do programa desenvolvido para o exercício 6) com o nome *exercício 7*,

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, definir:

- A variável:
    - tensao = 0.0
- O valor inicial do voltímetro de 0.0.

Para atribuir um valor fixo ao voltímetro é necessário inserir uma instrução Set e configurá-la para a opção *Set Analog Output*. Neste caso é necessário selecionar a saída que corresponde ao voltímetro (analog_out[0]) e definir o valor 0.0 Volts, conforme se ilustra na Figura 138.

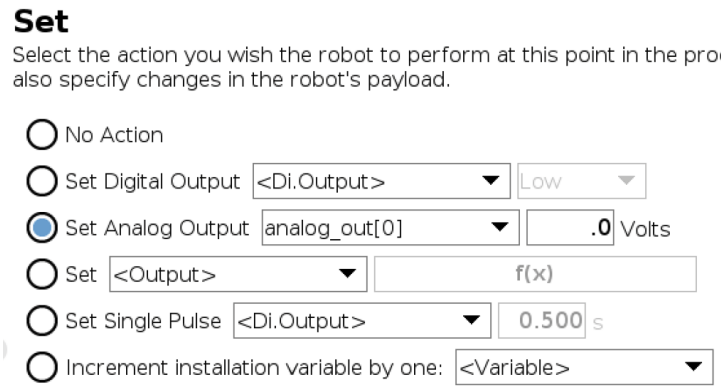


Figura 138 - Parametrização da instrução Set para atribuir valores fixos a saídas analógicas

Na Figura 139 apresenta-se a secção *Before Start* após a definição da variável e da instrução *Set*.

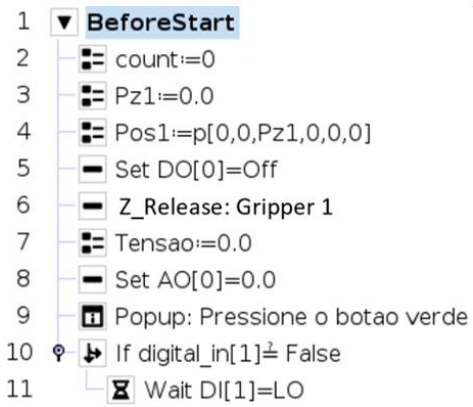
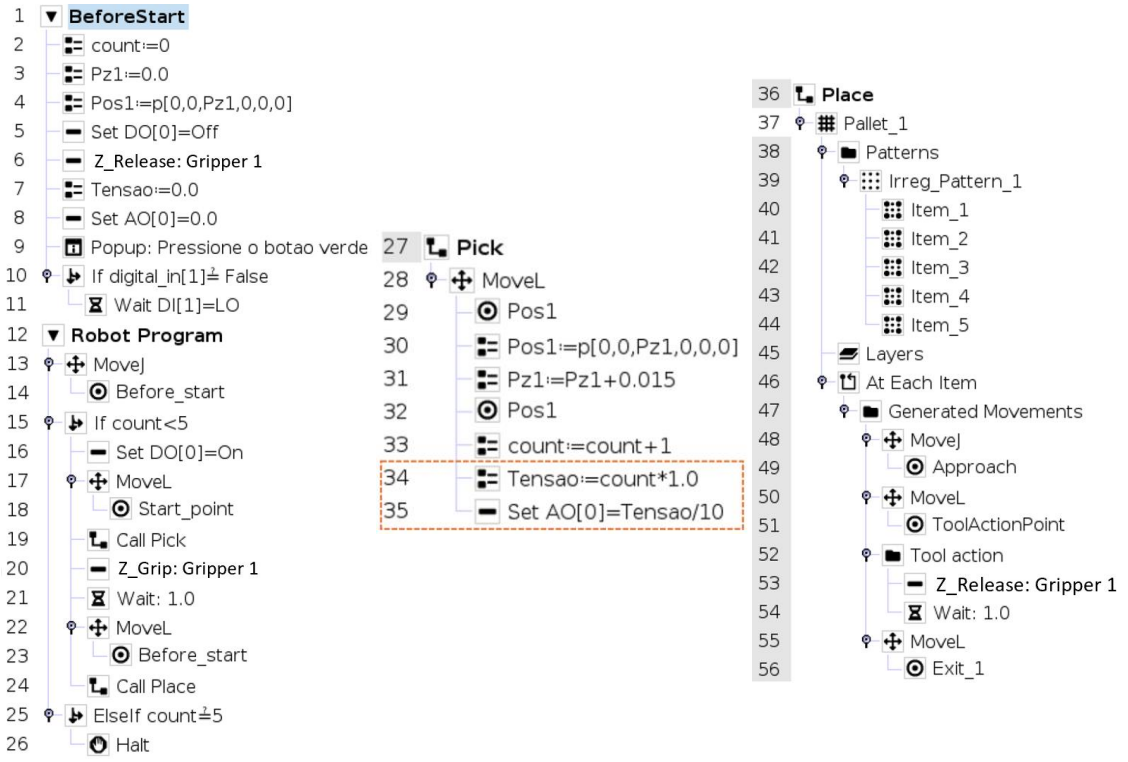


Figura 139 - Apresentação da secção Before Start

Passo 3 – Alteração do subprograma Pick

A alteração da programação ocorre no subprograma *Pick*, imediatamente após o incremento da variável contador, como se observa na Figura 140.



(a) (b) (c)

Figura 140 - Solução do exercício 4: Secção *BeforeStart* e corpo do programa principal (a); Subprograma de recolha *Pick* (b); Subprograma de posicionamento *Place* (c)

Como se demonstra na Figura 140, para a resolução do exercício devem-se acrescentar duas instruções:

- A primeira atribui à variável *Tensao* o valor do contador convertido para número real (Tensao := count * 1.0);
- A segunda envia este valor para a saída analógica (Set AO[0] := Tensao/10). A operação de divisão por 10 assegura que o valor de tensão transmitido ao voltímetro se encontra dentro da faixa operacional adequada, estabelecendo assim a relação matemática pretendida entre o número de ciclos executados e a medição do dispositivo.

Para a configuração do valor da saída analógica é necessário inserir uma instrução Set e parametrizá-la. Neste caso pretende-se que o valor do voltímetro (analog_out[0]) varie em função do valor do contador. Na Figura 141 apresenta-se a configuração da instrução Set que permite atribuir um valor variável a uma saída analógica.

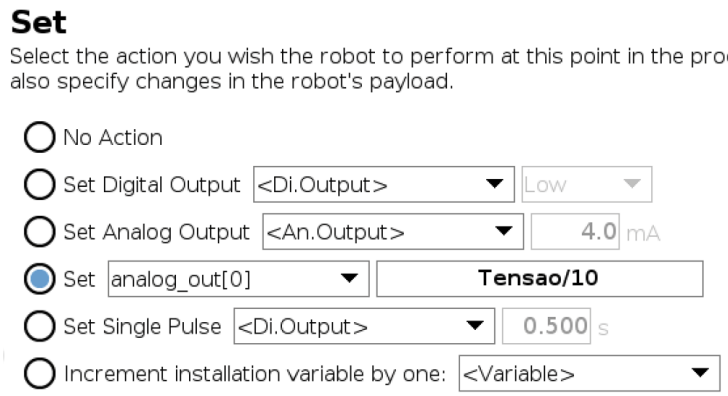


Figura 141 - Parametrização da instrução Set para atribuir valores variáveis a saídas analógicas

Passo 4 – Teste e validação

Testar o programa.

Exercício 8 – Paletização de 10 peças

O objetivo do exercício consiste em realizar a paletização de dez peças que se encontram empilhadas na zona A1 e A2. O robô deverá realizar operações de *Pick and Place* para transferir as peças da zona de recolha (zona A1 e A2) para a zona de paletização (posições numeradas de 1-10), de forma ordenada e repetitiva. A tarefa deverá ser executada até que dez peças sejam colocadas na zona de paletização, parando automaticamente após a última colocação. Na Figura 142 demonstra-se a organização da superfície de trabalho necessária para a realização do exercício.

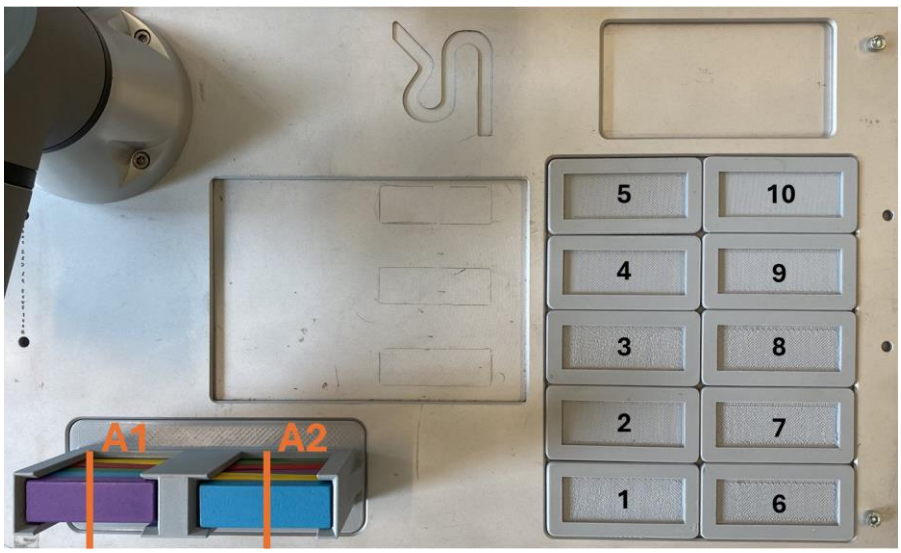


Figura 142 - Superfície de trabalho para a realização do exercício

Requisitos do exercício:

1. Após o sinal de início, o robô deve deslocar-se para a posição acima da zona A1, que deverá ser definida como *waypoint* *antes_recolha1*;

2. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher;
3. Acionar o *gripper* para agarrar a peça;
4. Aguardar 1 segundo;
5. Efetuar o movimento linear inverso até à posição *antes_recolha1*;
6. Transportar a peça até à zona B e colocá-la na próxima posição livre (de 1 a 10);
7. Desativar o *gripper* para largar a peça;
8. Aguardar 1 segundo;
9. Incrementar o contador de peças paletizadas;
10. Repetir o ciclo até que as 5 peças que se encontram na zona A1 tenham sido colocadas;
11. Após a colocação das primeiras 5 peças, o robô deve deslocar-se para a posição acima da zona A2, que deverá ser definida como *waypoint* *antes_recolha2*;
12. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher;
13. Acionar o *gripper* para agarrar a peça;
14. Aguardar 1 segundo;
15. Efetuar o movimento linear inverso até à posição *antes_recolha2*;
16. Transportar a peça até à zona B e colocá-la na próxima posição livre (de 1 a 10);
17. Desativar o *gripper* para largar a peça;
18. Aguardar 1 segundo;
19. Incrementar o contador de peças paletizadas;
20. Relacionar o valor do voltímetro com o do contador;
21. Após a colocação da 10ª peça, o programa deve terminar automaticamente.

Metodologia de resolução

Este exercício pode ser resolvido de diferentes formas. Para os subprogramas *Recolha 1* e *Recolha 2* pode-se utilizar a instrução Seek na função de desempilhamento (solução 1 do exercício 6) ou através de variáveis associadas a instruções de movimento como foi apresentado na solução 2 do exercício 6. Para o subprograma de posicionamento pode-se utilizar a instrução *Palletizing* e definir as 10 posições de colocação ou pode-se utilizar instruções de *Direction* e realizar movimentos nas coordenadas x e y, em relação a um ponto de referência (por exemplo o ponto central da placa).

Neste caso expõe-se a metodologia de resolução adotada que consistiu na utilização de variáveis para os subprogramas de recolha e instruções de direção para o subprograma de posicionamento.

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e com o nome *exercicio 8*.

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, definir (Figura 143):

- As variáveis:
    - *count* (contador) com valor inicial = 0;
    - Pz1 = 0.0;
    - Pz2 = 0.0;
    - Pos1 = p[0, 0, Pz1, 0, 0, 0];
    - Pos2 = p[0, 0, Pz2, 0, 0, 0];
    - tensao = 0.0
- O *gripper* com as garras abertas;
- O valor inicial do voltímetro de 0.0.

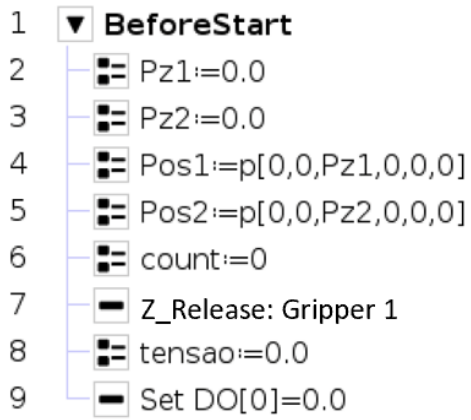


Figura 143 - Configuração da secção Before Start

Passo 3 – Criação de subprogramas

Criar dois subprogramas de recolha:

- *Recolha 1*;
- *Recolha 2*.

Criar um subprograma de posicionamento:

- *Pos.*

Passo 4 – Configuração dos subprogramas de recolha

Para configurar estes subprogramas é necessário seguir a metodologia utilizada na solução 2 do exercício 6. Na Figura 144 demonstra-se a programação desenvolvida para estes subprogramas. De notar que nesta secção já se encontra associado o valor do voltímetro ao valor do contador.

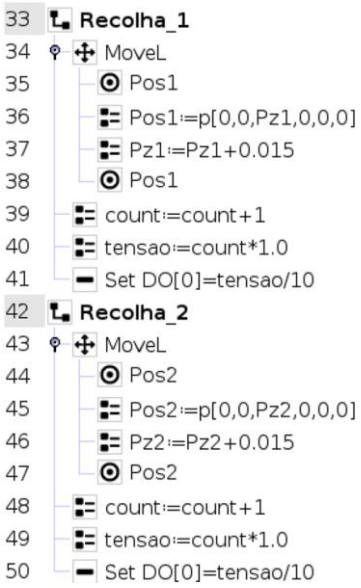


Figura 144 - Configuração dos subprogramas de recolha

Passo 5 – Configuração do subprograma de posicionamento

Para a configuração do subprograma de posicionamento (Pos), expõe-se uma metodologia que se baseia no deslocamento em três direções, tendo sempre como referência um ponto inicial e o valor do contador. Assim, cada valor do contador é associado a uma instrução condicional (If, Figura 145) que define a posição de destino da peça. Alternativamente, também é possível utilizar a instrução Switch, que permite estruturar o código de forma mais organizada e eficiente.

A sequência de posicionamento deve obedecer rigorosamente aos requisitos do enunciado. Dessa forma, quando o contador assumir o valor 1, a peça deve ser direcionada para a posição 1; quando assumir o valor 2, a peça deve ser transportada para a posição 2; e assim sucessivamente, até completar o posicionamento da décima peça. Este procedimento garante que todas as peças sejam corretamente alocadas nas respetivas posições de forma sistemática e ordenada.

```
51  Pos
52  If count=1
53      <empty>
54  If count=2
55      <empty>
56  If count=3
57      <empty>
58  If count=4
59      <empty>
60  If count=5
61      <empty>
62  If count=6
63      <empty>
64  If count=7
65      <empty>
66  If count=8
67      <empty>
68  If count=9
69      <empty>
70  If count=10
71      <empty>
```

Figura 145 - Apresentação do subprograma *Pos* antes da configuração das condicionais

Para o posicionamento das peças com as instruções *Direction* é necessário ter conhecimento das distâncias em **x** e **y** entre as placas, bem como da orientação do sistema de eixos. Na Figura 146 apresentam-se todas as medidas necessárias e a representação completa do sistema de eixos, servindo de base para a correta configuração dos movimentos.

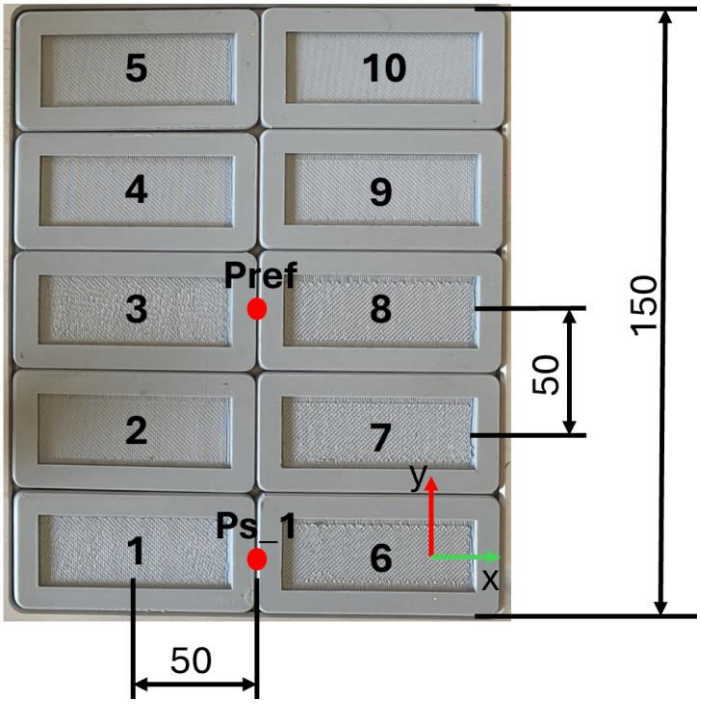


Figura 146 - Distância entre as posições e orientação do sistema de eixos

Para iniciar a configuração do subprograma ***Pos***, é fundamental definir previamente um ponto de referência (*Pref*, representado na Figura 146), situado a 100 mm da placa, de forma a evitar eventuais colisões ou problemas de trajetória. A partir deste ponto, programam-se as instruções *Direction* considerando as distâncias em **x** e **y** entre as posições de colocação e o ponto de referência.

Na configuração da **posição 1**, o primeiro passo consiste em inserir uma instrução *Direction* e selecionar a direção correspondente ao movimento pretendido. Neste caso, opta-se pelo movimento em **Y-**, com o sistema de eixos da **Base** selecionado (Figura 147), permitindo o deslocamento ao longo do eixo **y**, desde o ponto de referência até à posição 1.

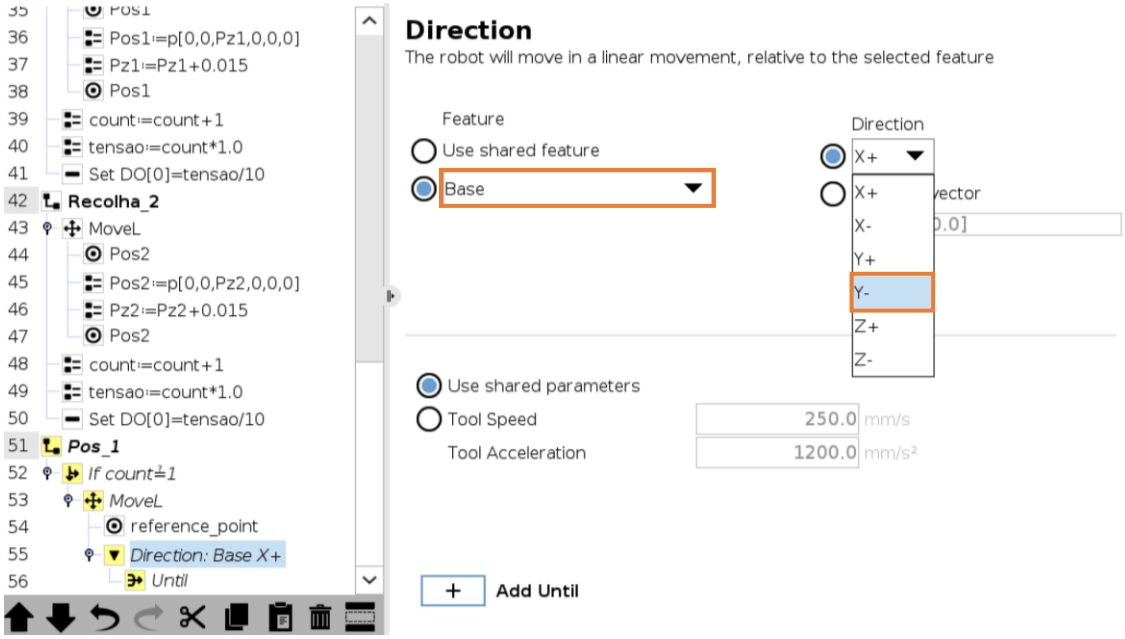


Figura 147 - Configuração da instrução Direction para a posição 1

De seguida, é necessário definir o tipo de restrição a aplicar ao movimento. Neste caso, seleciona-se a opção ***Distance*** (Figura 148), uma vez que o objetivo é deslocar o robô desde o ponto de referência até ao ponto central da posição 1.

Para concluir esta etapa, insere-se o valor da distância a percorrer, que neste caso corresponde a **100 mm**.

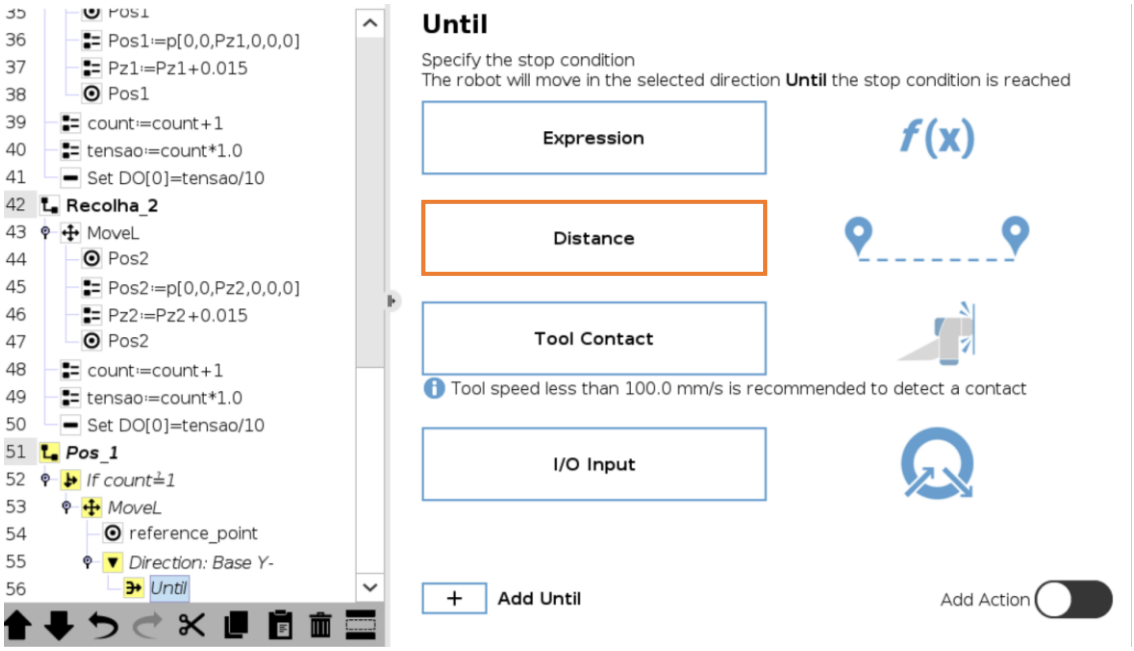


Figura 148 - Configuração da secção Until da instrução Direction para a posição 1

Para concluir esta etapa, insere-se o valor da distância a percorrer, que neste caso corresponde a 100 mm (Figura 149).

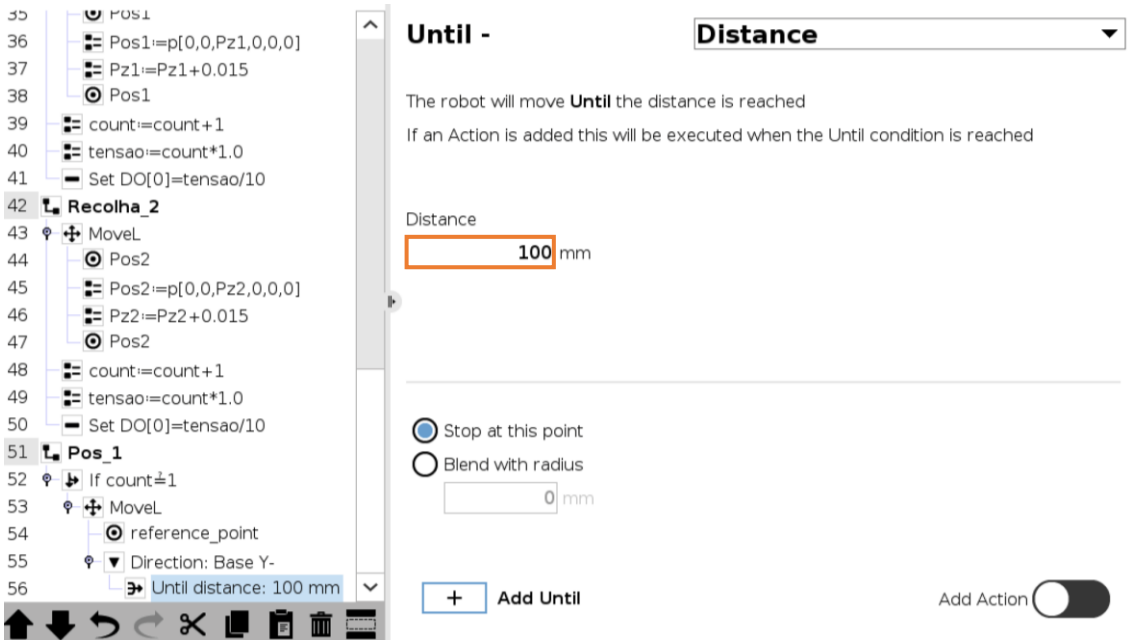


Figura 149 - Configuração da secção Until no modo Distance (para a posição 1)

A configuração do deslocamento no eixo **x** segue a mesma lógica adotada para o eixo **y**. Já o eixo **z** pode ser programado com uma restrição diferente: neste caso, utiliza-se a opção **Tool Contact** na secção *Until* da instrução *Direction* (Figura 150). Assim, a ferramenta desloca-se linearmente ao longo do eixo **z** até detetar uma força de contacto, que corresponde ao momento em que a peça transportada toca na placa de posicionamento.

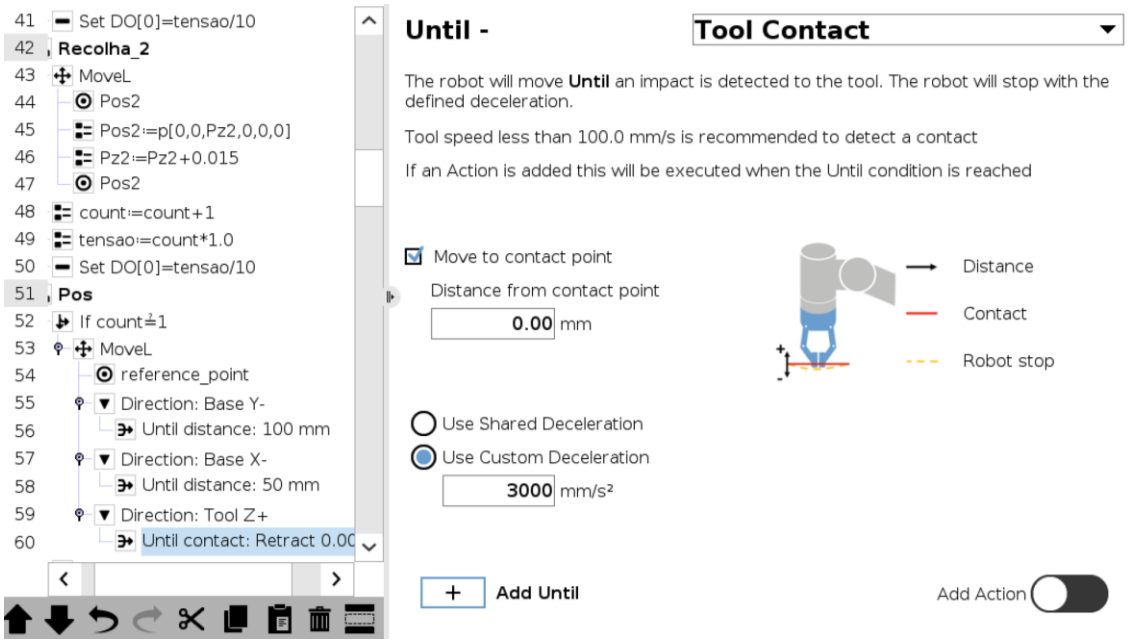


Figura 150 - Configuração da secção Until no modo Tool Contact

A Figura 151 apresenta a programação completa desenvolvida para o posicionamento da primeira peça na posição 1. Após esta etapa, e logo após o acionamento do gripper para libertar a peça, é essencial incluir uma instrução de movimento até um ponto de segurança, evitando que o gripper colida com a peça ao deslocar-se para a posição de recolha. Este ponto de segurança, denominado ***Ps_1*** (Figura 146), pode ser reutilizado para diferentes posições e deve estar localizado acima da placa, por exemplo, a **100 mm no eixo z**.

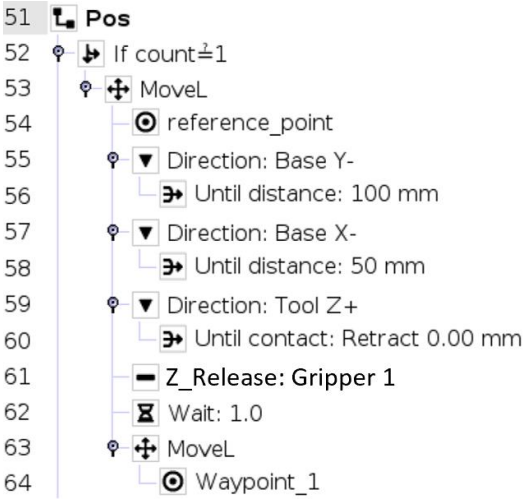


Figura 151 - Apresentação da programação desenvolvida para o subprograma *Pos* (posição 1)

Por fim, para completar a programação do subprograma ***Pos***, a metodologia aplicada na posição 1 deve ser replicada para todas as restantes posições, ajustando apenas as distâncias em **x** e **y** entre o ponto de referência e cada posição de colocação das peças.

Figura 152 - Programação das posições 2 a 10

Passo 6 – Configuração do programa principal

Para concluir a programação do exercício, é necessário desenvolver as instruções e restrições correspondentes ao programa principal. Nesta etapa devem ser definidas as condições que orientam o robô para as zonas de recolha das peças, conforme os valores assumidos pela variável *count:*

- Se $count < 5 \rightarrow$ recolher as peças na zona de recolha 1;
- Se $count \leq 5$ e $count < 10 \rightarrow$ recolher as peças na zona de recolha 2;
- Se $count = 10 \rightarrow$ mover o robô para a posição *antes_recolha1* e terminar o programa.

A solução final desenvolvida para esta secção do *Robot Program* encontra-se representada na Figura 153.

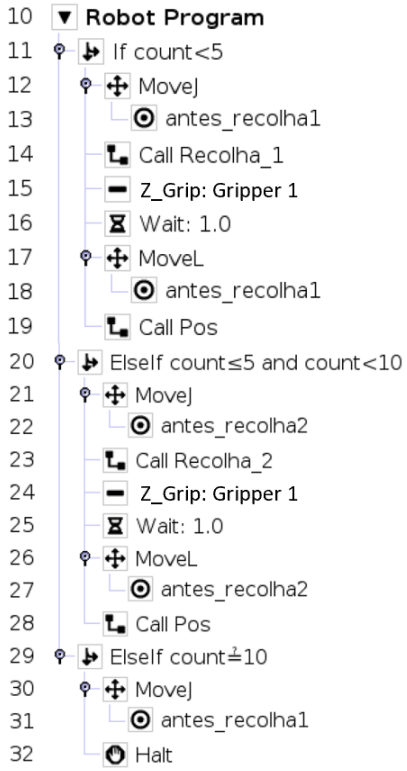


Figura 153 – Configuração da secção *Robot Program*

Passo 7 – Teste e validação

Testar o programa.

Exercício 9 – Controlo da velocidade do robô com AI[1]

O objetivo deste exercício é desenvolver uma aplicação capaz de controlar a velocidade de movimento do robô UR3e através da leitura de um valor analógico proveniente de um potenciómetro. O valor lido deve ser convertido numa fração de velocidade, o que permite ao operador ajustar dinamicamente a velocidade de execução do robô.

O robô deverá realizar um movimento contínuo entre dois pontos predefinidos, variando a sua velocidade conforme o valor do potenciómetro. O valor mínimo de velocidade deve ser de 0.1 (10% da velocidade máxima configurada) e o valor máximo de 1.0 (100% da velocidade máxima).

O programa deve estar continuamente a ler o valor do potenciómetro e a atualizar a velocidade em tempo real, de forma suave, garantindo que o robô não ultrapassa os limites de velocidade definidos.

Requisitos do exercício:

1. Definir as variáveis para armazenar o valor analógico lido (*ai_val*), a fração de velocidade (*vel_frac*) e o canal de leitura analógica (*ai_canal*);

2. Ler continuamente o valor do potenciómetro e convertê-lo para uma fração de velocidade no intervalo entre 0.1 e 1.0;
3. Garantir que valores abaixo de 0.1 são corrigidos para 0.1 e valores acima de 1.0 são corrigidos para 1.0;
4. Utilizar uma comunicação via *socket* para enviar o comando de alteração de velocidade ao controlador do robô;
5. Implementar um movimento repetitivo entre dois pontos definidos (*Waypoint 1* e *Waypoint 2*), cuja velocidade seja ajustada conforme a leitura do potenciómetro;
6. O sistema deve permanecer em funcionamento contínuo, atualizando a velocidade em tempo real até que seja interrompido manualmente.

Metodologia de resolução

Passo 1 – Criação de um programa

Criar um programa no PolyScope e denominá-lo como exercício 9.

Passo 2 – Configuração da secção *Before Start*

Criar as seguintes variáveis (Figura 154):

- ai_val = 0.0;
- vel_frac = 0.1;
- ai_canal = 1.

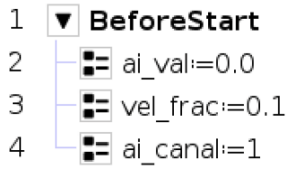


Figura 154 - Configuração da secção Before Start

Passo 3 - Configuração da secção *Robot Program*

Definir um movimento contínuo entre dois pontos (Figura 155).

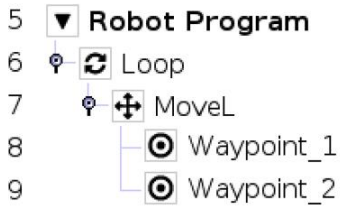


Figura 155 - Configuração da secção Robot Program

Passo 4 – Configuração da instrução *Thread*

1. **Inserção da instrução Thread**

Iniciar o processo adicionando uma instrução *Thread* ao programa, para garantir que a execução em paralelo permita monitorizar continuamente os parâmetros de velocidade.

2. **Abertura da porta 30002**

   Introduzir um *script* associado à abertura da porta **30002** do controlador (Figura 156), responsável pela receção dos valores do potenciómetro e pelo controlo da velocidade de movimento do robô.

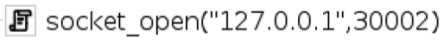


Figura 156 - Script associado à abertura da porta 30002 do controlador

3. **Leitura do potenciómetro**

   Definir as instruções necessárias para determinar o valor quase instantâneo do potenciómetro (Figura 157). Nesta fase devem ser estabelecidas restrições de velocidade que assegurem a conformidade com os parâmetros de segurança durante a execução do programa.

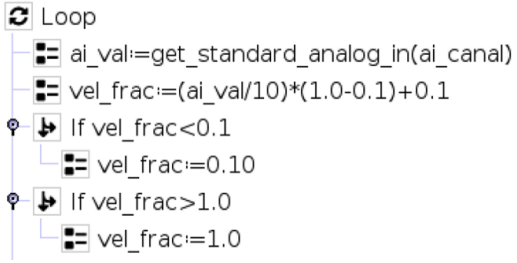


Figura 157 - Instruções que permitem determinar o valor quase instantâneo do potenciómetro

4. **Mapeamento entre velocidade e valor do potenciómetro**

   Introduzir os *scripts* responsáveis por estabelecer a relação direta entre o valor lido no potenciómetro e a velocidade aplicada ao robô (Figura 158), permitindo um controlo dinâmico e proporcional.

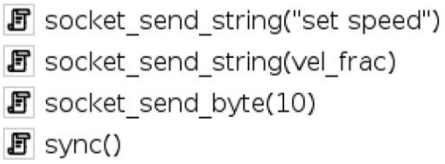


Figura 158 - Scripts associados ao mapeamento das velocidades

5. **Fecho da porta 30002**

   Por fim, incluir o *script* correspondente ao fecho da porta **30002** (Figura 159), garantindo o correto encerramento da comunicação e evitando falhas ou acessos indevidos após a execução do programa.

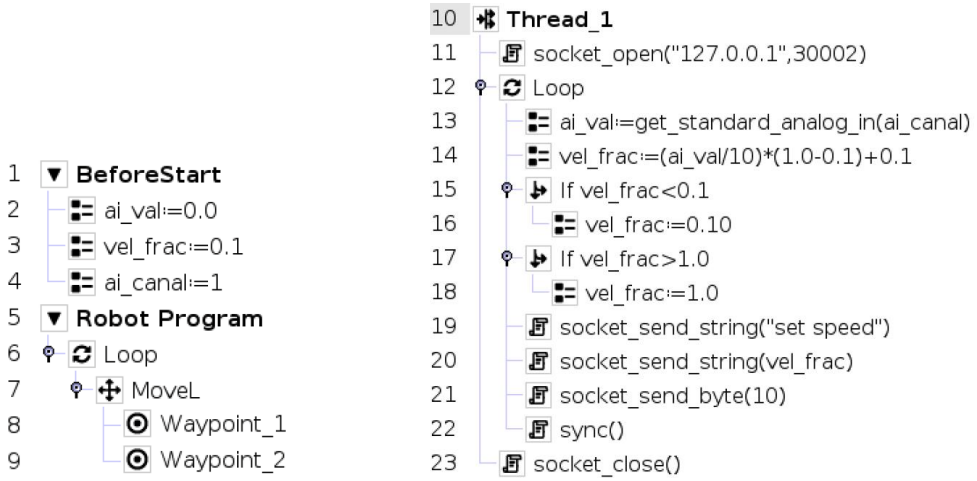
Figura 159 - Script associado ao fecho da porta 30002 do controlador

Passo 5 – Teste e validação

Testar o programa.

Explicação da solução

A solução apresentada para o controlo da velocidade do robô através de um potenciómetro é composta por duas partes principais: o programa de movimento do robô e uma *Thread* dedicada à leitura do valor analógico e ao envio da nova velocidade para o controlador. A Figura 160 apresenta-se a solução desenvolvida.



(a) (b)

Figura 160 - Solução do exercício 5: Secção *BeforeStart* e estrutura principal do programa (a); Subtarefa de controlo e atualização da velocidade em tempo real (b)

Através da análise a Figura 160 (a), verifica-se que antes do início do programa são atribuídas as variáveis cuja função é armazenar o valor analógico lido (*ai_val*), a fração de velocidade (*vel_frac*) e o número do canal de entrada analógica (*ai_canal*). Esta preparação garante que o programa é composto por uma base de dados consistente para receber e processar a informação do potenciómetro.

No programa principal, o robô executa um ciclo de movimentos lineares entre dois pontos definidos (*Waypoint 1* e *Waypoint 2*). A particularidade está no facto da velocidade destes movimentos não ser fixa. Esta é ajustada dinamicamente conforme o valor do potenciómetro.

Para que a velocidade seja atualizada em tempo real, foi implementada uma *Thread*, presente na Figura 160 (b), que executa as seguintes funções:

1. **Abertura de porta de comunicação com o controlador**

   É utilizada a função *socket_open("127.0.0.1", 30002)* para estabelecer uma conexão TCP/IP direta com o controlador do robô na porta 30002. Esta porta é específica para o envio de comandos de *script* ao robô, permitindo alterar parâmetros como a velocidade de movimento sem interromper o ciclo principal.

2. **Leitura do valor analógico**

A cada ciclo, o valor do potenciómetro é lido através da função *get_standard_analog_in(ai_canal)*. Este valor, entre 0 e 10 V, é convertido para um número decimal (*ai_val*) que serve de base para o cálculo da nova velocidade.

3. **Conversão para fração de velocidade**

   O valor lido é convertido para uma escala compreendida entre 0.1 e 1.0 através de uma expressão matemática. Estes limites são impostos para evitar velocidades extremamente baixas (que podem causar movimentos instáveis) ou demasiado altas (que podem comprometer a segurança).

4. **Envio de comandos via *socket***

   A comunicação com o controlador para alterar a velocidade é feita através do envio de *strings* formatadas através do *socket*. Utiliza-se *socket_send_string* para enviar o comando textual "*set speed*" seguido do valor calculado. Este método é mais avançado do que o uso de variáveis de programa, pois requer conhecimento da linguagem de URScript e da codificação de dados em formato *string* para controlo.

5. **Atualização em tempo real**

   A *thread* repete continuamente a leitura, conversão e envio do valor de velocidade enquanto o programa principal executa o movimento. Desta forma, qualquer alteração no potenciómetro reflete-se instantaneamente no comportamento do robô, sem necessidade de parar ou reiniciar o programa.

