UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Weslley da Silva Alves

MODELAGEM E IMPLEMENTAÇÃO DE UMA PLANTA DIDÁTICA DE MANUFATURA ROBÓTICA INSPIRADA EM PLANEJAMENTO AUTOMÁTICO

Goiânia
2023

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[X] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Weslley da Silva Alves

3. Título do trabalho

Modelagem e implementação de uma planta didática de manufatura robótica inspirada em planejamento automático

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Joao Paulo Da Silva Fonseca**, **Professor do Magistério Superior**, em 26/09/2023, às 12:58, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Weslley Da Silva Alves**, **Discente**, em 26/09/2023, às 13:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4073846** e o código CRC **CB52BE0F**.

Weslley da Silva Alves

MODELAGEM E IMPLEMENTAÇÃO DE UMA PLANTA DIDÁTICA DE MANUFATURA ROBÓTICA INSPIRADA EM PLANEJAMENTO AUTOMÁTICO

Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Engenharia Mecânica da Universidade Federal de Goiás, como requisito parcial para obtenção do Título de Mestre em Engenharia Mecânica. Área de Concentração: Ciências Mecânicas

Orientador: Prof. Dr. João Paulo da Silva Fonseca

Universidade Federal de Goiás
Escola de Engenharia Elétrica, Mecânica e de Computação

Goiânia
2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Alves, Weslley da Silva
Modelagem e implementação de uma planta didática de manufatura robótica inspirada em planejamento automático [manuscrito] / Weslley da Silva Alves. - 2023.
CVII, 107 f.

Orientador: Prof. Dr. João Paulo da Silva Fonseca.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-graduação em Engenharia Mecânica, Goiânia, 2023.
Bibliografia. Apêndice.
Inclui gráfico, lista de figuras, lista de tabelas.

1. Automação. 2. Manufatura. 3. Robótica. 4. Planejamento automático. 5. Bancada didática. I. Fonseca, João Paulo da Silva, orient. II. Título.

CDU 621

UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **03** da sessão de Defesa de Dissertação de **Weslley da Silva Alves**, que confere o título de Mestre em **Engenharia Mecânica**, na área de concentração em **Ciências Mecânicas**.

Aos **trinta dias do mês de agosto do ano de dois mil e vinte e três**, a partir das **14h00min**., realizou-se a sessão pública de Defesa de Dissertação intitulada **"Modelagem e implementação de uma planta didática de manufatura robótica inspirada em planejamento automático"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **João Paulo da Silva Fonseca - (EMC)** com a participação dos demais membros da Banca Examinadora: Professor Doutor **José Jean-Paul Zanlucchi de Souza Tavares - (FEMEC-UFU)**, membro titular externo e Professor Doutor **Anderson da Silva Soares - (INF-UFG)**, membro titular externo, **cujas participações ocorreram através de videoconferência pelo link:** meet.google.com/dys-xrre-sju. Durante a arguição os membros da banca **não fizeram** sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor João Paulo da Silva Fonseca, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos **trinta dias do mês de agosto do ano de dois mil e vinte e três** .

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **José Jean Paul Zanlucchi de Souza Tavares**, **Usuário Externo**, em 30/08/2023, às 16:14, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Anderson Da Silva Soares**, **Professor do Magistério Superior**, em 30/08/2023, às 16:14, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Joao Paulo Da Silva Fonseca**, **Professor do Magistério Superior**, em 30/08/2023, às 17:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3997581** e o código CRC **E8F5BBF7**.

Referência: Processo nº 23070.039916/2023-26 SEI nº 3997581

Weslley da Silva Alves

MODELAGEM E IMPLEMENTAÇÃO DE UMA PLANTA DIDÁTICA DE MANUFATURA ROBÓTICA INSPIRADA EM PLANEJAMENTO AUTOMÁTICO

Dissertação apresentada ao Programa de Pós-Graduação *Stricto sensu* em Engenharia Mecânica da Universidade Federal de Goiás, como requisito parcial para obtenção do Título de Mestre em Engenharia Mecânica.
Área de Concentração: Ciências Mecânicas.
Orientador: João Paulo da Silva Fonseca

Dissertação defendida e aprovada em 30 de agosto de 2023, pela seguinte Banca Examinadora:

Prof. Dr. João Paulo da Silva Fonseca
Orientador

Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares
Convidado 1

Prof. Dr. Anderson da Silva Soares
Convidado 2

Goiânia
2023

Este trabalho é dedicado a Deus, causa primordial de todas as coisas.

AGRADECIMENTOS

Agradeço a todos os professores da UFG que contribuíram para meu crescimento e qualificação profissional. Em especial, ao professor e amigo João Paulo pelo incentivo, motivação e orientação nesta caminhada acadêmica.

Os maiores e mais sinceros agradecimentos aos meus pais, Vandir e Natalício, meu alicerce, responsáveis pela minha educação e pela minha formação. À toda minha família pelo apoio, motivação e confiança durante todos esses anos. E em especial, à minha querida esposa Priscila, e meus filhos Samuel e Davi pela paciência e companheirismo e que sempre esteve ao meu lado.

Agradeço à Faculdade Senai Ítalo Bologna, pelo apoio nesta pesquisa.

"O futuro da manufatura está na combinação de automação inteligente, planejamento automático e robótica avançada para criar fábricas mais eficientes, flexíveis e sustentáveis.",

Juergen Maier, CEO da Siemens UK,

How robotics, automation and planning can boost manufacturing efficiency.

RESUMO

Os atuais sistemas de manufatura vêm sofrendo grandes avanços tecnológicos, sobretudo no ramo de planejamento e programação da produção em automação. A quarta revolução industrial, ou indústria 4.0, já começou a possibilitar inovações tecnológicas em vários domínios, especialmente na indústria de manufatura e serviços. Observa-se o surgimento de um volume crescente de pesquisas sobre técnicas de inteligência artificial envolvendo fabricantes que buscam eficiência e rapidez. Para alcançar um comportamento "inteligente", é inevitável implementar técnicas de resolução automática de problemas, a fim de propiciar algumas tomadas de decisões pelos próprios equipamentos e dispositivos. O planejamento automático, também conhecido em inglês como "automated planning" ou, ainda, "AI Planning", define um tipo específico de problema de transição de estado em que o objetivo é encontrar uma sequência admissível de ações para trazer o sistema de um determinado estado inicial para um estado final desejado. O uso do planejamento automático aplicado a processos de manufatura ganha mais destaque no meio acadêmico devido à possibilidade de ampliar o uso de técnicas em aplicações reais na indústria moderna. Este trabalho tem como objetivo principal analisar os dados da integração de soluções de planejamento automático em sistemas reais e descrever um exemplo prático baseado em uma planta didática que simula uma linha de produção em uma fábrica composta por células individuais. Para a realização do projeto, foi utilizado um controlador lógico programável como modelo de sistemas reais utilizado na automação industrial, e as etapas de montagem e descrição dos processos de cada estação são apresentadas. A linguagem utilizada para a modelagem do domínio de planejamento foi a de definição de domínios e problemas de planejamento "Planning Domain Definition Language" (PDDL), por meio da plataforma online "PDDL Editor", a qual faz parte de um conjunto de ferramentas online para AI Planning, denominado "Planning.Domains". A modelagem foi dispersa em cinco estações, de modo a facilitar a solução do problema proposto para cada estação, gerando assim cinco planos solução, um para cada estação. Os resultados alcançados do método de modelagem do domínio de manufatura utilizando uma ferramenta de planejamento automático em plataforma online, foram a geração de planos-solução para os cenários propostos e a implementação prática na bancada didática utilizando a linguagem de programação "Sequential Funcion Chart" (SFC). A abordagem proposta foi comparada com a abordagem clássica, que se baseia na lógica de programação centrada no CLP e no uso da linguagem de programação Ladder. Essa comparação destacou as vantagens do planejamento automático quando aplicado ao contexto da Indústria 4.0.

Palavras-chaves: Automação, manufatura, robótica, planejamento automático, bancada didática;

ABSTRACT

Current manufacturing systems have been undergoing significant technological advancements, particularly in the field of production planning and automation. The Fourth Industrial Revolution, or Industry 4.0, has already begun to enable technological innovations across various domains, especially in the manufacturing and services industries. There has been a noticeable surge in research on artificial intelligence techniques involving manufacturers seeking efficiency and speed. To achieve 'intelligent' behavior, it is inevitable to implement techniques for automated problem-solving, enabling some decision-making by the equipment and devices themselves. Automated planning, also known as 'automated planning' or 'AI Planning' in English, defines a specific type of state transition problem in which the goal is to find an admissible sequence of actions to bring the system from a certain initial state to a desired final state. The use of automated planning applied to manufacturing processes is gaining prominence in the academic sphere due to the potential to expand the use of techniques in real-world applications in modern industry. This work aims to primarily analyze data from the integration of automated planning solutions into real systems and describe a practical example based on a didactic plant that simulates a production line in a factory composed of individual cells. For the project's implementation, a programmable logic controller (PLC) was used as a model of real systems used in industrial automation, and the assembly and description steps of each station are presented. The language used for modeling the planning domain was the 'Planning Domain Definition Language' (PDDL), through the online platform 'PDDL Editor,' which is part of a set of online tools for AI Planning called 'Planning.Domains.' The modeling was distributed across five stations to facilitate solving the proposed problem for each station, thus generating five solution plans, one for each station. The results achieved from the domain modeling method in manufacturing using an online automated planning tool included the generation of solution plans for the proposed scenarios and practical implementation on the didactic bench using the 'Sequential Function Chart' (SFC) programming language. The proposed approach was compared with the classical approach, which concentrated programming logic on the PLC and used the Ladder programming language, highlighting the potential of using automated planning for Industry 4.0-related topics.

Keywords: Automation, manufacturing, robotics, AI planning, didactic testbench.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

1. INTRODUÇÃO

A Indústria 4.0 está transformando o cenário industrial de forma sem precedentes, com a integração de múltiplas tecnologias, incluindo “Cyber-Physical-System” (CPS). Essa nova era da manufatura é caracterizada pela digitalização e interconexão de produtos, serviços e sistemas de manufatura inteligentes. Um CPS, ou Sistema Ciber-físico em português, é uma entidade que integra de maneira profunda e intricada componentes computacionais (o "cyber") e físicos (o "physical"), com a capacidade de monitorar, controlar e interagir com o mundo real. Como definido por Lee *et al.* (2018), "CPSs são sistemas que incorporam redes de sensores, atuadores e elementos de processamento de dados interconectados, distribuídos e frequentemente hierárquicos, que monitoram e controlam dinamicamente sistemas físicos em tempo real." Esses sistemas desempenham um papel fundamental em uma variedade de domínios, desde manufatura avançada e cidades inteligentes até saúde e mobilidade, capacitando a tomada de decisões mais informadas e a automação inteligente para melhorar a eficiência e a qualidade de vida.

O uso do termo Indústria 4.0 é frequentemente associado a diversas tecnologias digitais, como veículos autônomos, impressão 3D, robôs inteligentes, nano materiais avançados como o grafeno, Internet das Coisas (IoT), “blockchain” e biologia sintética. A integração de CPSs no desenvolvimento de um domínio de planejamento é crucial para essa transformação, permitindo que a indústria possa criar um conjunto de planos que considerem as incertezas e variações do mundo físico. Nesse sentido, o uso de um planejador automático é uma ferramenta essencial para auxiliar na execução desses planos e garantir a eficiência e adaptabilidade dos sistemas ciber-físicos. (Yoo *et al*., 2021).

O domínio da automação de máquinas e instalações enfrenta uma demanda cada vez maior para garantir a adaptabilidade das instalações de fabricação no contexto da indústria 4.0. Abordagens de programação clássicas baseadas tipicamente em linguagens orientadas a sinais resultam em um esforço desproporcional para garantir a flexibilidade necessária. Cavalcanti e Nogueira (2017), destacam que a indústria 4.0 tem se mostrado substancialmente presente nas indústrias de manufatura, e seus processos podem ser associados às chamadas fábricas inteligentes, nas quais a tecnologia é utilizada para otimizar as atividades, buscando, portanto, garantir resultados precisos em comparação com empresas que não estão incluídas no modelo da quarta revolução industrial. Segundo Lu *et al*., (2017), fábricas inteligentes são ambientes de

produção altamente automatizados e conectados que incorporam tecnologias avançadas, como IoT, aprendizado de máquina e robótica, para otimizar processos de manufatura e melhorar a eficiência operacional

Os sistemas de manufatura se caracterizarão por uma utilização maciça de computadores, visando a automação das tarefas envolvidas no ciclo de projeto e manufatura de produtos. Nesse contexto, as indústrias têm feito grandes investimentos em sistemas CAD, CAM, máquinas CNC, Robôs, etc. O conceito de manufatura integrada por computador (CIM) surge da necessidade de que estes sistemas cooperem entre si a fim de que a indústria opere em um nível máximo de eficiência (AMICE, 1993). Para isto, deve ser possível transferir entre sistemas dados que tenham um significado comum entre eles de tal forma que possam ser utilizados pelo sistema que os recebe.

Wally *et al*. (2019) ressaltam que os sistemas de manufatura do futuro devem ser cada vez mais flexíveis, tanto em relação aos produtos que são fabricados, quanto aos próprios sistemas de produção. De acordo com os princípios da fabricação inteligente, os produtos e suas receitas não precisam ser conhecidos no momento do projeto, as variantes do produto podem ser editadas na hora da execução e o planejamento e programação da produção devem ser aplicados em tempo real, quando surgir uma nova ordem de produção.

Para a realização desses cenários de aplicação da indústria 4.0 em sistemas automatizados de manufatura, características como conectividade, orientação a serviços e autonomia desempenham um papel crucial. Em particular, para alcançar um comportamento inteligente, é inevitável implementar técnicas automatizadas de resolução de problemas, a fim de facilitar a tomada de decisões autônomas.

Dentre as metodologias possíveis de aplicar no processo de automatização dos sistemas de manufatura, está o planejamento automático que, embora tenha surgido há mais de 40 anos, ainda encontra certa resistência para ser utilizado em aplicações práticas (Fonseca, 2013). O planejamento automático (Ghallab *et al.*, 2016) é um subcampo da inteligência artificial (IA) dedicado a fornecer comportamento deliberativo orientado a objetivos para agentes físicos e virtuais, por exemplo, robôs ou controladores lógicos programáveis. Um planejador automático toma como entrada um domínio de planejamento, um estado inicial, uma lista de objetos e um objetivo e realiza um processo de busca, que retorna um plano (sequência de ações), que orienta o comportamento do agente, a fim de atingir o objetivo dado a partir do estado inicial.

O planejamento automático tem sido tradicionalmente uma das técnicas mais utilizadas em IA e tem sido aplicado com sucesso em aplicações do mundo real (Castillo *et al.,* 2008, Fdez-Olivares *et al.,* 2019). No entanto, para integrá-lo em sistemas de execução "online", ou seja, sistemas que utilizam cenários de tempo real que intercalam planejamento e atuação, existem várias questões que devem ser abordadas. Segundo Ferreira (2009), um exemplo dessas questões é que o planejamento geralmente é lento para aplicações em tempo real. Na maioria dos problemas práticos, o espaço de busca cresce exponencialmente com o tamanho do problema, portanto, apesar do uso de heurísticas, encontrar um plano adequado geralmente leva muito tempo.