6. **Encerramento da comunicação**

   Quando o programa termina ou a *thread* é encerrada, a porta de comunicação é fechada com *socket_close()* para libertar o recurso e evitar conflitos em execuções futuras.

Esta abordagem demonstra como é possível integrar entradas analógicas externas para um controlo dinâmico e flexível do robô, combinando conhecimentos de programação em robótica, comunicação de rede e manipulação de *strings* específicas da Interface de Programação de Aplicações (IPA) do fabricante.

Exercício 10 – Utilização de funções para traçar percursos

Um robô colaborativo UR3e, equipado com uma ferramenta de gravação (por exemplo, uma caneta), deve realizar um percurso constituído por dois círculos e meio de 50 mm de raio, dispostos horizontalmente no mesmo plano de trabalho, conforme ilustrado na Figura 161.

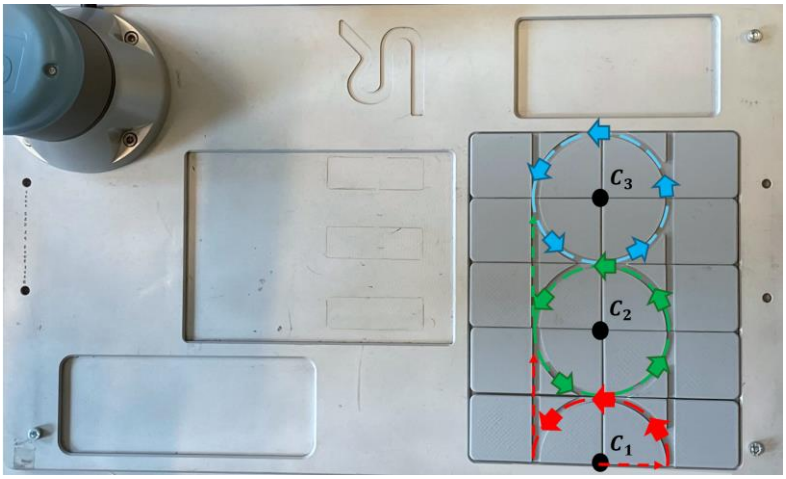


Figura 161 - Trajetória a percorrer

Requisitos do exercício:

1. Desenvolver um programa que permita desenhar dois círculos e meio, começando no C1 e terminando no C3.
2. Cada círculo deve ser aproximado por pontos sucessivos calculados por funções trigonométricas cos() e sin(), incrementando o ângulo θ de 10° em 10° (ou um valor definido pelo utilizador).

Metodologia de resolução

Passo 1 – Criação de um programa

Criar um programa no PolyScope e denominá-lo como exercício 10.

Passo 2 – Configuração da secção *Before Start*

Criar todas as variáveis necessária para a elaboração do programa e definir a posição do centro do círculo 1 como ponto de referência para o início do percurso (Figura 162).

```
1  ▼ BeforeStart
2      raio≔0.05
3      passo≔0.1
4      p0≔get_actual_tcp_pose()
5      X0≔p0[0]
6      Y0≔p0[1]
7      Z0≔p0[2]
8      rX0≔p0[3]
9      rY0≔p0[4]
10     rZ0≔p0[5]
11     centro_1≔p[X0,Y0,Z0,rX0,rY0,rZ0]
12     centro_2≔p[X0,Y0+0.1,Z0,rX0,rY0,rZ0]
13     centro_3≔p[X0,Y0+0.2,Z0,rX0,rY0,rZ0]
14     centro_atual≔centro_1
15     ponto≔centro_1
16     MoveJ
17         pos_centro_1
```

Figura 162 - Configuração da secção Before Start

Passo 3 – Configuração dos subprogramas de desenho

Cria dois subprogramas que permitam percorrer um círculo completo e meio círculo através de restrição angular (Figura 163).

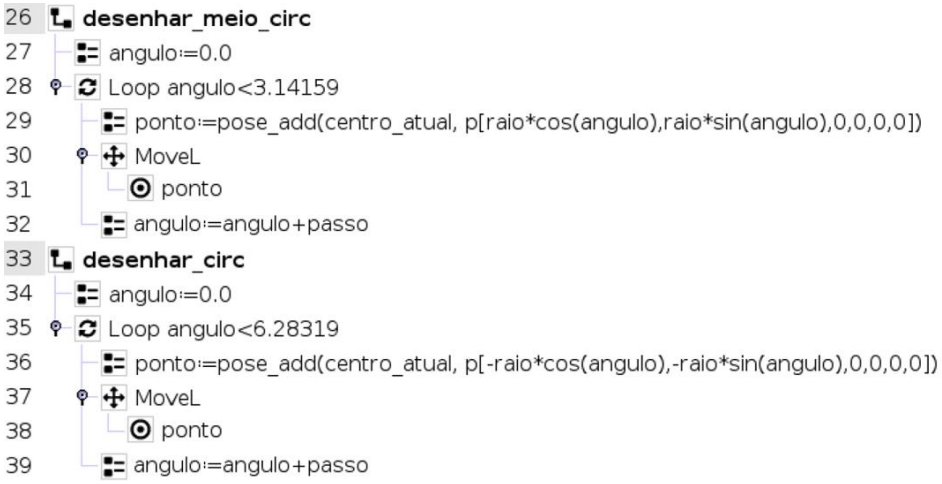


Figura 163 - Configuração dos subprogramas de desenho

Passo 4 – Configuração da secção Robot Program

Na árvore do programa principal inserir as instruções que definem o centro atual da ferramenta e chamadar os subprogramas de acordo com o trajeto que se pretende percorrer (Figura 164).

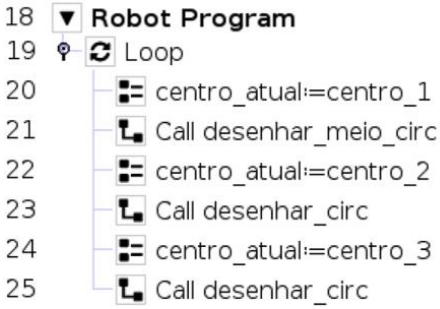


Figura 164 - Configuração da secção Robot Program

Passo 5 – Teste e validação

Testar o programa.

Explicação da solução

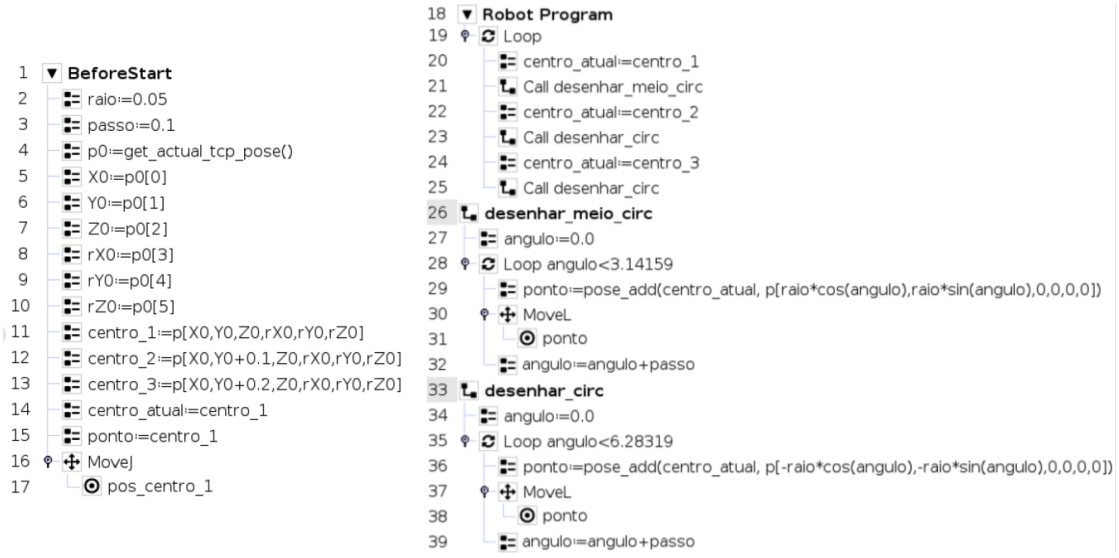
A solução desenvolvida baseia-se na definição matemática das coordenadas de pontos ao longo de um círculo, de forma a permitir que o TCP percorra trajetórias circulares de raio R em torno do *centro_atual*. A determinação da posição de cada ponto do círculo é feita através das expressões matemáticas (1) e (2), que calculam as coordenadas cartesianas x e y a partir do centro do círculo (X_C, Y_C), do raio R e do ângulo θ. O ângulo varia no intervalo $[0, 2\pi]$, sendo incrementado de forma constante com um passo $\Delta\theta$ equivalente a $10°$ (0.1745 rad).

$$x = X_c + R \,.\cos(\theta) \tag{1}$$

$$y = Y_c + R \,.\, \sin(\theta) \qquad (2)$$

Na implementação prática, representada na Figura 165 (a), são inicialmente definidos parâmetros como o raio do círculo e o incremento angular. Seguidamente, é obtida a posição atual do TCP, com a função *get_actual_tcp_pose*, armazenando as suas coordenadas e orientações. São também definidos diferentes centros possíveis para a execução do percurso, sendo selecionado um deles como referência inicial (*centro_atual*).

Os subprogramas ***desenhar_meio_circ*** e ***desenhar_circ***, apresentados na Figura 165 (b), traduzem as expressões matemáticas para o contexto da programação do robô. No subprograma ***desenhar_meio_circ***, o ângulo varia de 0 até π, o que permite percorrer exatamente metade da circunferência. Já no subprograma ***desenhar_circ***, o ângulo percorre todo o intervalo [0,2π], resultando na execução completa do círculo. Em ambos os casos, as coordenadas dos pontos são calculadas com base no centro de referência e no raio definido, recorrendo à função *pose_add* para gerar a nova posição do TCP, que é depois alcançada através do comando de movimento linear *MoveL*.



(a) (b)

Figura 165 - Solução do exercício 6: Secção *BeforeStart* (a); Programa principal e subprogramas que definem o trajeto (b)

A utilização de diferentes limites para o ângulo θ permite adaptar o percurso para desenhar arcos de qualquer dimensão, possibilitando a execução de meios círculos, quartos de círculo ou circunferências completas, de acordo com as necessidades da tarefa.

Após a definição das posições e da sequência de movimentos, o programa deverá ser executado para validar a trajetória. Durante esta fase, é recomendado que a operação inicie com uma velocidade de movimento reduzida, de modo a garantir maior precisão na trajetória e minimizar do risco de desvios. Após a validação, deve ser feita a afinação dos parâmetros de velocidade e aceleração para otimizar o tempo de execução, mantendo a precisão.

Exercício 11 – AI[1] a controlar a velocidade do percurso da ferramenta

Um robô colaborativo UR3e, equipado com uma ferramenta de gravação (por exemplo, uma caneta), deve executar um percurso constituído por dois círculos e meio com raio de 50 mm, dispostos horizontalmente no mesmo plano de trabalho, conforme ilustrado na Figura 166.

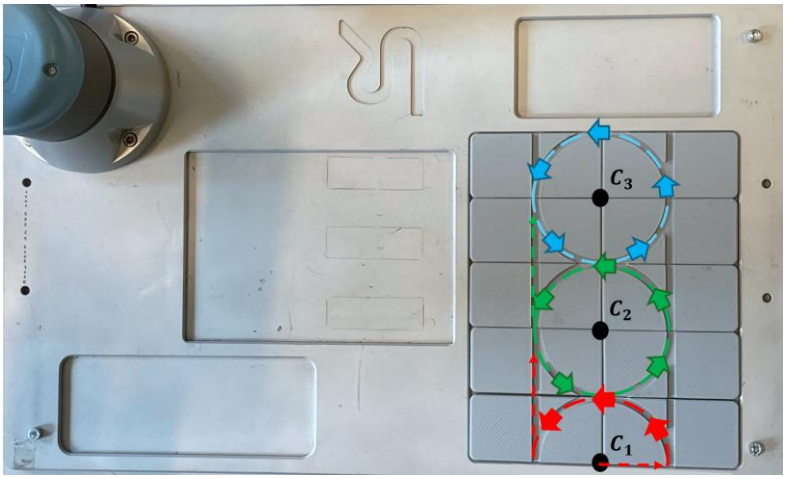


Figura 166 - Trajetória a percorrer

O robô deve realizar este percurso com velocidade ajustável em tempo real através da leitura analógica de um potenciómetro. O valor lido corresponde a uma fração da velocidade máxima, permitindo ao operador variar dinamicamente a velocidade de execução do robô durante o desenho.

A velocidade deve respeitar os seguintes limites:

- **Valor mínimo:** 0.1 (10% da velocidade máxima configurada);
- **Valor máximo:** 1.0 (100% da velocidade máxima configurada).

O programa deve efetuar a leitura contínua do potenciómetro e aplicar a conversão do valor para a fração de velocidade, garantindo que o robô nunca ultrapassa os limites definidos.

Requisitos do exercício:

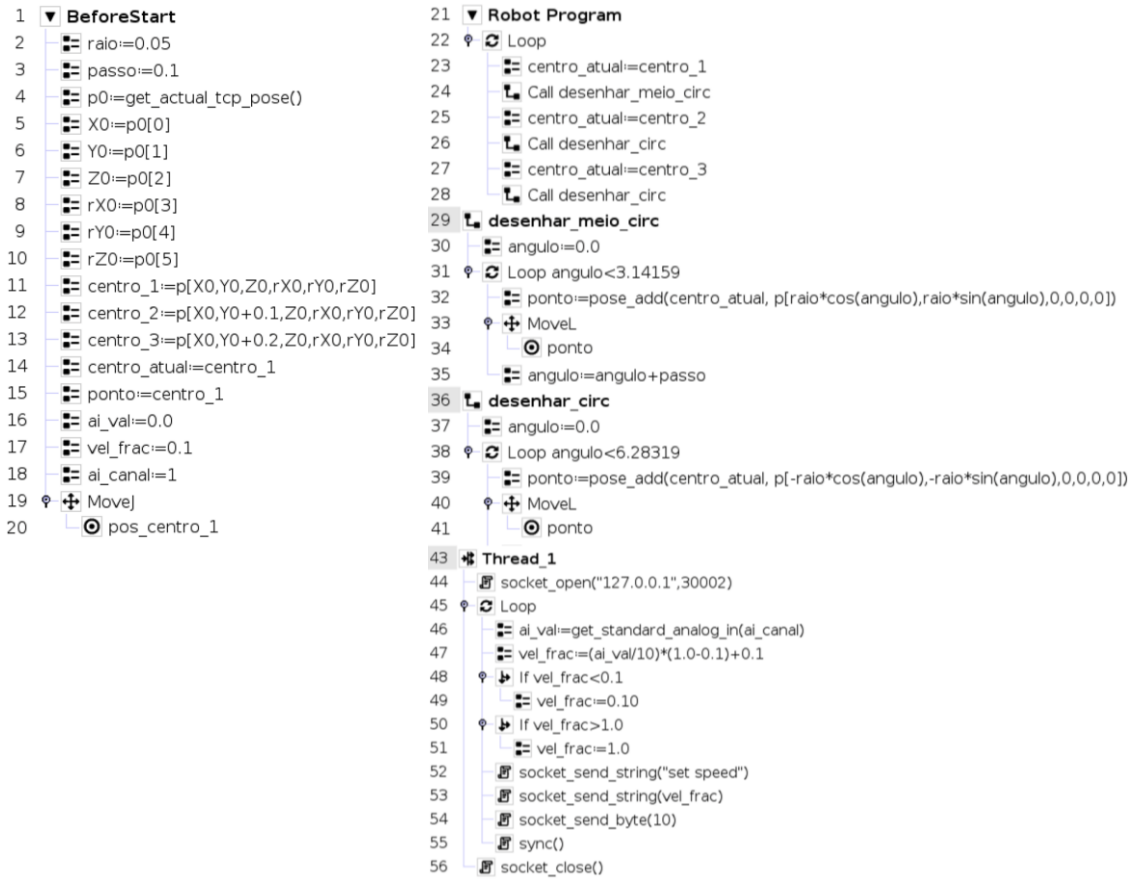
1. Desenvolver um programa que desenhe dois círculos e meio, começando no ponto C1 e terminando no ponto C3;
2. Aproximar cada círculo através de pontos sucessivos calculados com as funções trigonométricas cos() e sin(), incrementando o ângulo θ de 10° em 10° (ou outro valor definido pelo utilizador);
3. Definir as variáveis necessárias: *ai_val* (valor analógico lido), *vel_frac* (fração de velocidade) e *ai_canal* (canal de leitura analógica);
4. Ler continuamente o valor do potenciómetro e converter esse valor para uma fração de velocidade no intervalo [0.1, 1.0];

5. Corrigir valores inferiores a 0.1 para 0.1 e valores superiores a 1.0 para 1.0;
6. Implementar comunicação via *socket* para enviar ao controlador os comandos de ajuste de velocidade;
7. Executar o percurso circular definido, com movimento contínuo e velocidade ajustada em tempo real;
8. Manter o programa em funcionamento até ser interrompido manualmente pelo operador.

Metodologia de resolução

Passo 1 – Compilação de exercícios

Este exercício resulta da integração dos exercícios 9 e 10. Para a sua programação, é necessário compilar as soluções previamente desenvolvidas nestes exercícios e proceder à adaptação do tipo de movimento, de forma a garantir a coerência entre as lógicas de programação. Após esta integração e ajuste, deve-se obter a solução final representada na Figura 167.



(a) (b)

Figura 167 - Solução do exercício 11 com apresentação da secção *Before Start* (a), programa principal e subprogramas (b)

Passo 2 – Teste e validação

Testar o programa.

Exercício 12 - *Pick and Place* com inspeção manual de peças

O objetivo do exercício consiste na implementação de uma operação de pick and place, recorrendo à linguagem de programação do PolyScope e ao sistema de I/O digital do controlador do robô colaborativo UR3e. O robô deve recolher sucessivamente as peças existentes na zona A1 e transportá-las para as zonas de aceitação ou rejeição.

Pretende-se que o robô movimente as 5 peças presentes em A1 e as transporta até às zonas de aceitação e de rejeição. O robô inicia o ciclo de trabalho apenas quando o botão verde é pressionado. De seguida desloca-se para o ponto de recolha, agarra a peça com o atuador (gripper) e transfere-a para a zona pretendida. De duas em duas peças transportadas, o robô deve levá-las previamente até à zona de inspeção, onde permanecerá até existir decisão do operador. Essa decisão é introduzida através de uma variável binária que identifica se a peça está conforme (aceite) ou não conforme (rejeitada). As peças inspecionadas devem ser colocadas na zona correspondente de aceitação ou rejeição, consoante a decisão do operador. As restantes peças, que não são inspecionadas, devem ser sempre transportadas para a zona de aceitação.

Na Figura 168 expõe-se a disposição da superfície de trabalho necessária para a realização do exercício.

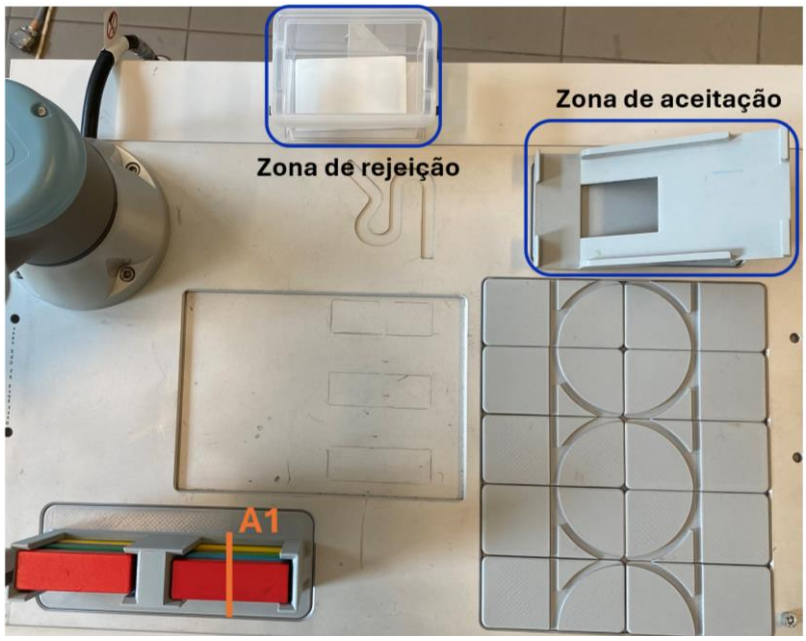


Figura 168 - Representação da superfície de trabalho

A colocação das peças na zona de aceitação deve respeitar a orientação e posicionamento exemplificados na Figura 169 para garantir que a peça desliza uniformemente entre as paredes laterais da rampa.

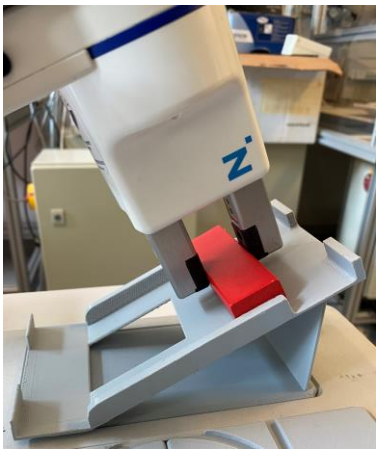


Figura 169 - Zona de colocação de peças

Requisitos do exercício:

1. Antes de iniciar o programa o utilizador deve garantir que o *gripper* está aberto, os LEDs estão desligados e o contador está com um valor inicial de zero;
2. A execução do programa deve iniciar-se apenas quando o botão verde (DI1) for pressionado;
3. Após o sinal de início, o robô deve deslocar-se para a posição acima da zona A, que deverá ser definida como *waypoint* *before_start*;
4. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher;
5. Acionar o *gripper* para agarrar a peça;
6. Aguardar 1 segundo;
7. Efetuar o movimento linear inverso até à posição *before_start*;
8. A cada duas peças transportadas, encaminhar a peça seguinte para a zona de inspeção;
9. Na zona de inspeção:
    1. Aguardar pela decisão do operador (variável binária: aceite / rejeitada);
    2. Transportar a peça para a zona de aceitação, caso seja aceite;
    3. Transportar a peça para a zona de rejeição, caso seja rejeitada;
10. Todas as peças que não passam pela inspeção devem ser transportadas diretamente para a zona de aceitação (zona A1);
11. Desativar o *gripper* para largar a peça;
12. Aguardar 1 segundo;
13. Incrementar o contador de peças transportadas;
14. Repetir o ciclo até que cinco peças tenham sido transportadas;
15. Durante todo o processo, manter o LED branco (DO3) ligado;
16. Durante a fase de aceitação de peças manter o Led verde (D00) ligado;
17. Durante a fase de rejeição de peças manter o Led vermelho (D02) ligado;

18. Durante a fase de inspeção de peças manter o Led amarelo (D01) ligado;
19. Após a colocação da 5ª peça, o programa deve terminar automaticamente regressar à posição *before_start*.

Metodologia de resolução

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e com o nome *exercício 12*.

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, definir:

- As variáveis:
  - *count* = 0;
  - Pz1 = 0.0;
  - Pos1 = p[0, 0, Pz1, 0, 0, 0];
- O *gripper* com as garras abertas;
- Os *LEDs* em estado desligado;
- O *Popup* para aviso enquanto o robô espera que o botão verde seja acionado;
- Condição que verifique o estado do botão verde.

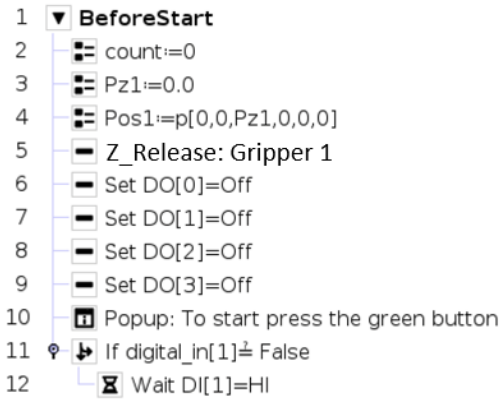


Figura 170 - Exposição da secção Before Start preenchida

Passo 3 – Criação de subprogramas

Devem ser criados quatro subprogramas, cada um associado a uma etapa específica do processo:

- ***Pick*** – responsável pela recolha das peças na zona A1;
- ***Inspection workpiece*** – responsável pelo transporte das peças até à zona de inspeção;

- ***Accepted workpiece*** – responsável pelo transporte das peças para a zona de aceitação;
- ***Rejected workpiece*** – responsável pelo transporte das peças para a zona de rejeição.

Passo 4 – Configuração dos subprogramas

A parametrização de cada subprograma deve contemplar, de forma sequencial, todas as ações necessárias para executar corretamente a tarefa atribuída.

- Subprograma *Pick* (Figura 171):
    - Desenvolver uma aplicação de recolha automática de peças;
    - Ativar o gripper para agarrar a peça;
    - Incrementar o contador (para registar o número de peças recolhidas) ;
    - Executar um movimento linear do robô até à posição ***before_start***, para garantir uma transição segura para a próxima operação.

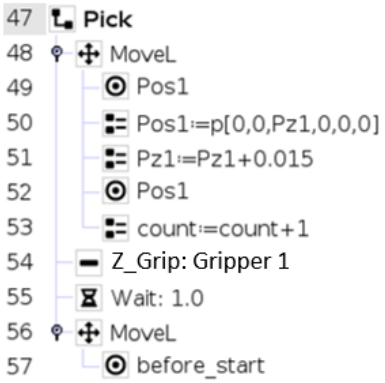


Figura 171 - Exposição da programação desenvolvida para o subprograma *Pick*

- Subprograma *inspection_workpiece* (Figura 172):
    - Ativar o *Led* amarelo (DO[1]);
    - Mover o robô para a posição de inspeção;
    - Inserir Popup com a mensagem *"Approve workpiece?".*

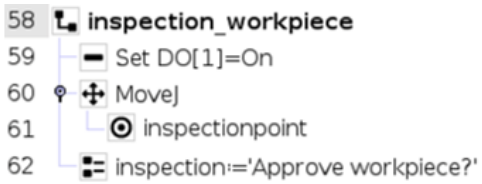


Figura 172 - Exposição da programação desenvolvida para o subprograma *inspection_workpiece*

- Subprograma *accepted_workpiece* (Figura 173):
    - Ativar o *Led* verde (DO[0]);
    - Mover o robô para a posição *before_exit_1*;
    - Mover linearmente o robô para a posição *exit_1*;

- Libertar a peça;
- Mover linearmente o robô para a posição *before_exit_1*.

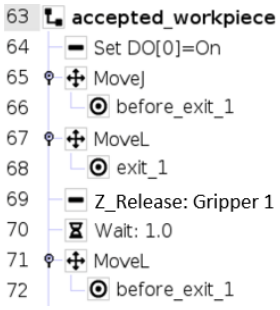


Figura 173 - Exposição da programação desenvolvida para o subprograma *accepted_workpiece*

- Subprograma *rejected_workpiece* (Figura 174):
  - Ativar o Led vermelho (DO[2]);
  - Mover o robô para a posição *before_exit_2*;
  - Mover linearmente o robô para a posição *exit_2*;
  - Libertar a peça.

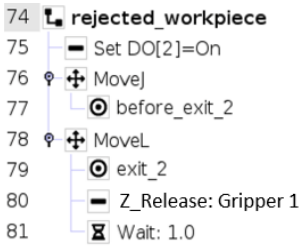


Figura 174 - Exposição da programação desenvolvida para o subprograma *rejected_workpiece*

Passo 5 – Configuração da secção Robot Program

É necessário desenvolver uma lógica de programação que combine diferentes **instruções de controlo de fluxo** (*Switch*, *If* e *ElseIf*) para gerir de forma eficiente os movimentos e ações definidos no enunciado do exercício. Esta abordagem permite estruturar as decisões de acordo com as condições impostas e assegurar que o robô executa corretamente cada tarefa em função do estado do processo.

A metodologia de programação adotada para a secção **Robot Program**, que integra esta lógica de controlo e cumpre todos os requisitos do exercício, encontra-se representada na Figura 175.

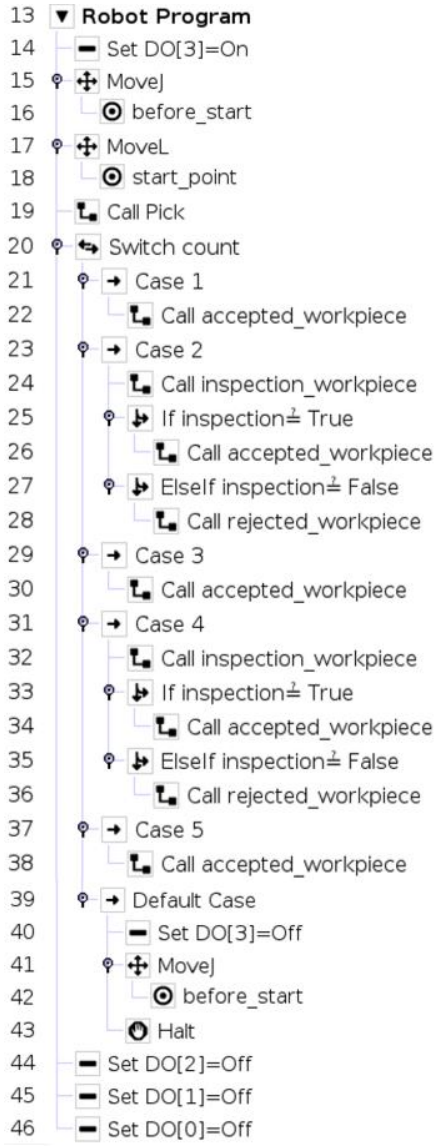


Figura 175 - Exposição da programação desenvolvida para a árvore do programa principal

Passo 6 – Teste e validação

Testar o programa.

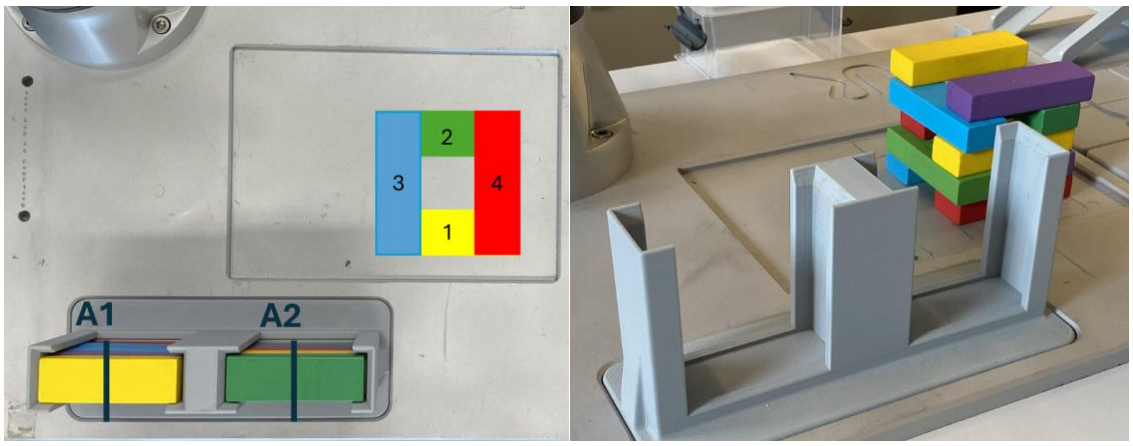
Exercício 13 – Empilhamento de peças

O objetivo deste exercício é desenvolver uma aplicação de manipulação e empilhamento de peças no UR3e. O robô deve recolher sequencialmente as peças existentes nas zonas A1 e A2, depositando-as nas posições 1 a 4, de modo a criar cinco camadas sobrepostas.

Na primeira fase, o robô deve desempilhar as 5 peças da zona A1 e colocá-las ordenadamente nas posições 1, 2, 3 e 4, criando a base do empilhamento. Na segunda fase, o robô deve desempilhar as 5 peças da zona A2 e colocá-las nas mesmas posições (1, 2, 3 e 4), de forma a completar as cinco camadas sobrepostas. Na Figura 176 (a) é apresentada a superfície de

trabalho com as zonas A1 e A2 assinaladas, bem como as posições de empilhamento (1 a 4). Na Figura 176 (b) consta a solução pretendida.

Para a execução da tarefa, devem ser criados subprogramas específicos para as operações de recolha e posicionamento, permitindo modularidade e reutilização do código.



(a) (b)

Figura 176 - Representação do exercício 7 (a) Superfície de trabalho e (b) Solução pretendida

Requisitos do exercício:

1. Inserir as variáveis para a contagem de peças movimentadas e para o registo das alturas (*offsets* no eixo Z) das pilhas em A1 e A2;
2. Criar funções para recolher peças nas zonas A1 e A2, ajustando automaticamente a altura de recolha a cada iteração;
3. Criar funções para posicionar as peças nas posições 1, 2, 3 e 4;
4. Implementar um ciclo que, para cada peça, execute:
   - Movimento para a zona de recolha;
   - Ativação do *gripper* para agarrar a peça;
   - Transporte até à posição de destino;
   - Libertação da peça;
   - Incremento dos contadores de altura e peças colocadas.
5. Garantir que, após movimentar todas as peças de A1, o robô inicia automaticamente o desempilhamento de A2 até completar as 5 camadas finais;
6. Otimizar os movimentos para evitar trajetórias desnecessárias e reduzir o tempo de execução.

Metodologia de resolução

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e com o nome *exercicio 8*.

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, são definidas as variáveis para armazenar as alturas de recolha em A1 (*Pz1*) e A2 (*Pz2*), bem como as posições base *Pos1* e *Pos2* para cada local. O contador *count* inicia com o valor zero e o *gripper* é aberto para garantir que começa livre de peças.

Nesta secção definir (Figura 177):

- As variáveis:
    - *count* (contador) com valor inicial = 0;
    - Pz1 = 0.0;
    - Pz2 = 0.0;
    - Pos1 = p[0, 0, Pz1, 0, 0, 0];
    - Pos2 = p[0, 0, Pz2, 0, 0, 0].
- O *gripper* com as garras abertas.

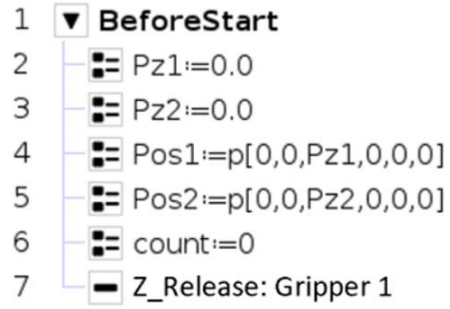


Figura 177 - Configuração da secção *Before Start*

Passo 3 – Criação de subprogramas

Criar dois subprogramas de recolha:

- *Recolha 1*;
- *Recolha 2*.

Criar quatro subprogramas de posicionamento:

- *Pos1;*
- *Pos2;*
- *Pos3;*
- *Pos4.*

Passo 4 – Configuração dos subprogramas de recolha

Os subprogramas de recolha movem o robô até à posição base da pilha, ajustam o eixo Z para compensar a altura já retirada e atualizam a variável de altura e o contador global. Na Figura 178 demonstra-se a programação destes subprogramas.

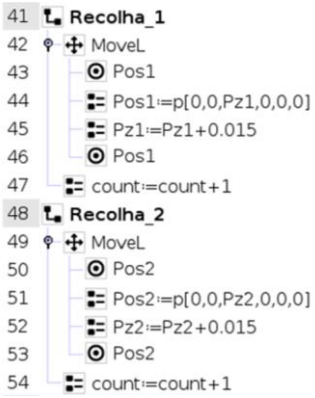


Figura 178 - Configuração dos subprogramas de recolha

Passo 5 – Configuração dos subprogramas de posicionamento

Os subprogramas de posicionamento movem o robô até à posição do empilhamento, ajustando a altura de acordo com a camada a ser formada (calculada pelo valor de *count*). A peça é libertada com o comando *Z_Release* quando a base da ferramenta deteta contacto. De seguida, o robô recua para evitar colisões.

Cada subprograma de posicionamento corresponde a uma posição específica de colocação da peça. Como existem quatro posições de empilhamento, devem ser criados quatro subprogramas, denominados Pos1, Pos2, Pos3 e Pos4. Assim, o subprograma Pos1 corresponde à posição 1, o Pos2 à posição 2, e assim sucessivamente. Para a configuração destes subprogramas, pode-se adotar uma lógica que considera que, a cada grupo de 4 peças, o robô realiza o empilhamento na mesma coordenada em x e y, mas em níveis diferentes no eixo z, ou seja, em patamares sucessivos.

Cada subprograma deve seguir a seguinte sequência de ações:

- Verificar o valor da variável *count* (condição de entrada);
- Definir um ponto de aproximação, localizado linearmente acima da posição de colocação (por exemplo, a 150 mm);
- Executar o movimento no eixo z até atingir a posição de depósito;
- Ativar a abertura do gripper para libertar a peça.

Este procedimento deve ser replicado para as quatro posições, garantindo a organização e o correto empilhamento das peças. Na Figura 179 apresenta-se a programação dos quatro subprogramas de posicionamento de peças, em função da metodologia descrita anteriormente.

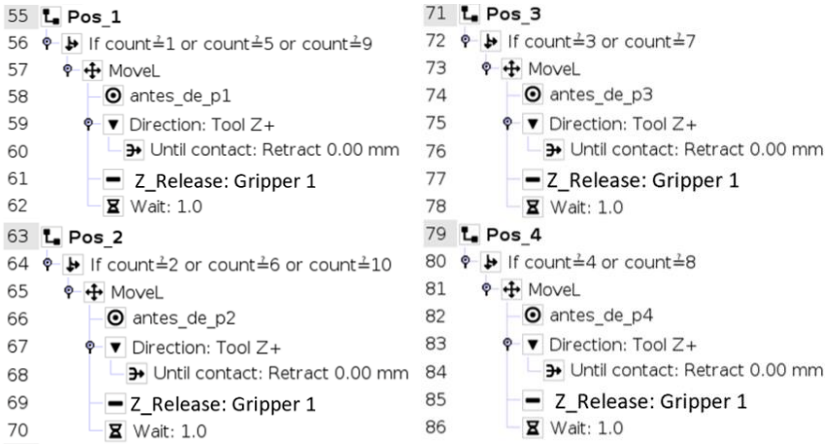


Figura 179 - Configuração dos subprogramas de posicionamento de peças

Passo 6 – Configuração do programa principal

Um conjunto de condições *If* e *Elseif* controlam as diferentes fases:

- Se $count < 5 \rightarrow$ o robô recolhe peças de A1 através do subprograma *Recolha_1*, ajustando a altura de recolha a cada iteração;
- Se $count \leq 5$ e $count < 10 \rightarrow$ o robô recolhe peças de A2 através do subprograma *Recolha_2*;
- Se $count = 10 \rightarrow$ mover o robô para a posição *antes_recolha1* e terminar o programa.
- Para cada peça recolhida, são chamados sequencialmente os subprogramas *Pos_1*, *Pos_2*, *Pos_3* e *Pos_4*, que depositam as peças nas posições de empilhamento.

No final deve-se obter a solução representada na Figura 180.

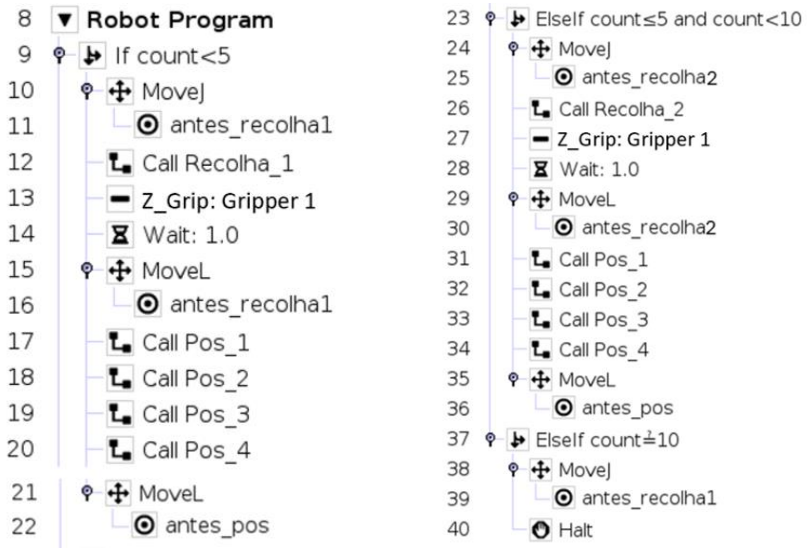


Figura 180 - Configuração do programa principal

Passo 7 – Teste e validação

Na execução do programa, é importante utilizar velocidades de movimento reduzidas para garantir maior precisão no posicionamento/empilhamento das peças. Dado que se recorre à função de posicionamento com deteção de contacto pela ferramenta, a utilização de velocidades mais baixas aumenta a sensibilidade na deteção. Desta forma, minimiza-se o risco de dano nas peças e assegura-se um posicionamento suave e controlado.

Referências

[1] ISO, «ISO/TS 15066:2016 Robots and robotic devices — Collaborative robots». Acedido: 10 de Dezembro de 2024. Disponível em: https://www.iso.org/standard/62996.html

[2] ISO, «ISO 10218-1:2025». Acedido: 1 de Setembro de 2025. Disponível em: https://www.iso.org/standard/73933.html

[3] ISO, «ISO 10218-2:2025». Acedido: 1 de Setembro de 2025. Disponível em: https://www.iso.org/standard/73934.html

[4] U. Robots, «UR3e». Acedido: 5 de Setembro de 2025. Disponível em: https://www.universal-robots.com/pt/produtos/ur3e/

[5] U. Robots, «Controlador». Acedido: 5 de Setembro de 2025. Disponível em: https://www.universal-robots.com/developer/hardware-and-motion/electrical-interfaces-control-box/

[6] U. Robots, «Teach Pendant». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/marketplace/products/01tP4000001uE4bIAE/

[7] U. Robots, «PolyScope 5». Acedido: 5 de Setembro de 2025. Disponível em: https://www.universal-robots.com/products/polyscope-5/

[8] Zimmer, «Gripper». Acedido: 11 de Agosto de 2025. Disponível em: https://www.zimmer-group.com/en-us/products/components/handling-technology/hrc-gripper/hrc-03/products/hrc-03-118505

[9] «Universal Robots». Acedido: 28 de Julho de 2025. Disponível em: https://www.universal-robots.com/pt/produtos/ur3-robot/

[10] U. Robots, «Control Boxes». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/developer/hardware-and-motion/electrical-interfaces-control-box/

[11] U. Robots, «Standard cable extension». Acedido: 29 de Julho de 2025. Disponível em: https://www.universal-robots.com/marketplace/products/01tP40000071NjSIAU/

[12] U. Robots, «Universal Robots». Acedido: 28 de Julho de 2025. Disponível em: https://www.universal-robots.com/br/blog/como-os-cobots-abordam-os-principais-desafios-da-producão-de-carros/

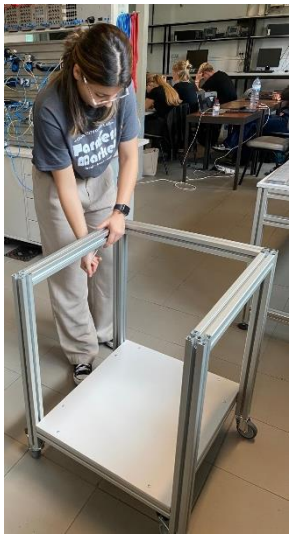
[13] U. Robots, «TCP». Acedido: 2 de Agosto de 2025. Disponível em: https://academy.universal-robots.com/modules/e-Series-core-track/English/module3/story_html5.html?courseId=2166&language=English&trainingId=5xCrknuMIKz

Apêndice C

Registo fotográfico do processo de montagem da bancada.

Etapas da montagem:

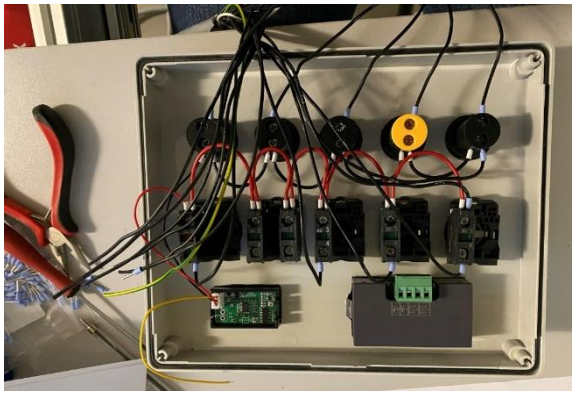
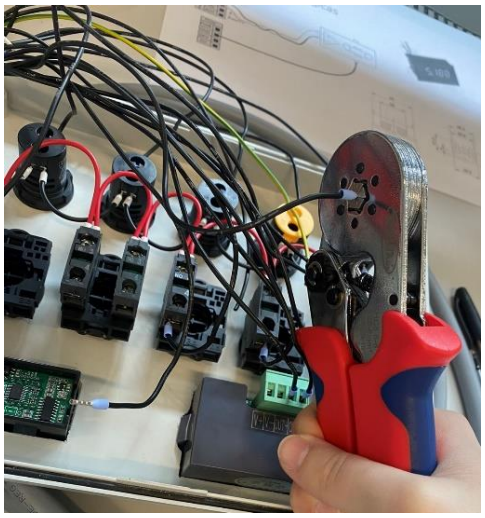
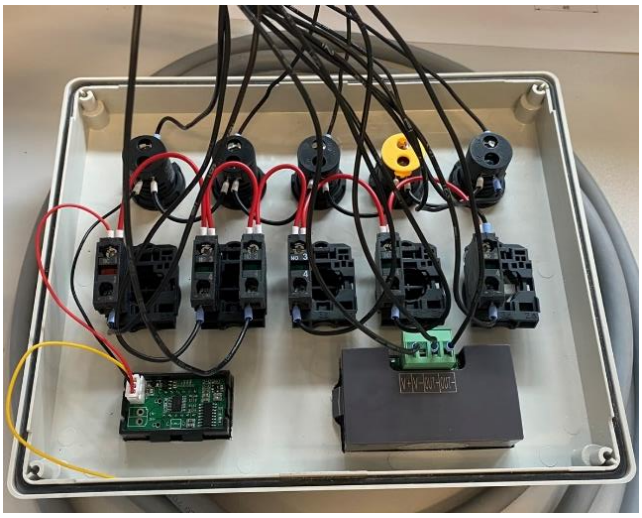
1. **Montagem da estrutura**



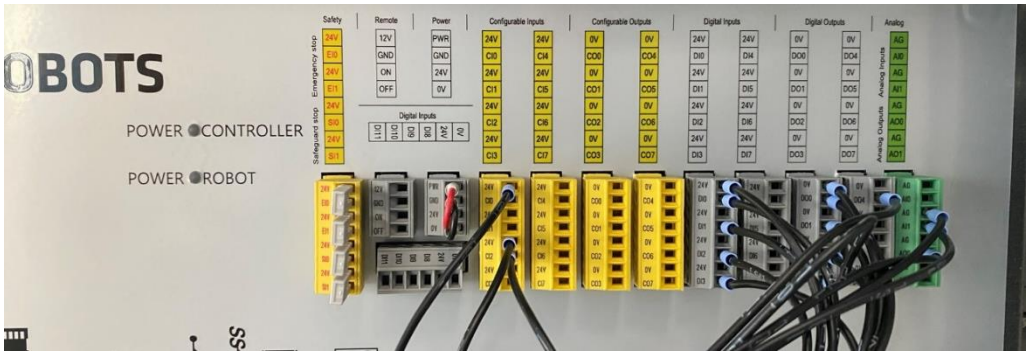

2. **Fixação do robô e periféricos**


3. **Montagem da caixa de I/Os digitais e analógicas**

4. **Ligações elétricas**
    a. **Na caixa de I/Os**

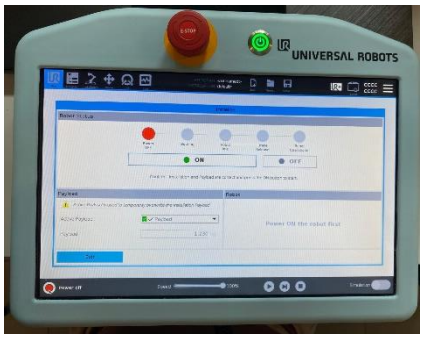





b. No controlador

5. Teste dos componentes

a. Ativação do braço robótico

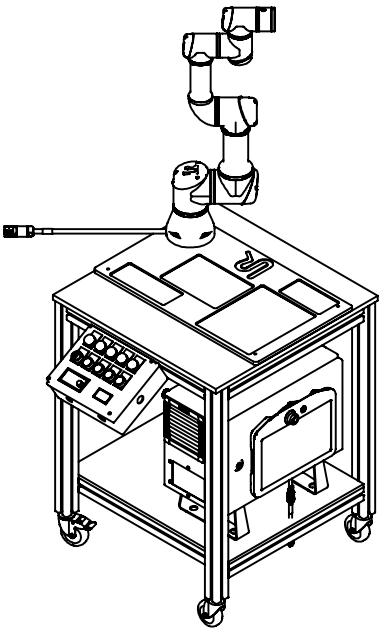


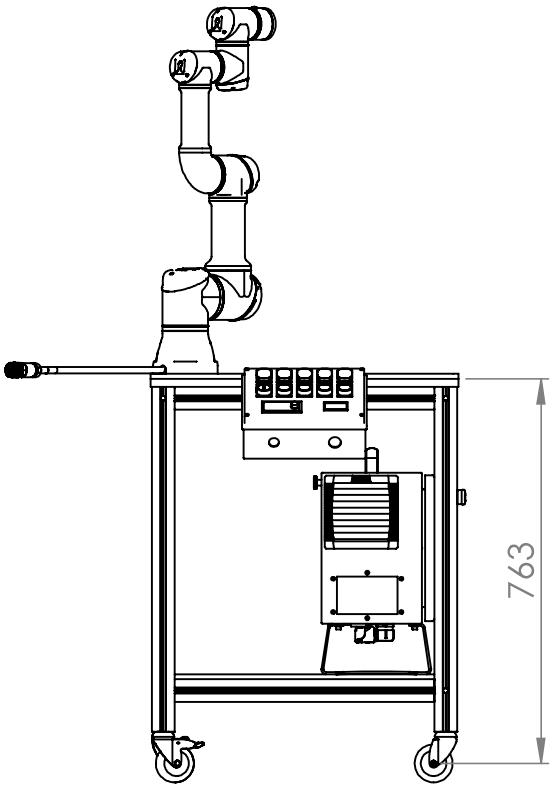
b. Verificação do funcionamento das entradas e saídas digitais e analógicas

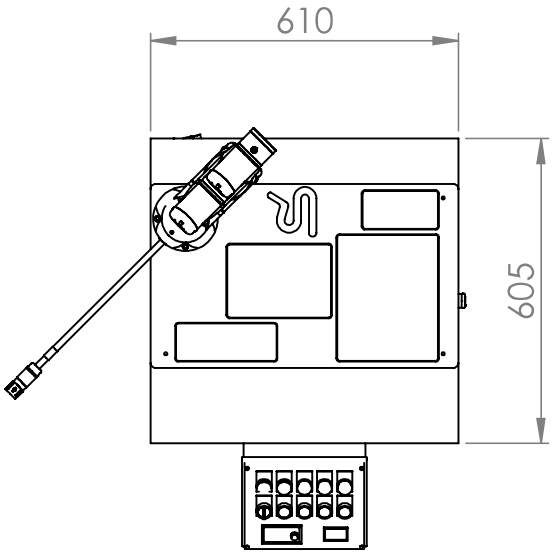
6. Resultado final

Apêndice D

Desenho de definição da bancada.







ESCALA

1:15

Pessoa responsável	Departamento responsável	Tipo de documento	Estado do documento			
1201027	www.dem.isep.ipp.pt	Desenho de definição	Publicado			
Proprietário legal		Título	Número			
DEPARTAMENTO DE ENGENHARIA MECÂNICA *INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO* *Rua Dr. António Bernardino de Almeida, 431* *4200-072 Porto*		Cotagem da bancada didática	1			
			Revisão	Data de edição	Lingua	Folha
			1	13/09/2025	PT	1/1

Apêndice E

Guião laboratorial.

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Departamento de Engenharia Mecânica

Ano letivo de 2024/2025

Laboratório de Automação

Guião de trabalho

Implementação de exercícios com robô colaborativo UR3e

Elaborado por: Margarida Pinto (1201027)

Porto, julho 2025

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

A robótica colaborativa tem vindo a assumir um papel central na modernização dos processos industriais, uma vez que promove uma maior eficiência, segurança e flexibilidade operacional.

É com base nestes pressupostos que no presente guião laboratorial se apresenta e desenvolvem uma série de exercícios baseados no robô colaborativo UR3e (*Universal Robots*), com integração de lógica de posicionamento cartesiano, manipulação de objetos e controlo sequencial com recurso ao software *PolyScope*. Utilizando movimentos lineares e polares pretende-se reproduzir os movimentos necessários ao controlo de processos industriais do tipo *Pick and Place*, por exemplo. O *PolyScope* é um *software* de programação que se encontra inserido no *Teach Pendant* do controlador do UR e como tal, utilizado como ambiente de desenvolvimento integrado (IDE). Na Figura 1 apresenta-se o esquema da bancada projetada para o desenvolvimento dos exercícios propostos, evidenciando-se e representando-se todos os elementos que a constituem.

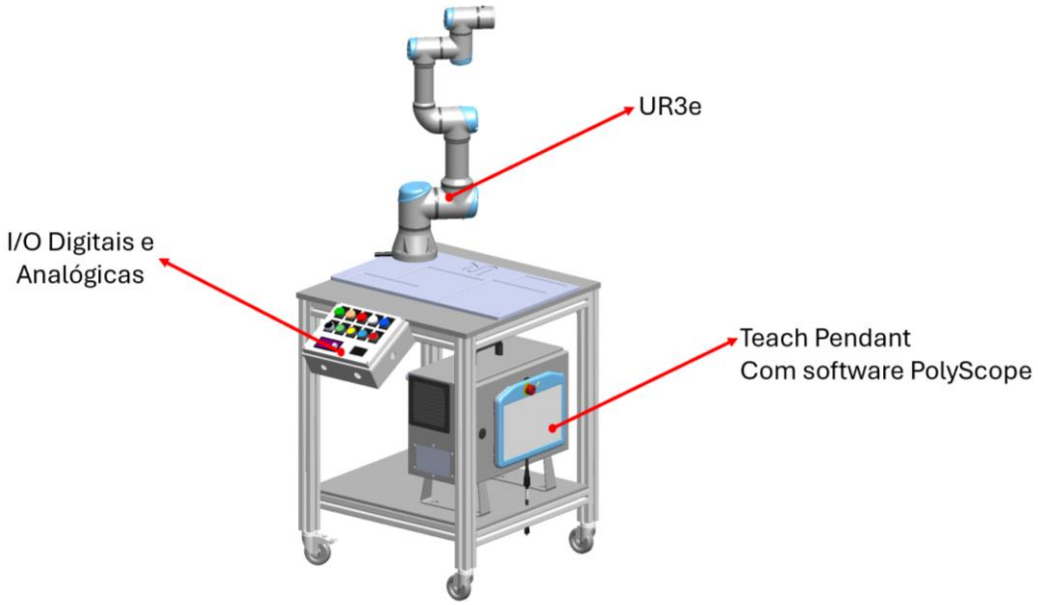


Figura 1 - Sistema UR3e + *Teach Pendant* + I/O

2. Objetivos

Os objetivos centrais deste guião prático focam-se na introdução de instruções básicas de programação e na familiarização dos utilizadores (alunos) com a linguagem de programação utilizada no *PolyScope*. Deste modo, pretende-se capacitar os alunos de competências para projetar, programar e testar diferentes exercícios com o robô UR3e, aplicando conhecimentos de programação robótica, lógica condicional e I/O digitais.

Pretende-se que, no final, os alunos desenvolvam competências nas seguintes áreas:

- Criação de um programa de *Pick and Place* utilizando a interface *PolyScope*;
- Definição e organização das posições de recolha e colocação de objetos;
- Organização da programação de forma modular;
- Teste e validação das aplicações.

Pretende-se, assim, que os alunos adquiram competências fundamentais na área da robótica colaborativa e da programação com o robô UR3.

2.1. Competências técnicas e operacionais

Na Tabela 1 estão descritas as competências técnicas e operacionais associadas a cada um dos exercícios propostos.

Tabela 1 - Competências técnicas e operacionais associadas a cada exercício

	Competências Técnicas	**Competências Operacionais**
Exercício 1	• Compreender e configurar diferentes tipos de movimento (*MoveJ*, *MoveL*, *MoveP*). • Definir *Waypoints*.	• Interpretar de forma clara as diferentes trajetórias.
Exercício 2	• Programar condições operacionais com instruções lógicas (*If/ElseIf*). • Configurar e testar entradas digitais (DI).	• Reconhecer as diferenças entre respostas a sinais externos.
Exercício 3	• Utilizar instruções de *Popup* para interação com o utilizador. • Gerar mensagens na interface.	• Comunicar de forma clara com o operador durante a execução do programa.
Exercício 4	• Configurar e testar saídas digitais (DO). • Programar ações de resposta complementares (ex: ativar *LED*).	• Representar o estado do sistema com sinais visuais.
Exercício 5	• Utilizar contadores e variáveis lógicas. • Implementar ciclos de *Pick and Place*.	• Executar sequências automáticas simples.

Tabela 1 - Competências técnicas e operacionais associadas a cada exercício

Exercício 6	• Programar subprogramas com *Palletizing* e *Seek*. • Utilizar parâmetros de desempilhamento e paletização.	• Automatizar processos com repetição estruturada.

3. Introdução Teórica

Neste capítulo apresentam-se algumas ferramentas de programação consideradas fundamentais para a construção do código de programação e consequente resolução dos exercícios.

3.1. Noções fundamentais de instruções de programação

Move

A função ***Move*** permite a definição de trajetórias que o manipulador deve seguir para mover o ponto TCP[1] (*Tool Center Point*) entre diferentes posições espaciais. Esta função comporta três tipos distintos de movimento:

- ***MoveJ* (Movimento em arco):** O ***MoveJ*** executa a trajetória do TCP utilizando interpolação entre as posições articulares (juntas) do robô, culminando num movimento em arco. Este tipo de movimento é caracterizado por ser **mais rápido e energeticamente eficiente**, embora **menos preciso em relação à trajetória cartesiana** percorrida pelo TCP.

  Indicação: *MoveJ* é utilizado sempre que não seja necessário seguir uma trajetória linear precisa, como em deslocações entre tarefas ou aproximações iniciais.

- ***MoveL* (Movimento Linear):** O ***MoveL*** garante que o TCP se desloque ao longo de uma linha reta no espaço cartesiano entre dois pontos. Este tipo de movimento é fundamental para **tarefas que exigem precisão na trajetória**, como soldadura, colagem ou inserção de componentes.

  Indicação: *MoveL* é utilizado em operações que exigem contacto contínuo ou movimento controlado em linha reta.

- ***MoveP* (Movimento Planeado):** O ***MoveP*** permite a definição de sequências de *Waypoints* que o robô percorre de forma **suave e contínua**, minimizando paragens entre os pontos intermédios. Através de **mistura de trajetórias**, garante-se uma transição fluída, ideal para aplicações que requerem velocidade e fluidez, como pintura ou polimento.

  Indicação: *MoveP* é utilizado quando é necessário um movimento fluido entre múltiplos pontos sem paragens intermédias.

If

Implementa uma condição lógica do tipo "se... então...". Através da instrução *If*, é possível executar blocos de código apenas quando determinadas condições são satisfeitas (por exemplo, se um sensor estiver ativado ou se um contador atingir um certo valor).

[1]**TCP:** O TCP é definido pelo ponto central entre garras do *Gripper*. Este ponto é fundamental para a programação e controlo dos movimentos do robô.

Wait

Esta função implementa uma **espera temporal** definida em segundos. É utilizada para temporizar ações ou aguardar a conclusão de eventos físicos externos.

Popup

Gera uma mensagem visível ao operador na interface gráfica. A mensagem pode ser informativa ou requerer interação do utilizador (e.g., confirmação para continuar).

Set

Permite ativar e desativar saídas digitais e analógicas.

Para ativar a saída digital é necessário defini-la no estado *High*, para desligar a saída digital utiliza-se o estado *Low*.

Assignment

Atribuição de valores a variáveis. Essencial para operações matemáticas, contagens (contadores) ou alterações de estado.

Instruções do *Gripper*:

- **Abrir:**
    - Z_Release: Gripper 1.
- **Fechar:**
    - Z_Grip: Gripper 1.

Seek

A instrução *Seek* permite realizar operações automáticas de empilhamento e desempilhamento de peças, através de parâmetros previamente definidos, como direção, distância máxima e sequência de movimentos, considerando alguns graus de liberdade como força e localização, por exemplo.

Palletizing

A instrução *Palletizing* permite realizar a paletização de peças em três padrões diferentes, designadamente linha, grelha e irregular (diferentes orientações). É constituída por uma estrutura de programação pré-definida que deve ser devidamente parametrizada pelo utilizador.

No software *PolyScope* do UR3e, é possível programar diferentes estratégias de paletização para otimizar o espaço e a eficiência do processo. A Figura 2 ilustra quatro configurações comuns.

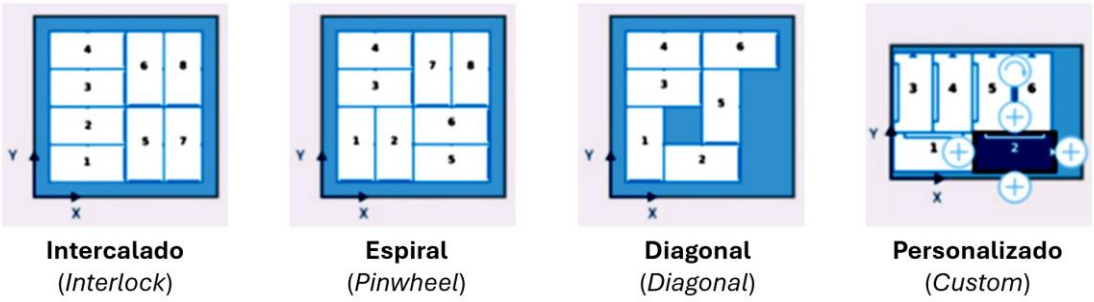


Figura 2 - Exemplos de padrões de paletização

Estas configurações podem ser implementadas diretamente no PolyScope, através da opção de paletização, definindo o número de camadas, filas, colunas, e deslocamentos nos eixos X, Y e Z.

4. Parte experimental

4.1. Segurança operacional (*safety*)

Antes de iniciar qualquer operação com robôs colaborativos, é essencial garantir condições seguras de trabalho. As seguintes boas práticas devem ser adotadas:

1. **Botão de emergência**: Sempre que necessário utilizar o botão de emergência (*E-Stop*) presente no *Teach Pendant* (Figura 3). Para reativar o robô siga as instruções apresentadas no *PolyScope*.
2. **Distância de segurança**: Evitar entrar na zona de trabalho do robô quando ele está em movimento;
3. **Modo *Freedrive* seguro**: Utilizar o modo *Freedrive* só quando pretende programar manualmente os pontos;
4. **Organização do espaço de trabalho**: Manter o espaço circundante ao robô livre, sem objetos que possam interferir no movimento do braço robótico;
5. **Evitar trajetórias imprevisíveis**: Nunca programar movimentos bruscos e rápidos que possam comprometer a segurança dos operados, provocar colisões com o meio envolvente ou danificar o braço robótico.

Figura 3 - Botão de emergência (*E-Stop*) presente no *Teach Pendant*

Estas medidas são fundamentais para prevenir acidentes e garantir um ambiente de trabalho colaborativo, seguro e eficiente.

4.2. Exercício 1 - Tipos de Movimento: *MoveJ*, *MoveL* e *MoveP*

Este exercício tem como objetivo analisar e comparar os três tipos de movimento disponíveis na instrução ***Move***: ***MoveJ***, ***MoveL*** e ***MoveP***, sendo que cada um corresponde a um método

distinto de interpolação dos eixos do robô. A implementação e execução de cada tipo de movimento permite observar o comportamento prático do robô.

Requisitos do exercício:

- Criar um programa e denominá-lo como *exercício_1*;
- Inserir 3 instruções *Move*;
- Associar os movimentos *MoveJ, MoveL* e *MoveP* às 3 instruções *Move*;
- Definir a posição dos *Waypoints*;
- Renomear os *Waypoints* como *ponto_1*, *ponto_2* e *ponto_3*, respetivamente.

4.2.1. Metodologia de resolução (exercício 1)

Passo 1 – Criação do Programa

Comece por criar um programa com o nome *exercicio_1*. Para isso, selecionar a opção ***Program***, localizada no segundo módulo da aba superior esquerda da interface (Figura 4).

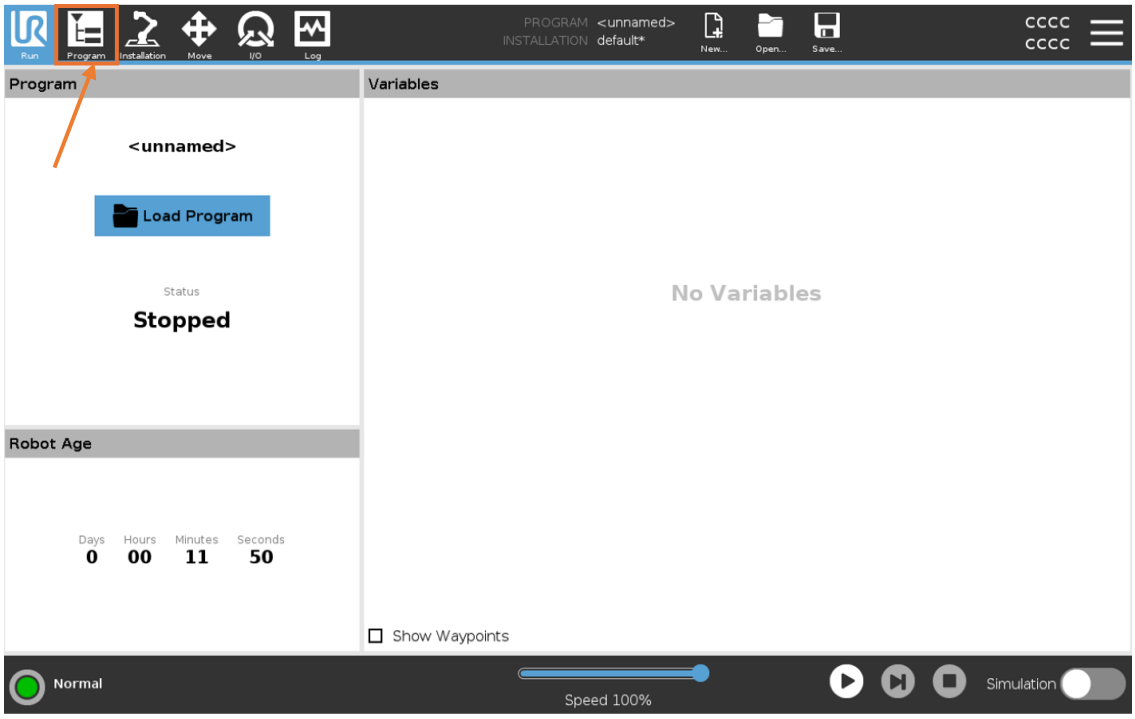


Figura 4 - Criação um programa

Em seguida, guardar o programa através da sequência ***Save > Save Program As > Filename > exercicio_1 > Submit*** (Figura 5, 6 e 7).

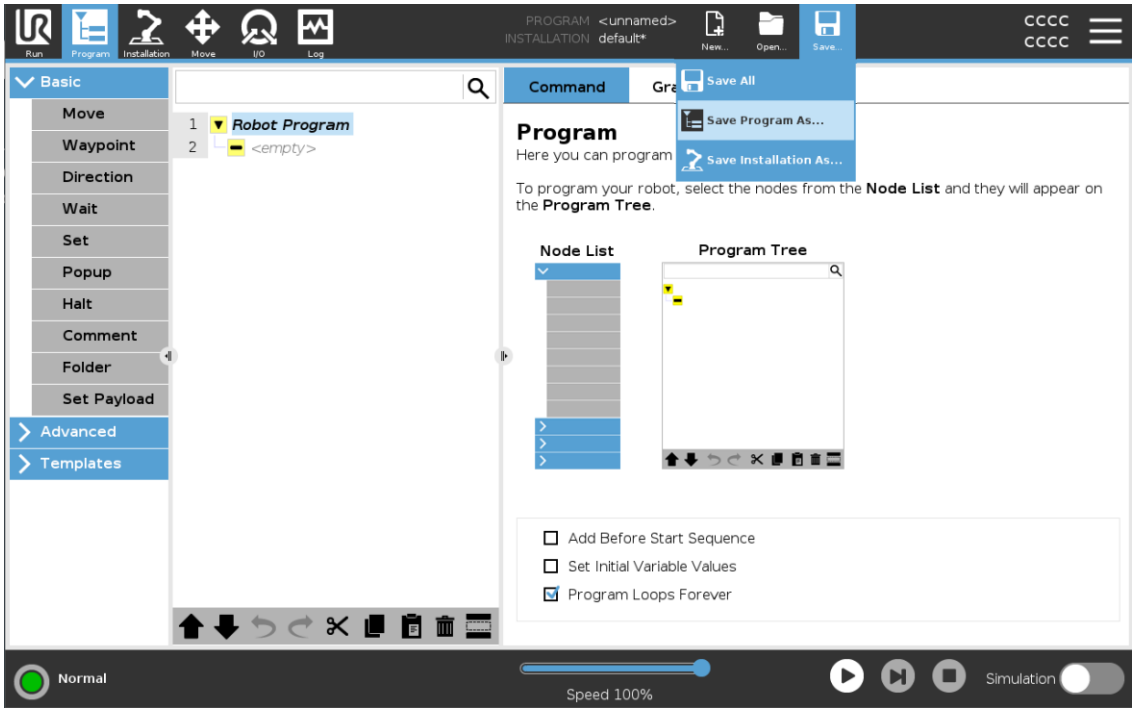


Figura 5 - Guardar um programa

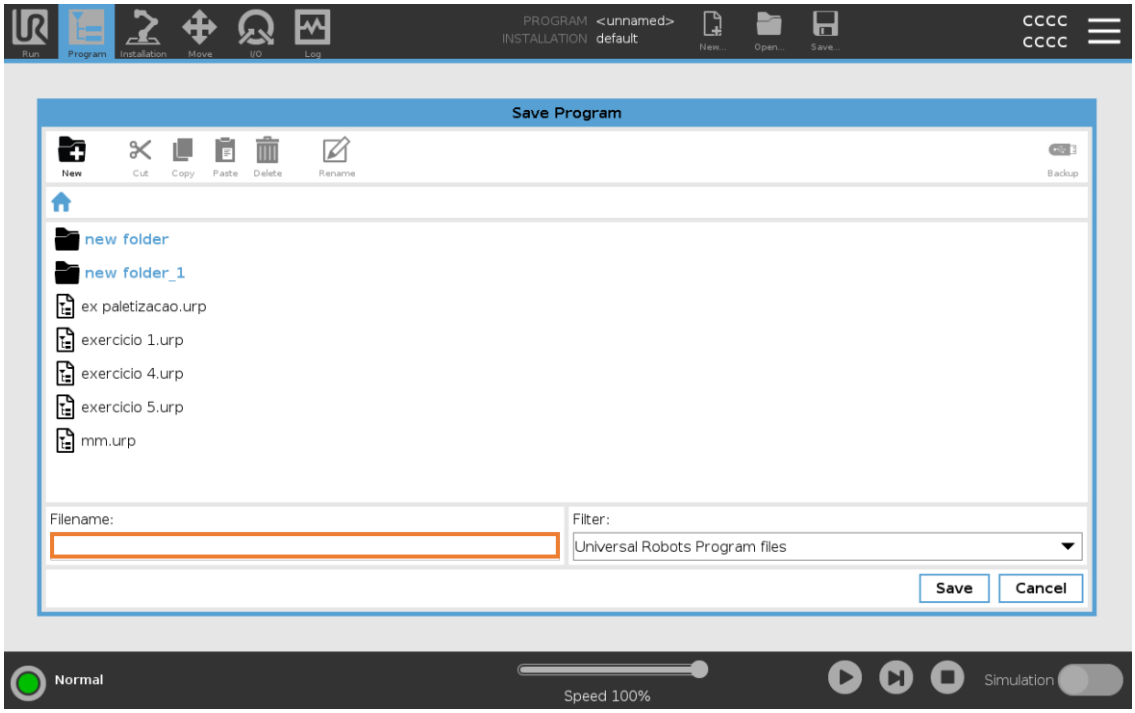


Figura 6 - Atribuir o nome a um programa (1/2)

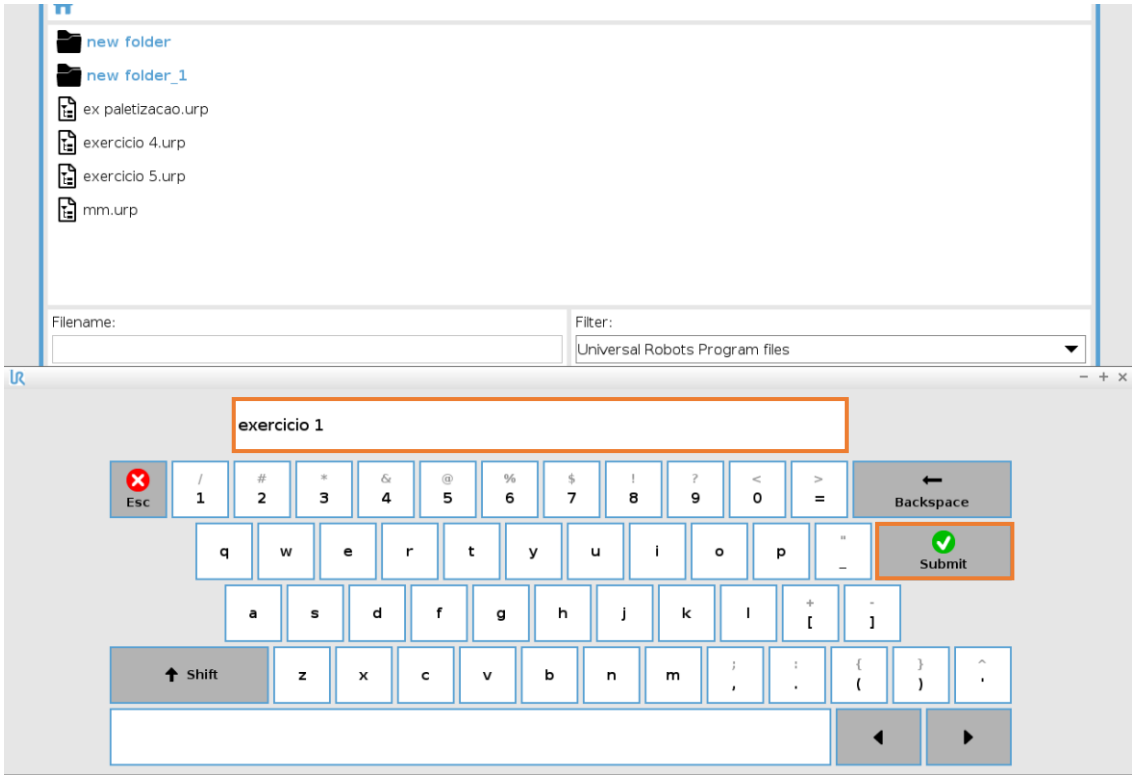


Figura 7 - Atribuir o nome a um programa (2/2)

Passo 2 – Inserção das Instruções de Movimento

Com a ramificação *empty* selecionada, inserir 3 instruções *Move* (Figura 8 e Figura 9).

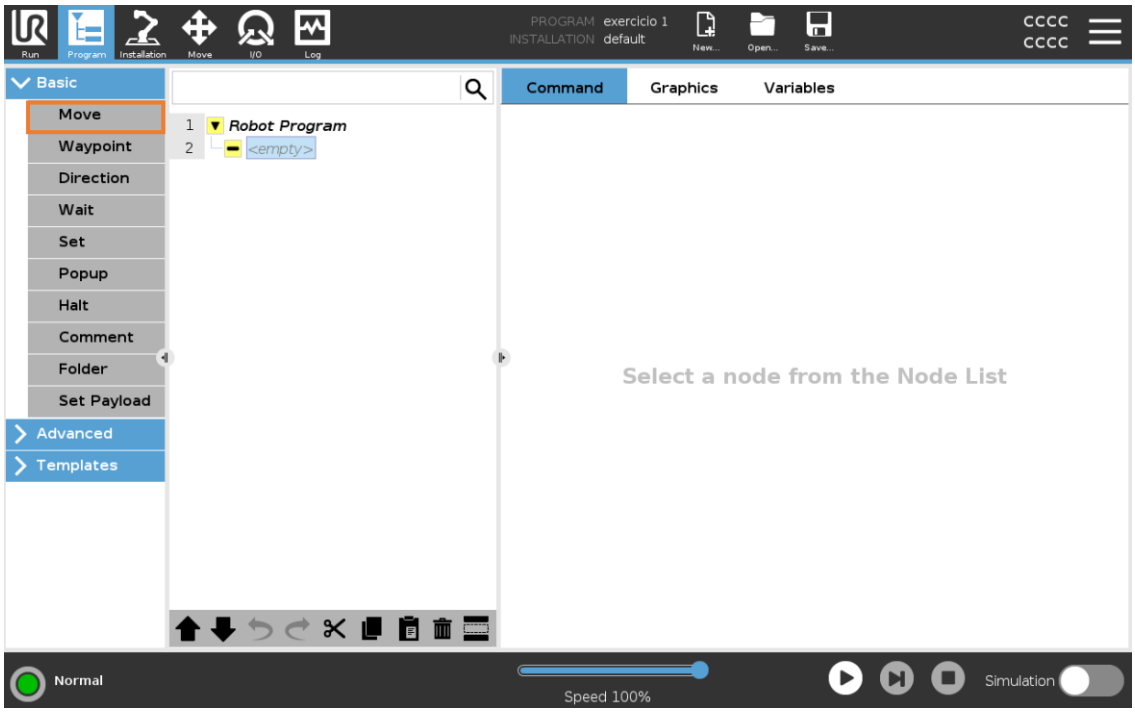


Figura 8 – Inserção da instrução *Move*

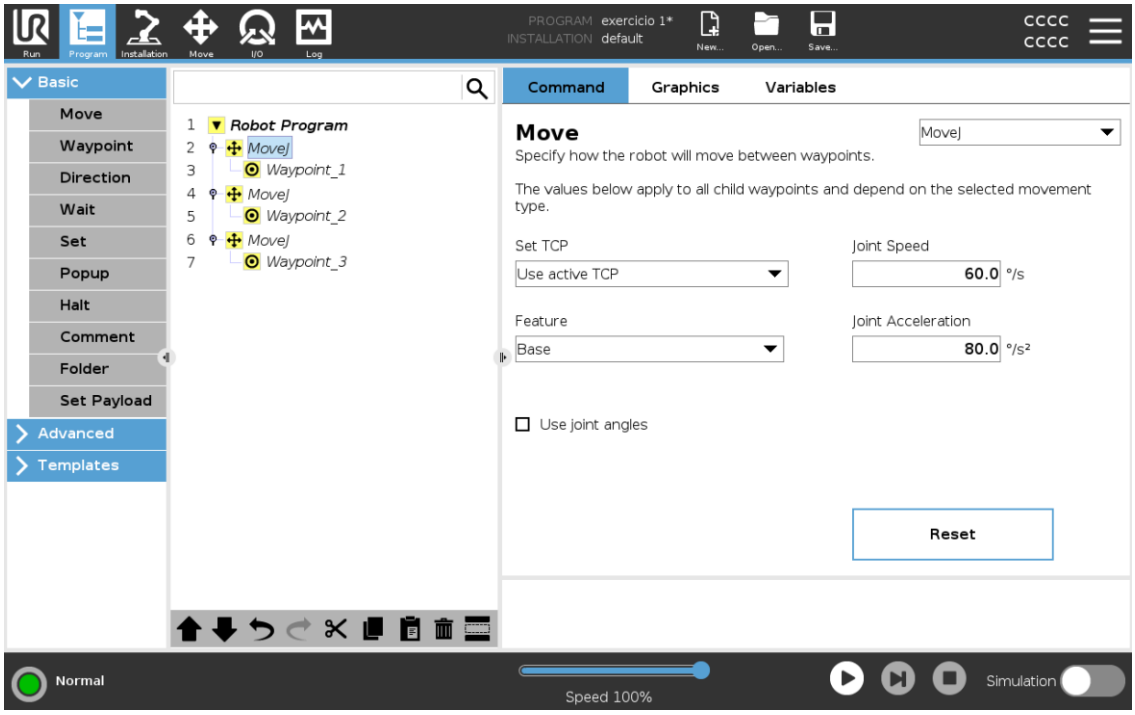


Figura 9 - Instruções *Move* inseridas no programa

Por padrão, a instrução *Move* assume o tipo ***MoveJ***, que pode ser alterado para ***MoveL*** ou ***MoveP*** no menu lateral, pasta comando (Figura 10).