Usualmente, é utilizada uma linguagem padrão para descrição do domínio de planejamento, a linguagem denominada *Planning Domain Definition Language* (PDDL) (Ghallab *et al*., 2016). A escolha pela linguagem PDDL ao invés de outras opções, como a PDDL 2, justifica-se pela compatibilidade e suporte oferecidos por diversas ferramentas e plataformas de planejamento automático existentes, o que facilita a replicação e validação dos resultados obtidos neste trabalho. Além disso, a PDDL é uma linguagem amplamente utilizada e reconhecida na comunidade de pesquisa em planejamento automático, o que garante a qualidade e relevância da modelagem realizada no estudo. A partir dessa linguagem, é possível especificar o domínio, formado pelos tipos de objetos, predicados, funções e ações, além da instância do problema a ser resolvido, constituída pela descrição dos estados inicial e final do sistema.

Com a introdução de modelos industriais trazidos dos conceitos da indústria 4.0 e o surgimento de aplicações de inteligência artificial no cotidiano, há uma reflexão de como essas tecnologias poderiam ser utilizadas na indústria. Um dos principais dispositivos utilizados no processo de automação são os controladores lógicos programáveis (CLPs), que sugiram com o objetivo de substituir os painéis de controle com relés, diminuindo assim o alto consumo de energia, a difícil manutenção e modificação de comandos. A grande vantagem dos CLPs é a possibilidade de reprogramação, permitindo substituir as modificações necessárias de hardware para modificações de software. Devido a essa característica, um mesmo CLP pode ser utilizado no controle de diversos sistemas automáticos, modificando sua programação de um sistema para outro (Stefani, 2012).

1.1 Objetivos

O objetivo geral deste trabalho é modelar e implementar um sistema de distribuição de produtos através de uma planta didática de manufatura robótica que não necessite de interferência humana e avaliar o uso da linguagem PDDL. Para alcançar este objetivo, a pesquisa irá explorar a utilização do planejamento automático como ferramenta para a manipulação do processo produtivo, buscando a criação de um domínio de manufatura, o qual pode adaptar-se a eventuais variações no ambiente de produção. A implementação de uma planta didática de manufatura robótica servirá como base para a avaliação da aplicabilidade do sistema proposto, permitindo a validação dos resultados obtidos e o estabelecimento de diretrizes para a aplicação de soluções similares em outras áreas da indústria.

Para atingir o objetivo geral que o trabalho propõe, destacam-se os seguintes objetivos específicos:

a) Levantar do estado da arte e revisão da literatura;

b) Desenvolver um projeto de automação para a bancada didática considerando o método clássico de programação de CLP com linguagem *Ladder*;

c) Implementar um método de modelagem de domínio de manufatura proposta por planejamento automático a partir dos modelos gerados na bancada didática;

d) Executar testes funcionais;

e) Avaliar os dois tipos de abordagens propostas;

f) Validar o funcionamento do sistema.

1.2 Justificativa

Para que uma empresa possa se manter competitiva no mercado, é necessário garantir uma elevada produtividade, o que resulta na modernização das plantas industriais, reposição de

equipamentos obsoletos e uso intensivo de sistemas automatizados. O princípio básico da automação industrial é melhorar as condições de trabalho e produtividade no seu ambiente de implementação. A partir da compreensão da importância de um sistema ciber-físico, como uma estrutura que integra a dimensão física e a dimensão virtual dos sistemas de manufatura, propõe-se a utilização do planejamento automático como um dos componentes essenciais desse sistema. Nesse contexto, o planejamento automático pode ser empregado como uma ferramenta capaz de aprimorar significativamente os processos de manufatura, permitindo que os componentes físicos e virtuais trabalhem em sincronia, tornando a produção mais eficiente e eficaz. Nesse sentido, a adoção do planejamento automático como parte integrante de um CPS é fundamental para garantir a competitividade das indústrias no atual cenário em constante evolução tecnológica.

Logo, este trabalho se destaca pela introdução de "inteligência" ao processo de manufatura, diferenciando-se da abordagem de Fonseca (2013). Essa distinção é evidente na utilização de uma bancada didática de manufatura robótica que simula uma linha de produção composta por células individuais. Além disso, a inovação se manifesta no método de modelagem do domínio de manufatura, que é proposto por meio da plataforma online "PDDL Editor". Os planos-solução obtidos também desempenham um papel fundamental, integrando-se à lógica de operação do sistema baseado em CLP.

Segundo Bazzo e Pereira (2000), a planta didática é importante como recurso de modelagem, pois representa um sistema físico real ou parte dele. Este recurso fornece maior proximidade com processos industriais, facilitando a detecção de erros e a otimização dos processos.

A utilização de CLPs justifica-se por diversos motivos. Em primeiro lugar, trata-se de um equipamento amplamente difundido na indústria, e frequentemente aplicado ao controle e monitoramento de processos produtivos. Dessa forma, seu uso garante maior fidelidade à realidade dos processos de manufatura industrial, possibilitando a aplicação prática do método proposto em outras aplicações semelhantes.

Além disso, é importante destacar que a substituição de CLPs por tecnologias mais modernas não ocorrerá imediatamente, dada a sua ampla utilização na indústria e a necessidade de um processo de transição gradual para a adoção de novas soluções. Nesse sentido, a utilização de CLPs na planta didática pode ser vista como uma forma de capacitar futuros profissionais para a utilização dessa tecnologia, garantindo que tecnologias antigas como o CLP possam ser integradas com as iniciativas da Indústria 4.0 sem que sejam descartados e substituídos por

novas tecnologias de imediato. Além disso essa integração é uma contribuição levando em consideração a redução de lixo industrial.

1.3 Estrutura da Dissertação

Este projeto traz uma proposta de integração de soluções de planejamento automático em sistemas reais com uso de controladores lógicos programáveis aplicado em um exemplo prático com uma planta didática de manufatura. Para tal, os capítulos deste trabalho estão organizados da seguinte forma.

No capítulo 2, apresenta-se toda fundamentação teórica dos temas envolvidos no projeto. Neste capítulo é apresentado o planejamento automático, incluindo os conceitos sobre domínios, problemas e planos. As linguagens de especificação e modelagem de domínios, tanto de planejamento automático, que é o caso do PDDL, quanto de sistemas gerais são descritas com a finalidade de especificar as necessidades de representação e de ferramentas que auxiliem no processo de aprendizado, descrição de requisitos do projeto, caracterização e modelagem de domínios que abranjam planejamento automático. Fala-se sobre controladores lógicos programáveis. Além desses conceitos, é apresentado a revisão da literatura.

É apresentado no capítulo 3, o estudo de caso utilizado para a aplicação do sistema proposto. É relatado o desenvolvimento do projeto, com detalhamento do processo desde o processo de montagem da bancada didática até a configuração do CLP e sua programação. Após a programação e configuração é demonstrado o resultado do processo implementado.

No capítulo 4 são apresentadas as fases da modelagem e análise de domínio de planejamento automático proposto neste trabalho, bem como a linguagem PDDL pode ser aplicada dentro do planejamento de uma planta didática de manufatura. Neste capítulo também são descritos os requisitos e as fases para implementação do projeto. O objetivo é mostrar ao leitor como modelar um domínio utilizando a PDDL.

Já no capítulo 5, os resultados e discussão final deste trabalho são discorridos, enquanto com capítulo 6 apresenta-se a conclusão do trabalho e os trabalhos futuros. No capítulo 7, apresentam-se as referências bibliográficas. A dissertação contém ainda, apêndices desenvolvidos.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentado o referencial teórico das principais áreas da engenharia associadas nesse projeto, principalmente os de planejamento automático. Em primeiro lugar, na seção 2.1, são abordados os conceitos da Industria 4.0 aplicados a sistemas de manufatura. Já na seção 2.2 traz as definições e características dos CLPs, desde o histórico, as linguagens de programação e suas aplicações. Na sequência, a seção 2.3 serão apresentados os conceitos do planejamento automático e sua importância dentro da IA, onde este trabalho está inserido. A seção 2.4 são abordadas as linguagens mais utilizadas para modelar domínios e problemas de planejamentos (STRIPS e PDDL). Por fim, na seção 2.5, é apresentado uma revisão abrangente da literatura relacionada a este campo, contextualizando ainda mais as contribuições deste projeto.

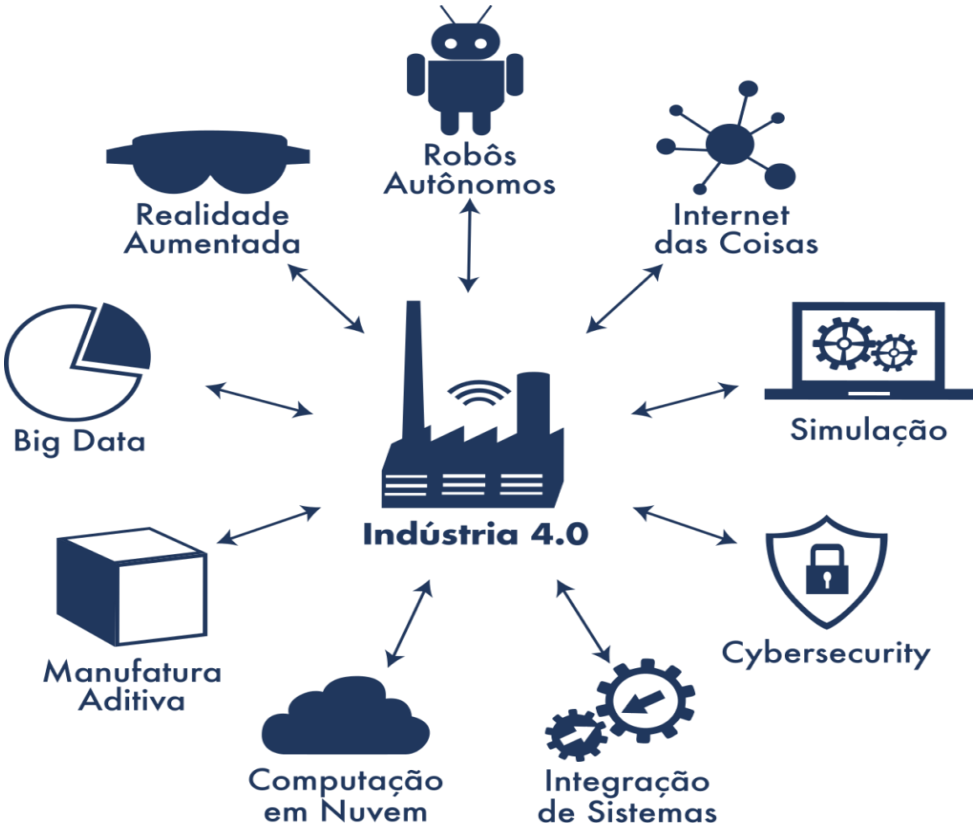
2.1 Indústria 4.0

O termo “Indústria 4.0” originou-se de um projeto iniciado pela estratégia de alta tecnologia do governo alemão para promover a informatização da manufatura. A Indústria 4.0 é considerada a próxima fase na digitalização do setor manufatureiro e é impulsionada por quatro disrupções: o aumento surpreendente de dados, poder computacional e conectividade, especialmente novas redes de longa distância e baixo consumo de energia; o surgimento de capacidades analíticas e da inteligência de negócios; novas formas de interação homem-máquina, como interfaces de toque e sistemas de realidade aumentada; melhorias na transferência de instruções digitais para o mundo físico, como robótica avançada e impressão 3D (Lee *et al*., 2013). Essas quatro tendências não são a razão para o “4.0”, no entanto, esta é a quarta grande revolução na manufatura moderna, após a revolução enxuta da década de 1970, o fenômeno da terceirização da década de 1990 e a automação que decolou na década de 2000 (Baur & Wee, 2015).

A maioria das tecnologias digitais mencionadas acima foi desenvolvida há anos e, embora algumas tecnologias ainda não estejam prontas para uso em larga escala, muitas estão agora em um ponto em que disponibilidade, confiabilidade e custos atraentes o suficiente para algumas aplicações industriais. De acordo com Kagermann *et al*. (2013), a Indústria 4.0 leva a

automação de manufatura a um novo nível ao introduzir tecnologias de produção em massa customizadas e flexíveis. Isso implica que as máquinas poderão operar sozinhas ou em colaboração com os seres humanos para produzir um tipo de manufatura focada no cliente, que é constantemente atualizada. Em vez disso, a máquina se torna uma entidade independente que pode coletar dados, analisá-los e processá-los. Isso se torna possível com a introdução de auto-otimização, auto-cognição e auto-personalização na indústria, e os fabricantes poderão se comunicar com computadores em vez de apenas operar máquinas.

Figura 1 . Pilares da Industria 4.0.



Fonte: Dias (2014)

A Fig. 1 mostra que a Indústria 4.0 é sustentada por diversos pilares que moldam a sua transformação digital. Os robôs autônomos representam a automatização avançada, capazes de realizar tarefas complexas sem intervenção humana. A Internet das Coisas (IoT) permite a interconexão de dispositivos, máquinas e sistemas, viabilizando a coleta e compartilhamento de dados em tempo real. A simulação facilita o teste e a otimização de processos de manufatura

virtualmente antes da implementação física. A cibersegurança é fundamental para proteger sistemas e dados contra ameaças digitais. A integração de sistemas promove a harmonização entre diferentes tecnologias e equipamentos. A computação em nuvem oferece escalabilidade e armazenamento de dados acessíveis de qualquer lugar. A manufatura aditiva (impressão 3D) revoluciona a produção com personalização e eficiência. O Big Data lida com a análise de grandes volumes de dados para insights estratégicos. A realidade aumentada aprimora a colaboração e o treinamento com informações digitais sobrepostas ao mundo real. Esses pilares juntos moldam a revolução da Indústria 4.0, impulsionando a eficiência e a inovação na manufatura.

A Indústria 4.0 está sendo cada vez mais explorada por acadêmicos, pesquisadores e profissionais da indústria. Há grandes expectativas, tanto nas comunidades de pesquisa quanto nas de negócios, de que tais tecnologias permeará as mais amplas cadeias produtivas e o setor de serviços (Wood *et al*., 2014). Um número considerável de estudos já foi publicado sobre o tema, propondo diversos cenários e benefícios da implantação da Indústria 4.0. Por exemplo, Ivanov *et al*. (2016), Kang *et al*. (2016), Lom *et al*. (2016), Thoben *et al*. (2017), Wollschlaeger *et al*. (2017) e Zhou *et al.* (2015) todos sugeriram que essas tecnologias permitem eficiência operacional, melhor controle das operações de dados e redução de desperdícios de energia de máquinas e processos. E por mais que sugiram, uma revolução industrial significa que novos paradigmas passam a vigorar e as soluções precisam ser desenvolvidas sobre esses. Sendo assim, a oportunidade de desenvolver soluções é mais ampla comparado a melhorias em tecnologias já sedimentadas. Essas sugestões não apresentam como aplicar as soluções, o que garante a trabalhos como esse a chance de estar na fronteira do conhecimento.

2.2 Controladores lógicos programáveis

O controlador lógico programável (CLP) originou-se no final da década de 1960 na indústria automotiva nos EUA e foi projetado para substituir os painéis de controle com relés. Antes, a lógica de controle para fabricação era composta principalmente de relés, temporizadores de came, sequenciadores de bateria e controladores dedicados de malha fechada, o que demandava um alto consumo de energia, a difícil manutenção e modificação de comandos.

Os primeiros CLPs tinham uma capacidade reduzida de processamento e suas aplicações se limitavam a máquinas e pequenos processos que necessitavam de operações repetitivas. A partir de 1970, com o surgimento da tecnologia dos microprocessadores, os CLPs passaram ter uma capacidade de processamento maior e alta flexibilidade de programação e expansão. Entre algumas características, cita-se: a capacidade de operar com números, realizar operações aritméticas com ponto decimal flutuante, manusear dados e se comunicar com computadores (Krakheche, 2007).

Segundo Mahato *et al*. (2015), os CLPs são amplamente utilizados no controle industrial por serem baratos, fáceis de instalar e muito flexíveis nas aplicações. Um CLP interage com o mundo externo através de suas entradas e saídas. Desde que a tecnologia para controle de movimento de acionamentos elétricos se tornou disponível, o uso de CLP com eletrônica de potência em aplicações de máquinas elétricas foi introduzido na automação de manufatura.

A automação no ambiente industrial requer dispositivos com hardwares robustos e um software confiável. Os CLPs cobrem a maioria desses requisitos e são amplamente utilizados na solução de tarefas de automação. As principais vantagens de sua aplicação são:

- Custo-benefício no controle de sistemas complexos;
- Flexibilidade - pode ser reprogramado de forma rápida e fácil para controlar outros sistemas;
- Os recursos do computador permitem um controle mais sofisticado;
- Ambientes de desenvolvimento integrados facilitam a programação e reduzem o tempo de inatividade;
- Confiabilidade no trabalho em condições industriais;
- Capacidades de comunicação para conexão com computadores e outros sistemas de controle;
- Oportunidades para construir sistemas de controle distribuído, redes de informação industrial e sistemas em nuvem.

Os CLPs são muito adequados para realizar operações repetitivas com flexibilização reduzida de escolhas, todavia casos não abarcados pela programação prévia não tem como ser atendidos. Nesse sentido a integração com Planejadores Automáticos amplia e muito a flexibilização dos CLPs. Com o avanço da tecnologia, principalmente na área da computação, surgiram novos paradigmas que têm impulsionado a chamada Quarta Revolução Industrial. Essa nova revolução se caracteriza pela integração de sistemas físicos, digitais e biológicos, bem como pela utilização de inteligência artificial e outras tecnologias avançadas. Portanto, pode-se dizer que sob o ponto de vista da descentralização, o CLP é uma tecnologia fadada a ser substituída pela nova revolução industrial. Se pensar em IoT, cada sensor e atuador comunicando-se e com processamento, o CLP deixa de ser necessário. Todavia há um parque fabril todo automatizado com CLP, forçando essa tecnologia a ser adaptada aos novos paradigmas da Quarta Revolução Industrial, permitindo o desenvolvimento de soluções inovadoras e cada vez mais conectadas.

2.2.1 Estrutura

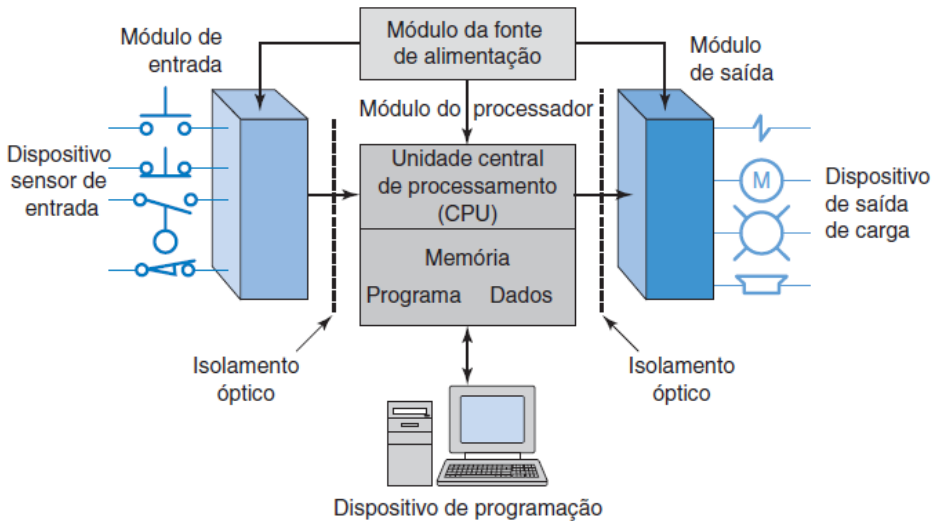
Um CLP é um sistema de controle baseado em microprocessador que usa uma memória programável para o armazenamento interno de instruções orientadas ao usuário para implementar funções específicas como aritmética, contagem, lógica, sequenciamento e temporização. O CLP pode ser caracterizado em função da sua estrutura de hardware. Para compreender melhor essa estrutura, é apresentado os principais componentes que formam um CLP, são eles:

- Módulos de Entradas;
- Módulos de Saídas;
- Unidade Central de Processamento – CPU;
- Memórias;
- Fonte de Tensão;
- Interface para Programação;

- Interface para Operação.

Dessa forma, com a junção desses componentes é possível montar uma estrutura e exemplificar o funcionamento de um CLP. Na Fig. 2 é mostrado todas as partes em digrama de blocos:

Figura 2. Diagrama esquemático representando as partes de um CLP.



Fonte: Krakheche (2007)

2.2.2 Programação

Linguagem de programação é um conjunto de símbolos e regras que formam um código capaz de permitir a comunicação entre o usuário e um determinado equipamento ou sistema, ou seja, é a forma de representação do programa de controle (Krakheche, 2007).

A IEC 61131–3 atualmente define cinco linguagens de programação para sistemas de controle programáveis: Diagrama de blocos de funções, diagrama *Ladder*, texto estruturado, (semelhante à linguagem de programação *Pascal*), lista de instruções (semelhante à linguagem *assembly*) e gráfico de funções sequenciais do inglês “Sequential Function Chart” (SFC) ou Grafcet. Essas técnicas enfatizam a organização lógica das operações.

Uma das linguagens mais utilizadas pelos programadores é a linguagem *Ladder*. A lógica *Ladder* para relé (RRL) é uma linguagem-padrão de programação usada com os CLPs, e

sua origem é baseada na lógica de relés. O programa com a linguagem da lógica *Ladder* representa graficamente os degraus de contatos, as bobinas e os blocos de instrução. A RRL foi projetada originalmente para facilitar o uso e o entendimento para seus usuários e tem sido modificada para acompanhar a crescente demanda de necessidades da indústria de controle. Esta é a linguagem é a mais popular e utilizada pelos fabricantes.

Krakheche (2007), destaca que linguagem *Ladder* possui três elementos básicos:

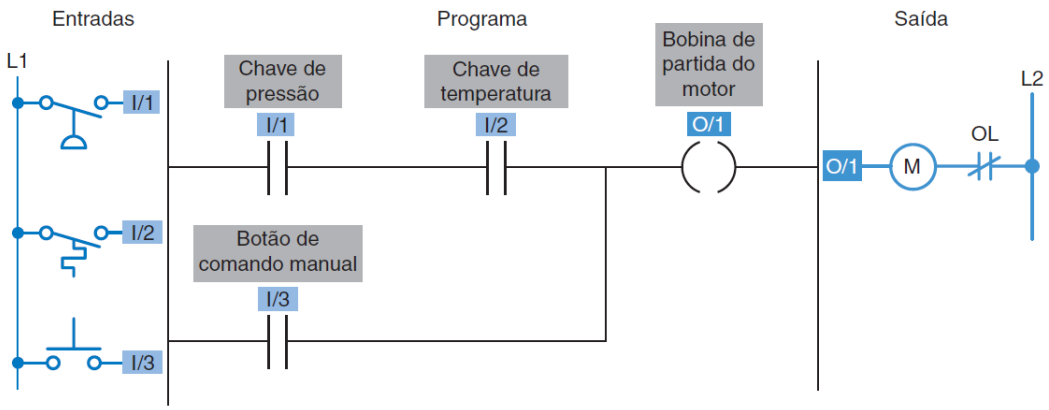
- Contato normalmente aberto (NA): representa o estado lógico de uma entrada digital. Quando a entrada está inativa (nível lógico 0), o contato apresenta-se aberto, de modo que, quando a entrada está ativa (nível lógico 1), o contato apresenta-se fechado;

- Contato normalmente fechado (NF) – também representa o estado lógico de uma entrada digital. Quando a entrada está ativa (nível lógico 1), o contato apresenta-se aberto e quando a entrada está inativa (nível lógico 0), o contato apresenta-se fechado;

- Bobina – a bobina representa os sinais das saídas digitais. Quando a mesma está inativa, a saída correspondente apresenta-se desligada (nível lógico 0) e quando a mesma está ativa, a saída correspondente apresenta-se ligada (nível lógico 1).

A Fig. 3 apresenta um exemplo de programa na linguagem *Ladder*.

Figura 3. Programa em lógica *Ladder* para controle do processo.



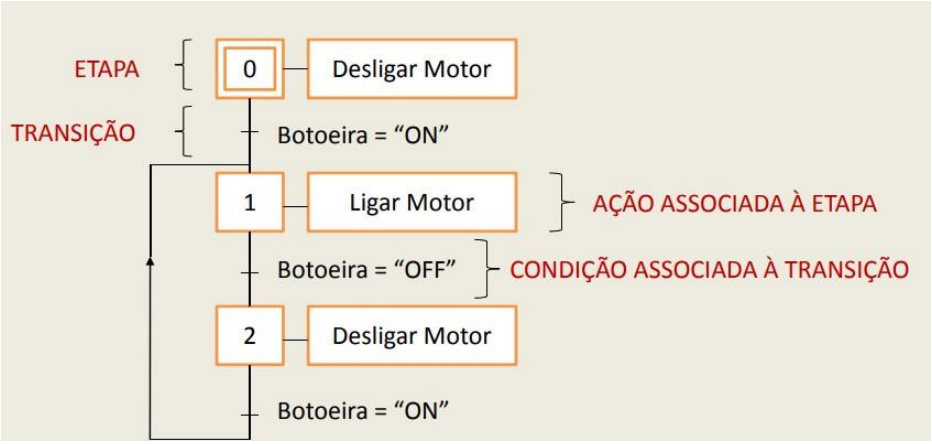
Fonte: Krakheche (2007)

No exemplo apresentado, a programação *Ladder* utiliza uma chave de pressão e uma chave de temperatura para monitorar as condições do sistema. Quando a chave de pressão é pressionada e a chave de temperatura está dentro de um limite aceitável, a bobina de partida do motor é energizada, acionando assim o motor. Esse acionamento pode ser realizado manualmente através de um botão de comando, permitindo ao operador controlar o início e a parada do motor conforme necessário. Essa programação permite o controle seguro e eficiente do motor, garantindo que ele seja acionado apenas quando todas as condições necessárias forem atendidas. É uma abordagem amplamente utilizada na automação industrial devido à sua simplicidade e facilidade de compreensão.

Outra linguagem muito difundida na programação de CLPs é o sequenciamento de gráficos de funções. O “Sequential Function Chart” (SFC) ou *GRAFCET* é uma linguagem de programação gráfica utilizada em automação industrial para descrever o comportamento sequencial de sistemas automatizados. Essa linguagem consiste em uma série de etapas, que são representadas por caixas retangulares, conectadas por transições, que são representadas por setas. Cada etapa representa uma ação ou conjunto de ações que devem ser executadas pelo sistema, enquanto as transições indicam as condições que devem ser atendidas para que o sistema passe para a próxima etapa. O SFC é amplamente utilizado em sistemas que requerem um alto grau de segurança, como em processos industriais, em que as etapas precisam ser executadas na ordem correta e com as condições adequadas. Além disso, a linguagem SFC é fácil de entender e visualizar, o que facilita a programação e a depuração dos sistemas automatizados (Hand, 2018).

É possível gerar automaticamente um programa do controlador de um sistema a partir do modelo em SFC. Construir o modelo gráfico é muito mais simples do que desenvolver o programa do controlador. Para exemplificar a Fig. 4 apresenta um exemplo de programa na linguagem SFC.

Figura 4. Exemplo de um diagrama SFC ou *GRAFCET*.



Fonte: Cassillo (2018)

O funcionamento da programação em SFC ocorre da seguinte forma: inicialmente, o sistema está na etapa 0 - Desligar motor. Quando é dado o comando "ON" para ligar o motor, ocorre uma transição da etapa 0 para a etapa 1 - Ligar motor. Nessa etapa, o motor é acionado e permanece ligado até que seja dado o comando para desligá-lo. Quando o comando "OFF" para desligar o motor é acionado, ocorre outra transição, dessa vez da etapa 1 para a etapa 2 - Desligar motor. Nessa etapa, o motor é desligado, e o sistema volta ao estado inicial, pronto para receber um novo comando para ligar o motor. Essa programação permite controlar o funcionamento do motor de forma sequencial e segura, garantindo que ele seja ligado e desligado conforme necessário. A representação gráfica facilita a compreensão do fluxo de operação do sistema e é amplamente utilizada em sistemas de automação para controlar diversos processos industriais.

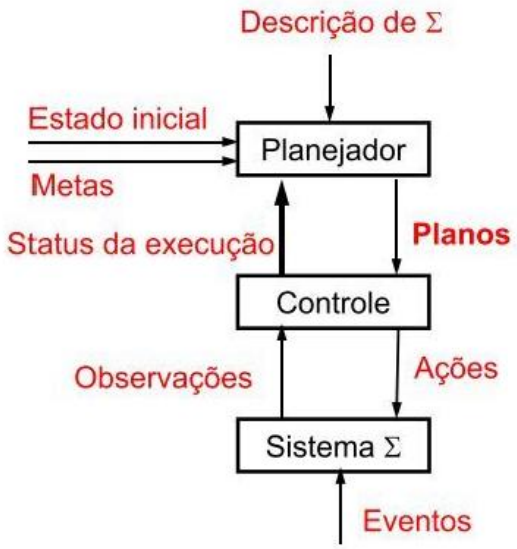
2.3 Planejamento Automático

Planejamento é o processo de escolha e organização de ações a partir da antecipação ou previsão dos seus efeitos. Esse processo de raciocínio tem o objetivo de satisfazer, através de ações, algumas metas previamente estabelecidas. Advindo da inteligência artificial, o planejamento automático é um processo de deliberação abstrato e explícito que escolhe e organiza

ações, antecipando os seus resultados esperados. Esta deliberação visa alcançar o melhor objetivo possível, e esse processo de deliberação é feita dentro de ambiente computacional, em sistemas que exigem comportamento autônomo. (Ghallab *et al*., 2016).

Basicamente, o objetivo é produzir uma sequência de ações que deverá ser executada por um ou mais agentes. Sendo assim, dado o estado inicial, um conjunto de operadores (ações que transformam o estado do mundo) e um estado final, o objetivo de um algoritmo de planejamento (denominado planejador) é encontrar a sequência de ações (plano) que conduz o agente do estado inicial para o estado final (Cantoni, 2010). As técnicas de planejamento automático são a base da pesquisa de IA e se concentram em algoritmos para derivar automaticamente uma estratégia para atingir um determinado objetivo. Uma estratégia neste contexto é um conjunto de ações e sua ordenação. O modelo conceitual de planejamento de IA (clássico), que é o foco neste trabalho, é representado na Fig. 5.

Figura 5. Modelo conceitual do planejamento clássico.