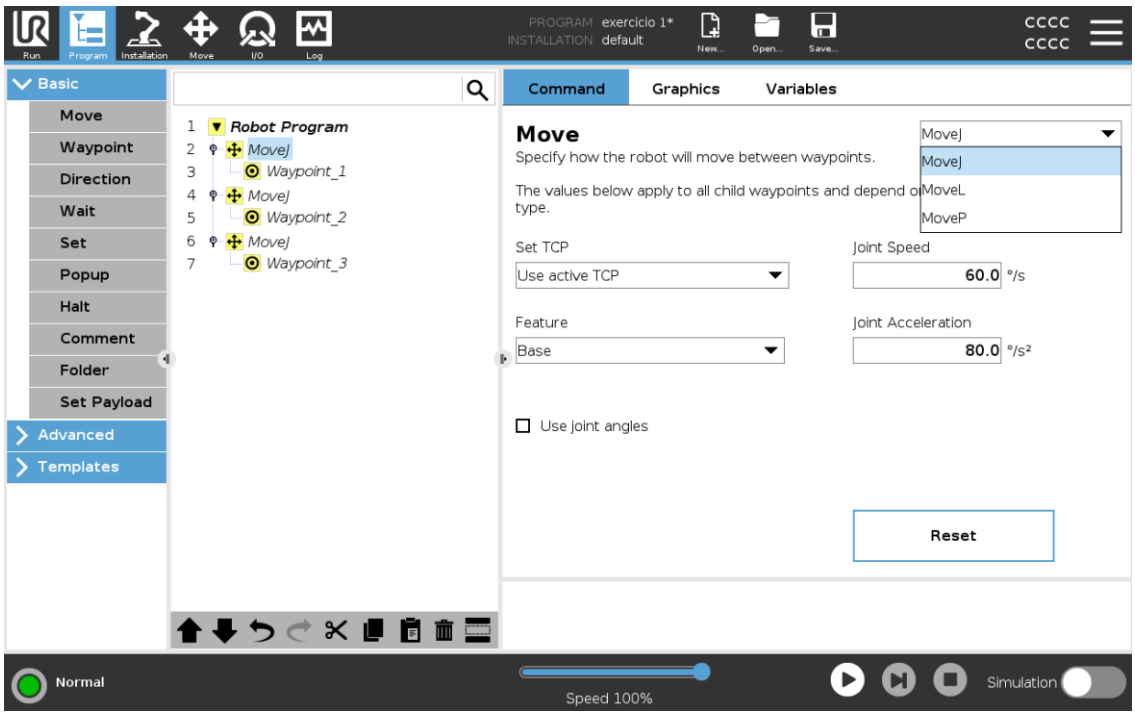


Figura 10 - Alterar o tipo de movimento (*MoveJ, MoveL* e *MoveP*)

De seguida, definir os movimentos como ***MoveJ***, ***MoveL*** e ***MoveP***, respetivamente (Figura 11).

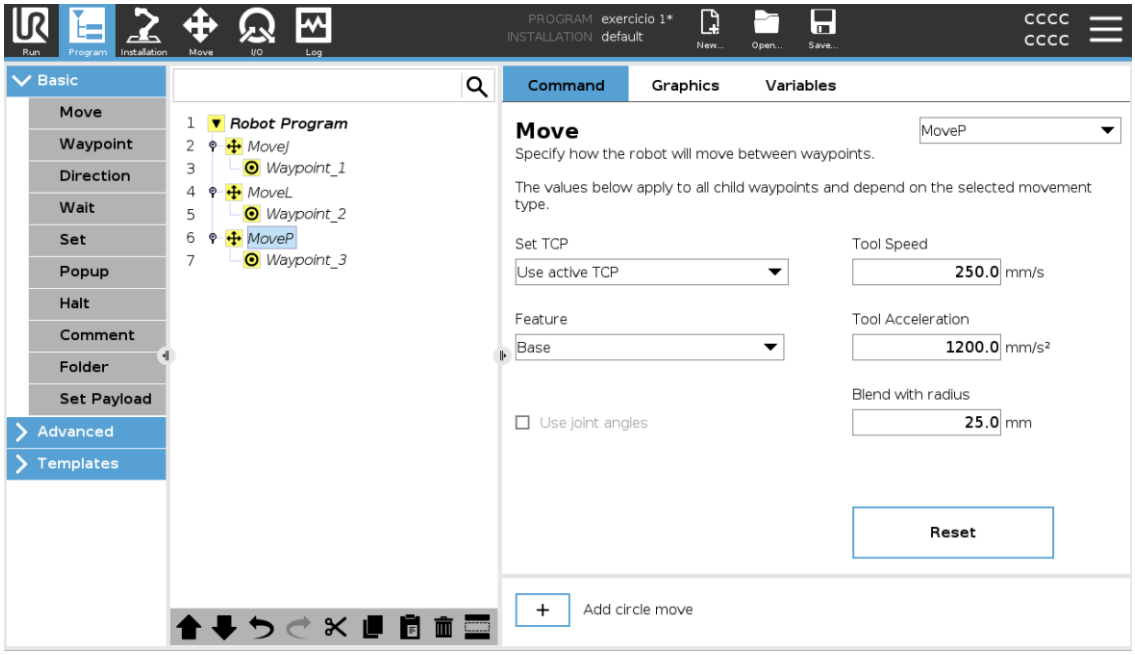


Figura 11 - Ramificação pretendida

Passo 3 – Configuração dos *Waypoints*

Cada instrução *Move* deve conter um *Waypoint* distinto. Para configurar o ***Waypoint_1***, seleciona-se a opção ***Set Waypoint*** (Figura 12).

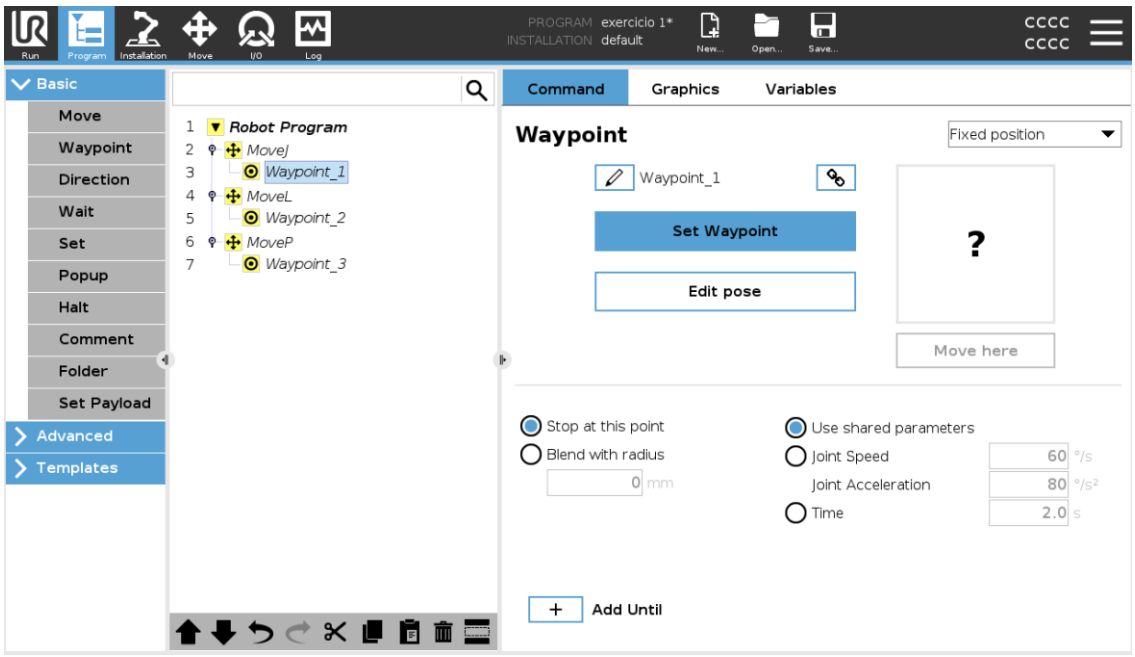


Figura 12 – Configuração do *Waypoint*

Com o modo ***Freedrive*** ativo (pressionar e manter pressionado), posicionar o robô no local pretendido e selecionar ***Ok*** para guardar as coordenadas do ponto (Figura 13). Repetir o

procedimento para os *Waypoints 2* e *3*. O movimento do robô pode também ser feito pelas setas existentes no ecrã da interface.

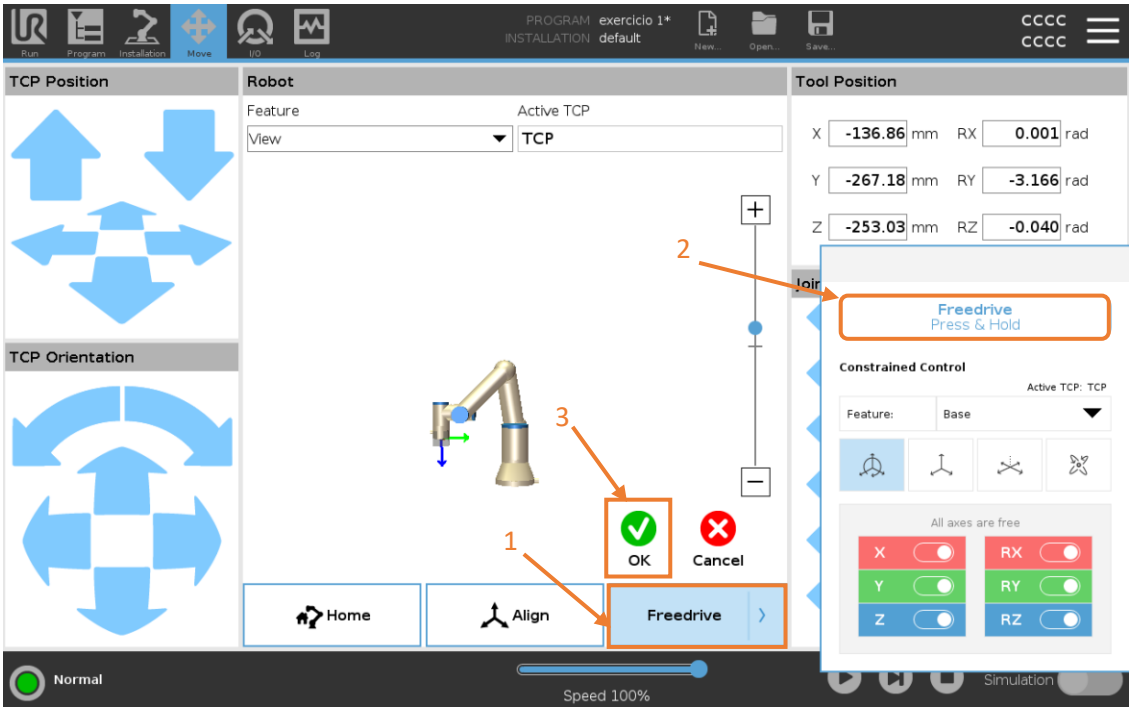


Figura 13 - Posicionar o robô para configurar o *Waypoint*

Passo 4 – Renomeação dos *Waypoints*

Através do ícone de edição que se encontra rodeado a laranja na Figura 14, alterar os nomes dos *Waypoints* para *ponto_1*, *ponto_2* e *ponto_3*, respetivamente.

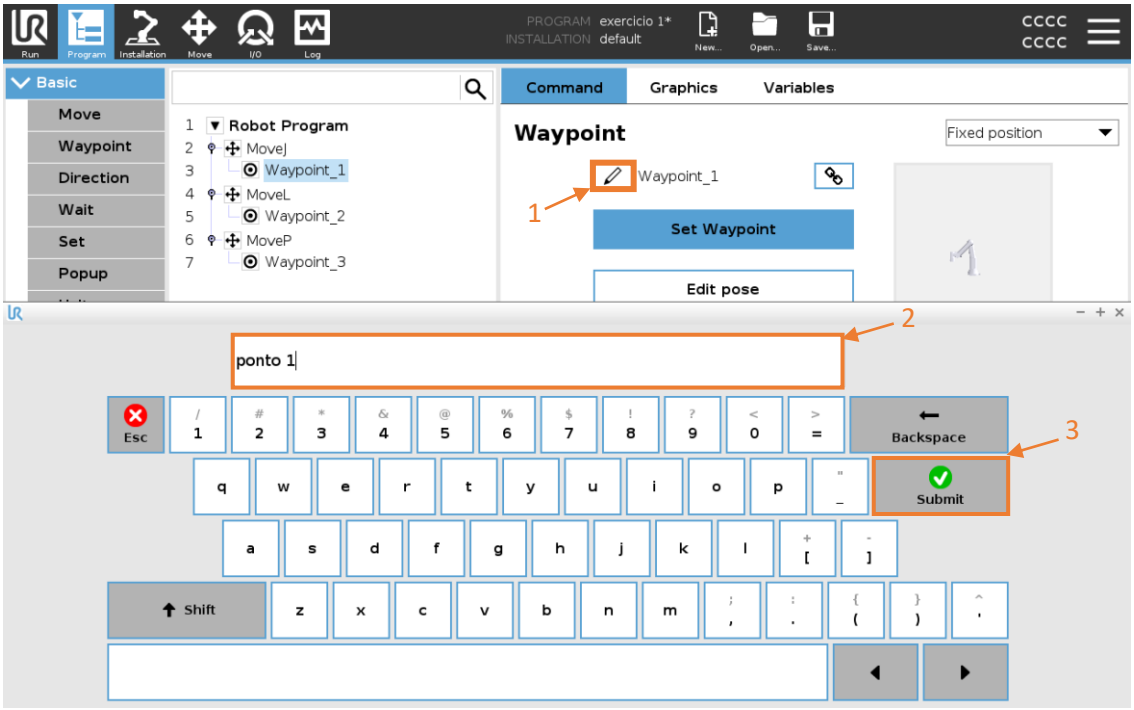


Figura 14 - Renomear o *Waypoint*

Passo 5 – Teste e Validação

Por fim, testar o programa utilizando os botões indicados na interface (Figura 15, 16 e 17).

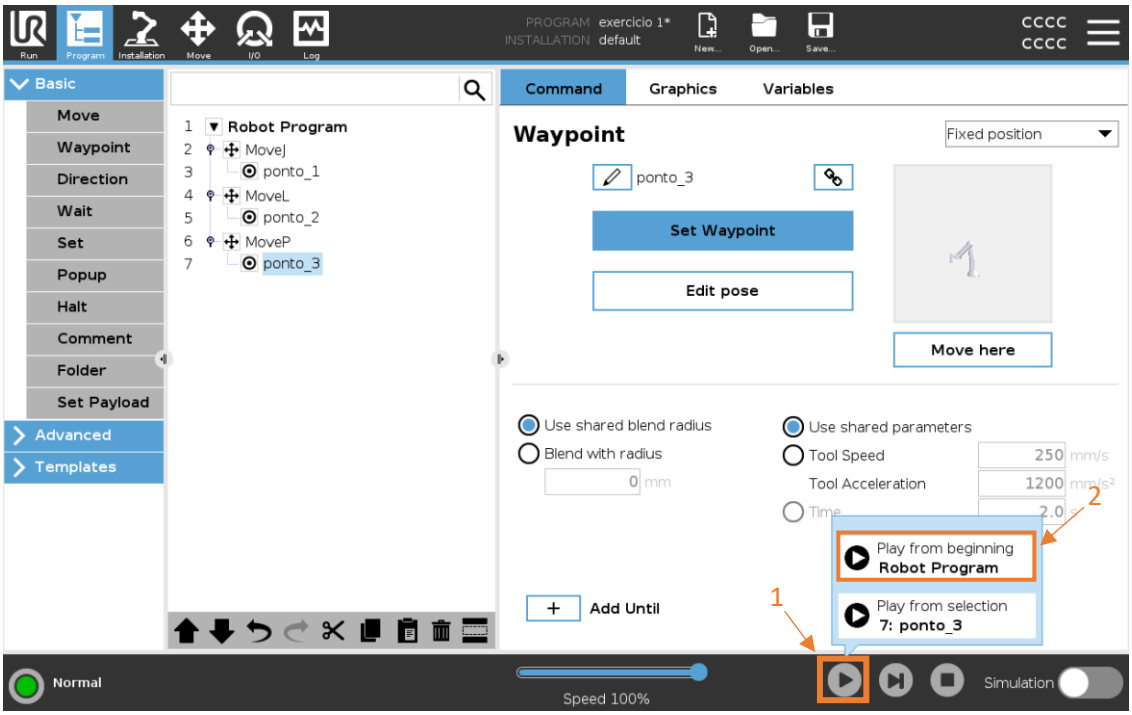


Figura 15 - Solução final e validação do exercício (1/3)

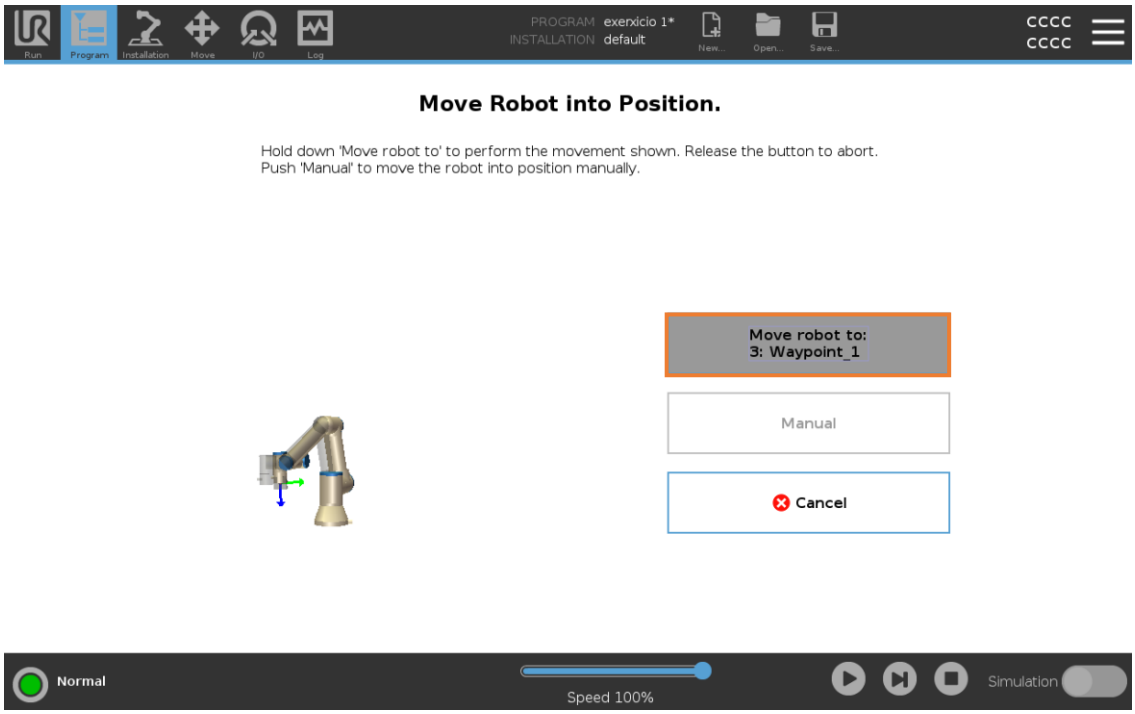


Figura 16 - Solução final e validação do exercício (2/3)

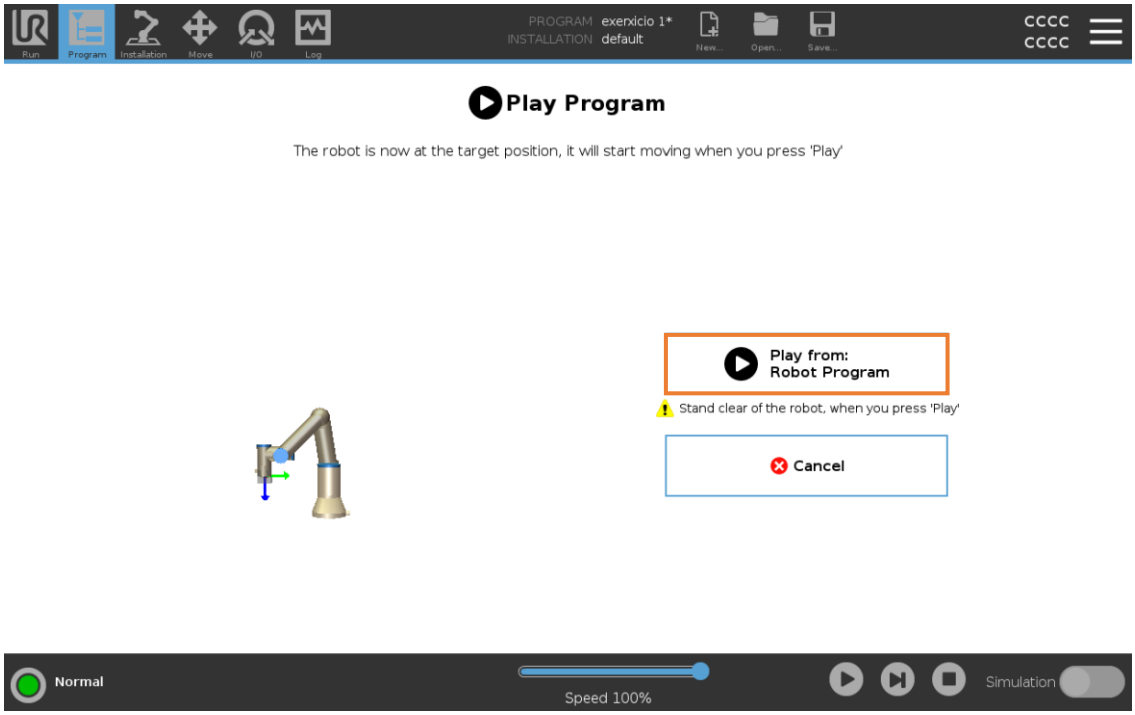


Figura 17 - Solução final e validação do exercício (3/3)

Para terminar o programa deve clicar no botão de parar, representado por um quadrado e assinalado na Figura 18.

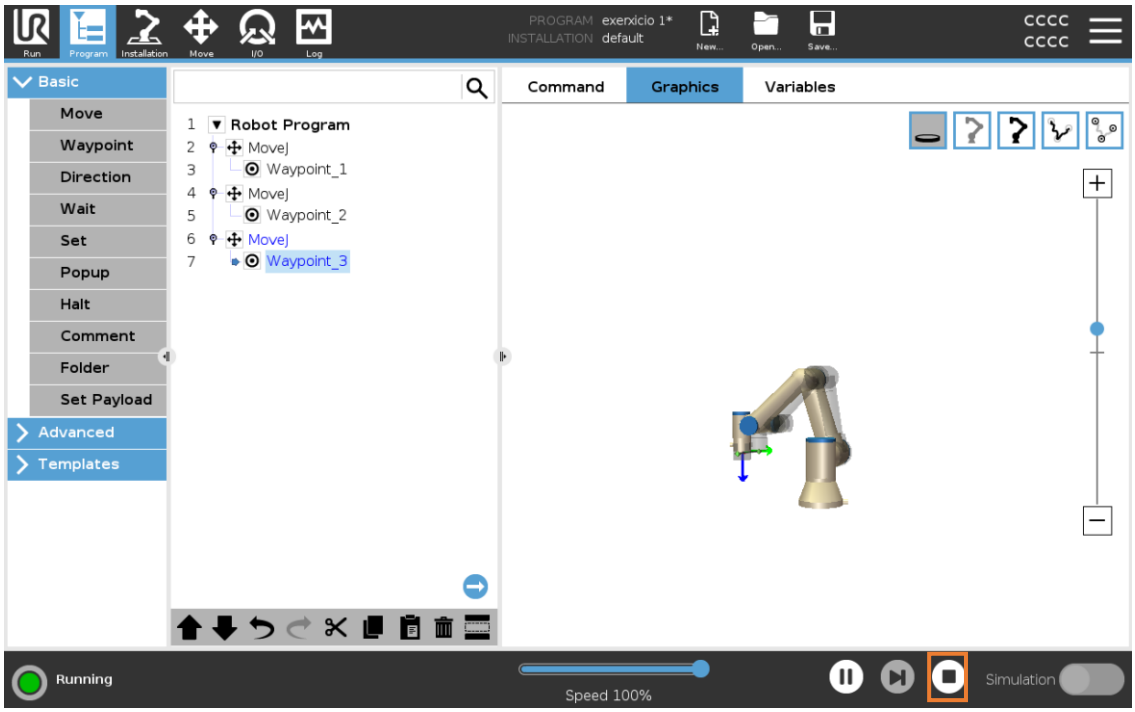


Figura 18 - Botão de parar

4.3. Exercício 2 - Interação com entradas digitais

Este exercício introduz o conceito de interação com o sistema de I/O (entradas/saídas digitais). Pretende-se que o robô reaja a uma entrada digital externa (DI1), ou seja, só pode executar o movimento quando o botão verde (DI1) é acionado. Caso o botão verde não seja acionado, o robô deve manter a posição.

Requisitos do exercício:

- Inserir uma secção de lógica condicional que verifique o estado da entrada digital:
    - Se *DI1 = True*, o programa realiza os movimentos definidos pelo utilizador;
    - Se *DI1 = False*, o programa espera até ser verdadeiro.

4.3.1. Metodologia de resolução (exercício 2)

Passo 1 – Criação de um novo programa

Para criar um novo programa deve-se selecionar o ícone *New* rodeado a laranja na Figura 19. De seguida, selecionar *Program* (ver Figura 20).

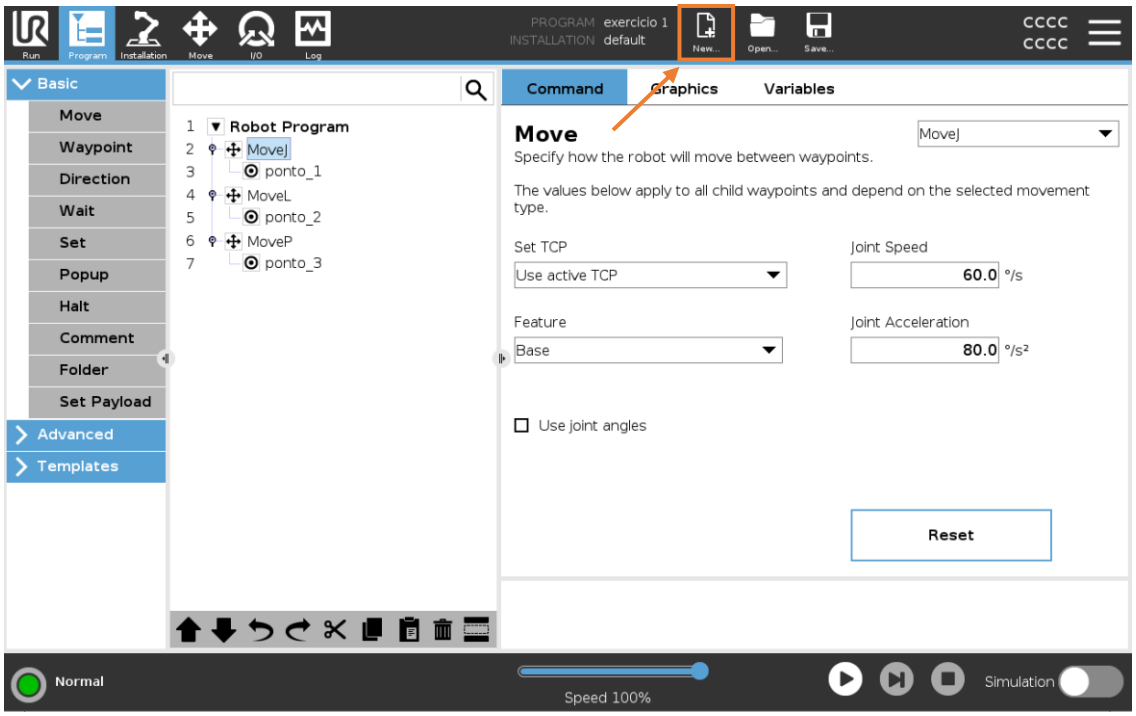


Figura 19 - Criação de um novo programa (1/2)

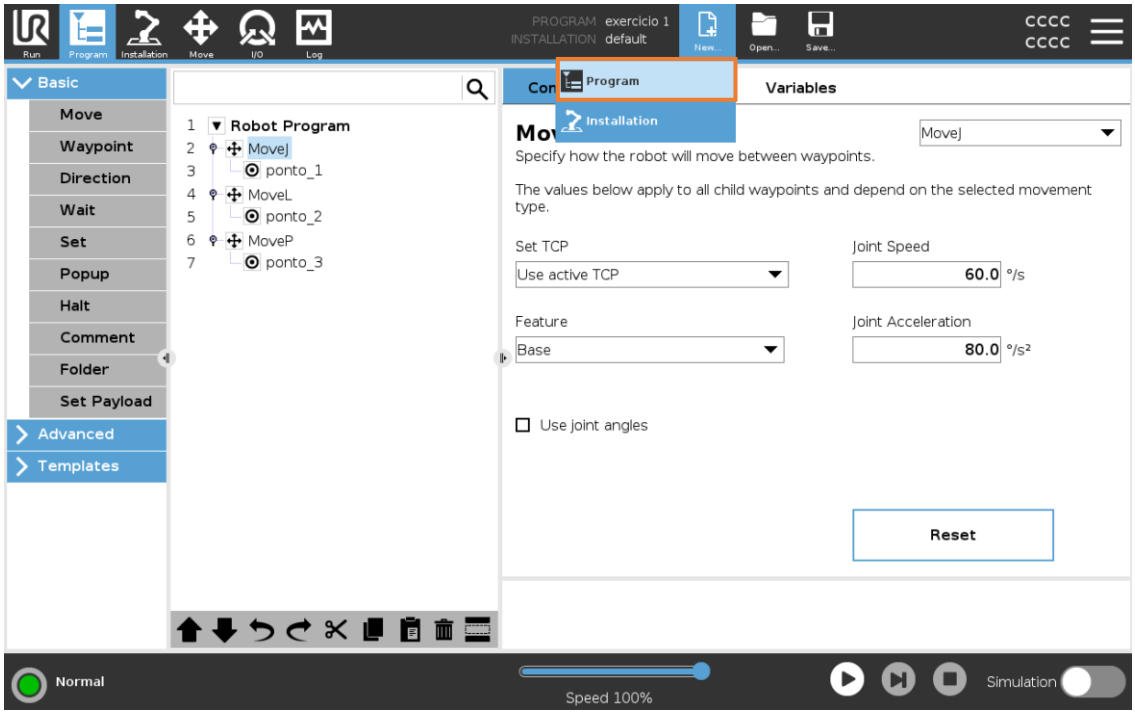


Figura 20 - Criação de um novo programa (2/2)

Por fim, guardar o programa com o nome "*exercicio 2*".

Passo 2 – Criação da Estrutura Condicional

Com a ramificação *empty* selecionada, aceder às instruções avançadas (*Advanced*) do *PolyScope* e inserir uma instrução *If*, conforme ilustrado na Figura 21.

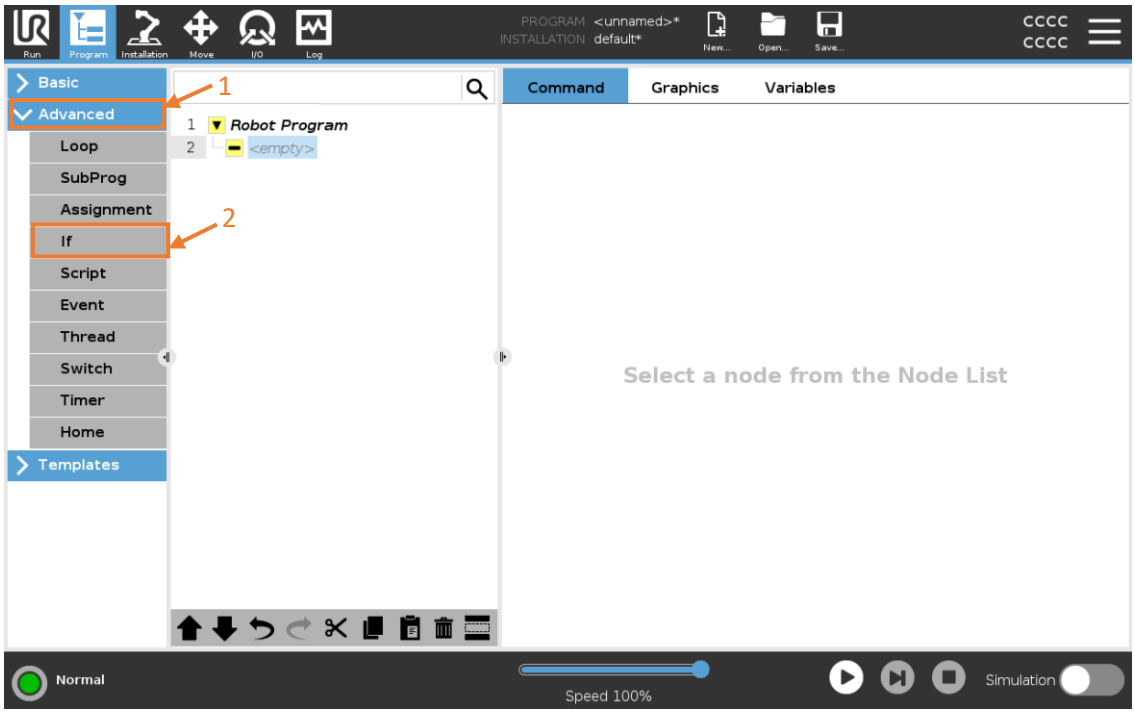


Figura 21 - Adicionar a instrução *If*

De seguida, para configurar a condição lógica seleciona-se o campo amarelo como indicado na Figura 22.

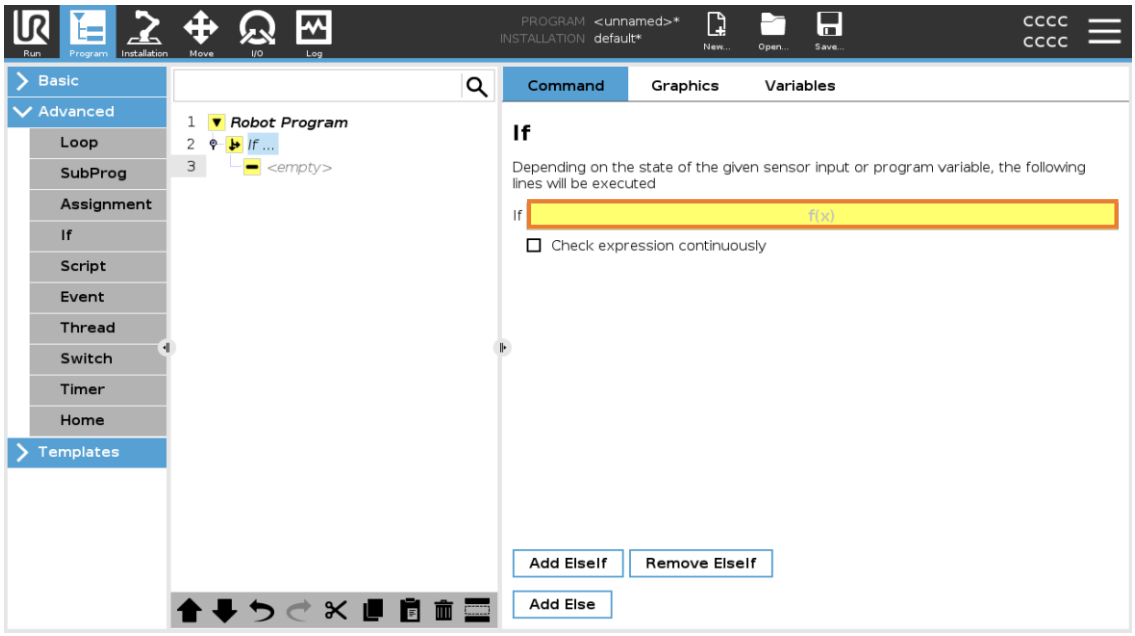


Figura 22 - Local de configuração da expressão *If*

Passo 2 – Definição da Condição *If / ElseIf*

A condição que se pretende programar visa verificar o estado da entrada digital 1 (***digital_in[1]***), associada ao botão verde. A expressão da condição deve verificar se essa entrada (*input*) está em estado ***High*** (ativa ou verdadeira). Para isso, deve-se preencher o campo com a expressão *digital_in[1] = True(HI)* e confirmar com ***Submit*** (Figura 23).

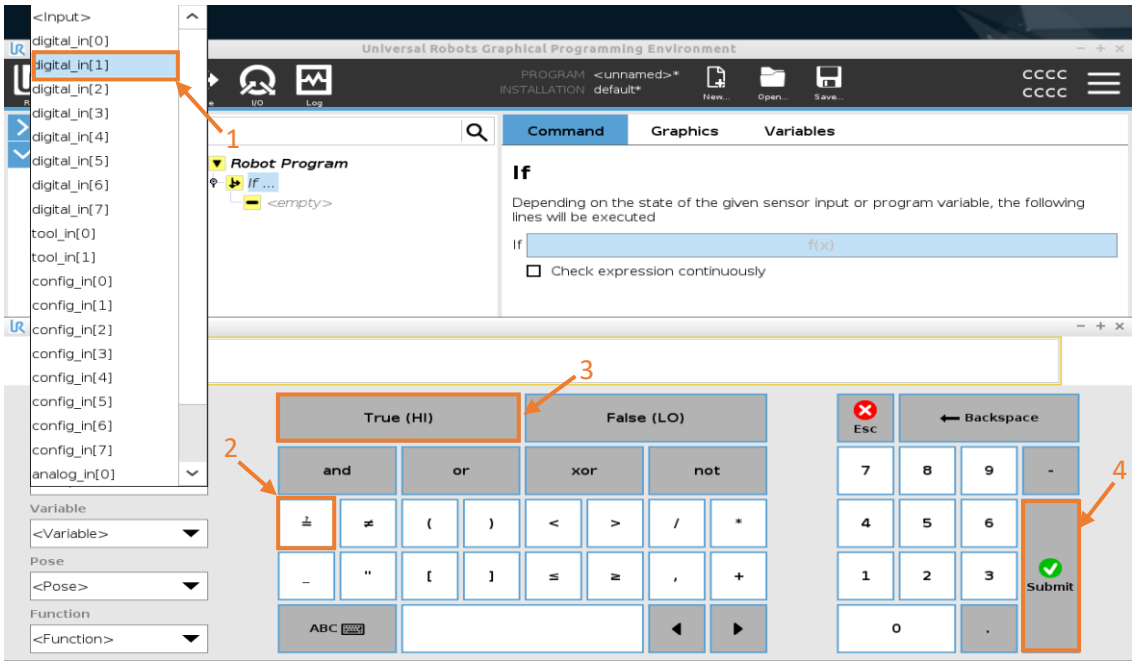


Figura 23 - Configuração da condição *If*

Existe ainda outra condição que tem de ser verificada. Enquanto o botão verde não é ativado, o robô espera. Neste sentido, é necessário adicionar um *ElseIf*. Para realizar esta ação, selecionar *Add ElseIf* conforme mostra na Figura 24.

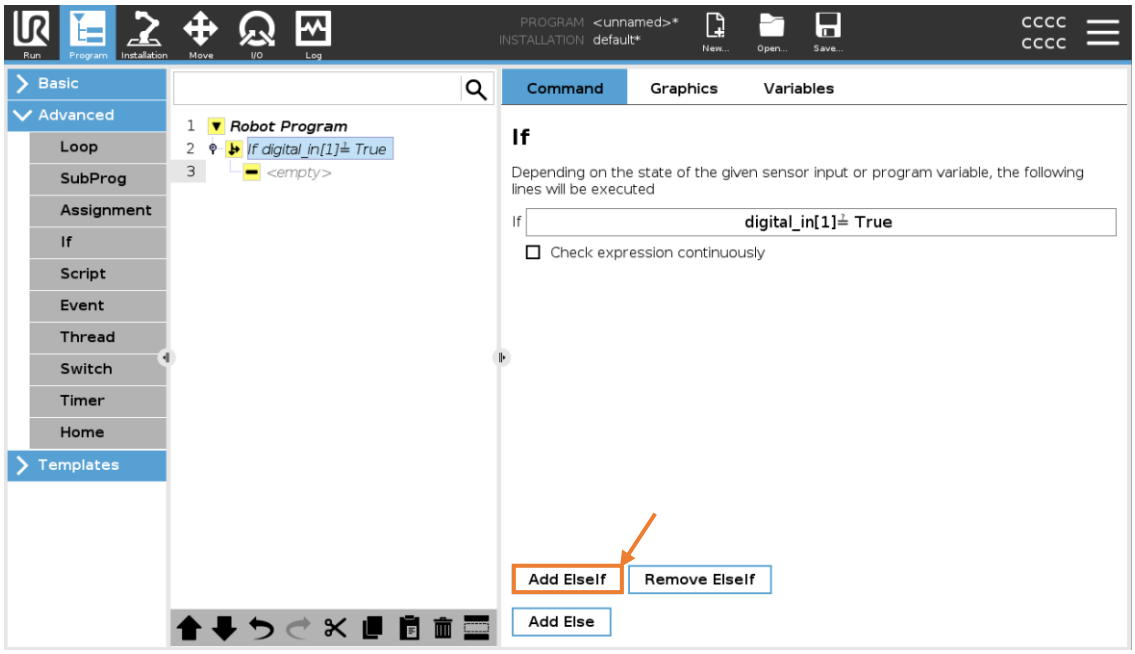


Figura 24 – Inserção da condição *ElseIf*

Neste caso, o campo de expressão é preenchido exatamente da mesma forma que o do *If*, com a exceção de que é necessário verificar se a entrada digital 1 (botão verde) não é acionada. Inserir *digital_in[1] = False (LO)* (Figura 25).