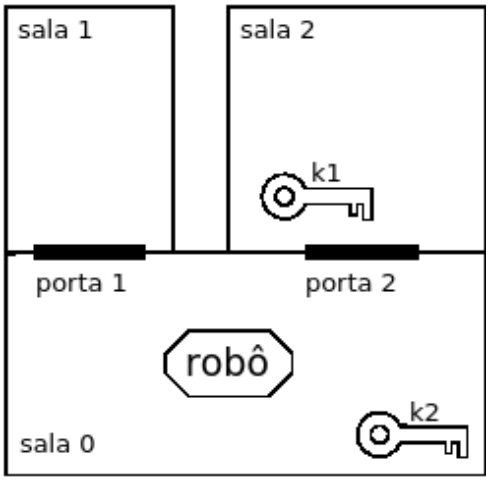
Fonte: Traduzido de Ghallab *et al.*, (2004)

No modelo conceitual é possível observar os três principais elementos que, normalmente, compõem um sistema de planejamento: o Planejador que recebe a descrição do sistema Σ, bem como o estado inicial e os objetivos para esboçar um plano e fornecê-lo ao controlador. O controlador recebe o estado atual do sistema Σ, e com base em observações realiza a ação de acordo com o plano definido pelo planejador. Um modelo de planejamento (clássico) é um

sistema de transição de estado discreto $\Sigma = \{S, A, \gamma\}$ com um conjunto de estados *S*, um conjunto de transições de estado *A*, (também referido como ações), e uma função de transição de estado $\gamma : S \times A \rightarrow S$. Com base nesse modelo de planejamento baseado em estados, um estado inicial define o estado do sistema antes que qualquer ação seja executada e o estado objetivo define o objetivo geral do planejamento automatizado: o estado que deve ser alcançado após a execução da estratégia a ser determinada pelo planejador. A combinação de modelo de planejamento, conjunto de objetos ou agentes, estado inicial e objetivo também é chamada de problema de planejamento. Quando um planejador identifica uma estratégia adequada, ela é passada para uma unidade de execução para executá-la e, finalmente, transferir o sistema para o estado objetivo.

As técnicas clássicas de planejamento são tipicamente caracterizadas por meio de seu domínio de aplicação (Ghallab *et al*., 2004), que é o próprio sistema e seu ambiente (ambos juntos também conhecidos como domínio do discurso). Os planejadores independentes de domínio não levam em consideração informações específicas de domínio e, portanto, afirmam ser aplicáveis a todos os domínios do discurso. Em contraste, os planejadores de domínio específico se concentram em desafios específicos por meio de algoritmos especializados dentro de um domínio de discurso bem definido, como um robô que tem o objetivo de sair de uma sala e chegar em outra, usando chaves para abrir as portas das salas. Esta tarefa pode ser modelada como um domínio de planejamento denominado "Domínio das Chaves" (Göbelbecker *et al*., 2010). Modelar o domínio de um problema é um trabalho complexo que vai depender do conhecimento que se tem sobre o problema e da representação desse conhecimento. A modelagem do domínio torna-se mais complexa à proporção que o número de variáveis de um problema aumenta. Um domínio de planejamento é constituído pelo ambiente em que o agente, nesse caso o robô, se encontra para executar uma tarefa e pelas ações que o agente pode executar dentro desse ambiente. No exemplo do "Domínio das chaves", demonstrado na Fig. 6, o ambiente é formado de salas, portas e chaves, e as ações que o agente pode realizar são: pegar uma chave, abrir uma porta e entrar em uma sala.

Figura 6. Estado inicial de um problema de planejamento.



Fonte: traduzido de Göbelbecker *et al*., (2010)

As ações de um domínio de planejamento são fornecidas por precondições e efeitos, essas precondições são proposições que precisam ter um valor verdadeiro para que a ação possa ser realizada. No exemplo citado anteriormente, para o robô realizar a ação de pegar uma chave em uma sala é necessário que exista essa chave nessa sala e que o robô nela esteja. Já os efeitos alteram os valores das proposições. No exemplo, quando o robô realiza a ação de abrir uma porta, a proposição que indica a condição daquela porta aberta passa de falso para verdadeiro.

Segundo Cardoso (2014) a utilização do planejamento automático dentro da IA possui duas motivações. Primeiro, consiste do desenvolvimento de ferramentas que auxiliem no processo de planejamento, onde no domínio industrial os planejadores de processos lidam com situações em que são necessárias mudanças nas tarefas a fim de atingir novos objetivos, seguindo a demanda requerida. Com isso, o planejamento automático tem o objetivo de acelerar esse processo de escolha e análise de alternativas.

A segunda motivação compreende uma abordagem mais teórica, se referindo ao estudo da “inteligência” em si. Com o desenvolvimento de tais tecnologias, é possível aproximar do que se trata de “essência da inteligência”, permitindo assim, que se desenvolvam tecnologias que consigam abstrair de forma mais completa o conceito de inteligência artificial. Nesse contexto, é correto afirmar que desta tecnologia pressupõe-se a criação de equipamentos e máquinas “inteligentes” que consigam ampliar a autonomia dos equipamentos de controle.

2.3.1 Planejamento automático independente de domínio

O domínio de automação de máquinas e instalações focalizado neste trabalho pode ser entendido como um domínio potencial de discurso para planejadores automáticos. Normalmente, a universalidade de planejadores independentes de domínio resulta em desempenho inferior em comparação com planejadores específicos de domínio que são otimizados para lidar com um problema específico. No entanto, desenvolver planejadores específicos de domínio é uma tarefa complexa, demorada e custosa. Os planejadores configuráveis, em conjunto com a utilização de hardware padronizado, são capazes de superar essa desvantagem, uma vez que aplicam conceitos e técnicas de planejamento independentes de domínio, que são configurados através de informações adicionais específicas do domínio, proporcionando uma maior flexibilidade e adaptação às demandas do processo produtivo. Dessa forma, a integração do hardware padronizado, como o CLP, com os planejadores configuráveis torna-se uma estratégia eficiente para a otimização dos processos produtivos, uma vez que permite uma maior personalização dos processos de produção, sem a necessidade de se preocupar com as especificidades do hardware.

As técnicas de planejamento automatizado podem ser enraizadas em solucionadores de problemas gerais já introduzidos na década de 1960 (Newell & Simon, 1961). Durante este tempo, várias abordagens foram propostas com base em diferentes técnicas e formalismos que podem ser divididos em duas categorias principais: abordagens de planejamento clássicas e neoclássicas.

Abordagens clássicas incluem algoritmos de busca em diferentes definições de espaço de busca, por exemplo, espaços de estado e espaços de plano. Fluxos de pesquisa ortogonais, como planejamento hierárquico e de menor comprometimento, são combinados com essas abordagens. Nesse aspecto, o planejamento clássico possui três formas principais para representar os domínios de planejamento: Teoria de Conjuntos, Variáveis de Estados e a Representação Clássica que é a mais utilizada (Ghallab *et al*., 2004).

Em contraste com as abordagens clássicas de planejamento, as abordagens neoclássicas incorporaram técnicas de áreas de pesquisa relacionadas, como programação de restrições ou teoria da satisfatibilidade em lógica proposicional. Vários trabalhos contam com a lógica proposicional para descrever o problema de planejamento de forma consolidada. Esse formalismo

é usado em vários planejadores ao considerar um plano como um conjunto de proposições lógicas. Um plano válido é então transformado em uma atribuição de "variáveis lógicas" (booleanas). O planejamento em si é levado de volta à prova de satisfatibilidade lógica. Esta abordagem é aplicada, por exemplo, dentro dos planejadores SATPLAN (Kautz & Selman, 1992) e WALKSAT (Selman et al., 1994). Em tese, as técnicas de planejamento neoclássicas contam com os fundamentos teóricos fornecidos pelas abordagens clássicas de planejamento. Eles utilizam formulações alternativas como satisfação de restrições e otimização para melhorar o desempenho do planejamento e permitir a determinação de uma solução ótima com critérios de otimização arbitrários.

2.4 Linguagens de planejamento

Os sistemas de planejamento geralmente adotam alguma linguagem de definição para modelar domínios e problemas de planejamento. Essa linguagem serve como entrada para o planejador. O núcleo conceitual das mais famosas linguagens de definição de problemas de planejamento para planejamento independente de domínio, são, STRIPS e PDDL, as quais serão brevemente apresentadas nesta seção.

A linguagem STRIPS (Fikes & Nilsson, 1972) foi desenvolvida como linguagem de entrada para o também denominado *Stanford Research Institute Problem Solver* (STRIPS) e pode ser vista como progenitora de cada linguagem de definição de problemas de planejamento. De acordo com Bylander (1994), um modelo STRIPS pode ser considerado como um modelo de 4 tuplas $\{P,O,I,G\}$ Onde P é um conjunto de proposições lógicas. Os estados inicial e objetivo do sistema em consideração são dados como $I \subseteq P$ e $G \subset P$, respectivamente. As transições de estado do sistema são dadas por meio do conjunto de operações O. Uma operação $o \in O$ é uma tupla $\{P_{RE}\ E_{FF}\}$ com um conjunto de proposições $P_{RE} \subseteq P$ definindo a execução de uma operação e um conjunto de proposições $E_{FF} \subseteq P$ descrevendo o efeito da execução da operação. Assim, as operações $o \in O$ definem a função de transição de estado $P_{RE} \rightarrow E_{FF}$ como descrito inicialmente nesta seção.

A linguagem mais conhecida e utilizada na representação clássica é a linguagem de de-

finição de domínios e problemas de planejamento, do inglês "*Planning Domain Definition Language"* (PDDL) (Mcdermont, 1998). A linguagem PDDL foi apresentada pela primeira vez em 1998, o objetivo era servir como base para especificação dos problemas de planejamento da competição AIPS-98. Trata-se de uma linguagem formal desenvolvida pela comunidade de pesquisa da área de planejamento automático (Ghallab *et al*., 2004). Essa linguagem destina-se a especificar o domínio, ou seja, os tipos de objetos, predicados existentes, quais possíveis ações e funções, além da instância do problema a ser resolvido, formado pela descrição dos estados inicial e final. Para a representação do problema de planejamento automático de forma computacional a informação é dividida em dois arquivos diferentes: o domínio e o problema. Os dois arquivos são inseridos como parâmetros de entrada para um planejador, o qual pode encontrar ou não uma solução ou sequência de ações que pode ser executada pelo agente.

A PDDL é o resultado da junção de diversas linguagens da área de planejamento, especialmente o STRIPS e ADL (Vaquero *et al.*, 2009). A versão inicial da PDDL é inspirada no STRIPS, mas conta com formalismo lógico de predicados. Assim, um problema de planejamento em PDDL é um conjunto de predicados (em vez de proposições em comparação com STRIPS). No entanto, a definição dos estados inicial e objetivo é semelhante. Como de costume na lógica de predicados, uma definição de problema PDDL é baseada em alguns objetos dentro do domínio do discurso. As ações também são descritas por meio de precondições e efeitos. Por se tratar de uma linguagem padronizada, a representação de um domínio em PDDL possibilita que planejadores diferentes possam tratar o mesmo problema de planejamento. Uma das vantagens dessa linguagem é o fato dela ser independente de domínio, podendo, então, ser aplicada a uma gama de problemas de naturezas distintas, desde problemas simples como o domínio das chaves, exemplificado anteriormente, a problemas mais complexos que envolvam as dimensões tempo e recursos.

A PDDL já passou por várias modificações desde a sua apresentação em 1998, ao longo dos anos, ela foi estendida e revisada, depois disso foi apresentada a PDDL 1.2, em seguida por exemplo, para suportar valores numéricos adicionais no PDDL 2.1 (Fox & Long, 2003) ou metas fracas (opcionais) no PDDL (Gerevini & Long, 2005). Também foram propostas versões não determinísticas, por exemplo, PPDDL (Younes & Littman, 2004), atualmente já se tem a versão 3.1. De acordo com Cantoni (2010), a cada versão novas características e mais possibilidades são adicionadas à linguagem, permitindo que uma diversidade maior de problemas seja especificada. Essas mudanças são propostas por um grupo de pesquisadores com vasto conhecimento da área. Após a publicação das novas características e da gramática, planejadores são

desenvolvidos ou modificados para suportarem a nova versão. A Tab. 2.1 exibe as versões, o ano de lançamento, a competição, a conferência onde a publicação descreve as modificações na linguagem.

Tabela 2.1. As cinco versões da PDDL.

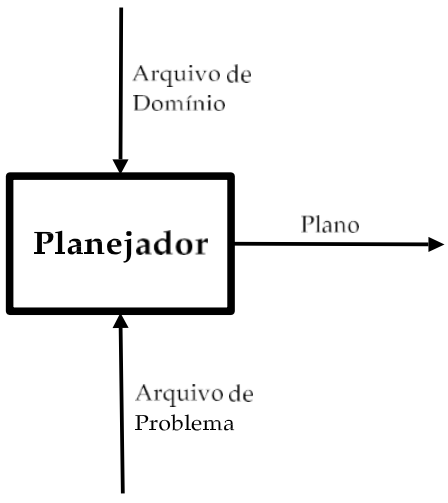
Versão	Ano	Competição	Conferência	Publicação que descreve a versão
1.2	1998	IPC-1	AIPS-98	Ghallab *et al*. (1998)
2.2	2002	IPC-3	AIPS-02	Fox & Long (2003)
2.2	2004	IPC-4	ICAPS-04	Edelkamp & Hoffmann (2004)
3.0	2006	IPC-5	ICAPS-06	Gerevini & Long (2005)
3.1	2008	IPC-6	ICAPS-08	Não existe publicação

Fonte: Cantoni (2010)

2.4.1 Representação do Domínio e do Problema em PDDL

A linguagem PDDL possui um formalismo para definições do domínio e do problema. O arquivo de domínio tem a finalidade de expressar a física de um domínio, sendo necessário especificar quais predicados existem, as ações possíveis, a estrutura das funções, e quais os efeitos das ações. Já no arquivo de problema, é descrita uma instância de um problema a ser resolvido, onde a instância é composta pelos estados inicial e final. A Fig. 7 apresenta uma visão geral de um problema de planejamento em PDDL segundo Cantoni (2010).

Figura 7. Visão geral da resolução de um problema de planejamento em PDDL.



Fonte: Cantoni, (2010)

2.5 Revisão da Literatura

Realizou-se uma busca na literatura sobre planejamento automático aplicado à processos de manufatura. A estratégia de busca se deu nas seguintes bases de dados: Portal de Periódicos da CAPES, Google Acadêmico, *Scopus* (Elsevier Science), *Scientific Eletronic Library Online* (SciELO). O acesso ocorreu entre os meses de fevereiro a outubro de 2021. Os descritores utilizados para a busca foram: planejamento automático; automação; manufatura. Foram incluídos estudos teóricos e práticos sobre planejamento automático aplicado à processos de manufatura, escritos em português ou inglês, publicados entre os anos de 2016 e 2021. Na etapa de busca e seleção da literatura científica, foi feita uma busca avançada nas bases de dados mencionadas, resultando em 69 artigos: 34 disponíveis no Google Acadêmico; 18, no Portal de Periódicos da CAPES; 11, no *Scopus* e 6, no SciELO. Após a leitura dos títulos, resumos ou *abstracts* dos textos, foram excluídos 50 artigos que não atendiam aos critérios de elegibilidade. Esses critérios incluíam a pertinência ao tema de pesquisa, ou seja, os artigos deveriam abordar o planejamento automático aplicado a processos de manufatura. Além disso, os artigos foram avaliados quanto à qualidade e relevância dos conteúdos. As informações consideradas relevantes diziam respeito à contribuição significativa para o campo de pesquisa em planejamento automático, a inclusão de resultados empíricos, experimentos e estudos de casos que comprovassem a eficácia das abordagens propostas, bem como a fundamentação teórica sólida e a clareza na exposição dos conceitos abordados. Artigos que não atendiam a esses critérios de qualidade e relevância foram excluídos da análise.