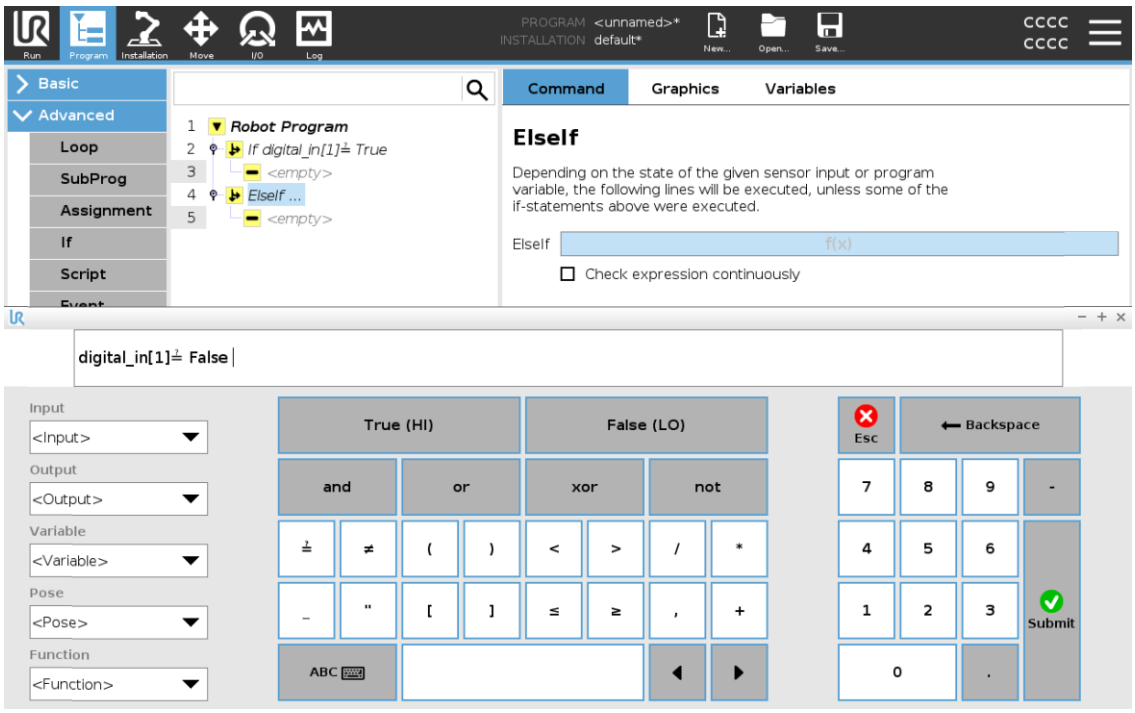


Figura 25 – Expressão para o estado inativo

Passo 3 – Comportamento Condicional do Robô

Consoante o estado da entrada digital, definem-se os seguintes comportamentos:

- **Se DI1 estiver ativa (*True*):** O robô executa duas ou mais instruções de movimento previamente definidas. Devem ser inseridas as instruções ***Move*** com os respetivos *Waypoints*, na secção *empty* da condição *if*.
- **Se DI1 estiver inativa (*False*):** O robô permanece em espera. Para isso, insere-se uma instrução ***Wait*** (localizada em *Basic*), configurando-a para aguardar pela entrada digital 1 em estado *High*. Para configurar a instrução, é necessário selecionar a opção ***Wait for Digital Input***, seguida da escolha da entrada ***digital_in[1]*** e da condição ***High*** (Figura 26).

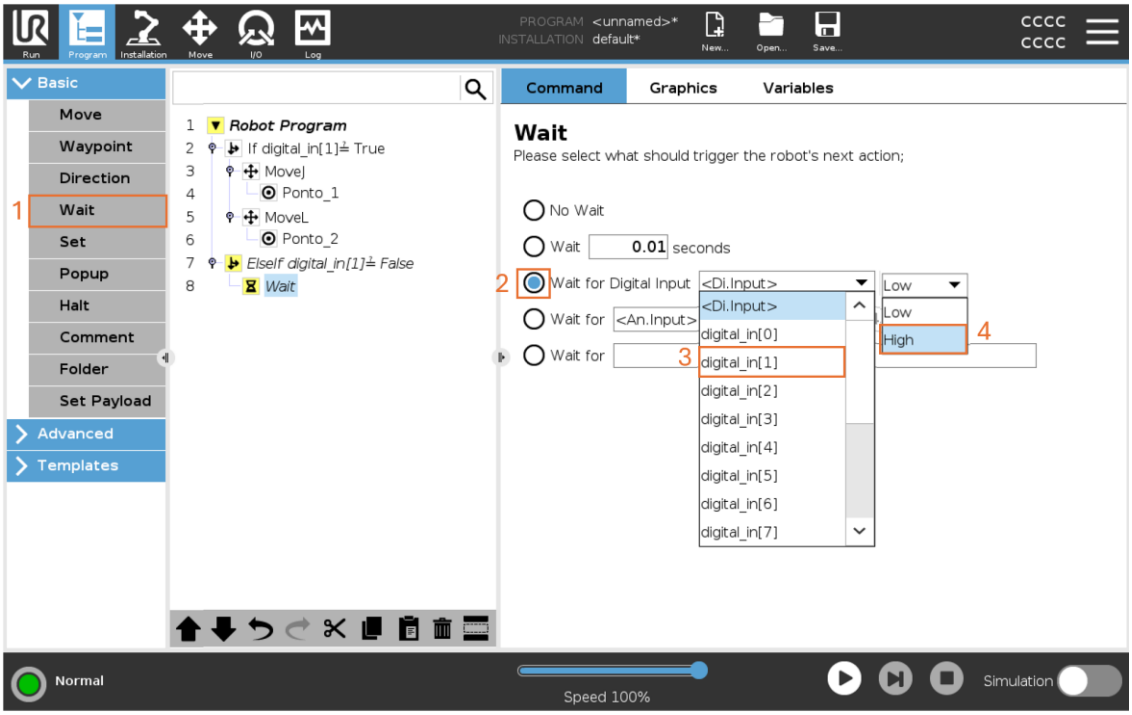


Figura 26 - Configuração da instrução *Wait*

Passo 4 – Teste e Validação

Após a definição da lógica condicional e inserção dos movimentos e tempos de espera, o programa encontra-se pronto para teste (Figura 27).

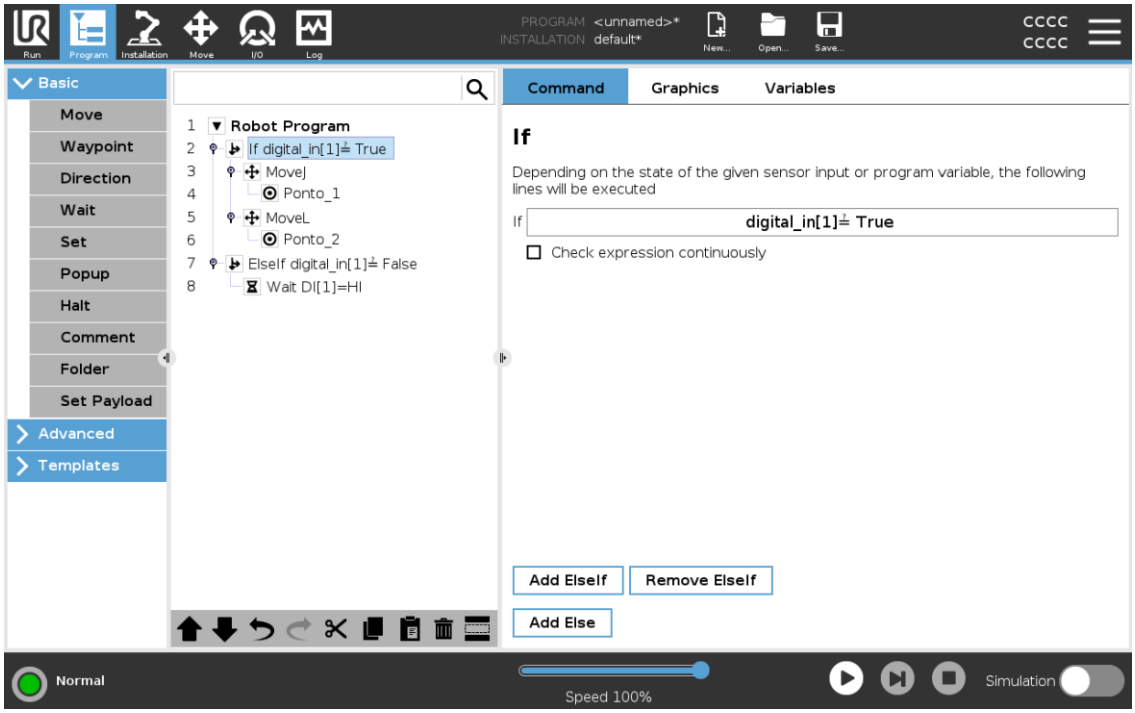


Figura 27 - Solução final e teste do exercício 2

4.4. Exercício 3 - Instrução de Mensagem

Este exercício tem como objetivo adicionar uma mensagem de alerta ao programa desenvolvido no exercício anterior. Enquanto o robô aguarda que o botão verde seja pressionado, deve surgir uma mensagem no ecrã do *Teach Pendant* a alertar o utilizador. A interação deve ser feita através de uma janela de mensagem com o texto: "Pressione o botão verde". Deve ser utilizado o programa desenvolvido no exercício anterior. Proceda com as alterações necessárias.

Requisitos do exercício:

- Inserir uma instrução de mensagem quando *DI1 = False*:
    - Inserir a mensagem "Pressione o botão verde", para alertar o utilizador;
    - O programa continua à espera até que *DI1 = True*.

4.4.1. Metodologia de resolução (exercício 3)

Passo 1 – Inserção da instrução *Popup*

Mantendo a lógica do exercício anterior, acrescenta-se uma instrução ***Popup*** na secção *ElseIf* (Figura 28).

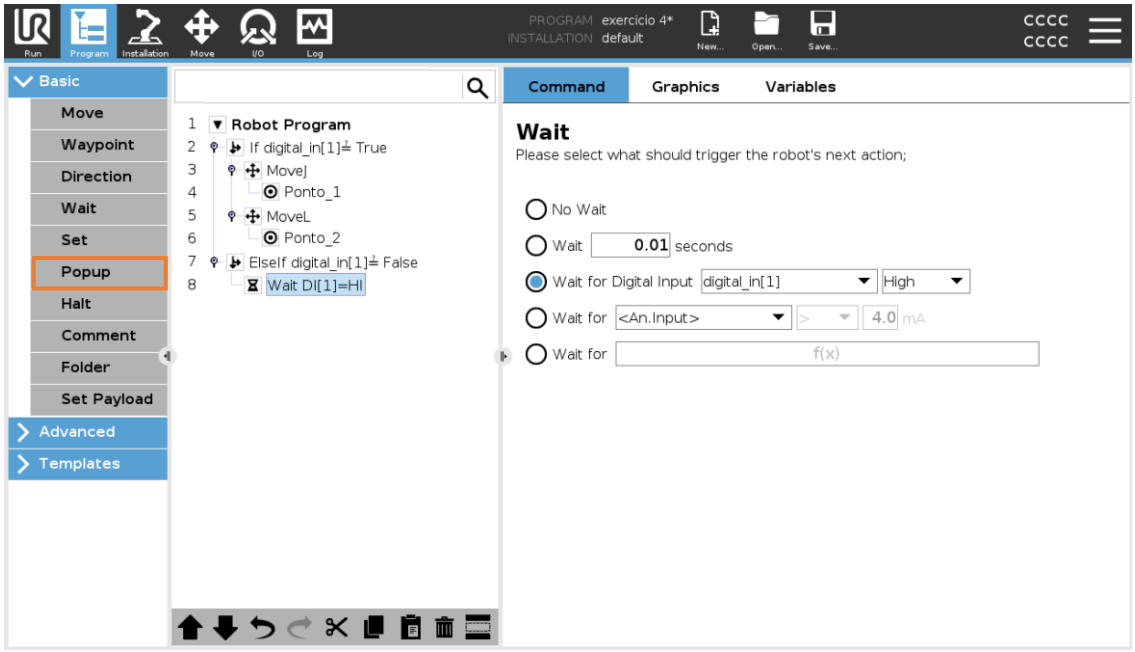


Figura 28 – Inserção da instrução *Popup*

Passo 2 – Reposicionamento da instrução

Como se pode verificar a instrução *Popup* ficou inserida depois da instrução *Wait*. Pode-se alterar o seu posicionamento clicando nas setas assinaladas a laranja na Figura 29. Ao clicar na seta que tem sentido ascendente, a instrução sobe. Ao clicar na seta que tem sentido descendente a instrução desce. Neste caso pretende-se que a **primeira ação** seja ativar a mensagem enquanto o botão verde não é pressionado.

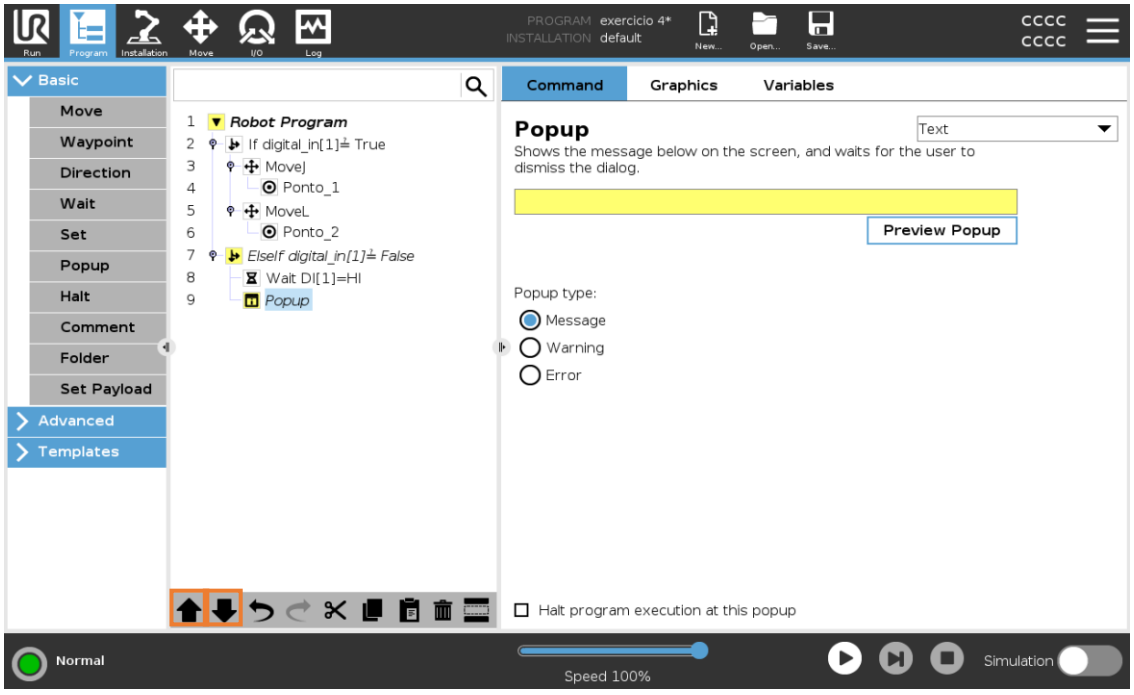


Figura 29 - Alteração do posicionamento de instruções

Passo 3 – Definição da mensagem

No comando da instrução verificar se o tipo de *Popup* selecionado é o *Message* (Figura 30).

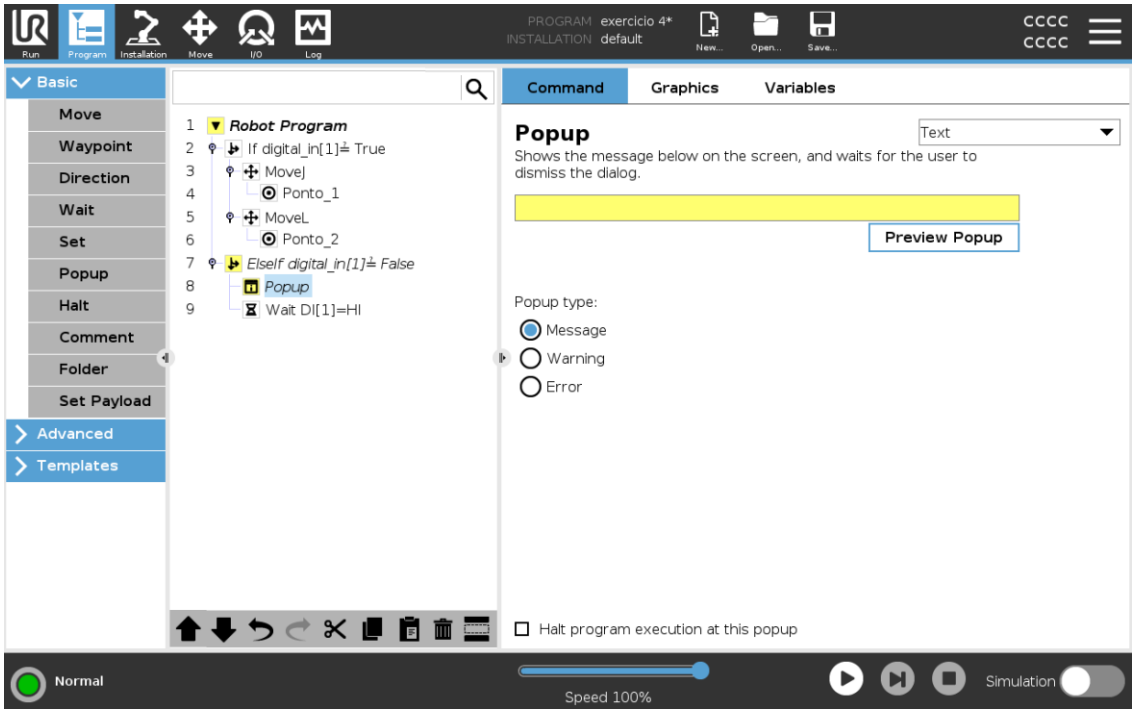


Figura 30 - Instrução *Popup*

De seguida, inserir a mensagem “Pressione o botão verde” no campo de texto correspondente (Figura 31) e validar com *Submit*.

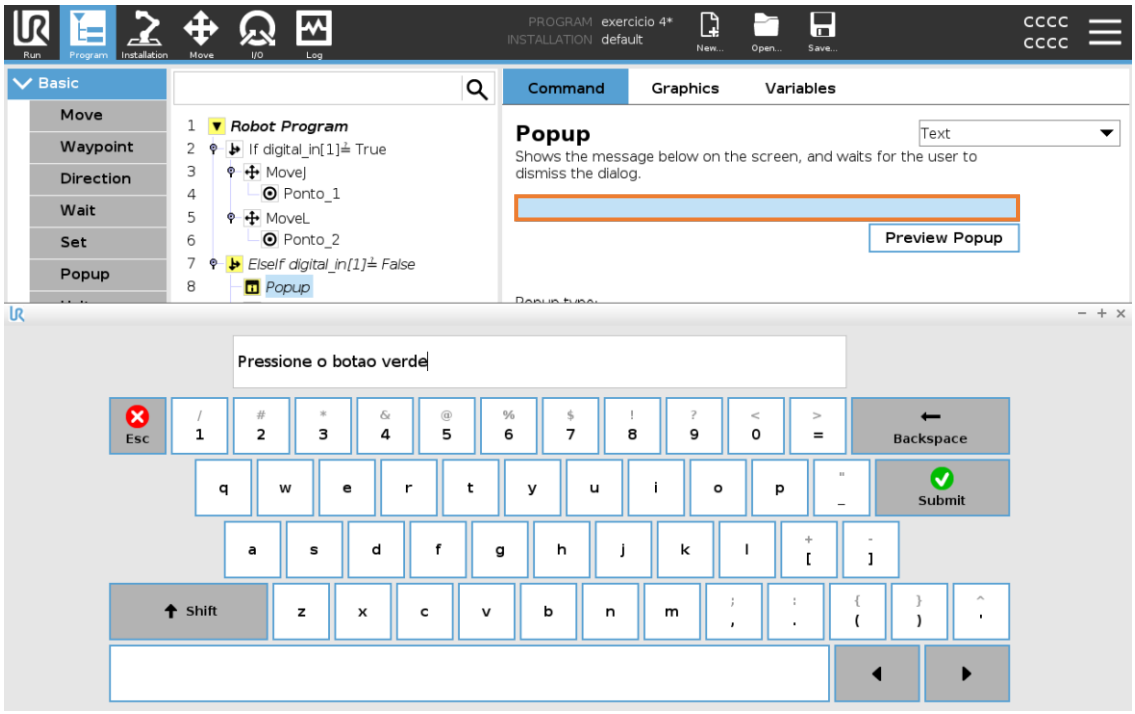


Figura 31 - Configuração da mensagem no *Popup*

Passo 4 – Teste e Validação

Após a alteração, testar o programa e verificar se a mensagem exibida está conforme a Figura 32.

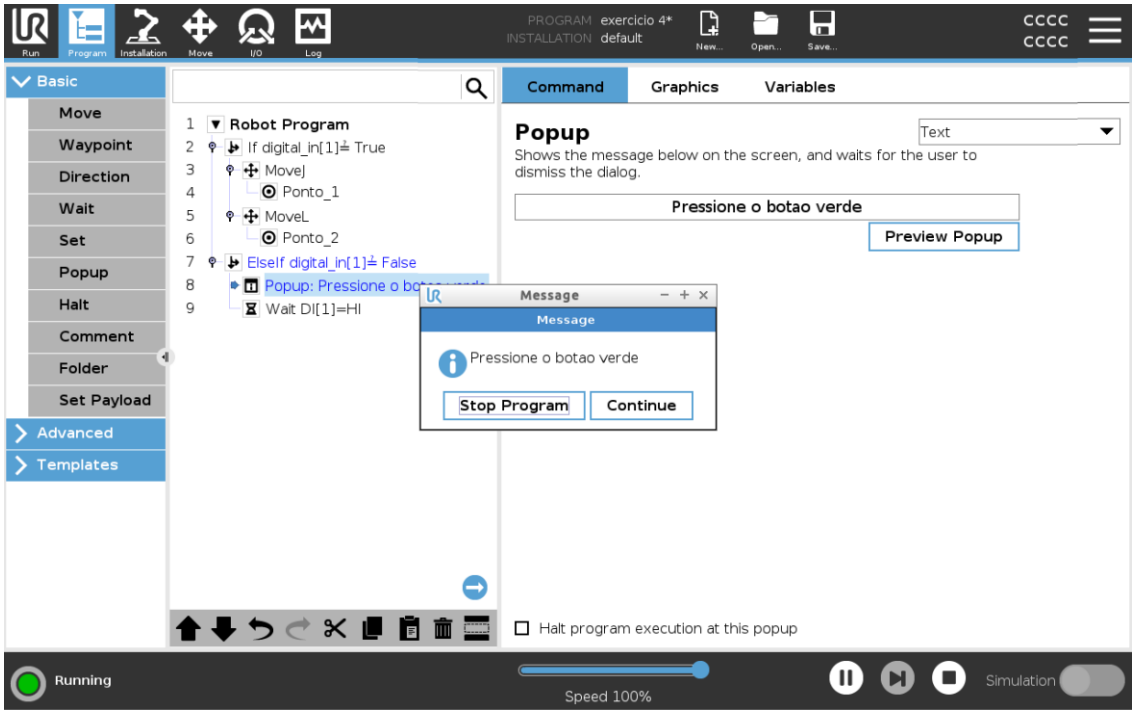


Figura 32 - Solução final e validação do exercício 3

4.5. Exercício 4 – Interação com saídas digitais

Pretende-se acrescentar a saída digital (DO0) – *LED* verde ao exercício 3. Utilize o programa anterior e ative o *LED* verde quando o botão verde for pressionado. Proceda com as alterações necessárias.

Requisitos do exercício:

- Se *DI1 = True*, ativar o *LED* verde (DO0) como primeira ação.

4.5.1. Metodologia de resolução (exercício 4)

Passo 1 – Inserção da Instrução Set

No interior do bloco ***If*** e com a primeira instrução ***Move*** selecionada, inserir uma instrução ***Set*** para ativar a saída digital (Figura 33).

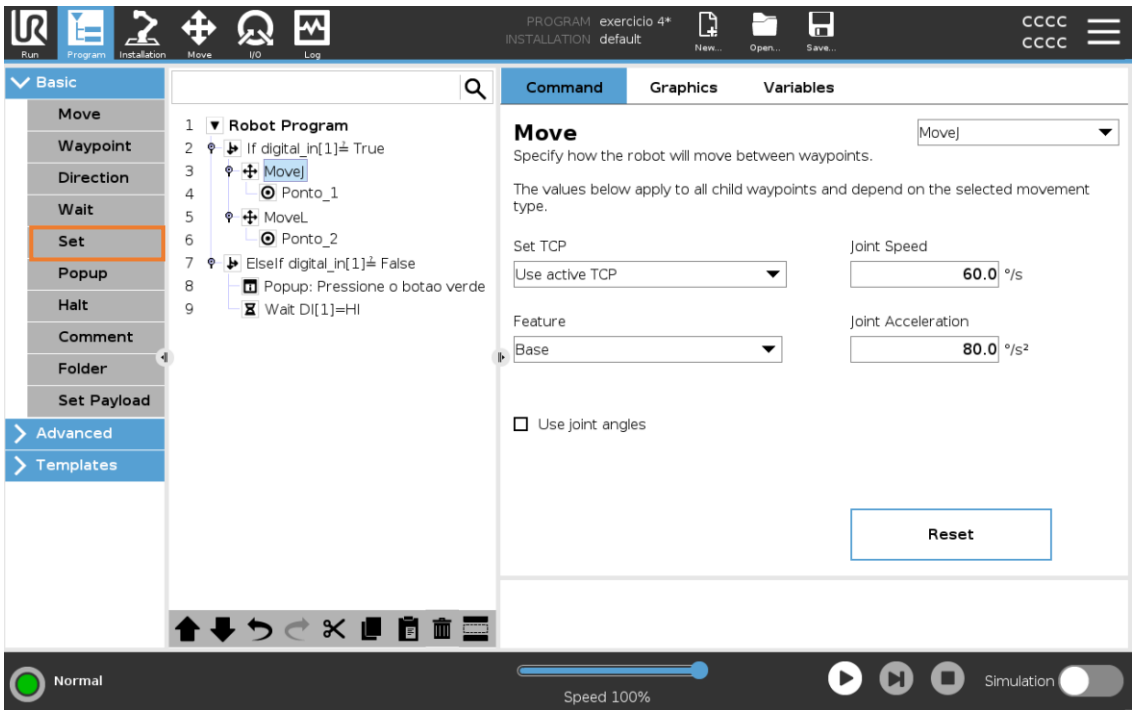


Figura 33 – Inserção da instrução *Set*

Pretende-se que a **primeira ação** a realizar seja o acionamento do Led verde. Para isso deve posicionar corretamente a instrução *Set* (Figura 34).

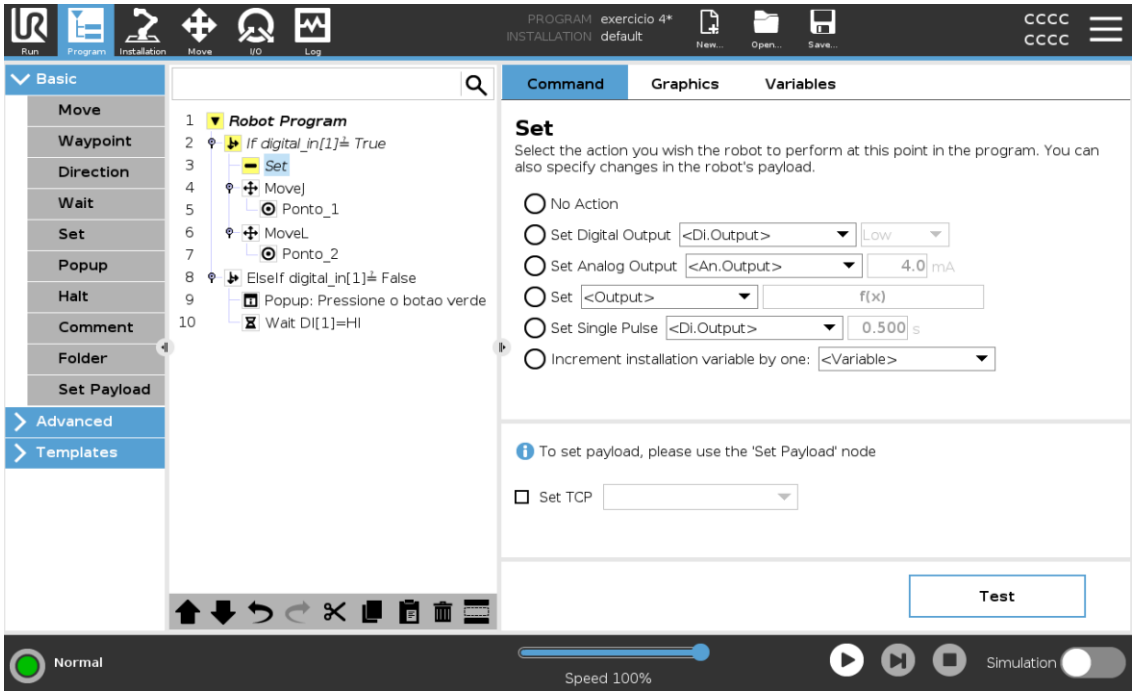


Figura 34 – Posicionamento da instrução *Set*

Passo 2 – Configuração da saída digital

O objetivo é ativar o *LED* verde. Para isso definir *Set Digital Output* > *digital_out[0]* > *High* (Figura 35).

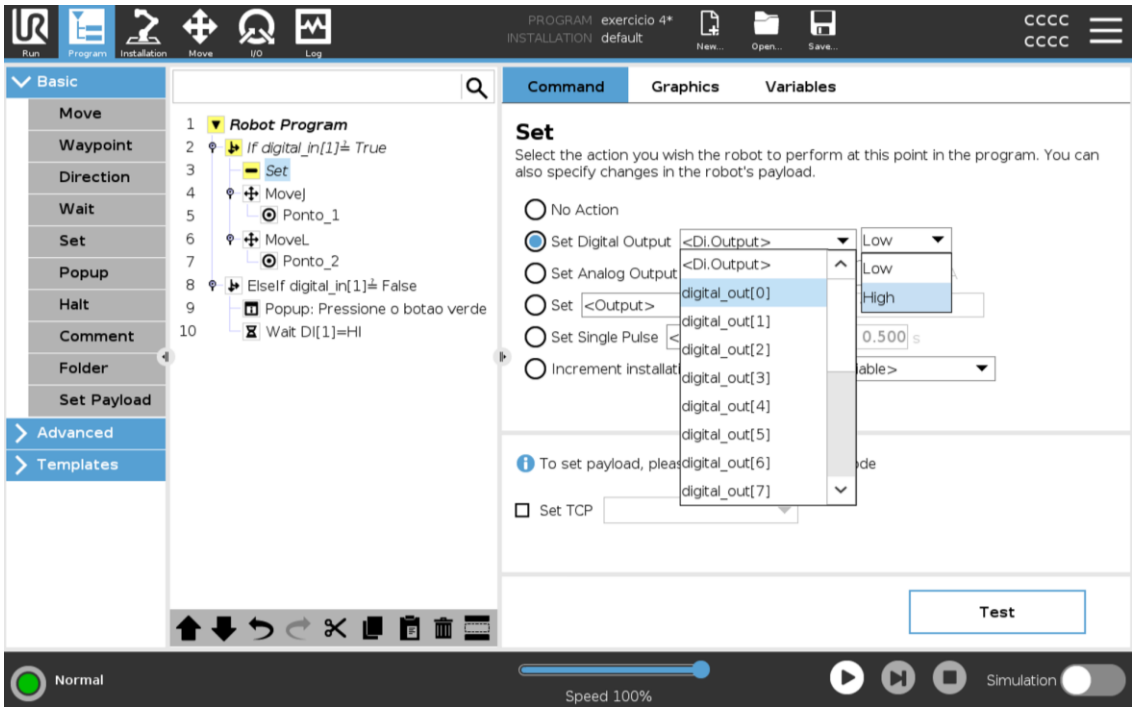


Figura 35 - Configuração da instrução *Set* para ativar o *LED*

Passo 3 – Desativação da saída digital (*Before Start*)

Para evitar que a saída permaneça ativa após o fim do programa, deve-se adicionar uma instrução ***Set***, com o estado da saída definido como ***Low***. Com a segunda instrução *Move* selecionada, inserir a instrução *Set* (Figura 36) e parametrizá-la (Figura 37). Esta ação garante que o *LED* verde é **desligado** no fim do programa.

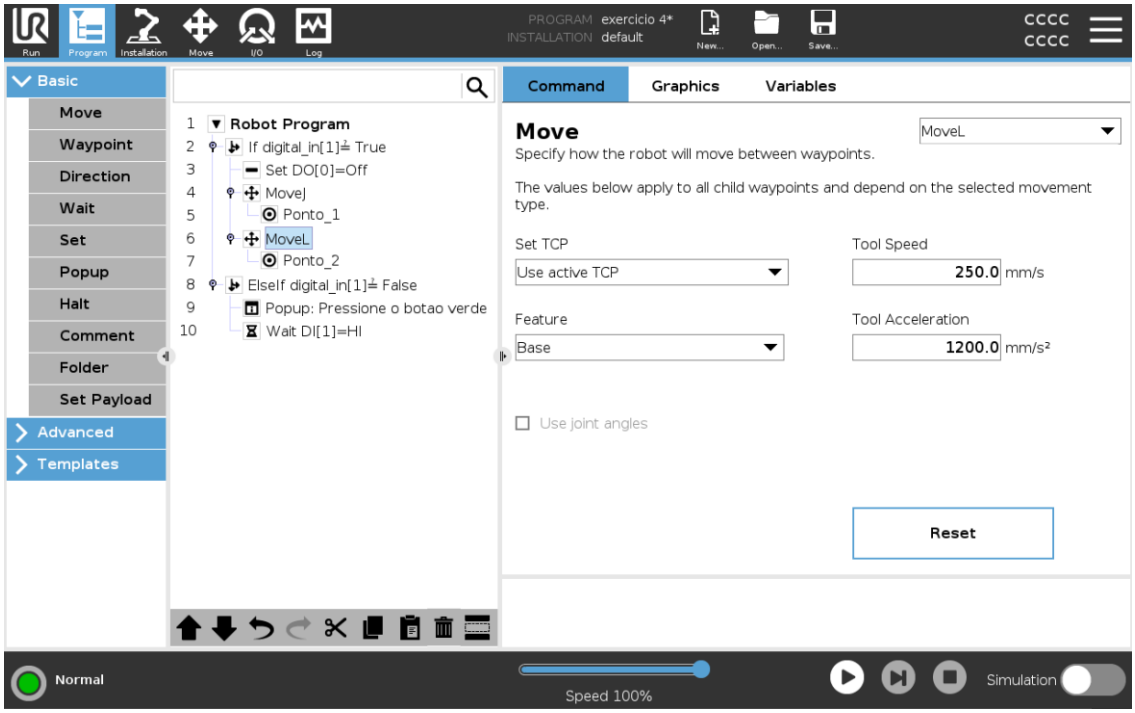


Figura 36 – Inserção da instrução *Set*

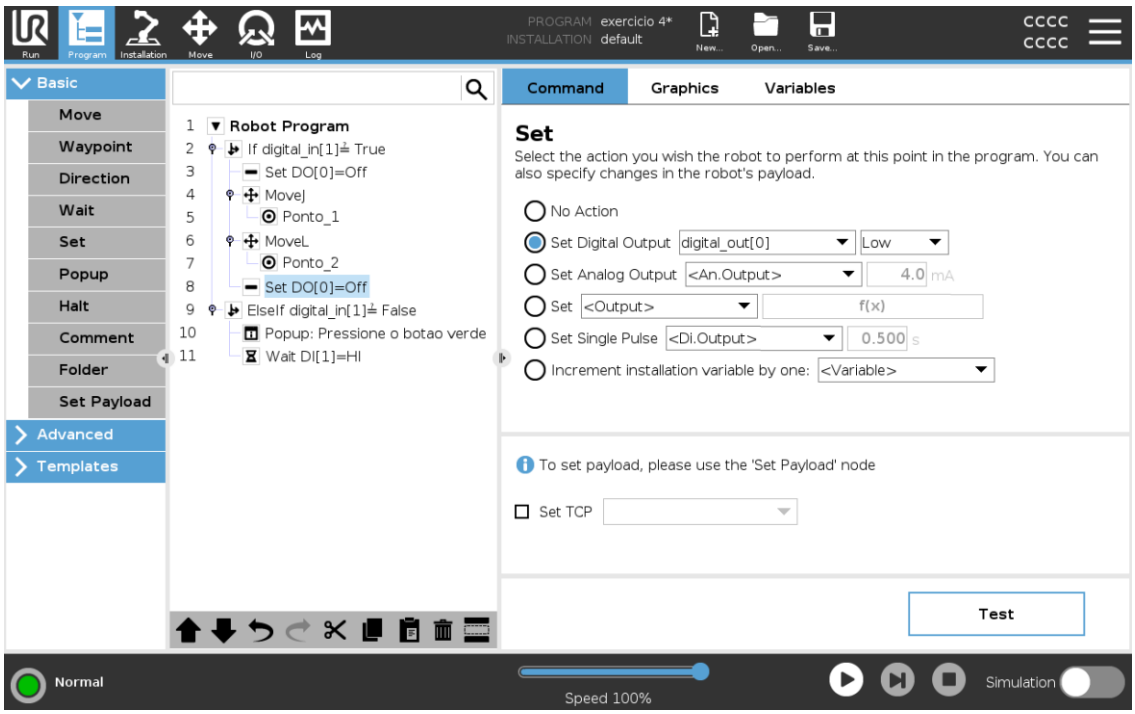


Figura 37 - Parametrização da instrução *Set*

Passo 4 – Teste e Validação

Testar o programa.

4.6. Exercício 5 - Ciclo *Pick and Place* com Contador

O objetivo do exercício consiste em desenvolver um programa no *PolyScope* que permita ao robô UR3e executar uma sequência de *Pick and Place*. Na Figura 38, está presente a área de trabalho do robô. Na posição inicial (zona A), encontram-se empilhadas cinco peças, mas apenas a primeira peça deve ser transportada até à posição 1 que se situa na zona B.

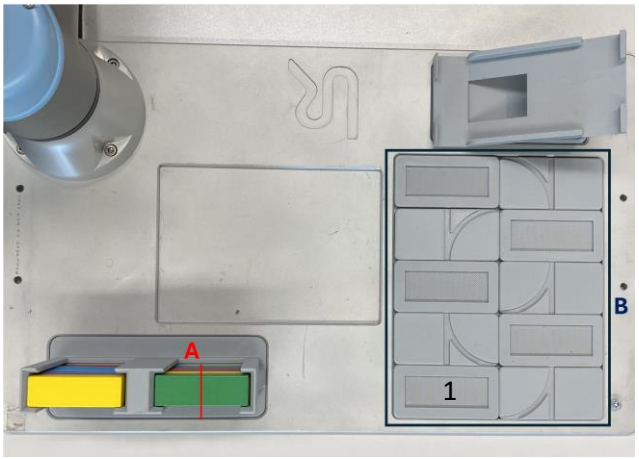


Figura 38 - Área de trabalho do robô

Requisitos do exercício:

1. O robô deve deslocar-se para a posição acima da zona A, que deverá ser definida como *waypoint* *before_start*;
2. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher (definir como *waypoint* *start_point*);
3. Acionar o *gripper* para agarrar a peça;
4. Aguardar 1 segundo;
5. Efetuar o movimento linear inverso até à posição *before_start*;
6. Transportar a peça até um ponto acima da zona B e defini-lo como *before_exit*;
7. Descer em movimento linear até à posição 1 e definir o ponto como *exit_point*;
8. Desativar o *gripper* para largar a peça;
9. Aguardar 1 segundo;
10. Incrementar o contador de peças paletizadas;
11. Após a colocação da 1ª peça, o programa deve terminar automaticamente
12. Durante todo o processo, manter o LED verde (DO0) ligado.

4.6.1. Metodologia de resolução (exercício 5)

Passo 1 – Configuração Inicial

Criar um programa denominado "*exercício 5*". Inserir uma secção ***Before Start***.

Para inserir a secção *Before Start*, deve-se ativar a opção *Add Before Start Sequence* situada no comando da secção *Robot Program* (ver Figura 39 e Figura 40). Esta secção permite definir todas as variáveis, condições de segurança e instruções, antes do programa começar.

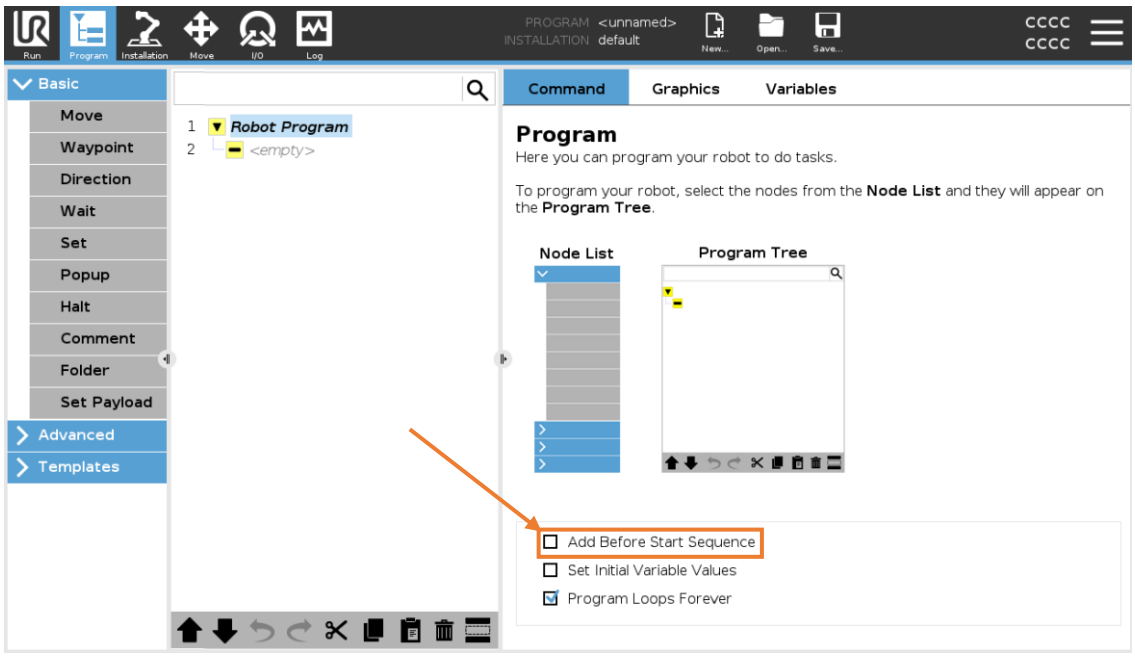


Figura 39 - Inserção da secção *Before Start*

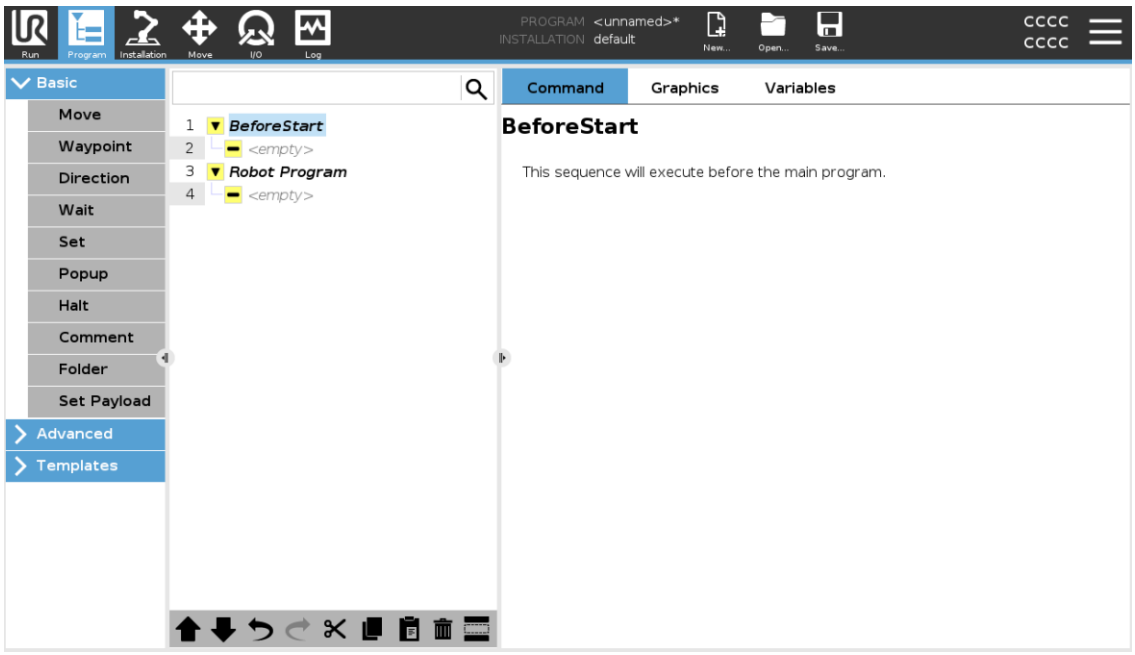


Figura 40 - Secção *Before Start* inserida

Definir as condições iniciais:

- Variável *count* com valor inicial = 0 (Figura 41-43), utilizar a instrução *Assignment*.
- Abertura das garras do *gripper*, através do comando *URCaps > Zimmer > Z_Release* (Figura 44).

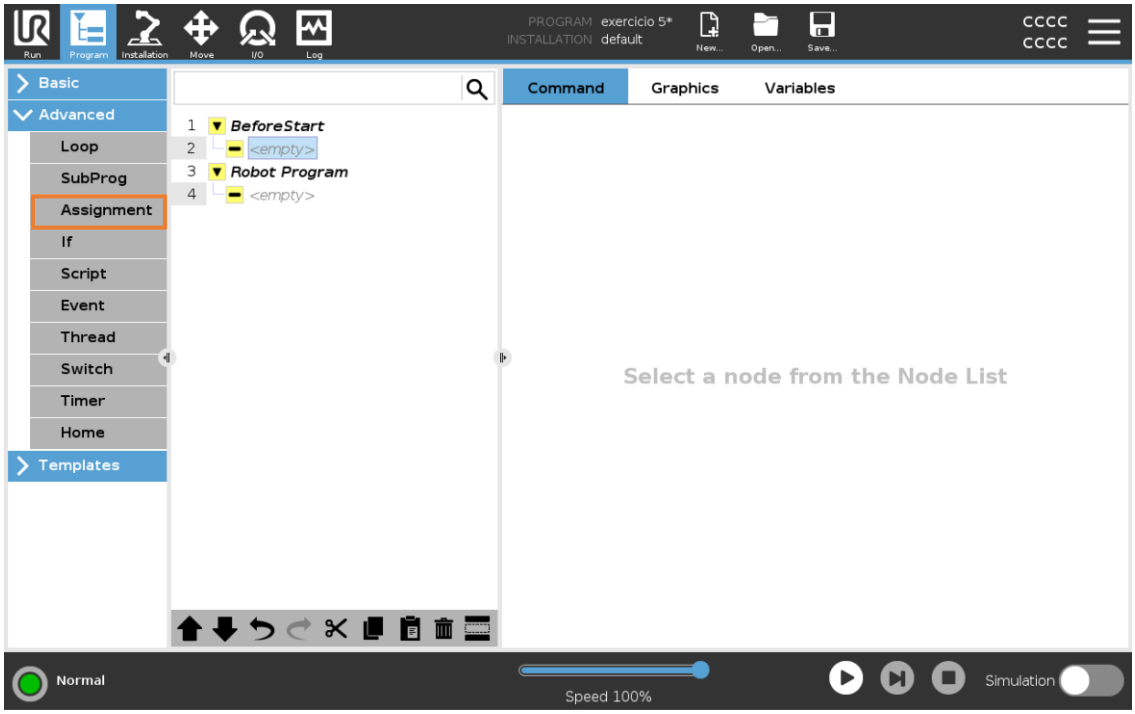


Figura 41 – Inserção da instrução *Assignment*

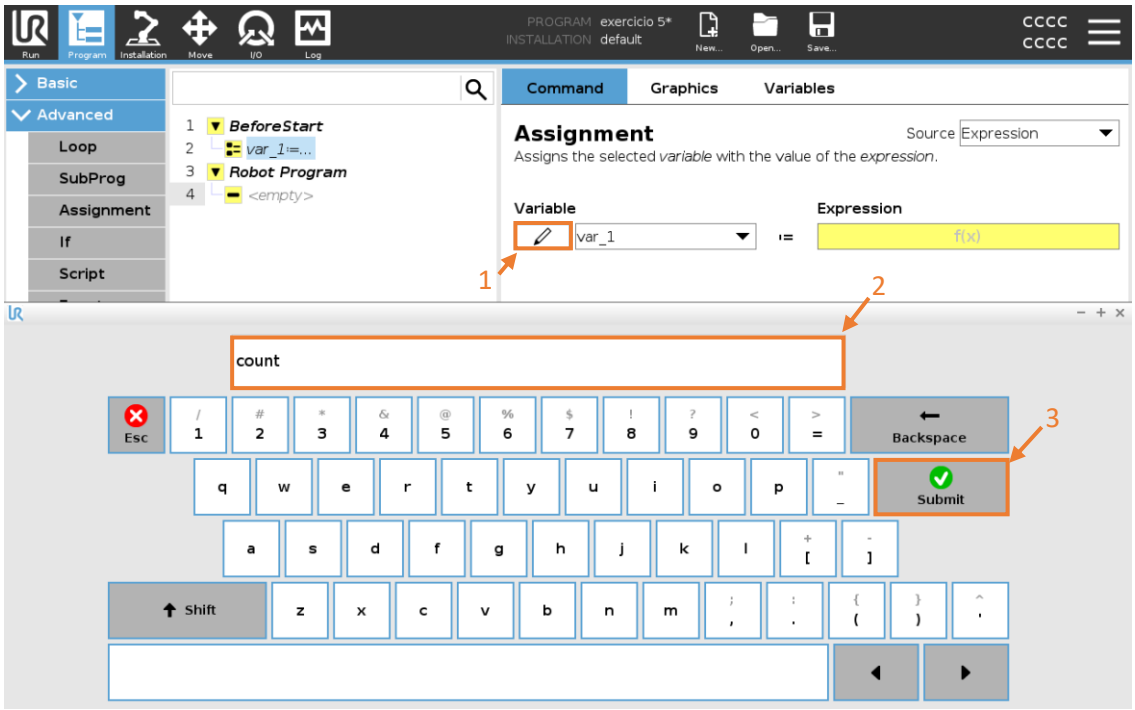


Figura 42 - Configurar a instrução *Assignment*

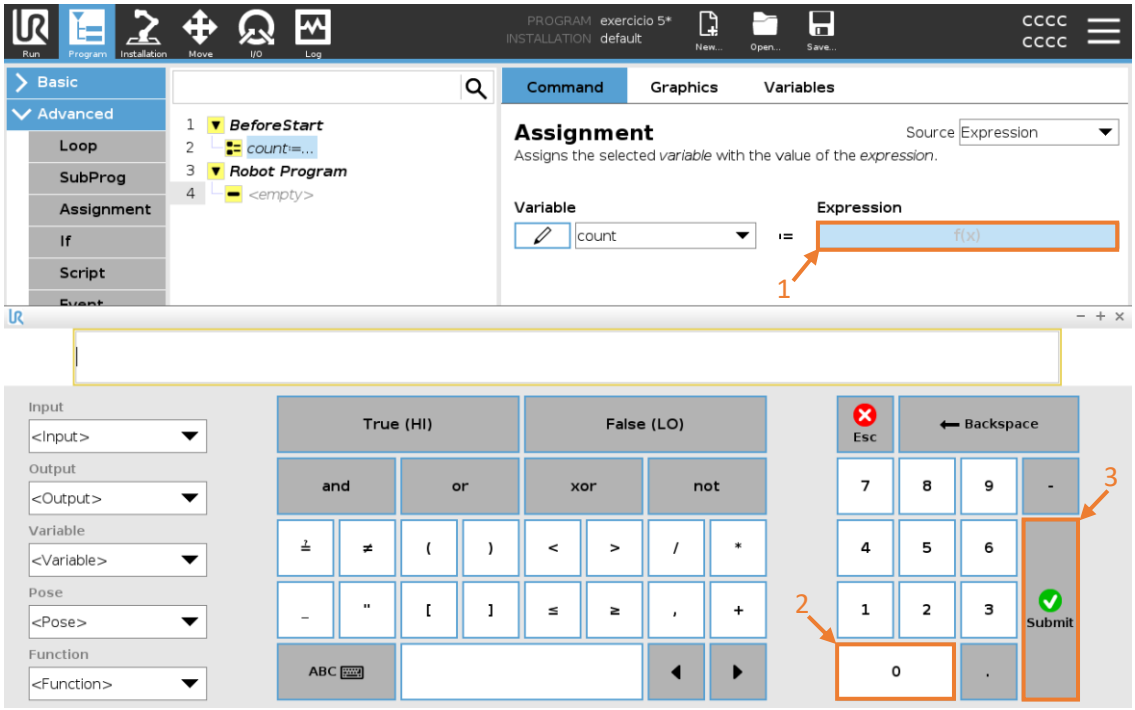


Figura 43 - Definir o valor inicial do contador

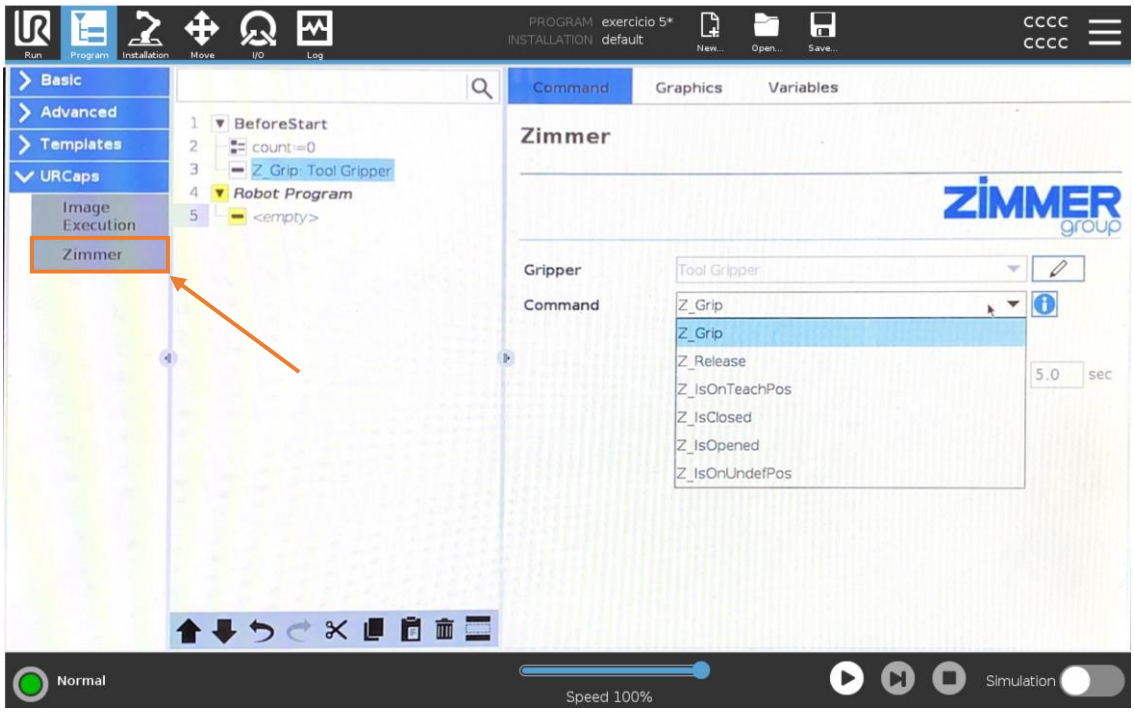


Figura 44 - Comandos do *Gripper*

Passo 2 – Programação do Ciclo *Pick and Place*

O programa principal inicia com uma instrução ***If***, verificando se ***count* < 1**. Numa fase inicial definir a sequência de recolha da peça:

1. Ativar o *LED* verde;

2. Mover o robô para uma posição acima da peça (zona A), posição *before_start*;
3. Mover o robô linearmente até à peça, posição *start_point*;
4. Acionar o Gripper para agarrar a peça, através do comando *URCaps > Zimmer > Z_Grip*.
5. Aguardar 1 segundo.

O objetivo é obter uma ramificação igual à que está presente na Figura 45.

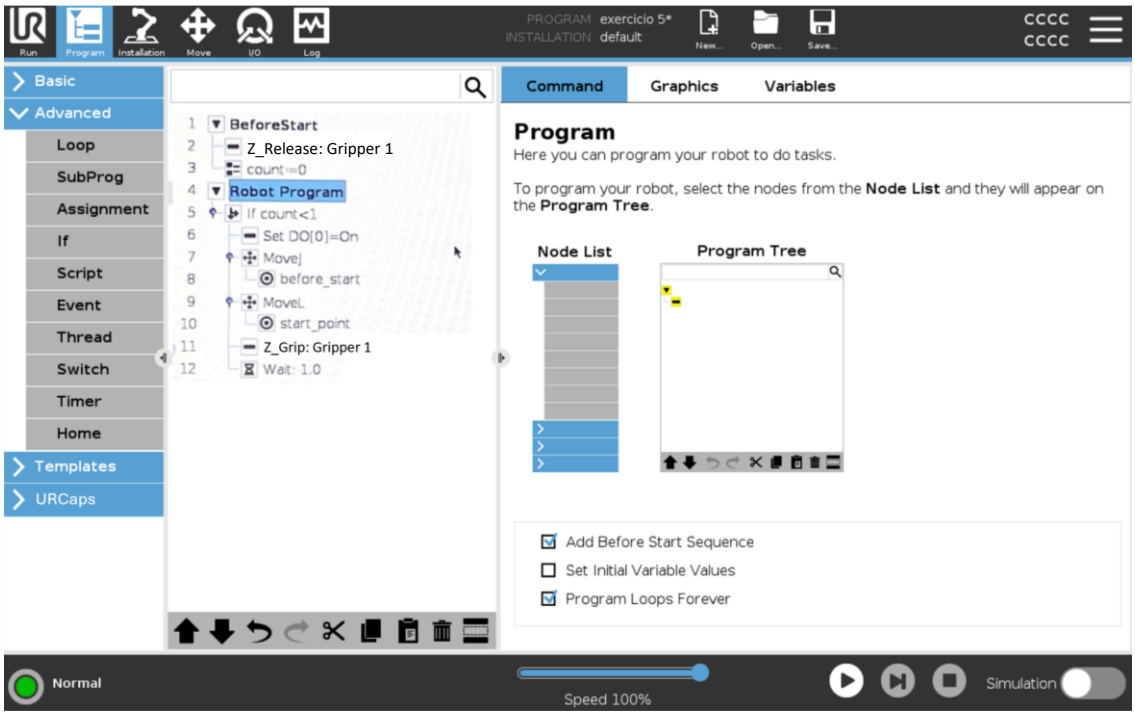


Figura 45 - Sequência de programação 1

De seguida, é necessário definir a sequência de colocação (*Place*) da peça:

1. Mover o robô em movimento linear até a posição *before_start;*
2. Mover o robô até à zona B para depósito, posição *before_exit*;
3. Descer o robô em movimento linear até à posição 1 (*exit_point*);
4. Libertar a peça;
5. Aguardar 1 segundo;
6. Incrementar o contador.

No final obtém-se uma sequência de programação semelhante à que está presente na Figura 46.

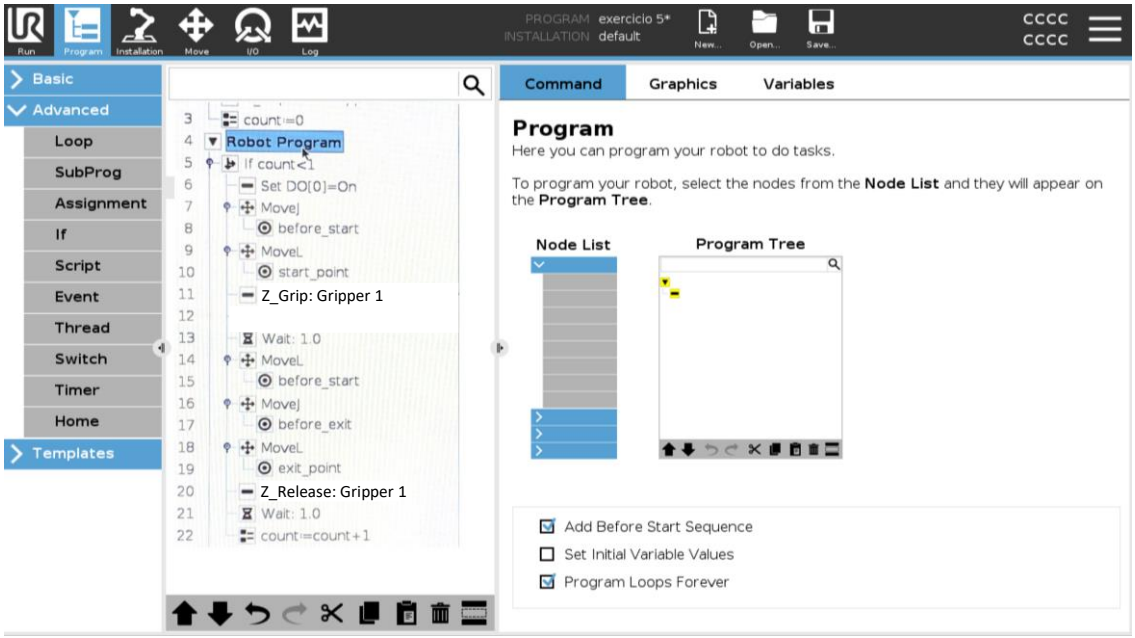


Figura 46 – Conclusão da lógica de movimentação

Passo 3 – Instrução *ElseIf* e Paragem do Programa

Adicionar uma instrução ***ElseIf*** para verificar se ***count = 1*** (Figura 47). Neste caso:

- Mover o robô para a posição *before_start*;
- Desativar o *LED* verde;
- Utilizar a instrução ***Halt*** para parar o programa (a instrução situa-se na secção *Basic*).

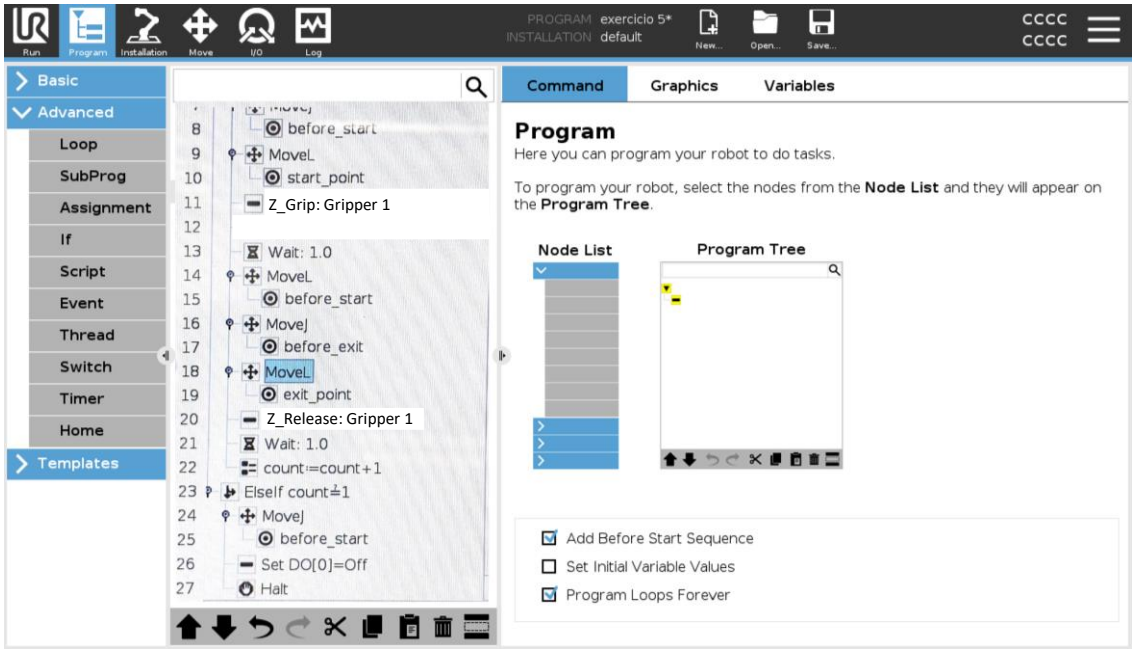


Figura 47 - Instrução *ElseIf* para terminar o programa

Passo 4 – Teste e Validação

4.7. Exercício 6 - Paletização

O objetivo deste exercício é desenvolver uma aplicação de paletização automática utilizando o robô colaborativo UR3e. O robô deverá realizar operações de *Pick and Place* para transferir peças da zona de recolha (zona A) para a zona de paletização (zona B), de forma ordenada e repetitiva. A tarefa deverá ser executada até que cinco peças sejam colocadas na zona B (Figura 48), parando automaticamente após a última colocação.

Requisitos do exercício:

1. A execução do programa deve iniciar-se apenas quando o botão verde (DI1) for pressionado;
2. Após o sinal de início, o robô deve deslocar-se para a posição acima da zona A, que deverá ser definida como *waypoint before_start*;
3. De seguida, deve ser executado um movimento linear até ao centro da peça a recolher (definir como *waypoint start_point*);
4. Acionar o *gripper* para agarrar a peça;
5. Aguardar 1 segundo;
6. Efetuar o movimento linear inverso até à posição *before_start*;
7. Transportar a peça até à zona B e colocá-la na próxima posição livre (de 1 a 5);
8. Desativar o *gripper* para largar a peça;
9. Aguardar 1 segundo;
10. Incrementar o contador de peças paletizadas;
11. Repetir o ciclo até que cinco peças tenham sido colocadas;
12. Durante todo o processo, manter o LED verde (DO0) ligado;
13. Após a colocação da 5ª peça, o programa deve terminar automaticamente.

Na Figura 48 encontra-se a representação da superfície de trabalho com as zonas A e B assinaladas. A zona A contém 5 peças empilhadas de diferentes cores. A zona B está subdividida em cinco compartimentos onde as peças deverão ser colocadas.

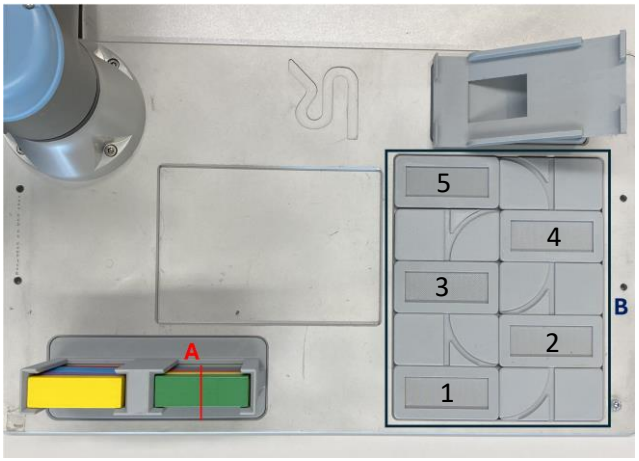


Figura 48 - Sinalização da superfície de trabalho

Materiais e equipamentos necessários:

- Robô colaborativo UR3e e controlador;
- Interface de programação *PolyScope* para desenvolvimento, execução e monitorização do código;
- *LED* verde – DO0;
- Botões: de 2 posições (DI0) e verde (DI1);
- *Gripper Zimmer*.
- Estrutura física com:
    - zona A: local de recolha de peças;
    - zona B: local de paletização de peças;

Instruções adicionais:

- A espessura/altura das peças a considerar é de **15 mm**.
- A paletização deve seguir a **ordem crescente das posições** na zona B (1 → 5)

4.7.1. Metodologia de resolução (exercício 6)

Passo 1 – Criação de um novo programa

Criar um programa no *PolyScop*e com o nome *ex_paletização*.

Passo 2 – Definição de variáveis (*Before Start*)

Na secção *before start*, definir:

- A variável count (contador) com valor inicial = 0;
- O *gripper* com as garras abertas;
- O *LED* verde em estado desligado;
- O *Popup* para aviso enquanto o robô espera que o botão verde seja acionado.

Passo 3 – Criação de subprogramas

Criar dois subprogramas:

- *Pick*;
- *Place*.

Para criar subprogramas é necessário inserir a instrução *SubProg* com a secção *Robot Program* ativa (Figura 49 e Figura 50).

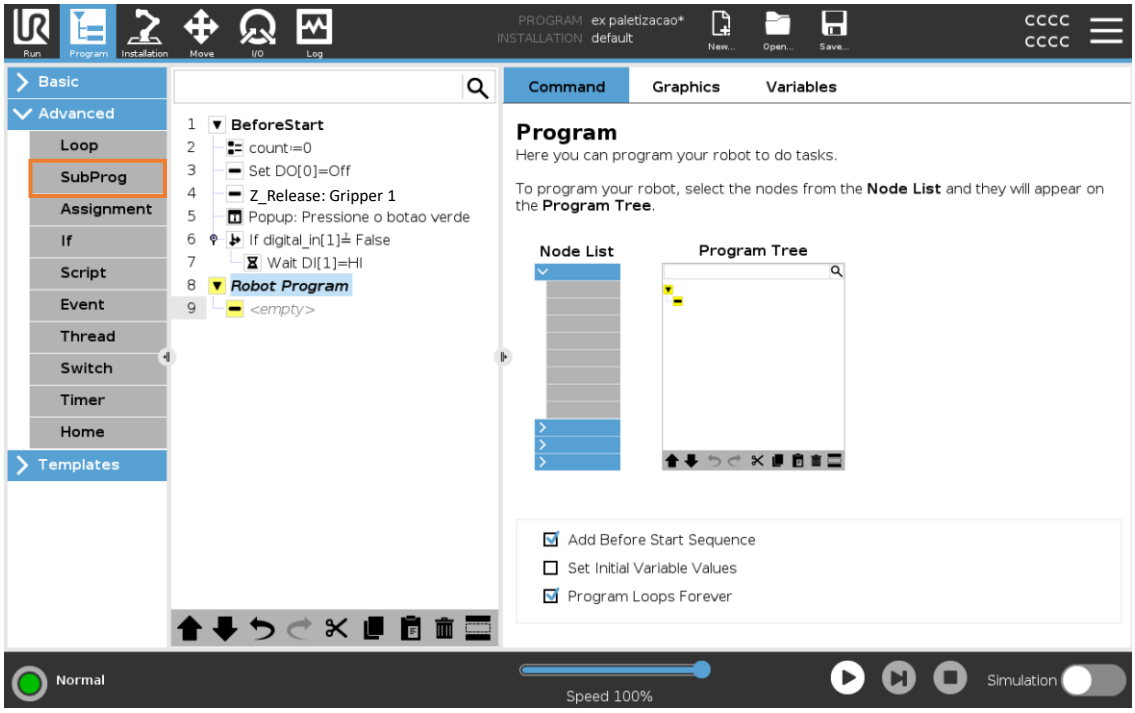


Figura 49 - Inserção da instrução *SubProg*

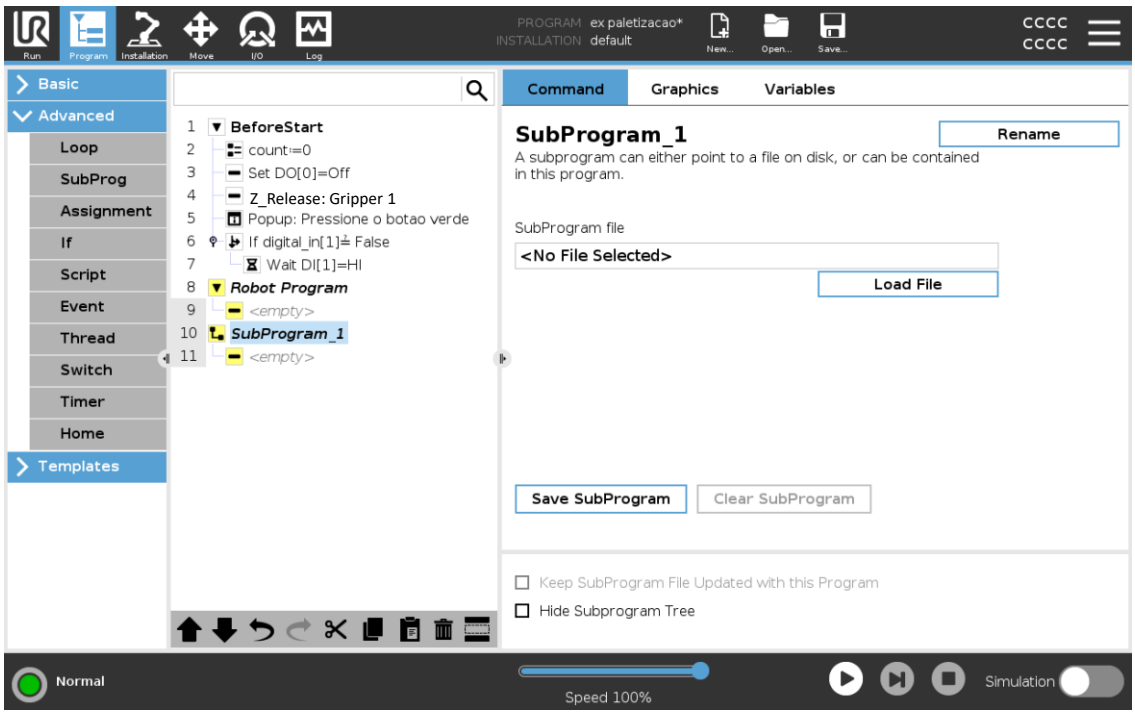


Figura 50 - Instrução *SubProg*

De seguida, atribuir o nome *Pick* ao subprograma 1 (Figura 51) e *Place* ao subprograma 2 (Figura 52).

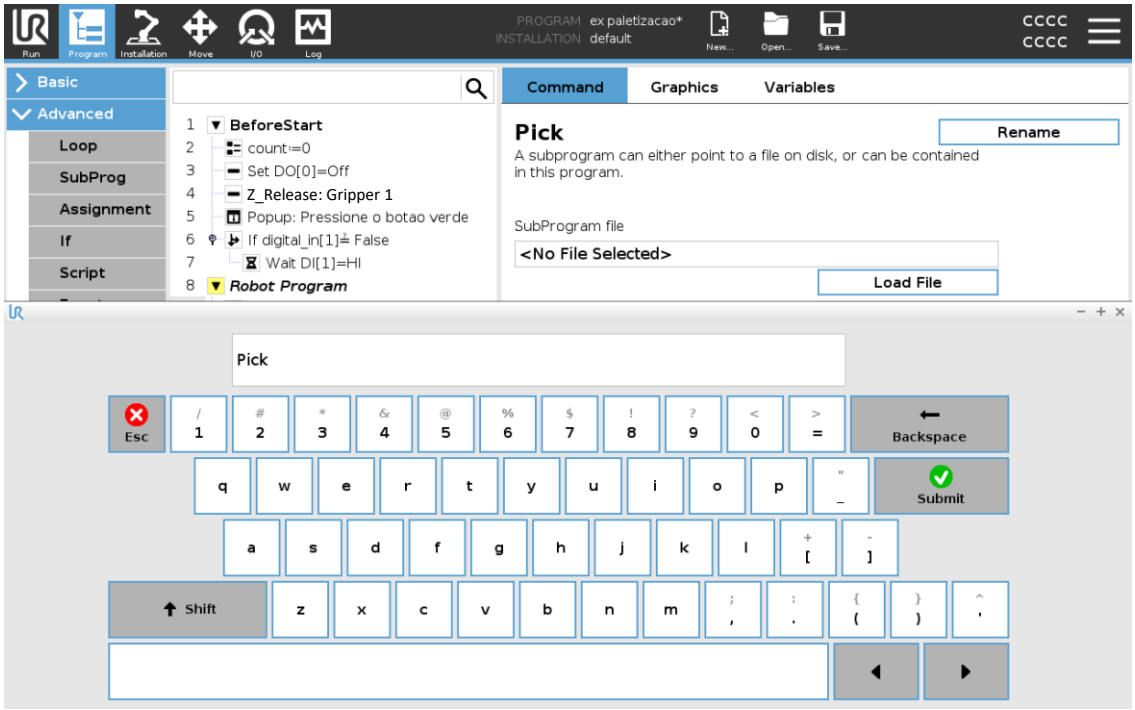


Figura 51 - Denominação da instrução *SubProg*

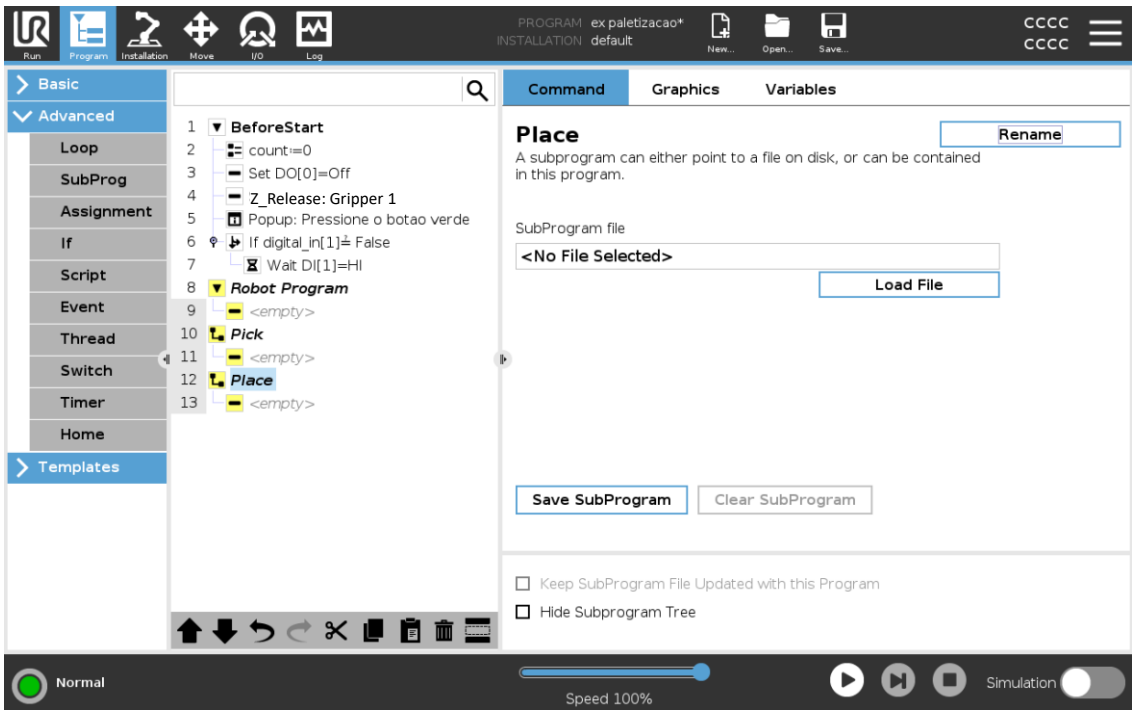


Figura 52 - Ramificação obtida com os dois subprogramas criados

Passo 4 – Configuração do subprograma "*Pick*"

A instrução *Seek* permite realizar o processo de desempilhamento de peças de forma automática, sendo apenas necessário definir as secções da instrução. Como na zona A estão 5 peças empilhadas pode-se utilizar esta instrução. Para a inserir aceda à área de programação *Templates*, selecionando a opção *Seek* (Figura 53).

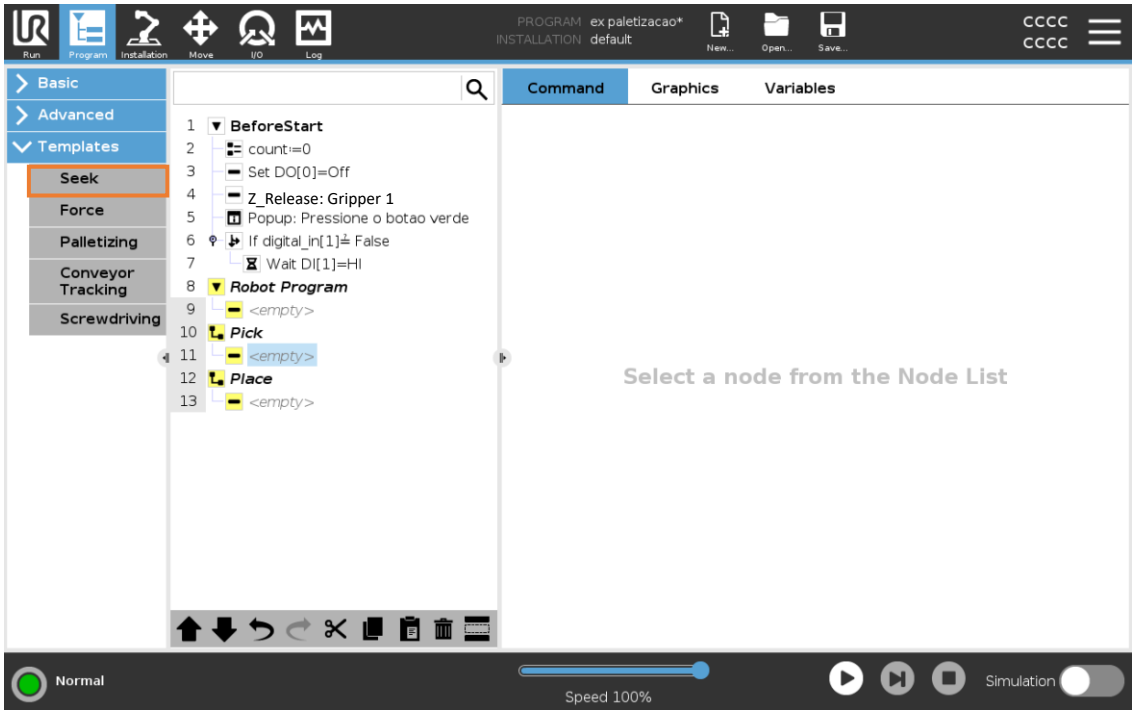


Figura 53 - Inserção da instrução *Seek*

De seguida, selecionar o tipo *Destacking* que corresponde à opção de desempilhamento (Figura 54).