Com isso, 19 artigos foram elegíveis para compor a amostra da presente revisão. Estes foram dispostos em um quadro para síntese, contendo informações como: periódico, autores, ano de publicação, título, tipo de estudo, objetivo, considerações e resultados de interesse. Os estudos incluídos nesta revisão destacam a pesquisa e o desenvolvimento de soluções de automação inteligentes baseadas em técnicas de planejamento e programação de processos integrados automatizados em nível de controle de manufatura discreta.

Após a primeira etapa de seleção, já na segunda fase, passou-se à leitura na íntegra dos artigos selecionados. Os dados relativos a cada um deles estão identificados na Tab. 2.2 e Tab. 2.3, que contemplam, respectivamente, a distribuição dos artigos selecionados, considerando-se o ano de publicação, autor(es) do texto, título, objetivo e método (tipo de estudo), e principais resultados.

Tabela 2.2. Distribuição das referências incluídas na revisão integrativa, 2016-2021.

Artigo/Ano	Autor(es)	Título	Objetivo	Tipo de Estudo
1/2021	JI, S. *et al*.	Learning-Based Automation of Robotic Assembly for Smart Manufacturing	Abordagem para a autoprogramação de tarefas de montagem robótica.	Estudo de caso
2/2017	LEGAT, C.; VOGEL-HEUSER, B.	A configurable partial-order planning approach for field level operation strategies of PLC based industry 4.0 automated manufacturing systems	Estudo intensivo sobre as obras existentes sobre planejamento automatizado, tanto teóricas quanto práticas.	Revisão bibliográfica
3/2016	MICHNIEWICZ, J.; REINHART, G.	Cyber-Physical-Robotics – Modelling of modular robot cells for automated planning and execution of assembly tasks	Abordagem para autoprogramação de tarefas de células modulares de robôs utilizando Sistemas Cibernéticos-Físicos (CPS).	Estudo de caso
4/2018	SIERLA, S. *et al*.	Automatic assembly planning based on digital product descriptions	Planejamento de montagem e fabricação orientados por descrições de produtos digitais.	Estudo de caso
5/2018	FRIEDRICH, C. *et al*.	Efficient Task and Path Planning for Maintenance Automation Using a Robot System	Abordagem para planejamento automático de tarefas e caminhos de manutenção usando um sistema de robô	Estudo de caso
6/2019	WALLY, B. *et al*.	Flexible Production Systems: Automated Generation of Operations Plans Based on ISA-95 and PDDL	Abordagem baseada em modelo para transformar automaticamente uma especificação de sistema de manufatura em um plano de produção por meio de planejamento automatizado.	Estudo de caso

7/2018	OSTROUKH, A. *et al*.	Automated process control system of mobile crushing and screening plant	Pesquisa o problema de automatizar a planta móvel de britagem e peneiramento como um sistema complexo de vários níveis utilizando o planejamento automático.	Estudo de caso
8/2021	MALHAN, R. K. *et al*.	Automated planning for robotic layup of composite prepreg	Uma célula robótica capaz de executar layup usando os planos gerados automaticamente.	Estudo de caso
9/2018	ROGALLA, A. *et al*.	Improved Domain Modeling for Realistic Automated Planning and Scheduling in Discrete Manufacturing	Desenvolvimento de um conceito de Planejamento e programação de processos integrados automatizados em nível de controle em manufatura discreta.	Bibliográfico
10/2018	WHITEHEAD, E. *et al*.	Automated Planning Enables Complex Protocols on Liquid-Handling Robots	Desenvolvimento de um sistema de software Roboliq, que usa métodos de inteligência artificial (IA) para integrar protocolos padronizados e linguagens de programação dedicadas para operações de manuseio de líquidos.	Estudo de caso
11/2018	BEHANDISH, M. *et. al*.	Automated process planning for hybrid manufacturing	Uma nova abordagem sistemática para o planejamento automatizado do processo de manufatura híbrida é apresentada.	Estudo de caso

12/2019	KOU, X. *et al*.	Sub-assembly recognition algorithm and performance analysis in assembly sequence planning	O foco está concentrado em um algoritmo de reconhecimento de submontagem e análise de desempenho no planejamento da sequência de montagem.	Estudo de caso
13/2021	YOO, J. J. *et al*.	An intelligent learning framework for Industry 4.0 through automated planning	É apresentado um mecanismo para facilitar a aquisição de conhecimentos e habilidades. Especificamente, uma ferramenta de software de planejamento automatizado, iCoursePlanner.	Estudo de caso
14/2017	KABIR, A. M. *et al*.	Automated Planning for Robotic Cleaning Using Multiple Setups and Oscillatory Tool Motions	O artigo apresenta algoritmos de planejamento para limpeza robótica de manchas em superfícies não planas.	Estudo de caso
15/2016	BORGO, S. *et al*.	A Planning-Based Architecture for a Reconfigurable Manufacturing System	O artigo descreve o uso de um novo sistema planejamento na Manufatura Reconfigurável.	Estudo de caso
16/2017	MCCLUSKEY, T. L. *et al*.	Engineering Knowledge for Automated Planning: Towards a Notion of Quality	Este artigo levanta algumas questões relacionadas à engenharia de conhecimento de aplicação para planejamento automatizado, com foco no modelo de domínio.	Bibliográfico

17/2019	SILVA, J. M.; SILVA J. R.	A new hierarchical approach to requirement analysis of problems in automated planning	É apresentado abordagem para a análise de requisitos de problemas no planejamento automatizado através de uma ferramenta de software desenvolvida por um dos autores	Estudo de caso
18/2020	WALLY, B. *et al*.	Leveraging Iterative Plan Refinement for Reactive Smart Manufacturing Systems	Neste artigo, é apresentado uma nova abordagem para projetar e controlar sistemas de manufatura inteligentes.	Estudo de caso
19/2019	DJE-ZAIRI, S. *et al*.	Automated Planning for Mobile Manipulators using passive RFID tags	Este artigo propõe uma abordagem baseada em um Planejamento de Tarefas em Camadas para manipuladores móveis.	Estudo de caso

Fonte: Próprio autor (2021)

Tabela 2.3. Principais resultados dos artigos selecionados, 2016-2021.

Artigo/ Ano	Resultados
1/2021	Os resultados verificam que o sistema de montagem robótica automatizado proposto é viável, pois está associado a uma redução dramática no esforço humano necessário para automatizar a montagem robótica. Para planejar as ações do robô foi predefinido um conjunto de ações a ser usado no domínio PDDL a partir de objetos em 3D de seus arquivos CAD.
2/2017	Mostra-se que a abordagem proposta é capaz de resolver com eficiência grandes problemas de planejamento, exibindo características de escalabilidade positivas, o que indica sua aplicabilidade para plantas de tamanho real. Uma visão geral dos fundamentos conceituais da nova abordagem de planejamento foi proposta, chamada HiTraP-AT. O núcleo conceitual das linguagens de definição de problemas de planejamento mais famosas para o planejamento independente de domínio, estudadas neste trabalho foram STRIPS e PDDL.

3/2016	O método apresentado mostra as capacidades de sistemas de robôs modulares equipados com Cyber-Physical-Devices e –Products a partir de objetos em 3D de seus arquivos CAD. Para facilitar a reconfiguração das células do robô, a abordagem Plug & Produce foi introduzida. O artigo se concentra nas informações trocadas e armazenadas por vários CPS na célula do robô e não aborda as tecnologias de troca de dados.
4/2018	O artigo investiga como um designer poderia enviar uma descrição de produto digital a vários fabricantes em potencial com a capacidade de realizar um planejamento de montagem virtual totalmente automático. Foram apresentados dois estudos de caso e aplicações em um contexto industrial real. O objetivo foi definir o conceito em vez de focar protocolos. Os métodos ASP e APP foram usados para implementar essas funcionalidades dentro do framework.
5/2018	Para o planejamento do caminho foram utilizados algoritmos globais de última geração com um método que permite a adaptação do tamanho das etapas de exploração para reduzir o tempo de planejamento, o que permite que um sistema de robô planeje de forma autônoma diferentes tarefas de manutenção. Para o planejamento de caminho foram aplicados planejadores baseados em amostragem com RRT e RRT bidirecional.
6/2019	Foi apresentado uma abordagem baseada em modelo para transformar automaticamente uma especificação de sistema de manufatura em um plano de produção por meio de planejamento automatizado. Foram utilizados um mapa conceitual e um fluxo de trabalho para a transformação de modelos ISA-95 em um formalismo PDDL. A abordagem de um caso de uso foi testada com sucesso, na qual verificou-se que o layout do sistema de transporte escolhido para atenderia aos requisitos de logística. Foi usado uma ferramenta de software simples que é capaz de executar o plano computado com máquinas reais.
7/2018	Um software para o sistema de controle de processo automatizado de materiais de pedra de britagem foi proposto, referido software torna possível integrar um conjunto de informações de unidades tecnológicas em um hardware de gestão complexa. Desenvolveu-se um sistema de controle automatizado Metso IC para britagem e processo de classificação de materiais de pedra com base no sistema SCADA, o qual fornece uma utilização simples e segura da unidade de britagem. O sistema permite o indivíduo controle o alimentador, triturador e transportadores com um único toque na tela.
8/2021	O sistema pode gerar planos de maneira computacionalmente eficiente, além de lidar com uma grande variedade de peças complexas. Foi demonstrado o layup automatizado conduzindo experimentos físicos em um molde inspirado na indústria usando os planos gerados. Esta abordagem gerou, com sucesso, planos de agarrar para peças com vários níveis de complexidade. O MATLAB foi usado para a implementação do planejador de aperto. O software utilizado para gerar os algoritmos é o GenGraspPlan.
9/2018	O artigo constrói uma ponte entre a automação e o planejamento de IA e permite um planejamento e programação automatizados realistas em manufatura discreta. Diferentes abordagens de modelagem (MAs) discutidas são avaliadas por um estudo de caso. O artigo foca sua análise na linguagem de planejamento Planning Domain Definition Language (PDDL).

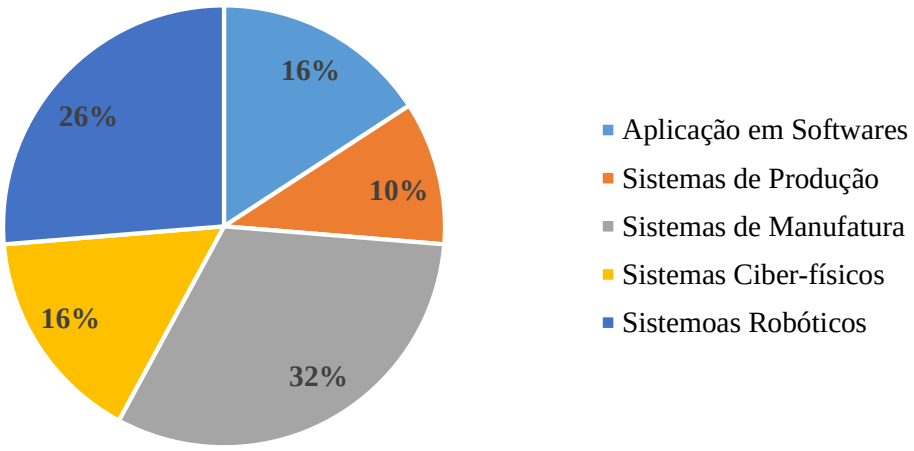
10/2018	Neste artigo foi desenvolvido o Roboliq, um sistema de software que usa métodos de inteligência artificial (IA) para integrar descrições de protocolo genéricas formais, porém intuitivas, recursos de programação completos, e interações usuário-sistema para gerar automaticamente programas de robô otimizados e executáveis. O protocolo integrado foi o JSON (JavaScript Object Notation), um padrão moderno para intercâmbio de dados na web que também é usado na linguagem Autoprotocol da Transcriptic e GUIs relacionadas.
11/2018	Este artigo apresenta fundamentos teóricos e algoritmos computacionais para permitir a construção automática de planos de processo de HM válidos e econômicos para uma coleção arbitrária de recursos AM / SM, fornecidos pela mesma máquina ou por máquinas diferentes, com formas e movimentos de complexidade geométrica arbitrária.
12/2019	A modelagem matemática foi realizada de acordo com o processo e princípio de reconhecimento de submontagem, o algoritmo de reconhecimento de submontagem foi projetado e compilado com base no Matlab, e a viabilidade do algoritmo de reconhecimento de submontagem foi verificada por um exemplo.
13/2021	Neste artigo foram identificadas as competências-chave que os acadêmicos das áreas de engenharia devem desenvolver para a era da Indústria 4.0 seguindo a literatura existente e, sequencialmente, apresentado um software inteligente, iCoursePlanner, para o estudo de caso de planejamento automático.
14/2017	Foram apresentados algoritmos para automatizar a limpeza robótica de superfícies curvas com manchas duras. A abordagem foi aplicável a tarefas como polimento e decapagem de tinta. O envolvimento de múltiplos reposicionamentos e reorientações tornou o método adequado para tarefas práticas de limpeza. Foram apresentadas heurísticas que permitem ao algoritmo fornecer soluções em tempo real. Foi utilizado como algoritmo de busca o DFBnB. Referido algoritmo foi adotado para resolver o problema de planejamento de configuração.
15/2016	Este artigo descreve um novo aplicativo baseado em planejamento para o domínio de fabricação. Um estudo de caso para testar as soluções integradas foi utilizado para avaliar as capacidades de planejamento. A planta visa reciclar Placas de Circuito Impresso (PCBs). O Artigo não fornece a linguagem utilizada.
16/2017	Neste artigo, foi investigado a função e a natureza do modelo de domínio de planejamento e discutido sua importância no processo de criação de um aplicativo de planejamento. O artigo explana a linguagem de codificação de domínio de planejamento mais comum, PDDL. Um modelo de domínio usando diagramas também é apresentado, como na Unified Modeling Language (UML).
17/2019	Este artigo propõe um caminho para resolver uma lacuna entre a abordagem hierárquica e o estado da arte da análise de requisitos para permitir que os recursos previstos pela abordagem da Engenharia do Conhecimento realmente apareçam nos requisitos de um processo de planejamento. A princípio, na linguagem semiformal (diagramática) convencional - UML - traduzida para Redes de Petri Hierárquicas (HPNs) por um novo algoritmo aprimorado. O processo proposto foi instalado em um software - desenvolvido por um dos autores que analisa o desempenho do modelo de planejamento: itSIMPLE (Interface de Software de Ferramentas Integradas para Modelagem de Ambiente de Planejamento). Uma das linguagens de transferência usadas foi o PDDL.

18/2020	Os resultados da avaliação indicam que a abordagem apresentada é viável e capaz de fortalecer significativamente a flexibilidade dos sistemas de produção durante a execução. Foi apresentada uma abordagem para a utilização de planejamento automatizado para a criação de planos de produção contrastando com a abordagem tradicional, na qual as receitas são programadas no sistema de produção com antecedência. Este trabalho foi concentrado em problemas de planejamento clássicos baseados na Linguagem de Definição de Domínio de Planejamento (PDDL).
19/2019	A proposta abordada apresentada neste artigo foi validada em simulação. Foi apresentado um planejador de tarefas autônomo para manipuladores móveis operando em um ambiente industrial para a realização da tarefa de alimentação de peças. O robô verificava continuamente seu ambiente em busca de tags. Quando uma nova tag era detectada, seus dados eram lidos e analisados. O robô atuava no objeto de acordo com os objetivos especificados nos dados da etiqueta RFID. Os resultados mostraram que o método é bem-sucedido em tornar o planejamento de manipuladores móveis autônomo, a fim de que os robôs operassem sem a intervenção de um operador humano ou de um programa mestre.