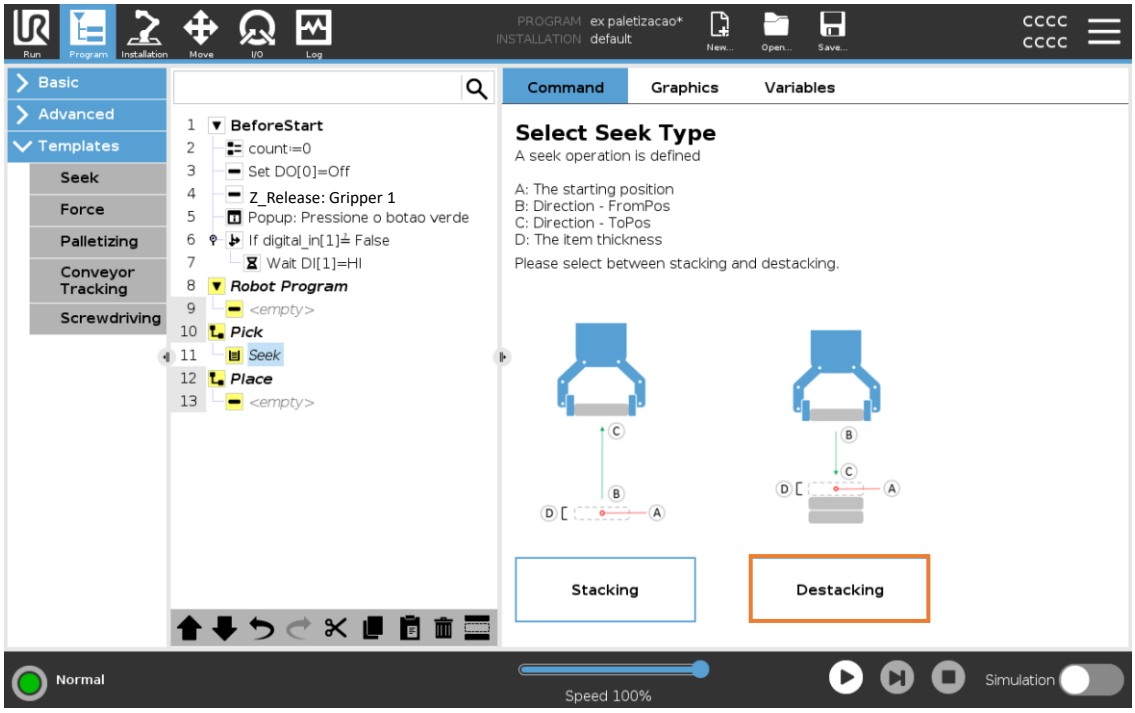


Figura 54 - Seleção do tipo da instrução *Seek*

Para ser possível visualizar todos os parâmetros da instrução é necessário abrir a ramificação presente na secção *Direction* e *PickSequence* (Figura 55). Deve-se obter uma ramificação igual à que está representada na Figura 56.

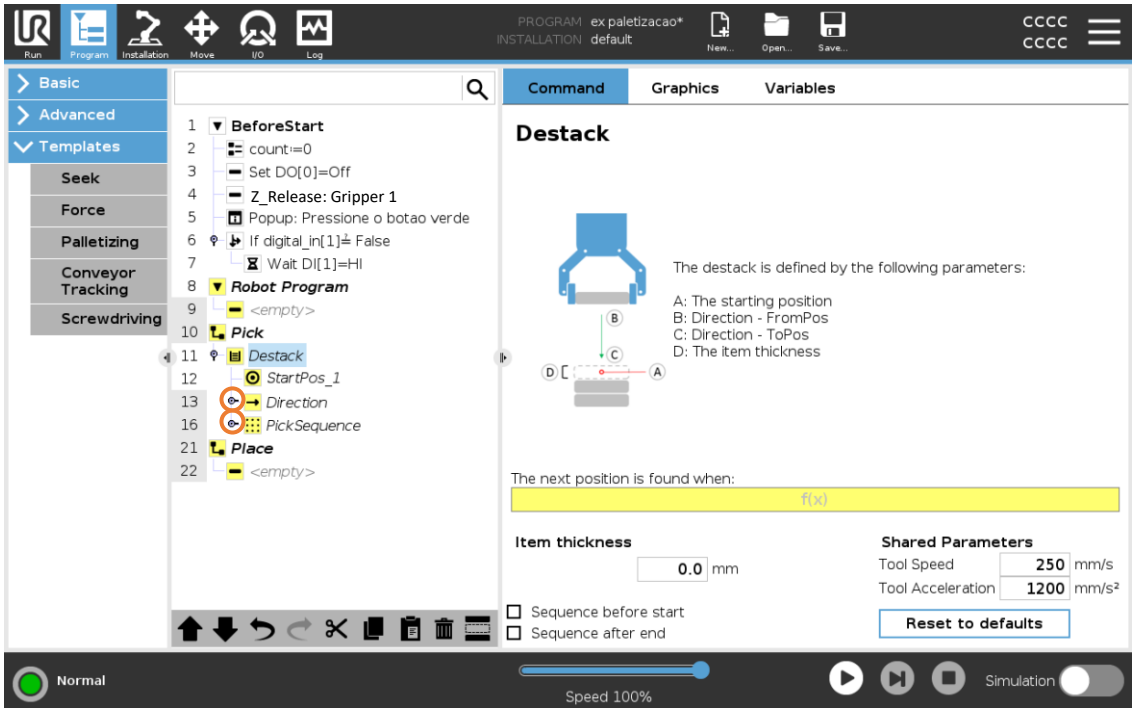


Figura 55 - Tipo *Destack*

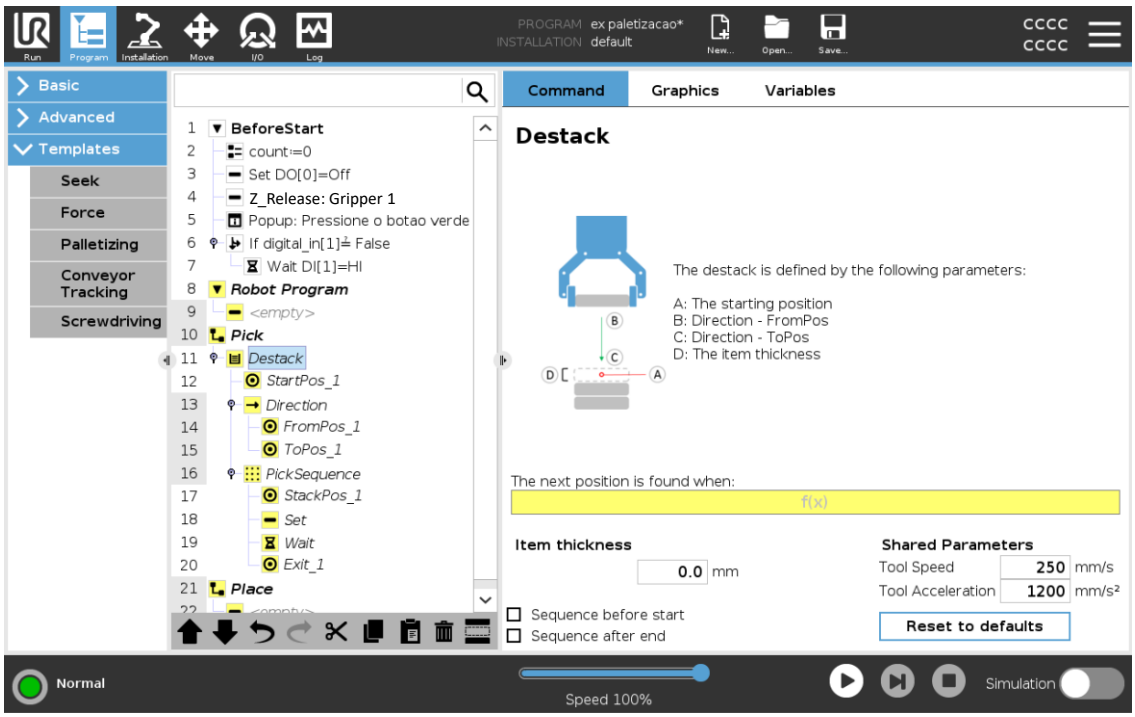


Figura 56 - Parâmetros a configurar na instrução *Seek*

Inicialmente é necessário definir qual é a condição para encontrar a próxima peça e a espessura da peça (ver Figura 57). Neste caso, pode utilizar o botão DI0 de 2 posições como condição. Se o botão estiver ativo serve de sinal de "peça disponível". Desta forma, o robô desce **15 mm** sempre que volta à posição inicial (correspondente à espessura da peça) e pega na peça seguinte.

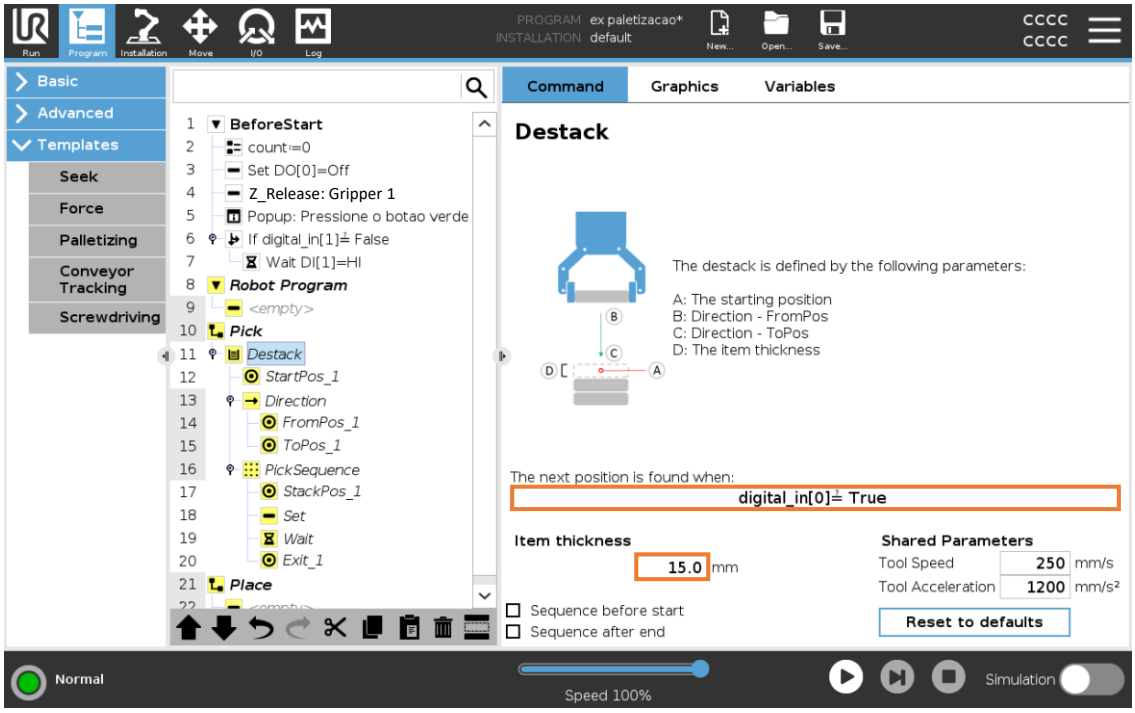


Figura 57 - Parâmetros iniciais da instrução *Seek*

De seguida, define-se o ponto inicial do ciclo *StartPos_1* (Figura 58). Este ponto corresponde ao local de recolha da primeira peça na zona A (aproximadamente localizado no centro da peça).

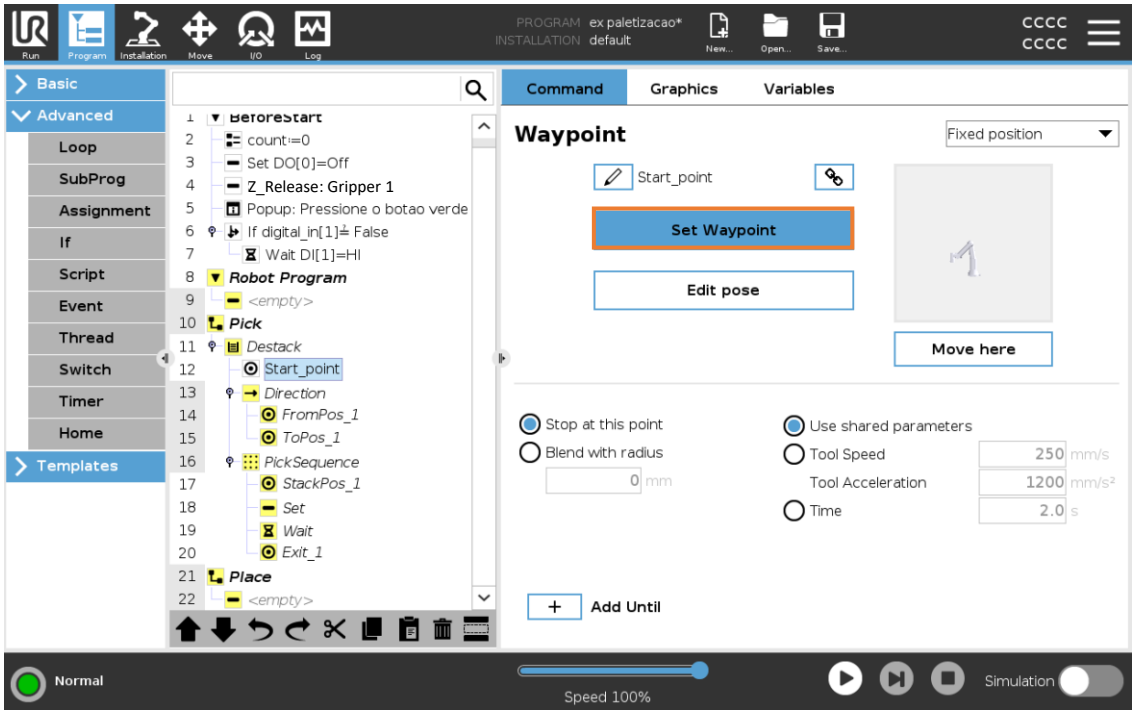


Figura 58 - Definição do ponto de início do movimento

Posteriormente, define-se a direção cartesiana do movimento (descendente no eixo z). O primeiro ponto (*FromPos_1*) corresponde ao ponto linearmente acima da peça a ser recolhida

(*before_start*) (Figura 59) e o segundo ponto (*ToPos_1*) corresponde ao local de recolha (*start_point*) (Figura 60).

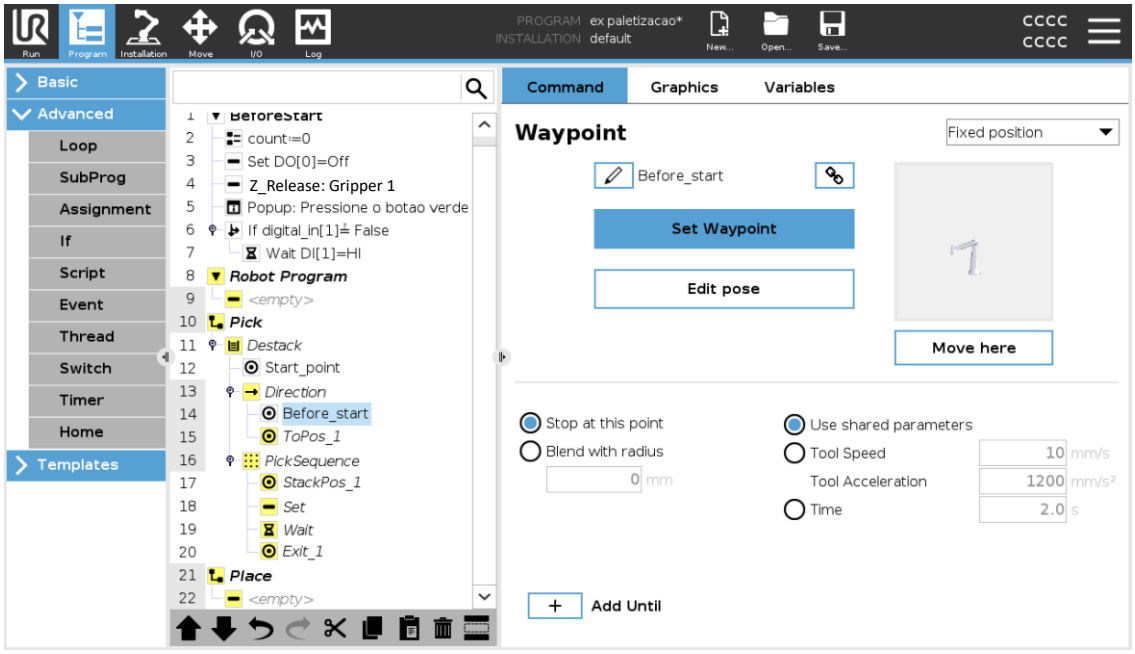


Figura 59 - Definição da direção do movimento_1

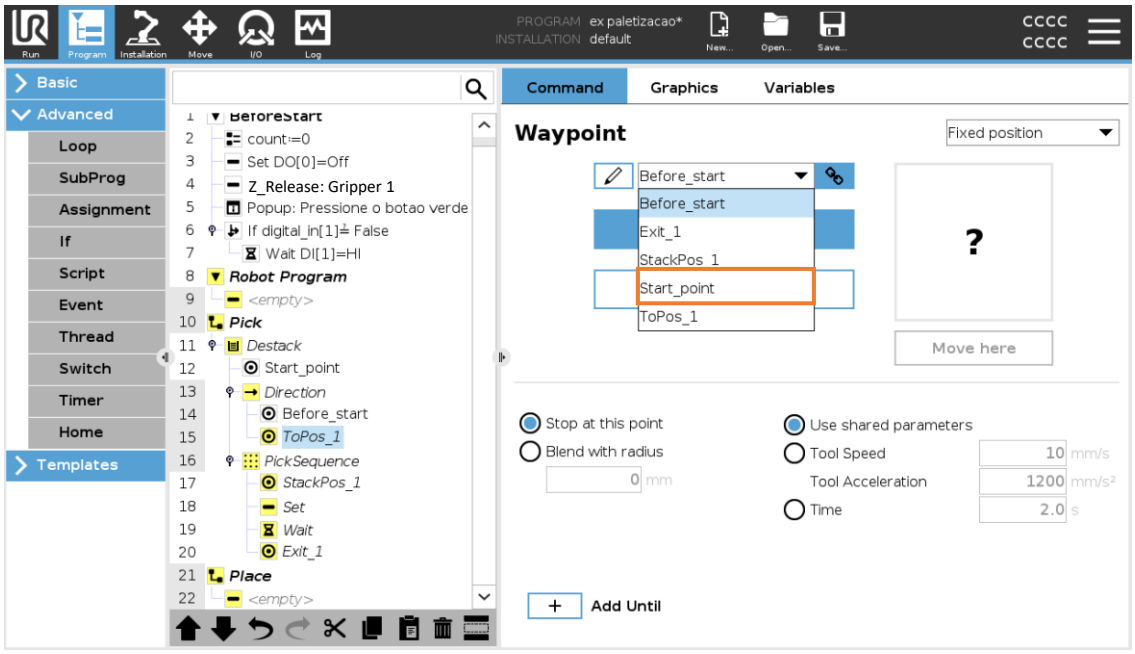


Figura 60 - Definição da direção do movimento_2

Para finalizar a secção da direção, basta definir que o robô para quando atingir 130 mm (Figura 61).

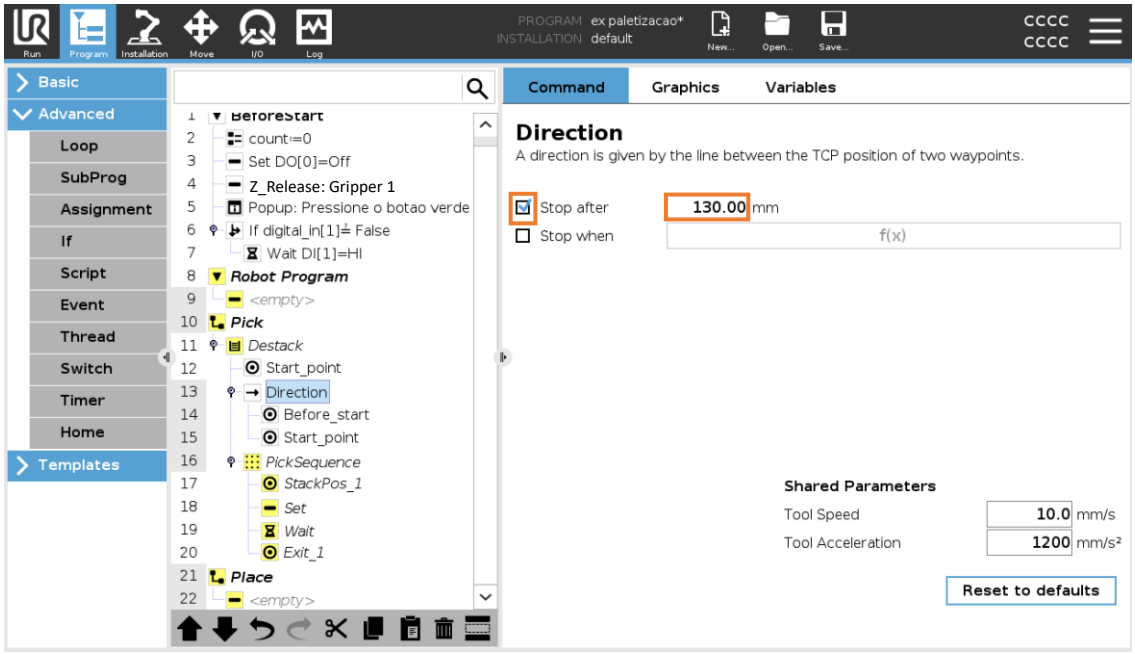


Figura 61 - Definição da direção do movimento_3

A próxima secção corresponde à sequência de movimentos (*PickSequence*) que se pretendem realizar no movimento de recolha. Deve-se seguir a seguinte lógica:

- Definir o *StackPos_1*, que corresponde ao ponto inicial de recolha *start_point*;
- Acionar o *gripper* para agarrar a peça;
- Esperar um segundo;
- Subir em movimento linear até ao ponto *before_start*;
- Incrementar 1 ao contador.

Após a parametrização das secções, finaliza-se o subprograma de recolha das peças (*Pick*) (Figura 62).

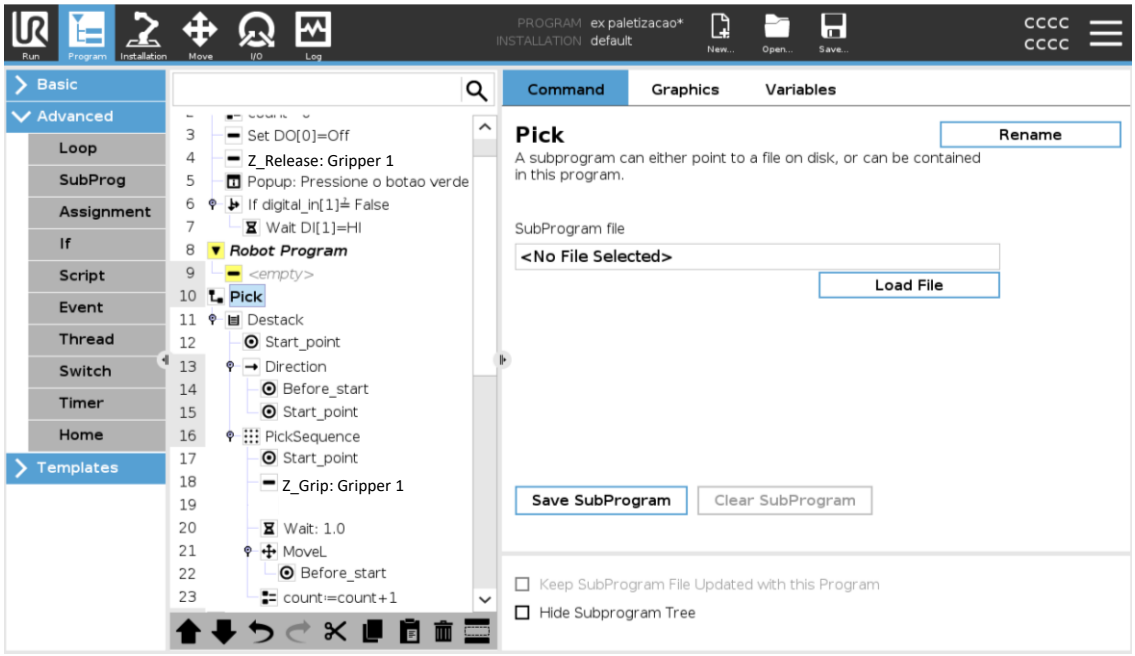


Figura 62 - Subprograma *Pick* finalizado

Passo 5 – Configuração do subprograma "*Place*"

A instrução *Palletizing* permite realizar a paletização de peças de forma mais simplificada. Para começar, insere-se a instrução no subprograma 2 (*Place*) (Figura 63).

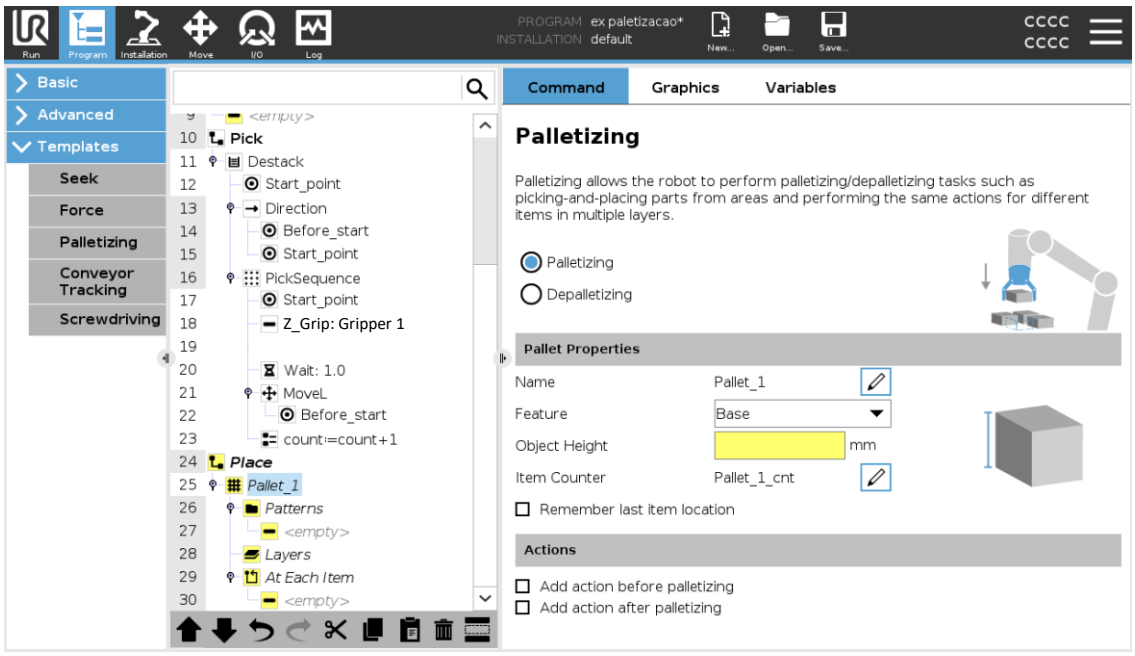


Figura 63 - Instrução *Palletizing*

Numa primeira fase, definir a altura do objeto **15 mm** e ativar a opção *Remember last item location* na secção *Pallet_1* (Figura 64).

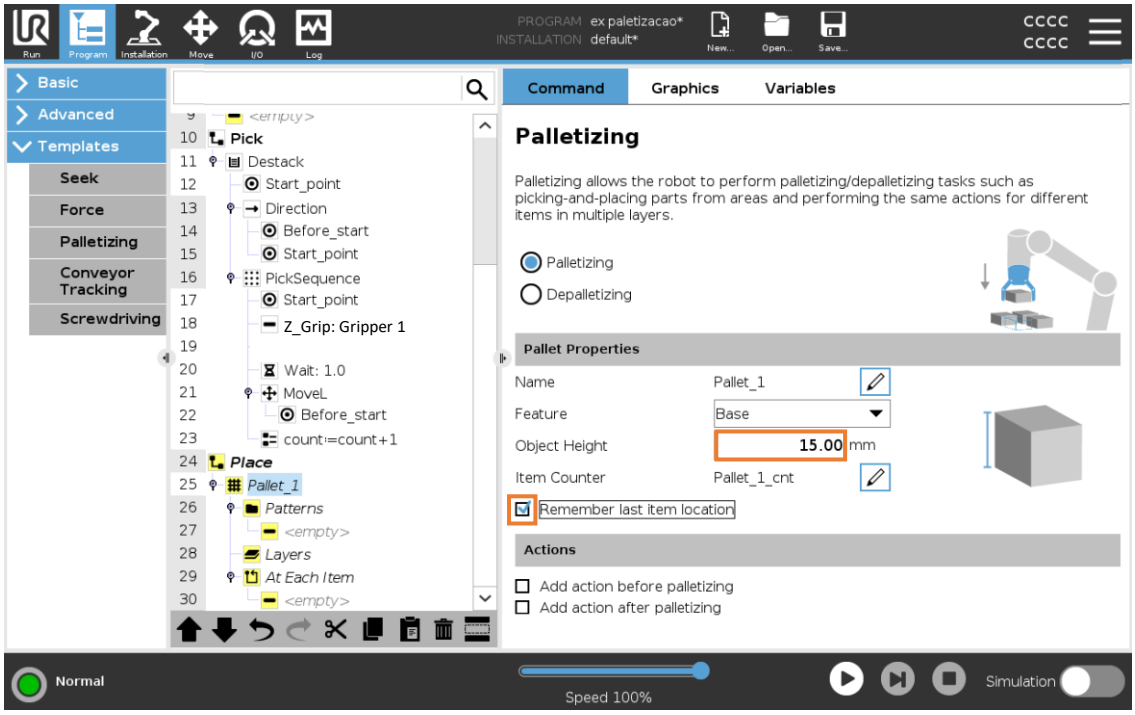


Figura 64 - Parametrização da secção Pallet_1

Numa segunda fase, é necessário definir o tipo de paletização que se pretende realizar. A instrução *Palletizing* permite realizar a paletização de peças em três padrões diferentes, designadamente linha, grelha e irregular. Abaixo segue uma breve explicação das mesmas:

1. **Em Linha (*Line*):**
    - As peças são depositadas sequencialmente ao longo de uma única linha (horizontal ou vertical);
    - Útil quando há apenas uma fila de posições disponíveis.
2. **Em Grelha (*Grid*):**
    - As peças são distribuídas numa matriz com várias linhas e colunas, seguindo um padrão uniforme;
    - Ideal para paletização com organização regular e geométrica.
3. **Irregular (*Irregular*):**
    - As posições de depósito são definidas manualmente ou com base num percurso não sistemático.
    - Indicado para casos onde o layout das paletes ou o posicionamento dos objetos não segue uma simetria.

Neste caso, o tipo de paletização é irregular (Figura 65).

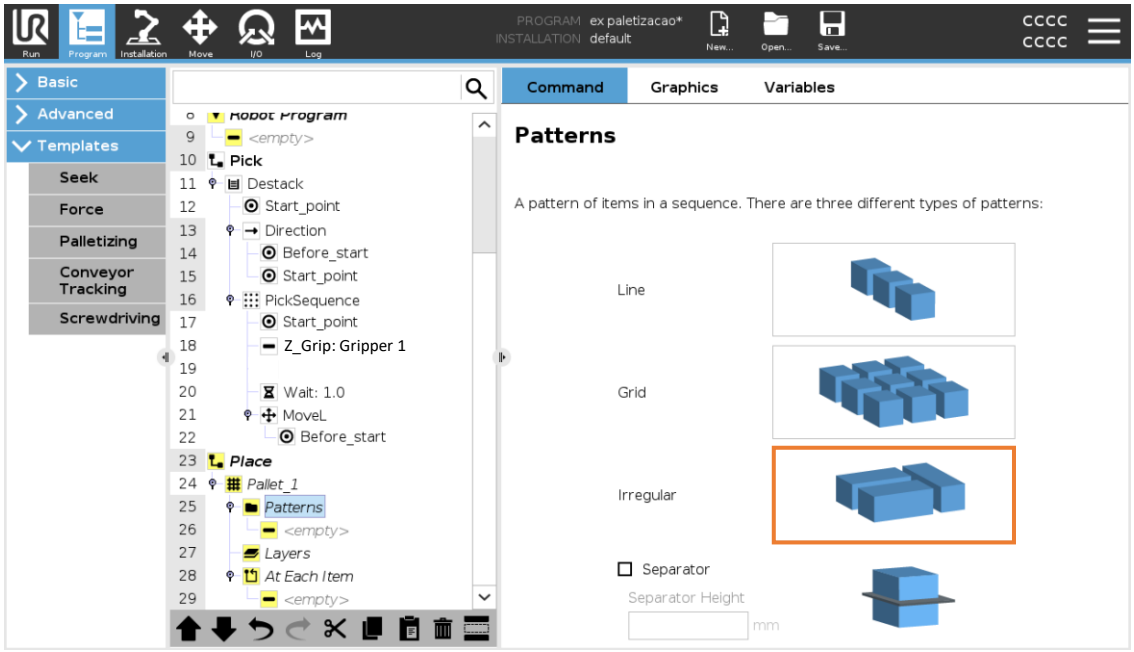


Figura 65 - Opções de Paletização

De seguida, parametriza-se a secção *Irreg_Pattern_1*. Por padrão, a sequência irregular acrescenta de forma automática 3 itens (*Item_1, Item_2* e *Item_3*). Cada Item corresponde a uma posição de deposição. Como o objetivo do exercício é realizar a paletização de 5 peças, é necessário definir 5 itens na sequência. Para inserir os itens, é necessário clicar na opção *Add Item,* conforme indicado na Figura 66.

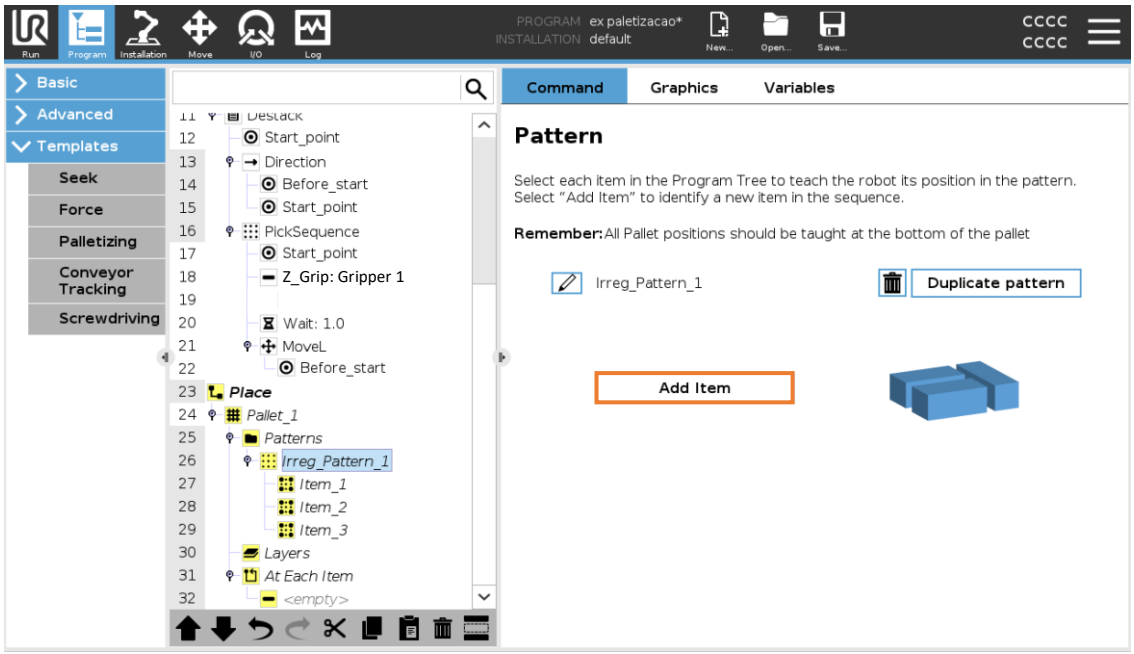


Figura 66 - Adição de itens

Posteriormente, define-se a localização de cada item. A posição de origem de paletização (*Item_1*) deve coincidir com a célula que está assinalada com o número 1 e assim sucessivamente. Na Figura 67 estão presentes a distância entre posições de paletização em

milímetros e a numeração das mesmas. Após a definição do *Item_1* pode definir os restantes itens alterando apenas as coordenadas x e y dos pontos.

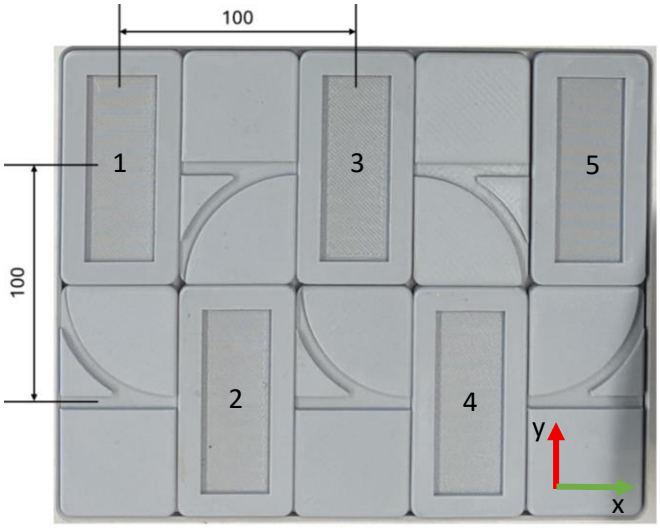


Figura 67 - Distância entre paletes e numeração das mesmas

Numa terceira fase seleciona-se o tipo de camada. Definir o *Irreg_Pattern_1* para a camada 1 (*Layer_1*) (Figura 68).

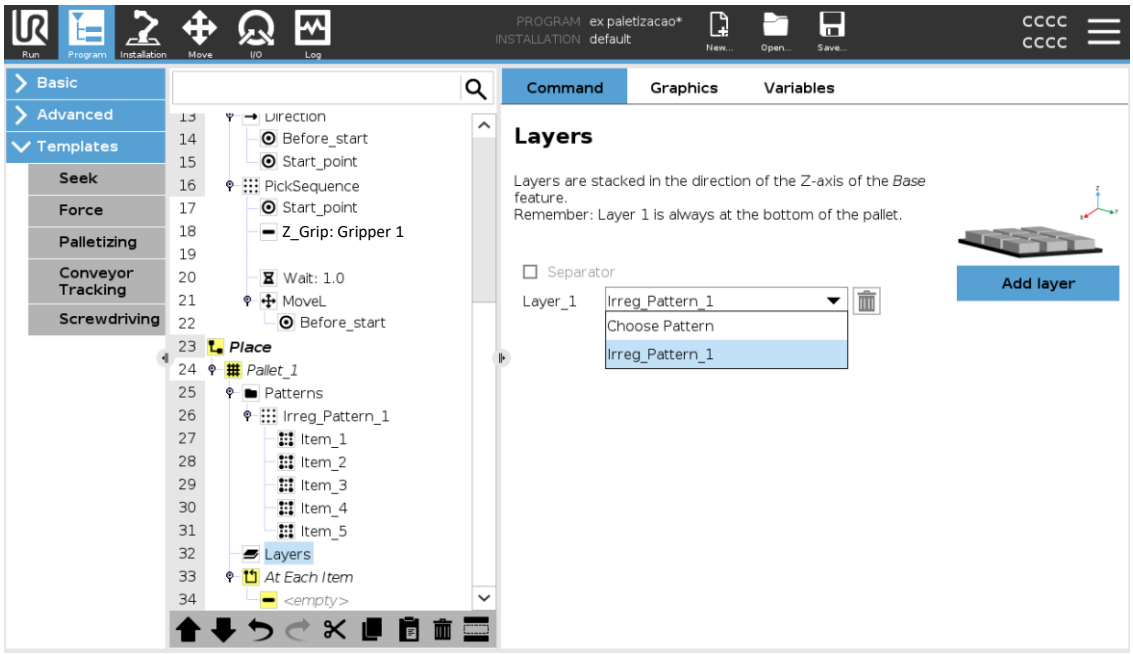


Figura 68 - Parametrização da *Layer*

Numa quarta fase, é definida a sequência de movimentos que vai ser realizada em cada item. Nesta secção segue-se um procedimento inicial de 6 passos:

- Passo 1: Clicar em *Next* (Figura 69);
- Passo 2: Mover o robô para o ponto de referência (Figura 70);

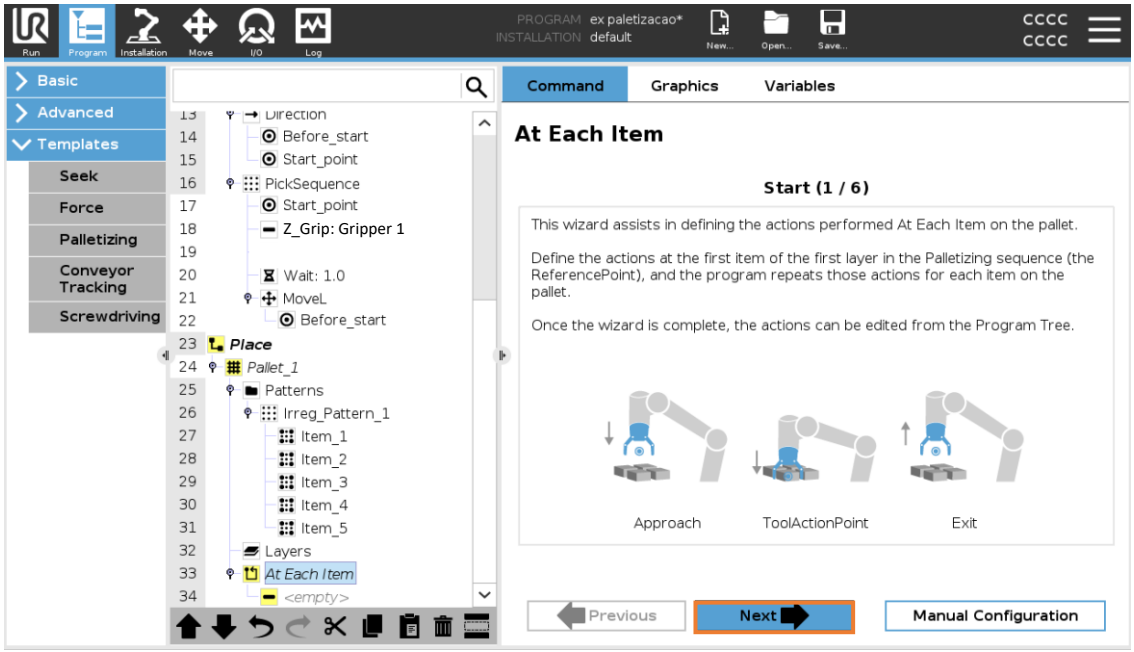


Figura 69 - Primeiro passo da sequência

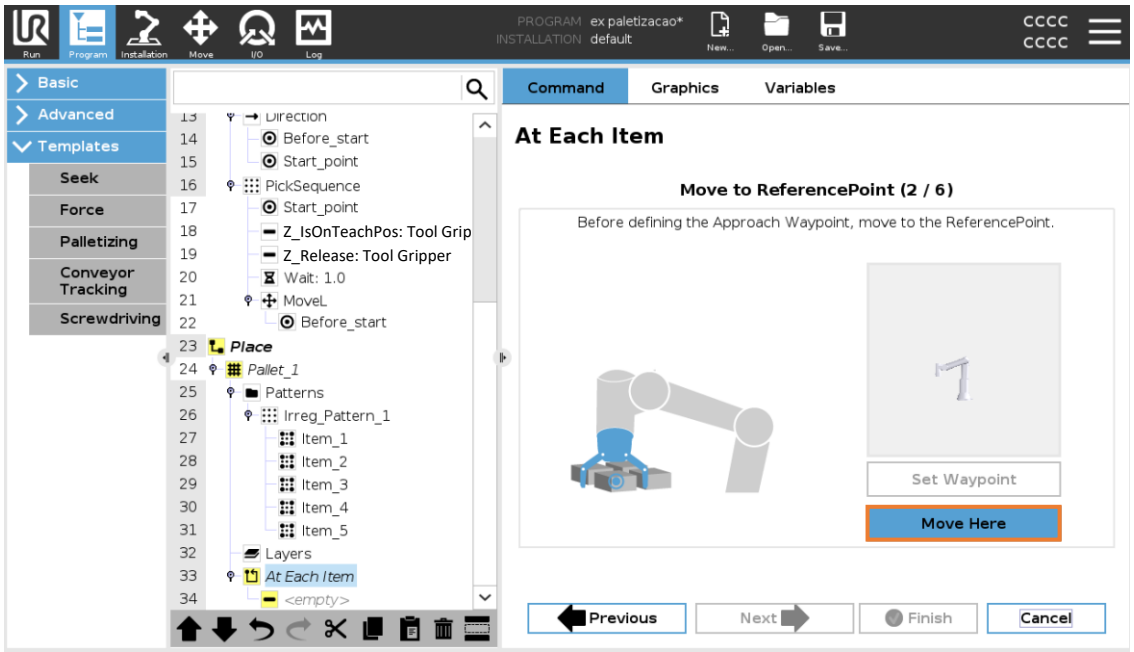


Figura 70 - Segundo passo da sequência

- Passo 3: Definir um ponto de segurança quando o robô se dirige para a palete. Pode-se definir um ponto linearmente acima da posição 1 (cota z superior) (Figura 71);
- Passo 4: Mover o robô para o ponto de referência (Figura 72);

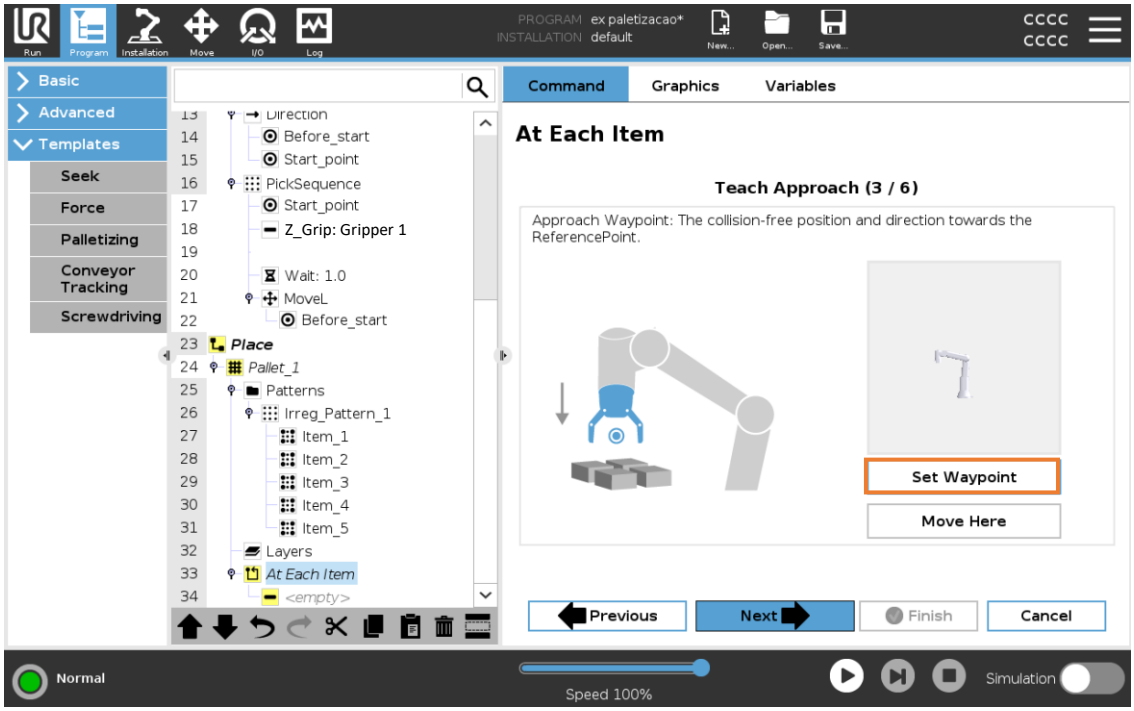


Figura 71 - Terceiro passo da sequência

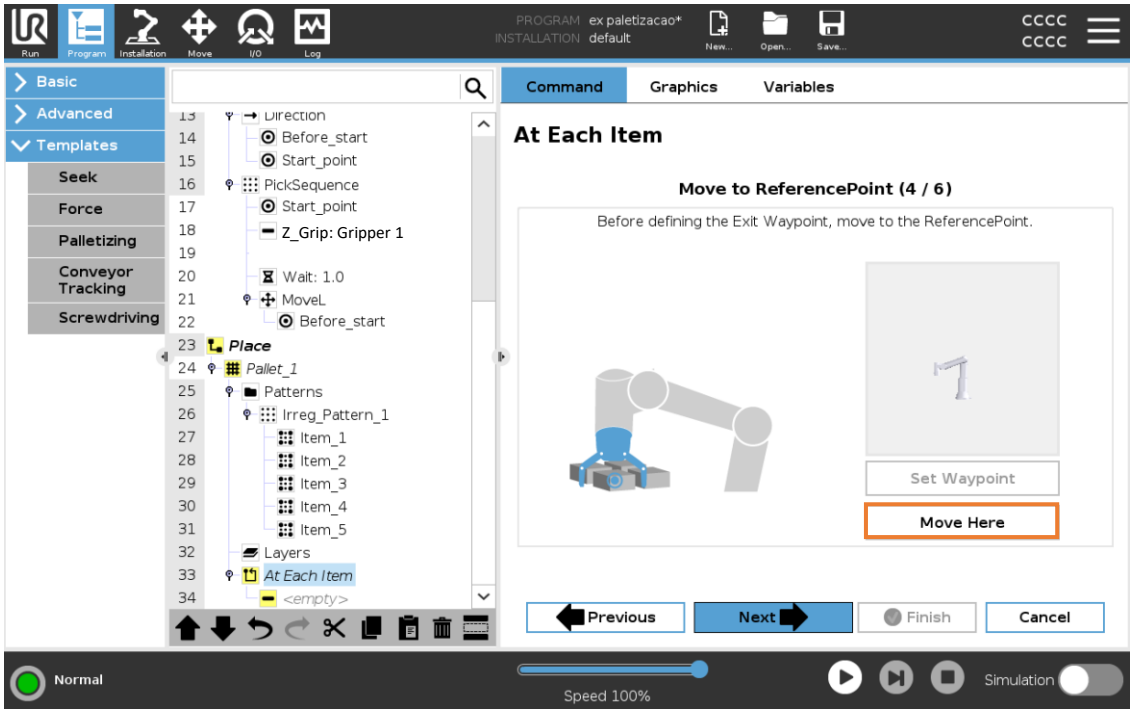


Figura 72 - Quarto passo da sequência

- Passo 5: Definir o ponto de saída. Seguir a mesma lógica do passo 3 (Figura 73);
- Passo 6: Clicar em *Finish* (Figura 74).

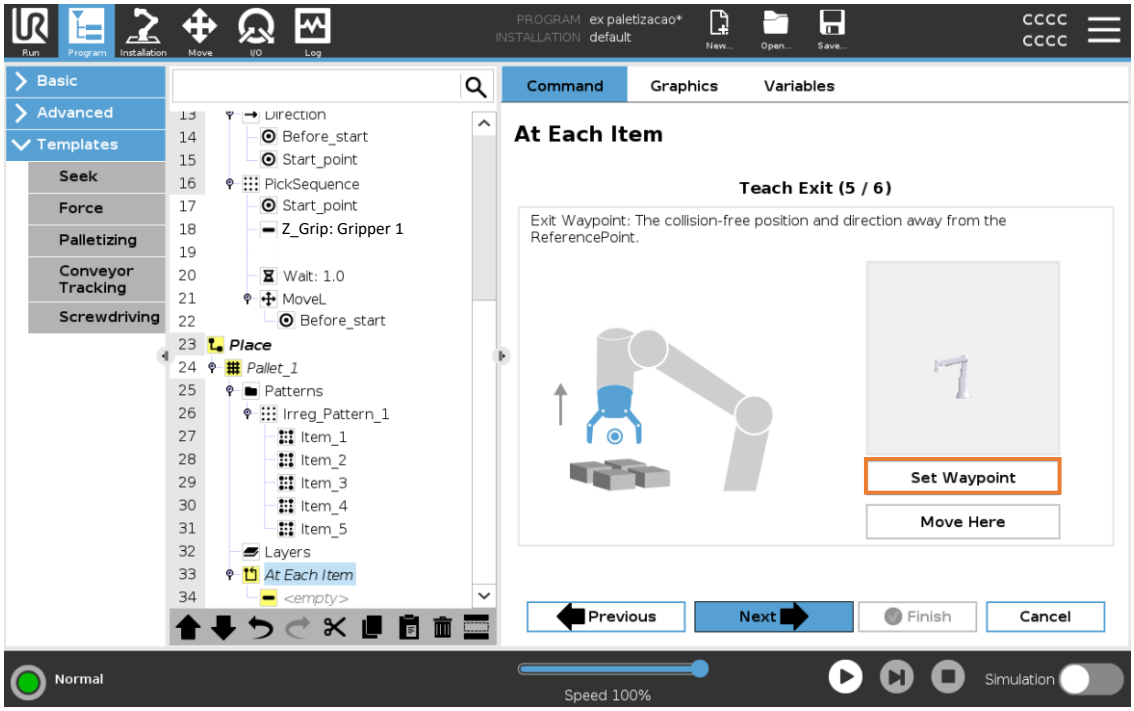


Figura 73 - Quinto passo da sequência

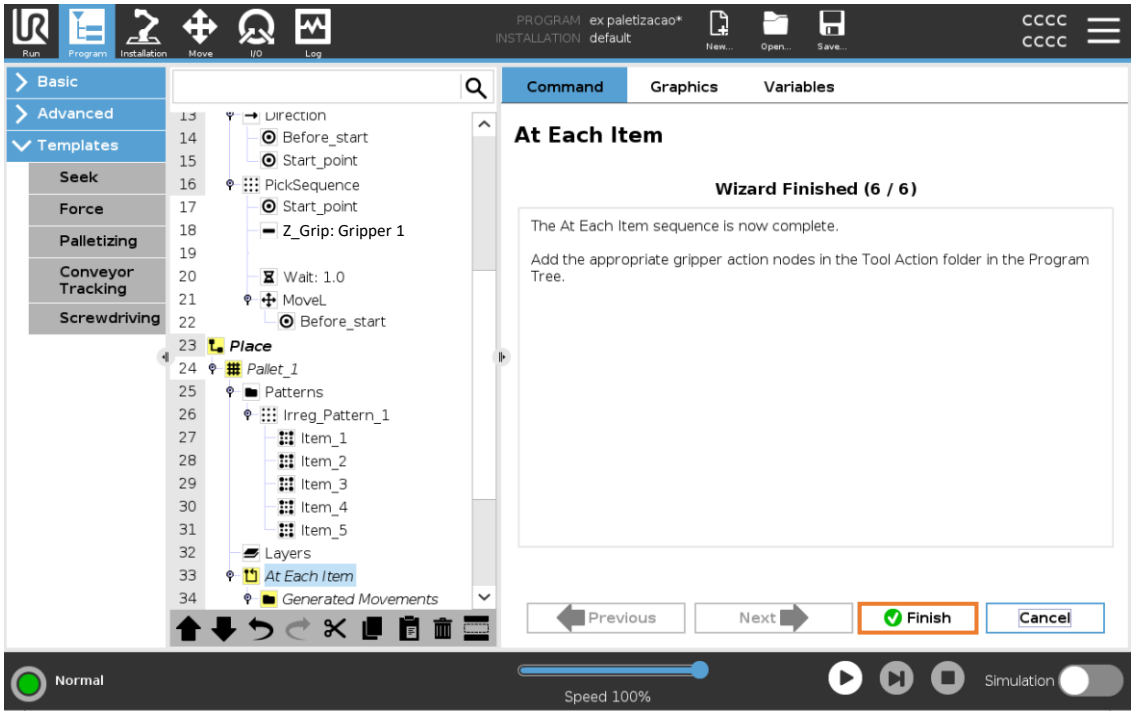


Figura 74 - Sexto passo da sequência

Para completar a secção de movimentos deve inserir a ação que a ferramenta (*gripper*) vai realizar. Quando se atinge a posição de deposição a peça é largada, ou seja, o *gripper* abre as garras (Figura 75).

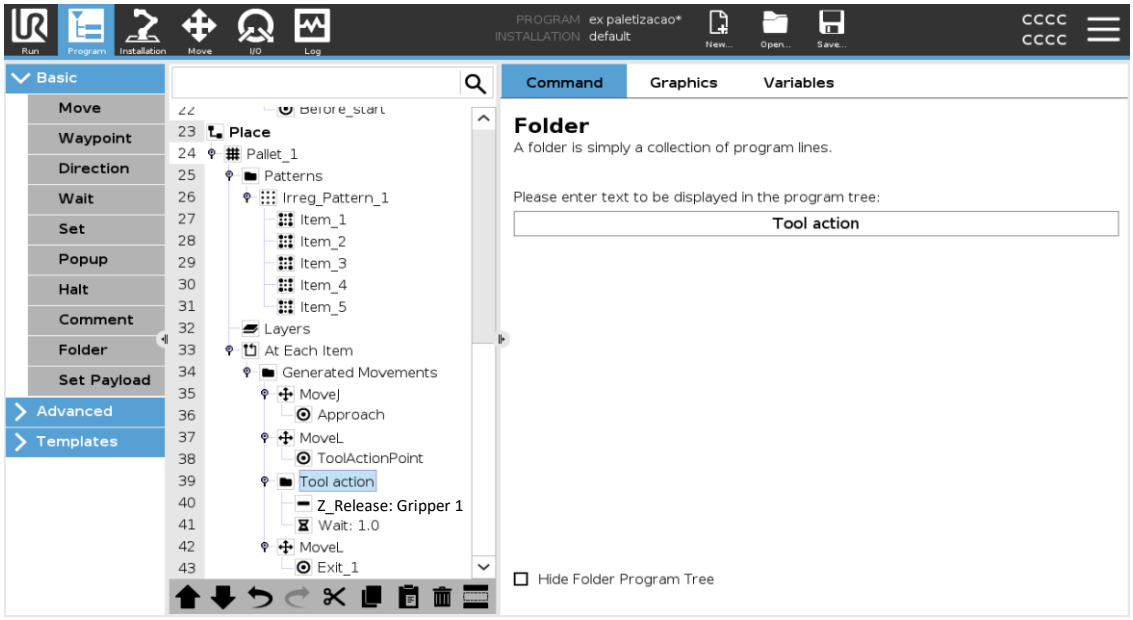


Figura 75 - Parametrização da secção *Tool Action*

Passo 6 – Estruturação do programa principal

Para a estruturação do programa principal é necessário definir as seguintes condições:

- Garantir que o robô começa e termina o programa na posição *Before_start*;
- Inserir uma instrução *If* para *count < 5* e uma instrução *ElseIf* para *count = 5*;
- Na instrução *If* deve-se:
  - Ativar o *LED* verde (DO0);
  - Mover o robô em movimento linear para a posição inicial (*Start_point*);
  - Inserir o subprograma *Pick* (conforme Figura 76) e *Place* (Figura 77).

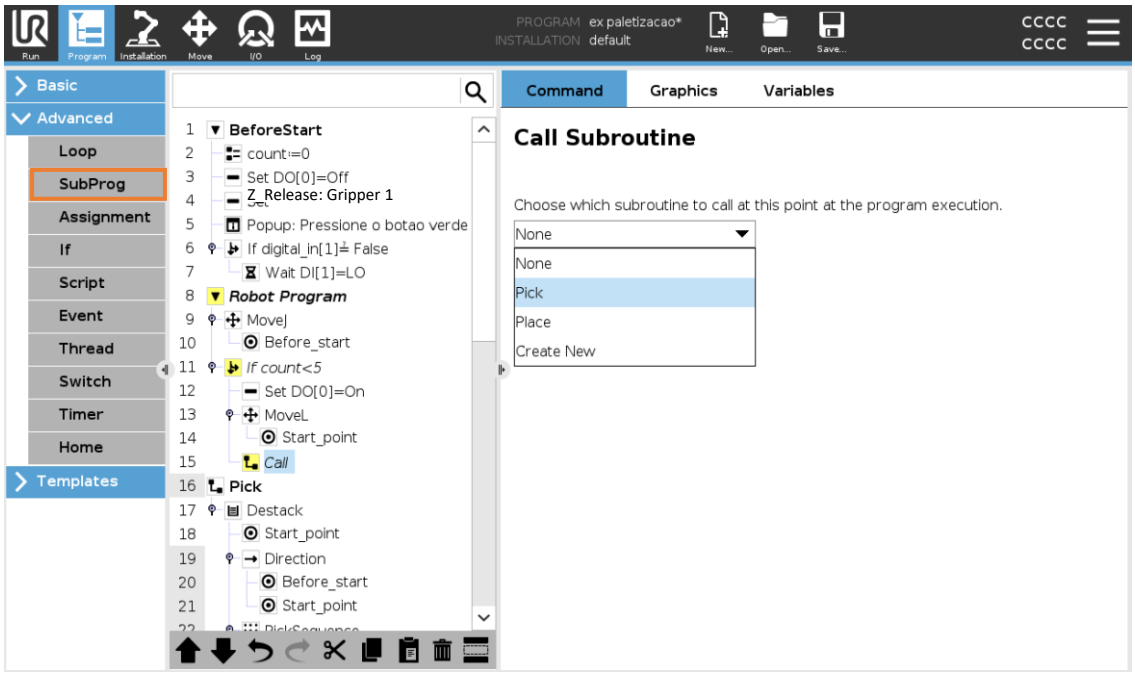


Figura 76 - Inserção do subprograma *Pick* no programa principal

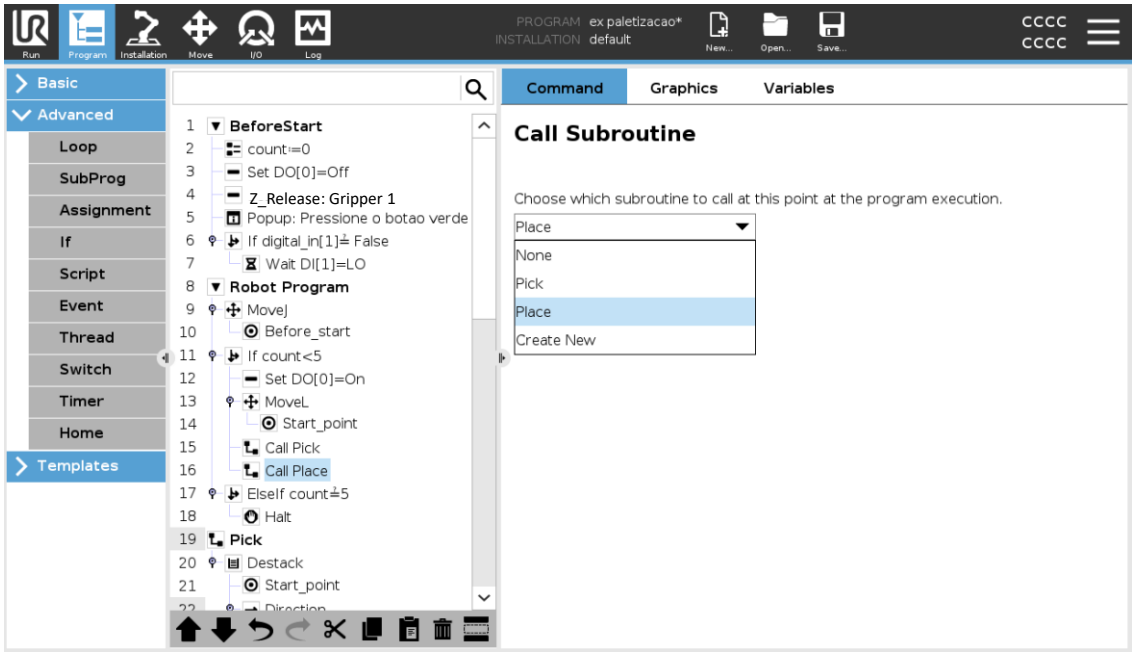


Figura 77 - Inserção do subprograma *Place* no programa principal

- Na instrução *ElseIf* deve-se:
  - Terminar o programa com a instrução *Halt* (Figura 78).

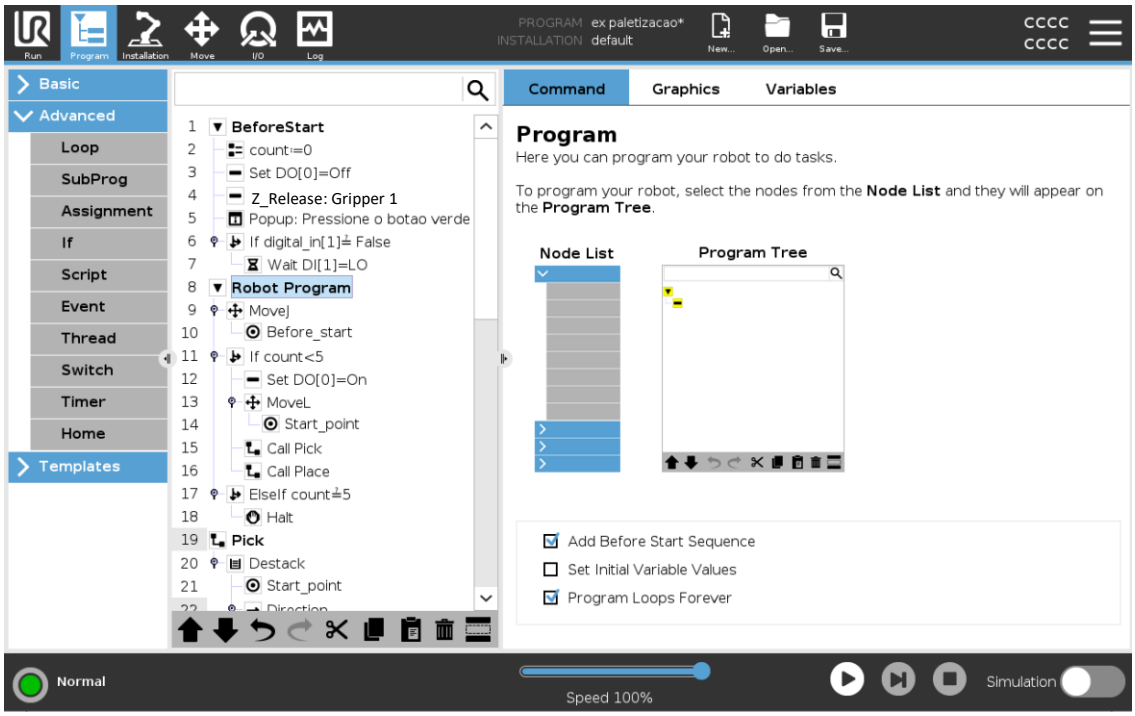


Figura 78 - Estruturação final do programa principal

Passo 4 – Teste e Validação

Testar o programa.

5. Discussão

O conjunto de exercícios apresentados neste guião deve permitir aos alunos adquirir e desenvolver competências fundamentais na programação de robôs colaborativos, utilizando o robô UR3e e a interface gráfica *PolyScope*. A execução progressiva das tarefas conduzirá à consolidação de conhecimentos relacionados com movimentos, lógica condicional, integração de *I/O Digital* e manipulação de objetos. Deve ainda potenciar a autonomia do aluno na criação, teste e validação de soluções robóticas eficazes, seguras e adaptáveis, que lhe permita recriar abordagens diferentes aos exercícios propostos ou mesmo desenvolver novos exercícios.

Apêndice F

Inquérito.

Implementação de exercícios com robô colaborativo UR3e

Questionário de Avaliação em robótica colaborativa

1. **Idade:** _____
2. **Qual é o seu nível de experiência com robótica colaborativa antes dos exercícios?**
   - ○ Nenhum
   - ○ Básico
   - ○ Intermédio
   - ○ Avançado

Organização da atividade

3. **Considera que os objetivos dos exercícios práticos foram apresentados com clareza?**
   - ○ Sim
   - ○ Não
   - ○ Parcialmente
4. **Em que medida considera que o material e as instruções fornecidas foram suficientes para compreender e concluir os exercícios?**
   - ○ Nada suficientes
   - ○ Pouco suficientes
   - ○ Suficientes
   - ○ Muito suficientes
   - ○ Totalmente suficientes
5. **Avalie o nível de clareza das explicações fornecidas.**
   - ○ Nada claras
   - ○ Pouco claras
   - ○ Claras
   - ○ Muito claras
   - ○ Totalmente claras

Conteúdo técnico

6. **Assinale os exercícios que resolveu.**
   - ○ Exercício 1
   - ○ Exercício 2
   - ○ Exercício 3
   - ○ Exercício 4
   - ○ Exercício 5
   - ○ Exercício 6
7. **Como avalia a dificuldade dos exercícios?**

Exercício 1
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

Exercício 2
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

Exercício 3
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

Exercício 4
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

Exercício 5
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

Exercício 6
- ○ Muito difícil
- ○ Difícil
- ○ Moderada
- ○ Fácil
- ○ Muito fácil

8. **Considera que o *PolyScope* foi fácil de utilizar?**
   - ○ Sim
   - ○ Não
   - ○ Neutro

9. **Existiram dificuldades técnicas com o UR3e?**
    - ○ Sim, não foi possível resolver os exercícios
    - ○ Sim, mas foram resolvidas rapidamente
    - ○ Não
10. **Houve algum aspeto dos exercícios que considerou especialmente difícil ou confuso? Se sim, qual?**

_____
_____
_____
_____
_____
_____

Desenvolvimento pessoal

11. **Como avalia a utilidade dos exercícios práticos para a sua aprendizagem em robótica colaborativa?**
    - ○ Nada úteis
    - ○ Pouco úteis
    - ○ Neutro
    - ○ Úteis
    - ○ Muito úteis
12. **Em que medida considera que os exercícios foram desafiantes para si?**
    - ○ Nada desafiantes
    - ○ Pouco desafiantes
    - ○ Desafiantes
    - ○ Muito desafiantes
13. **Em que medida considera que os exercícios práticos contribuíram para o seu entendimento em programação de robôs colaborativos (via *PolyScope*)?**
    - ○ Não contribuíram nada
    - ○ Contribuíram pouco
    - ○ Contribuíram muito

14. **Avalie o quanto os exercícios práticos melhoraram a sua confiança na área da robótica colaborativa?**
    - ○ Diminuíram muito
    - ○ Diminuíram um pouco
    - ○ Não teve impacto
    - ○ Aumentaram
    - ○ Aumentaram muito
15. **Avalie o quanto se sente capaz de programar um sistema simples após a atividade.**
    - ○ Nada capaz
    - ○ Pouco capaz
    - ○ Capaz
    - ○ Muito capaz
    - ○ Totalmente capaz

Ergonomia e envolvimento emocional

16. **Como classificaria a ergonomia e o conforto da bancada?**
    - ○ Muito pouco ergonómica e confortável
    - ○ Pouco ergonómica e confortável
    - ○ Ergonómica e confortável
    - ○ Muito ergonómica e confortável
17. **Que parte dos exercícios práticos lhe despertou mais interesse ou lhe pareceu mais útil?**

_____
_____
_____
_____
_____
_____

18. **Gostaria de sugerir melhorias para os próximos exercícios práticos? Se sim, quais?**

_____
_____
_____
_____
_____
_____

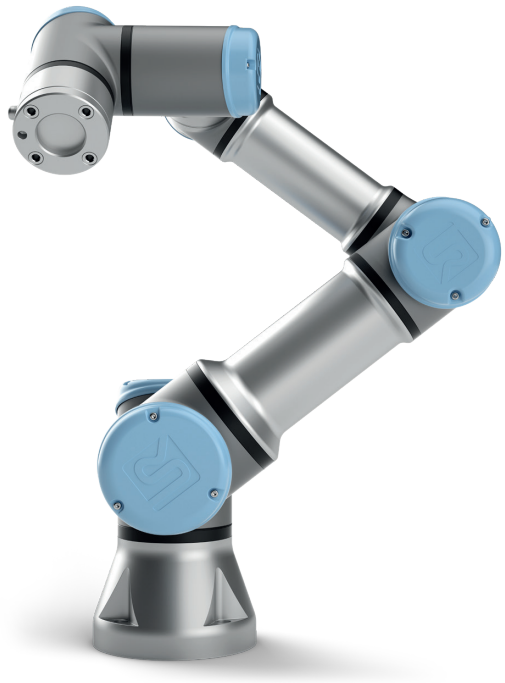
19. **De modo geral, quão interessante considera a sua participação nesta atividade?**
    - ○ Nada interessante
    - ○ Pouco interessante
    - ○ Neutro
    - ○ Interessante
    - ○ Muito interessante
20. **Gostaria de realizar outras atividades práticas sobre robótica colaborativa no futuro?**
    - ○ Sim
    - ○ Não
    - ○ Talvez
21. **Gostaria de acrescentar algum comentário ou feedback sobre a atividade?**

_____
_____
_____
_____
_____
_____

Anexo A

Ficha técnica do robô colaborativo UR3e.

UR3e technical details



Performance

Power consumption	Approx. 100 W using a typical program
Safety System	All 17 advanced adjustable safety functions incl. elbow monitoring certified to Cat.3, PL d. Remote Control according to ISO 10218
Certifications by TUV Nord	EN ISO 13849-1, Cat.3, PL d, and full EN ISO 10218-1
F/T Sensor - Force, x-y-z	
Range	30 N
Resolution	1.0 N
Accuracy	3.5 N
F/T Sensor - Torque, x-y-z	
Range	10 Nm
Resolution	0.02 Nm
Accuracy	0.10 Nm

Specification

Payload	3 kg / 6.6 lbs
Reach	500 mm / 19.7 in
Degrees of freedom	6 rotating joints DOF
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting

Movement

Pose Repeatability	+/- 0.03 mm, with payload, per ISO 9283	
Axis movement robot arm	Working range	Maximum speed
Base	± 360°	± 180°/s
Shoulder	± 360°	± 180°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 360°/s
Wrist 2	± 360°	± 360°/s
Wrist 3	Infinite	± 360°/s
Typical TCP speed		1 m/s / 39.4 in/s

Features

IP classification	IP54	
ISO Class Cleanroom	5	
Noise	Less than 60 dB(A)	
Robot mounting	Any Orientation	
I/O ports	Digital in Digital out Analog in Tool communication	2 2 2 RS-485
I/O power supply in tool	12V/24V 600mA continuous, 2A peak	
Ambient temperature range	0-50°C*	
Humidity	90%RH (non-condensing)	

Physical

Footprint	Ø 128 mm
Materials	Aluminium, Plastic, Steel
Tool (end-effector) connector type	M8 | M8 8-pin
Cable length robot arm	6 m / 236 in
Weight including cable	11.2 kg / 24.7 lbs

* The robot can work in a temperature range of 0-50°C. At high continuous joint speeds the maximum allowed ambient temperature is reduced.

Control box

Features

IP classification	IP44	
ISO Class Cleanroom	6	
Ambient temperature range	0-50°	
I/O ports	Digital in Digital out Analog in Analog out 500 Hz control, 4 separated high speed quadrature digital inputs	16 16 2 2
I/O power supply	24V 2A	
Communication	Control frequency: 500 Hz ModbusTCP: 500 Hz signal frequency ProfiNet and EthernetIP: 500 Hz signal frequency USB ports: 1 USB 2.0, 1 USB 3.0	
Power source	100-240VAC, 47-440Hz	
Humidity	90%RH (non-condensing)	

Physical

Control box size (WxHxD)	475 mm x 423 mm x 268 mm 18.7 in x 16.7 in x 10.6 in
Weight	13 kg / 28.7 lbs
Materials	Steel

Teach pendant

Features

IP classification	IP54
Humidity	90%RH (non-condensing)
Display resolution	1280 x 800 pixels

Physical

Materials	Plastic
Weight including 1 m of TP cable	1.6 kg / 3.5 lbs
Cable length	4.5 m / 177.17 in

UNIVERSAL ROBOTS

universal-robots.com

Anexo B

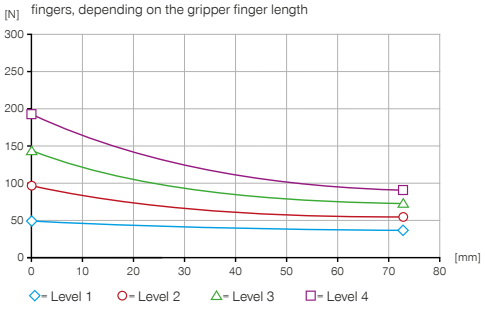
Ficha técnica do *gripper* HCR-03-118505 Zimmer.

2-JAW PARALLEL GRIPPERS
HRC-03-118505

▶ PRODUCT SPECIFICATIONS

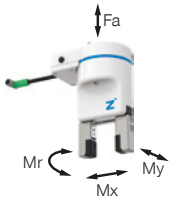
▶ Gripping force diagram

Shows the arithmetic total of the individual forces that occur on the gripper fingers, depending on the gripper finger length



▶ Forces and moments

Displays static forces and moments that can also have an effect, besides the gripping force.



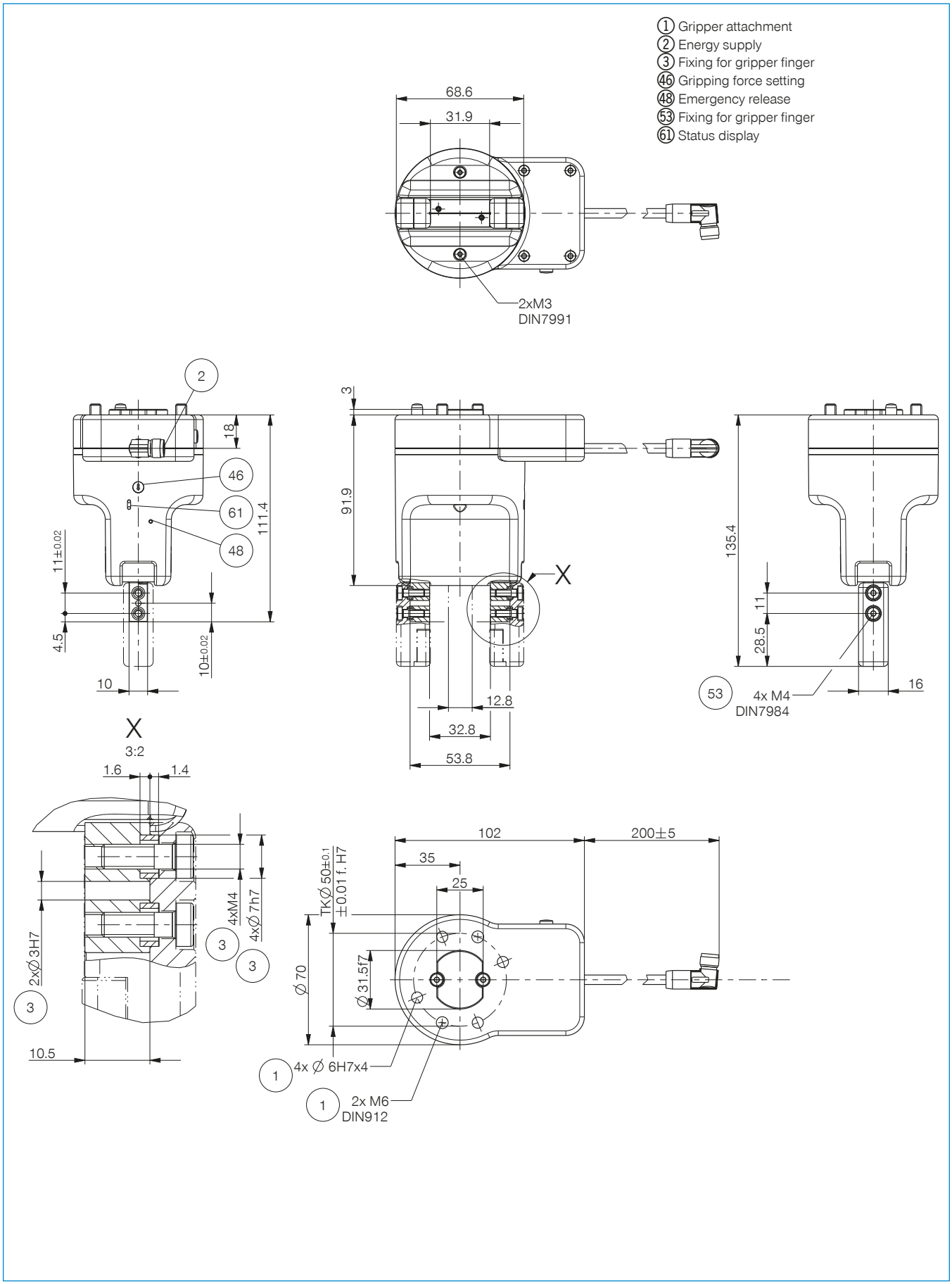
Mr [Nm]	7
Mx [Nm]	7
My [Nm]	5.5
Fa [N]	200

▶ TECHNICAL DATA

Order no.	HRC-03-118505
Suitable for robot type	Universal Robots e-Series / HANWHA HCR with M8 Tool IO
Multifunction button	freely programmable / Freedrive
MRK design according to ISO/TS 15066	Yes
HRC form	collaborative
Cable management	internal
Type of drive	electrical
Control	I/O
Position sensing	integrated
Position sensing analog 0 to 10 V	Yes
Switching output	NPN
Stroke per jaw [mm]	10
Self locking mechanism	mechanical
Gripping force in closing (adjustable) max. [N]	190
Gripping force in opening (adjustable) max. [N]	190
Gripping force in accordance with ISO/TS 15066 [N]*	<140
Closing time [s]	0.19
Opening time [s]	0.19
Control time [s]	0.03
Dead weight of mounted gripper finger max. [kg]	0.1
Length of the gripper fingers max. [mm]	80
Repetition accuracy +/- [mm]	0.05
Operating temperature [°C]	5 ... +50
Voltage [V]	24
Current consumption max. [A]	1
Minimum positioning path per jaw [mm]	0.5
Protection to IEC 60529	IP40
Weight [kg]	0.68

*Value based on the parameters described in the ISO/TS 15066, determined with a force measuring device certified by the DGUV (German Social Accident Insurance)

▶ TECHNICAL DRAWINGS



Glossário

Latência

Período decorrido entre o momento em que um dado comando é transmitido e recebido.