Fonte: Próprio autor (2021)

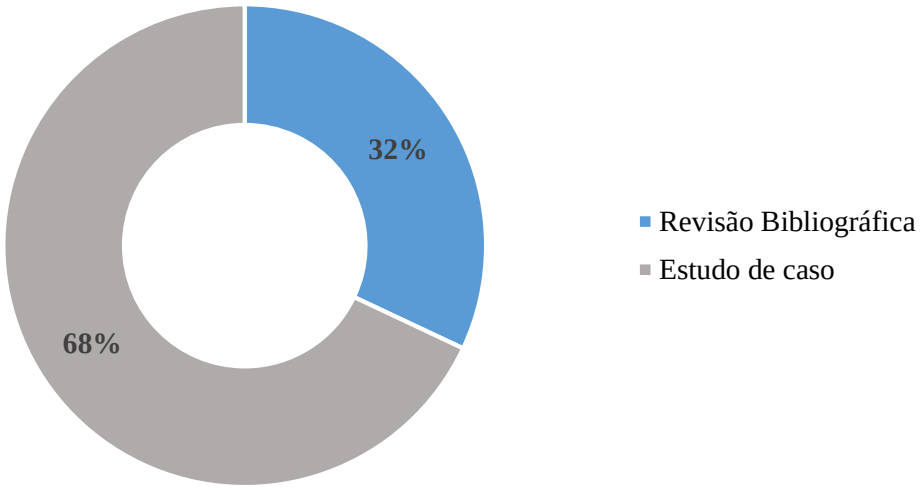
Resumidamente, o Gráf. 1 apresenta a distribuição percentual dos artigos classificados por tema. Cada fatia representa uma categoria temática específica (aplicação em software, sistemas de produção, manufatura, sistemas ciber-físicos ou sistemas robóticos) e sua proporção em relação ao total de artigos analisados. Já o Gráf. 2 detalha a classificação por porcentagem dos artigos lidos, categorizando-os em dois tipos de estudo distintos: revisão bibliográfica e estudo de caso.

Gráfico 1. Classificação dos artigos por tema.



Fonte: Próprio autor (2022)

Gráfico 2. Classificação dos artigos por tipo de estudo.



Fonte: Próprio autor (2022)

Diante dos trabalhos apresentados, a análise crítica dos 19 artigos revela uma diversidade de abordagens e contribuições para o campo do planejamento automático aplicado à manufatura e robótica. Os estudos de caso foram a abordagem mais comum, proporcionando uma visão prática da aplicação das técnicas de planejamento automático em situações reais. A revisão bibliográfica também foi presente, contribuindo para a compreensão mais ampla das teorias e práticas existentes no campo. Os artigos demonstram que a autoprogramação de tarefas de montagem robótica é uma área promissora, assim como a utilização de sistemas ciber-físicos para a modelagem e execução de tarefas. Além disso, o uso de técnicas de inteligência artificial, como o planejamento baseado em PDDL, mostrou-se relevante para a automação de processos de manufatura. As abordagens para planejamento de tarefas de manutenção, controle de sistemas de britagem e peneiramento e otimização de sistemas de produção também foram amplamente exploradas. No entanto, embora haja avanços significativos em cada artigo, a área ainda apresenta desafios, como a necessidade de aprimorar a modelagem de domínio e o refinamento iterativo de planos. Portanto, sugere-se que futuras pesquisas se concentrem em aspectos como a integração eficiente de sistemas, a aplicação de técnicas de aprendizado de máquina e a consideração de fatores de qualidade na engenharia do conhecimento para o planejamento automatizado. Dessa forma, será possível impulsionar ainda mais o desenvolvimento e a aplicação de sistemas de manufatura inteligentes e eficientes no contexto da indústria 4.0.

3. ESTUDO DE CASO

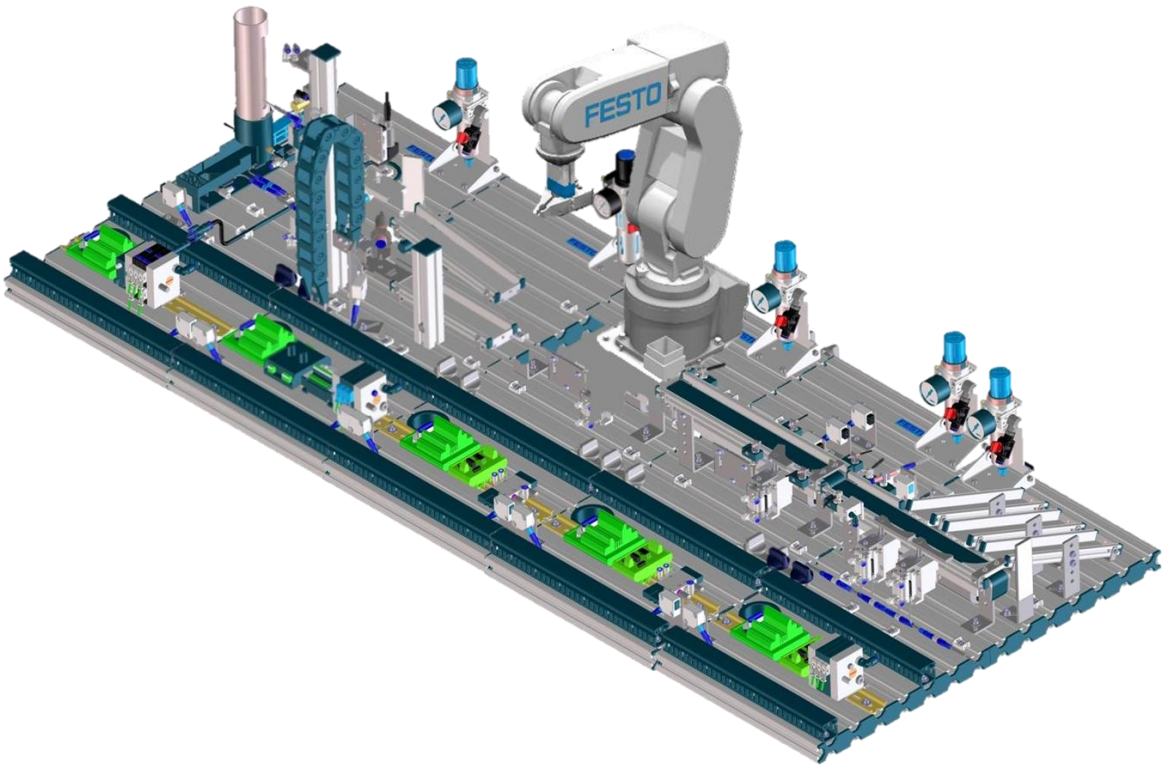
O presente estudo visa apresentar a modelagem e implementação de uma planta didática de manufatura robótica inspirada em planejamento automático. A planta didática foi projetada para fornecer uma plataforma de testes e simulação para avaliar as capacidades de controle e automação, bem como a eficiência do planejamento automático na produção. O estudo abordará o método utilizado na implementação da planta, desde a modelagem do sistema, seleção de sensores e atuadores até a programação do controlador e testes de validação. O objetivo é fornecer uma visão geral do processo de desenvolvimento da planta e das técnicas utilizadas para implementação de um sistema de manufatura robótica automatizado e eficiente.

3.1 Descrição geral da planta didática de manufatura

Para a aplicação do planejamento automático em sistemas práticos baseados em CLPs foi utilizada uma bancada didática de manufatura robótica que simula uma linha de produção composta por células individuais utilizando atuadores e válvulas pneumáticas controladas por CLP e um manipulador robótico de cinco graus de liberdade, responsável por movimentar peças cilíndricas entre duas células vizinhas. Cada célula tem uma função específica no processo, a saber: recebimento, processamento, distribuição, espera e separação.

Plantas didáticas ou plantas piloto são plataformas tecnológicas utilizadas para o aprendizado prático de controle de processos. Diversos trabalhos científicos fazem uso de plantas piloto ou didáticas. Os trabalhos de Ji *et al*. (2021), Malhan *et al*. (2021), Sioto & Schiavento (2018), Fonseca *et al*. (2016), Michniewicz & Reinhart (2016) por exemplo, fazem sua utilização no estudo de controle de processos. Na Fig. 8 é ilustrado um modelo tridimensional da bancada didática parcialmente instalada.

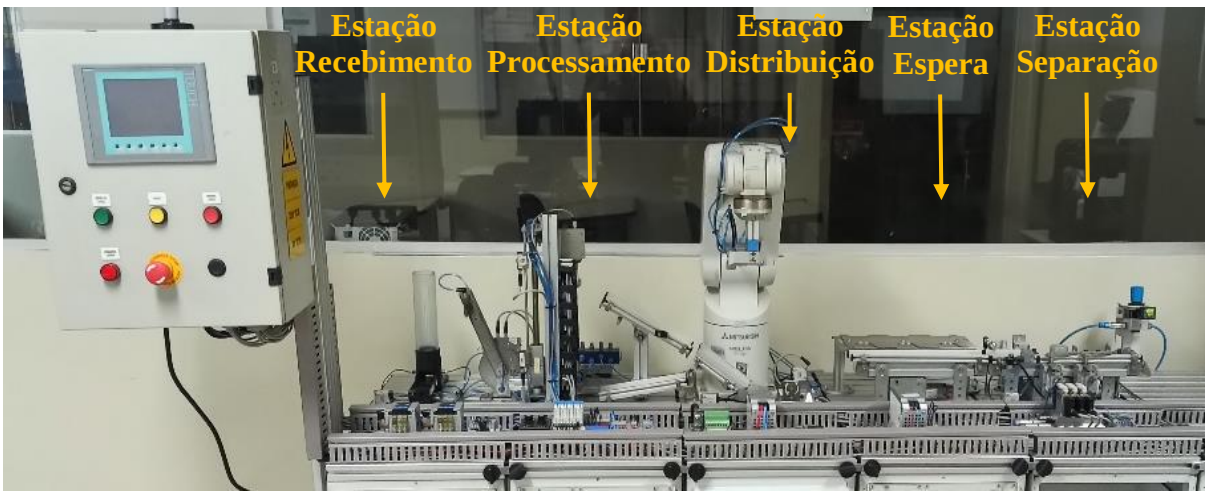
Figura 8. Modelo 3D da bancada de manufatura robótica.



Fonte: Próprio autor (2021)

O processo de manufatura da bancada é dividido em cinco etapas sequenciais, executadas respectivamente por cinco estações de trabalho. A Fig. 9 apresenta a foto da bancada física devidamente instalada. As seções 3.2 a 3.6 detalham cada estação prevista no processo produtivo considerado.

Figura 9. Fotografia com a visão geral da bancada de manufatura robótica instalada.



Fonte: Próprio autor (2021)

3.2 Estação Recebimento

A estação Recebimento é responsável por fornecer peças de trabalho ao sistema e é composta por dois módulos: o módulo magazine e o módulo de transferência de peças. O módulo magazine é responsável por disponibilizar as peças de trabalho, podem ser armazenadas até nove peças no magazine, um cilindro pneumático de dupla ação é responsável por disponibilizar novas peças de trabalho para o sistema, pode-se observar essas partes na Fig. 10. De uma forma geral, a estação magazine possui:

- Um cilindro de dupla ação instalado na parte inferior, responsável por empurrar as peças para que sejam distribuídas;
- Dois sensores magnéticos que detectam as posições de início e fim de curso de atuação do cilindro;
- Um sensor de barreira para detecção de presença de peças;
- Duas válvulas reguladoras de fluxo para o controle de velocidade de atuação do cilindro pneumático;

O módulo de transferência de peças é composto por um atuador giratório, responsável pelo transporte de peças de trabalho para a estação seguinte. As peças são apanhadas por meio de sistema a vácuo e transferidas por meio de um atuador rotativo pneumático:

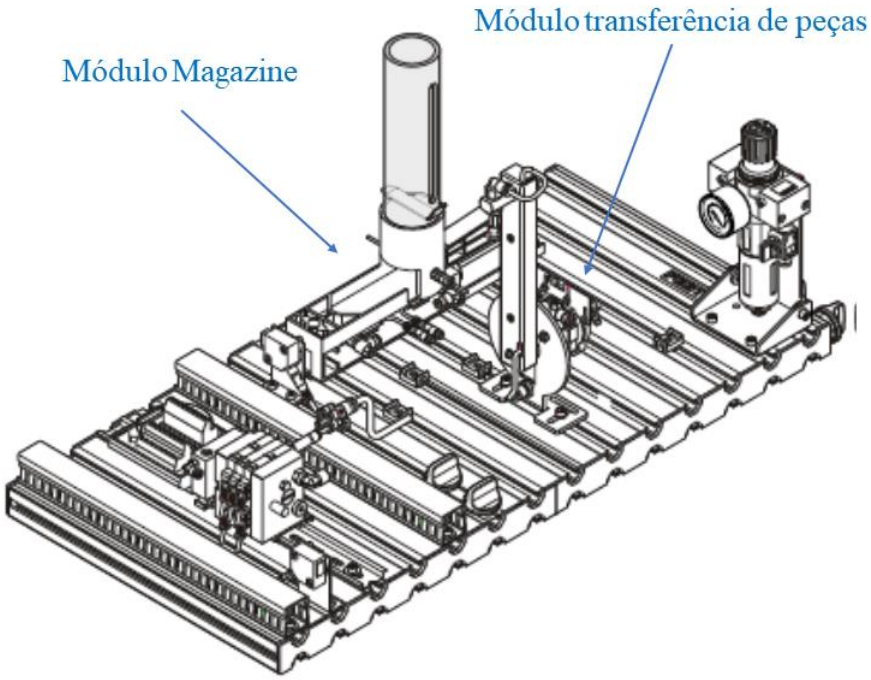
- Possui um atuador semi-rotativo pneumático de duplo efeito, com angulação de giro variando de 0º a 180º, limitado por um fim de curso mecânico;
- Possui duas chaves eletromecânicas de fim de curso para a detecção das posições extremas de atuação;
- Possui sistema de sucção por meio de ventosa.

Além dos módulos magazine e transferência de peças, a estação é composta pelos seguintes componentes:

- Vacuostato, sensor responsável pela detecção de peças por meio da sucção gerada pelo sistema de vácuo;

- Terminal de válvulas compacto (CPV), composto de: Uma válvula direcional 3/2 vias simples solenoide, utilizada para acionar o módulo atuador giratório; uma válvula direcional 5/2 vias simples solenoide, utilizada para acionar o cilindro pneumático do módulo magazine; uma válvula direcional (gerador de vácuo) - 2/2 vias, utilizada para alimentação da ventosa;

Figura 10. Representação tridimensional da estação Recebimento.



Fonte: Festo (2003)

3.3 Estação Processamento

A estação Processamento recebe peças provindas da estação Recebimento, verifica se a mesma atende aos critérios estabelecidos e direciona a peça para a rampa de peças aprovadas ou para a rampa de peças rejeitadas. Por meio de sensores, é possível realizar a classificação, de acordo com a verificação da reflexão da peça e altura. A estação é composta por módulos, o módulo elevador é utilizado para movimentar a peça até o módulo de medição, que por sua vez direciona a peça para a rampa de peças aprovadas ou de peças rejeitadas. Possui alguns componentes como:

- Um atuador pneumático de duplo efeito para deslocamento do eixo vertical;
- Um atuador linear pneumático de duplo efeito;
- Quatro sensores de fim de curso para detecção dos limites de curso dos atuadores;
- Duas válvulas reguladoras de fluxo.

O módulo de verificação identifica presença de material e a cor da peça baseado na reflexão do sinal óptico. É composto por dois diferentes sensores:

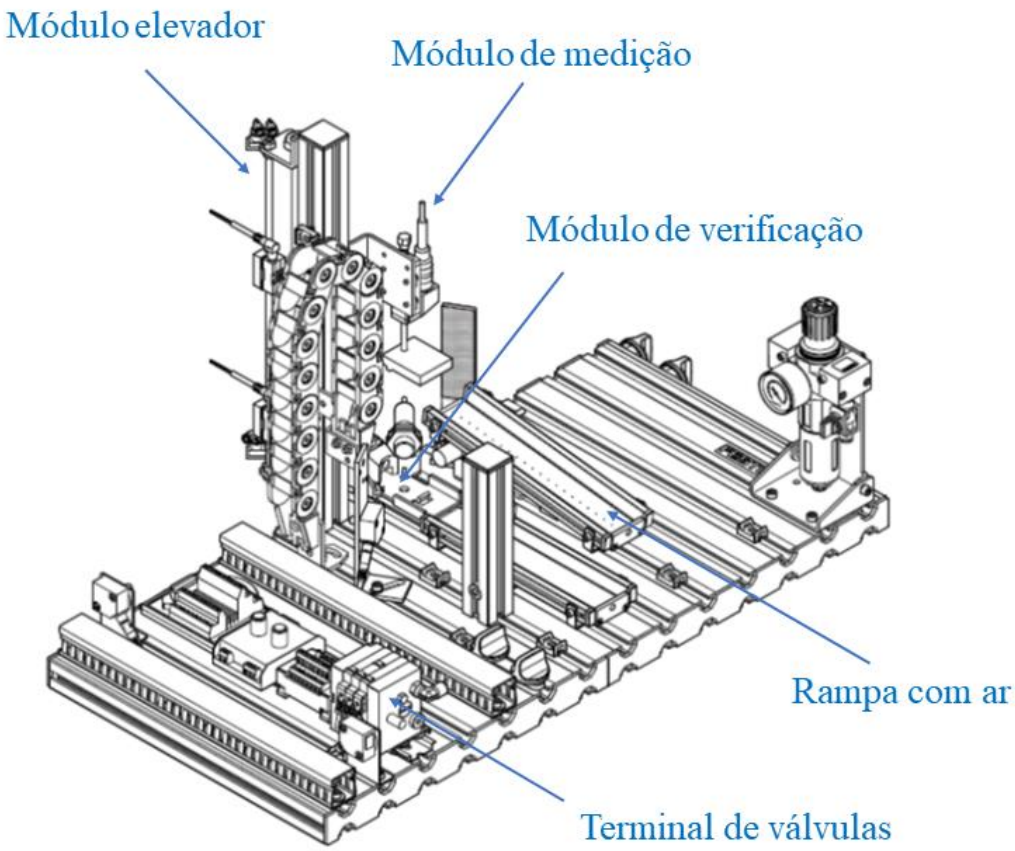
- Sensor óptico por reflexão difusa;
- Sensor capacitivo.

Além dos módulos elevador e medição, a estação Processamento é composta pelos seguintes componentes:

- Sensor de deslocamento linear, utilizado para realizar a medição da altura das peças;
- Rampa de deslize para separação das peças aprovadas e rejeitadas. Para reduzir o atrito no movimento das peças passando pela rampa, há um sistema de colchão de ar.
- Terminal de válvulas compacto (CPV), composto por duas válvulas direcionais 5/2 vias simples solenoide utilizadas para o acionamento do cilindro de expulsão de peças, do sistema de colchão de ar utilizado na rampa de deslize das peças; uma válvula direcional 3/2 vias simples solenoide utilizado para o acionamento do módulo elevador;

Na Fig. 11 é possível visualizar todas as partes referentes a esta estação.

Figura 11. Representação tridimensional da estação Processamento.



Fonte: Festo (2003)

3.4 Estação Distribuição

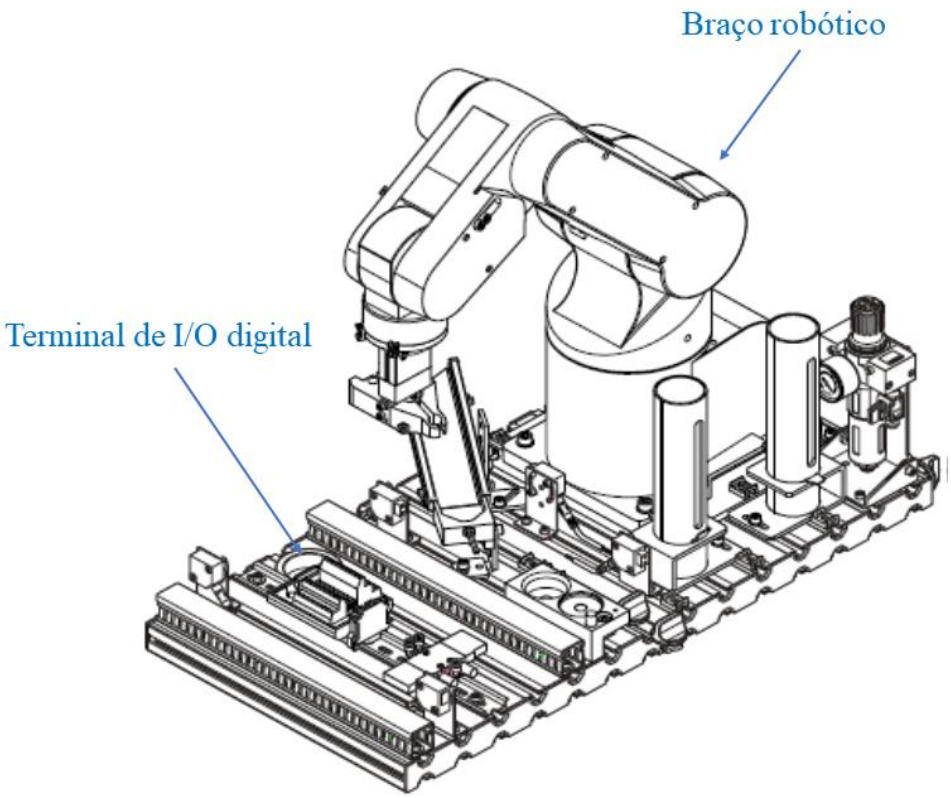
Na estação Distribuição, representada pela Fig. 12, a peça recebida é manipulada pelo braço robótico que a direciona para a próxima estação. O manipulador robótico Mitsubishi RV-2AJ, equipado com uma garra pneumática e sensores optoeletrônicos, é responsável por realizar os movimentos necessários para a distribuição das peças. Trata-se de um robô manipulador de cinco graus de liberdade, com estrutura do tipo angular e oferece capacidade de carga máxima de 2kg. Os componentes que compõem a estação Distribuição, são:

- Braço robótico RV-2AJ;
- Controlador CR1-571, responsável pelo sistema de controle do robô;

- “Teach Pendant”, caixa de controle disponível ao operador, ligada diretamente por um cabo ao controlador do braço robótico.

- Terminal de I/O digital, utilizado como interface entre o controlador e a estação para conexão dos sinais de entradas e saídas digitais.

Figura 12. Representação tridimensional da estação Distribuição.



Fonte: Festo (2003)

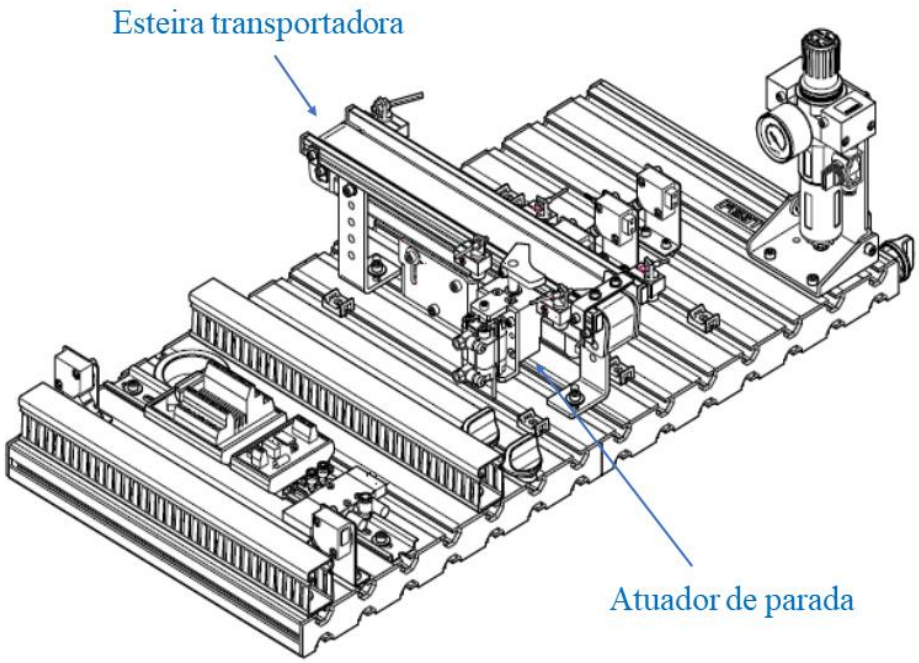
3.5 Estação Espera

Na estação Espera pode-se armazenar até cinco peças, as quais são liberadas pelo atuador de parada, na medida em que a estação subsequente estiver livre, na Fig. 13 é possível verificar a estrutura desse módulo. A estação é composta pelos seguintes componentes:

- Atuador linear pneumático duplo efeito, utilizado para realizar a parada e a liberação de peças na esteira;

- Sensores magnéticos para detecção das posições limite do atuador;
- Válvulas reguladoras de fluxo para controle de velocidade de atuação do cilindro;
- Esteira transportadora, utilizada para movimentar peças;
- Válvula direcional 5/2 vias simples solenoide, utilizada para o comandar o atuador linear.

Figura 13. Representação tridimensional da estação Espera.



Fonte: Festo (2003)

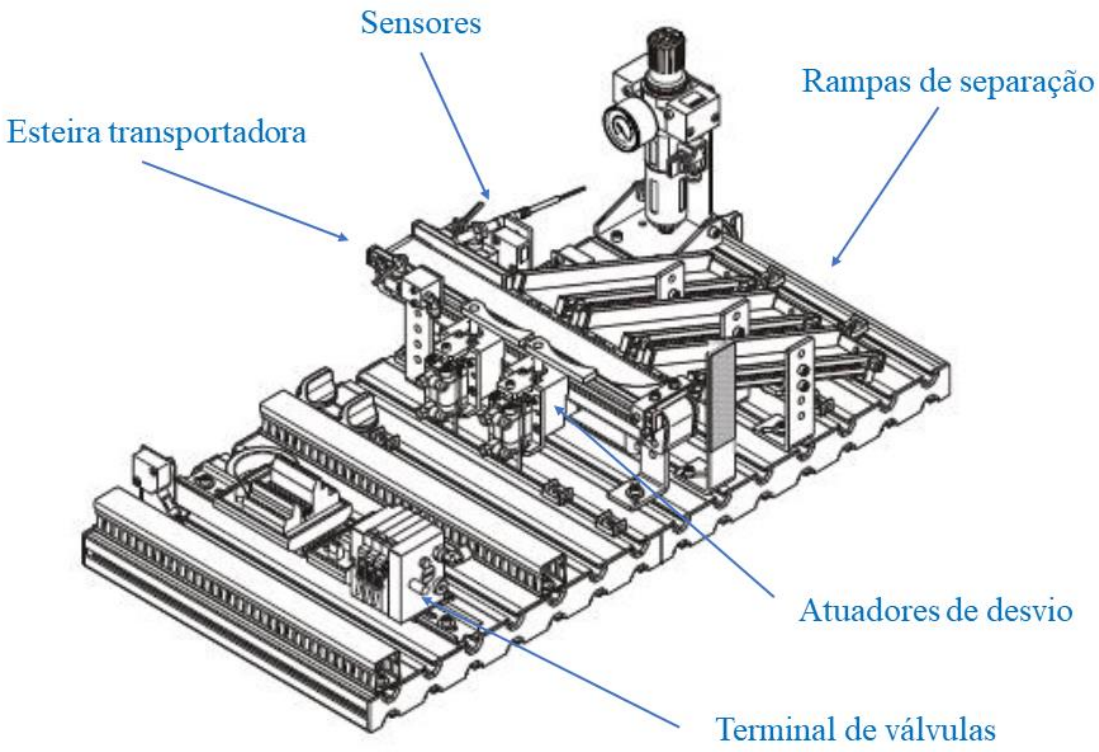
3.6 Estação Separação

Por último, a estação Separação classifica as peças, e as separa por tipo ou por cor. As peças são detectadas no início da esteira e levadas até o módulo de parada, pelo qual é classificada pelos sensores ópticos e indutivo, como mostrado na Fig. 14.

Após a classificação, as peças são separadas nas três rampas, de acordo com os critérios estabelecidos na programação do usuário. A estação é composta pelos seguintes componentes:

- Sensor retro reflexivo, utilizado para detecção de presença de peça quando o feixe de luz é interrompido;
- Sensor indutivo, utilizado para detectar presença de peças metálicas;
- Sensor óptico difuso, utilizado para detecção de peças baseado na reflexão do feixe de luz emitido;
- Dois atuadores lineares pneumáticos de duplo efeito acoplados a um sistema se engrenagens internas com giro de até 90º. Utilizado para realizar o desvio de peças para as rampas.
- Sensores magnéticos para detecção das posições finais dos atuadores.
- Esteira transportadora, utilizada para movimentar peças;
- Terminal de válvulas compacto (CPV) com 2 válvulas direcionais 5/2 vias simples solenoide, utilizadas para o acionamento dos dois desviadores.:

Figura 14. Representação tridimensional da estação Separação.



Fonte: Festo (2003)

Uma abordagem prática para o projeto de sistemas complexos é dividi-los em problemas ou tarefas menores. Os sistemas de automação das fábricas se enquadram naturalmente nos sistemas modulares que consistem em uma estação, máquina ou parte da fábrica e processo. Cada seção pode ser programada separadamente, tendo a mentalidade de integrar todo o sistema em um processo unificado. A mesma filosofia foi seguida neste trabalho, descrita com mais detalhes na seção a seguir.

3.7 Programação

A bancada está equipada com um CLP *Siemens SIMATIC S7- 1500* CPU 1516-3 PN/DP com 32 entradas/32 saídas digitais e 8 entradas analógicas, além de uma *Human Machine Interface* (HMI) Siemens modelo KTP600 Basic Color. Os CLPs e a HMI estão conectados entre si via rede de comunicação industrial PROFINET. A configuração dos dispositivos e programação das operações são executadas com o auxílio do software *Totally Integrated Automation (*TIA Portal V13*)*. Pela Fig. 15 é possível visualizar uma fotografia do painel da bancada didática.

Figura 15. Fotografia do painel da bancada didática.

Fonte: Próprio autor (2021)

A bancada didática está conectada ao CLP por meio do seu cartão de entradas digitais. Todos os sensores operam com 24 VDC como uma tensão padrão e estão conectados ao CLP utilizando as entradas digitais. Os atuadores são controlados de maneira semelhante, pelas saídas digitais do CLP.

Antes de iniciar a programação da planta didática foi feito o levantamento dos sensores e atuadores utilizados, bem como o mapeamento de todas as entradas e saídas digitais conforme apresentado na Tab. 3.1.

Tabela 3.1. Mapa de entradas e saídas digitais do CLP.

Terminal de I/O Entradas digitais (IN)	Descrição	Identificação Tag no CLP	Terminal de I/O Saídas digitais (OUT)	Identificação Tag no CLP	Descrição
DI0.0	Sensor peça disponível no magazine estação Recebimento	SENSOR_PECA_MAG	DO0.0	ATUADOR_MAG_AVANCADO	Avança atuador magazine estação Recebimento
DI0.1	Sensor atuador magazine avançado estação Recebimento	S_AVANC_01	DO0.1	VACUO_LIGADO	Liga vácuo atuador giratório estação Recebimento
DI0.2	Sensor atuador magazine recuado estação Recebimento	S_RECUO_AT01	DO0.3	ATUADOR_GIR_POS_MAG	Move módulo giratório posição magazine estação Recebimento
DI0.3	Sensor atuador giratório posição elevador estação Recebimento	AT_GIR_POS_ELEV	DO0.4	ATUADOR_GIR_POS_ELEV	Move módulo giratório posição elevador estação Recebimento
DI0.4	Sensor atuador giratório posição magazine estação Recebimento	AT_GIR_POS_MAG	DO0.5	RAMPA_AR	Aciona rampa de ar estação Processamento
DI0.5	Sensor atuador de expulsão recuado estação Processamento	SENSOR_RECU_ATUA_MEDICAO	DO0.6	ATUADOR_ELEVADOR_AVANÇADO	Sobe módulo elevador estação Processamento
DI0.6	Sensor atuador de expulsão avançado estação Processamento	SENSOR_AVAN_ATUA_MEDICAO	DO0.7	ATUADOR_MEDICAO_RECUADO	Recua cilindro de expulsão de peças estação Processamento

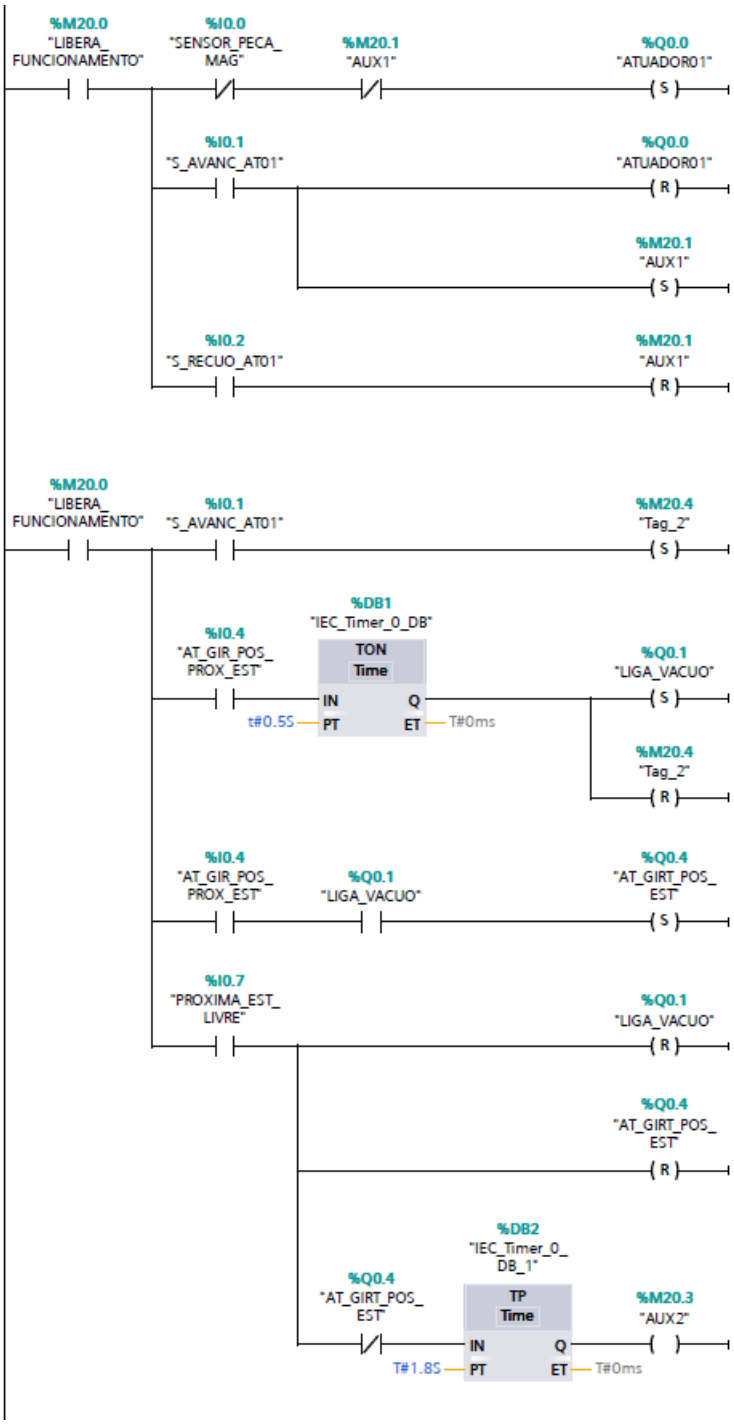
DI0.7	Sensor peça disponível (sensor reflexivo) estação Processamento	SENSOR_PECA_REFLEXIVA_PROCESSAMENTO	DO1.0	ATUADOR_MEDICAO_AVANCADO	Avança cilindro de expulsão de peças estação Processamento
DI1.0	Sensor módulo elevador posição inferior estação Processamento	SENSOR_RECU_ELEVADOR	DO1.3	ATUADOR_VERM_AVANCADO	Avança cilindro separador de peças vermelhas estação Separação
DI1.1	Sensor módulo elevador posição superior estação Processamento	SENSOR_AVAN_ELEVADOR	DO1.4	ATUADOR_VERM_RECUADO	Recua cilindro separador de peças vermelhas estação Separação
DI1.2	Sensor atuador peça metálica recuada estação separação	SENSOR_ATUA_METAL_RECUA	DO1.5	ATUADOR_METAL_AVANCADO	Avança cilindro separador de peças metálicas estação Separação
DI1.3	Sensor atuador peça vermelha avançada estação separação	SENSOR_ATUA_VERM_AVANC	DO1.6	ATUADOR_METAL_RECUADO	Recua cilindro separador de peças metálicas estação Separação
DI1.4	Sensor atuador peça vermelha recuada estação Separação	SENSOR_ATUA_VERM_RECUA	DO1.7	ATUADOR_ESPERA_AVANCADO	Avança atuador estação espera
DI1.5	Sensor atuador peça metálica avançada estação separação	SENSOR_ATUA_METAL_AVANC	DO2.0	ATUADOR_ESPERA_RECUADO	Recua atuador estação espera
DI1.7	Sensor indutivo peça metálica estação separação	SENSOR_INDUTIVO_SEP	DO2.1	ESTEIRA_ESPERA_LIGADA	Aciona esteira estação Espera
DI2.0	Sensor capacitivo peça vermelha estação separação	SENSOR_CAPACITIVO_SEP	DO2.2	ESTEIRA_SEPARACAO_DESLIGADA	Desliga esteira estação Espera
DI2.1	Sensor rampa de peças estação separação	SENSOR_RAMPA_SEP			
DI2.2	Sensor peça não reflexiva estação processamento	SENSOR_PECA_N_REFLEXIVA_PROCESSAMENTO			
DI2.3	Sensor do vácuo estação recebimento	SENSOR_VACUO			

Fonte: Próprio autor (2021)

Após o levantamento dos sensores e atuadores utilizados e mapeamento de todas as entradas e saídas digitais partiu-se para a programação dos CPL e da HMI, como evidenciado nas Fig. 16, 17, 18 e 19 utilizando o software *TIA Portal V13*, seguindo os critérios e sequências estabelecidas pela planta didática.

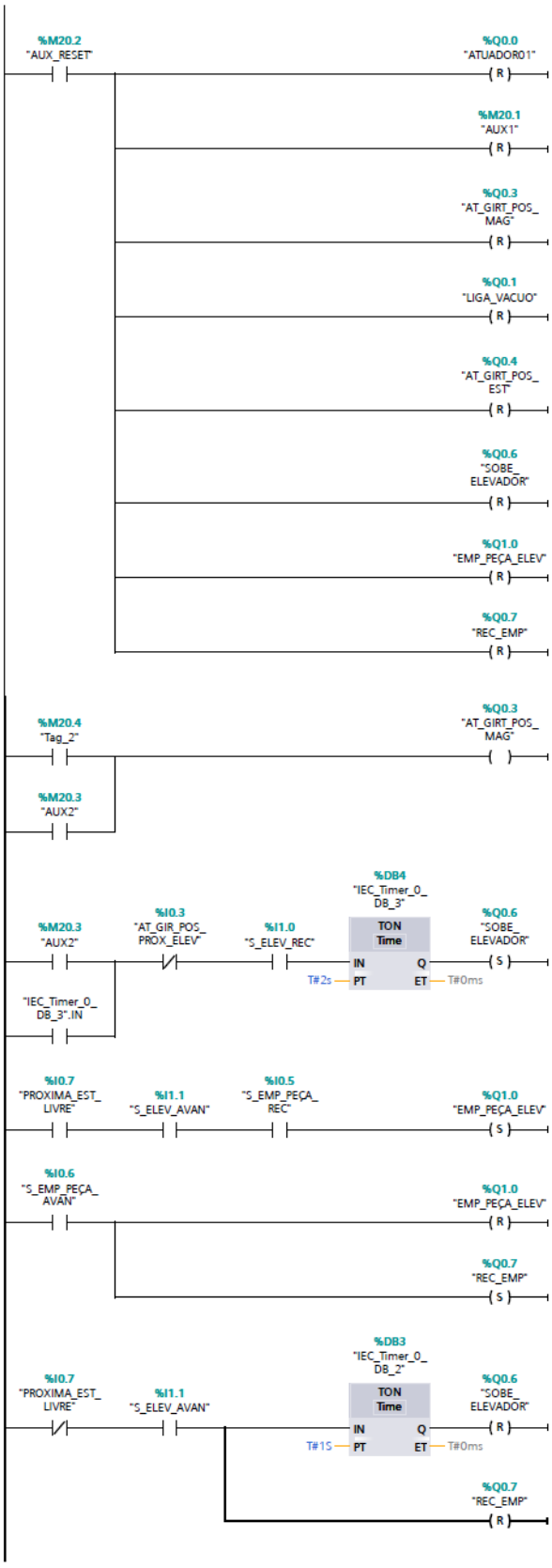
Para a programação da planta didática de manufatura robótica, foi adotada a linguagem *Ladder*, que é uma das linguagens definidas pela norma IEC 61131-3. Essa linguagem é uma ferramenta gráfica utilizada para desenvolver programas ou softwares para CLPs. Ela é responsável pela lógica de controle, indicando ao controlador quais ações devem ser realizadas com base nos valores de entrada.

Figura 16. Programação em *Ladder* CLP – Parte 1.



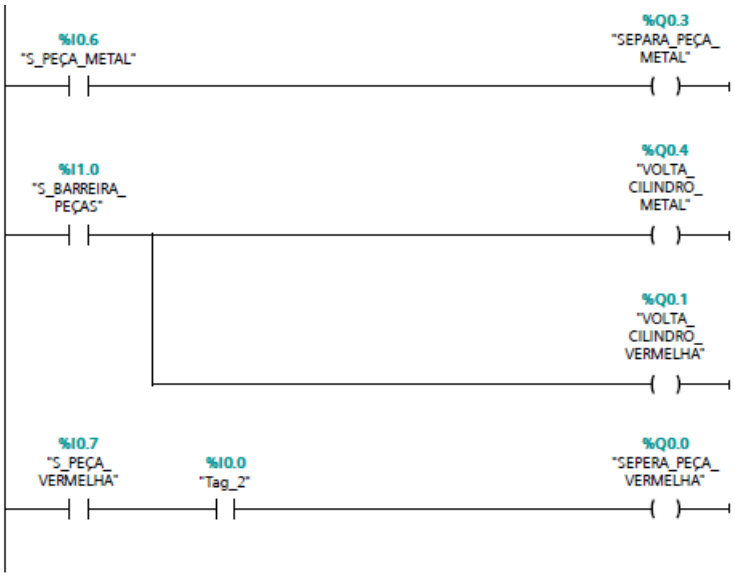
Fonte: TIA Portal V13 (2021)

Figura 17. Programação em Ladder CLP – Parte 2.



Fonte: TIA Portal V13 (2021)

Figura 18. Programação em Ladder CLP – Parte 3.



Fonte: TIA Portal V13 (2021)

O CLP se comunica com uma HMI Siemens KTP600, os equipamentos foram conectados utilizando uma rede Ethernet com protocolo de comunicação *PROFINET*. As memórias M20.0 e M20.2 mostradas na programação em *Ladder* na Fig. 16 (M20.0) e Fig. 17 (M20.2) foram utilizadas para liberar o funcionamento e para reset das condições iniciais, respectivamente. A programação da HMI é feita no mesmo software, e sua tela preliminar é apresentada na Fig. 19.

Figura 19. Tela HMI Planta Didática.

Fonte: TIA Portal V13 (2021)

4. MODELAGEM DO DOMÍNIO

Diante dos vários tipos de contextos e problemas que possa se apresentar em um ambiente industrial ou até situações cotidianas, existem vários tipos de ações e em consequência vários tipos de planejamento. Ghallab *et al.* (2004), cita algumas dessas formas de planejamento, que são: planejamento de trajetória e movimento, planejamento de percepção, planejamento de navegação e planejamento de manipulação em que o objetivo é a manipulação de objetos, cujas ações envolvam força, torque, sensoriamento e alcance em ações como pegar uma peça e depositá-la em locais predeterminados. O objetivo deste trabalho está justamente no planejamento da manipulação, sendo que a proposta consiste em um sistema de distribuição de produtos que não necessite de interferência humana.

Neste trabalho a modelagem do domínio foi separada por estações para facilitar a solução do problema proposto para cada estação. Utilizou-se como editor o "PDDL Editor" – "software online" [1] gratuito que apresenta variedade de ferramentas e recursos para escrever e testar domínios e arquivos de problemas.

O "PDDL Editor" faz parte de um conjunto de softwares denominado "Planning.Domains" (PD). A iniciativa PD foi desenvolvida no ICAPS-2015 em Jerusalém, Israel. Desde a sua criação foram envolvidas mais de sete instituições, bem como apoio financeiro da "Industrial Cyber-Physical Systems" (ICPS), uma entidade comprometida com o avanço das tecnologias de sistemas ciber-físicos industriais, fortalecendo assim o impacto e a relevância do PD na indústria moderna. O objetivo da iniciativa PD é ser um recurso criado por pesquisadores de planejamento, e também, um caminho para alcançar outras comunidades não familiarizadas com a tecnologia de planejamento (Muise, 2016).

O editor online de PDDL é semelhante a outras ferramentas de modelagem de domínios como o "itSIMPLE" (Vaquero *et al.,* 2009), o "PDDL Studio" (Plch *et al*., 2012) e o editor básico online "myPDDL" (Strobel 2015). O "PDDL Editor" se baseia no editor "myPDDL" para unir as outras iniciativas de PD e expandir os recursos encontrados no "PDDL Studio" e no "itSIMPLE". A escolha da ferramenta "PDDL Editor" utilizada neste trabalho deu-se durante a revisão da literatura e levantamento do estado da arte, quando obteve-se acesso a um

[1] http://editor.planning.domains/#

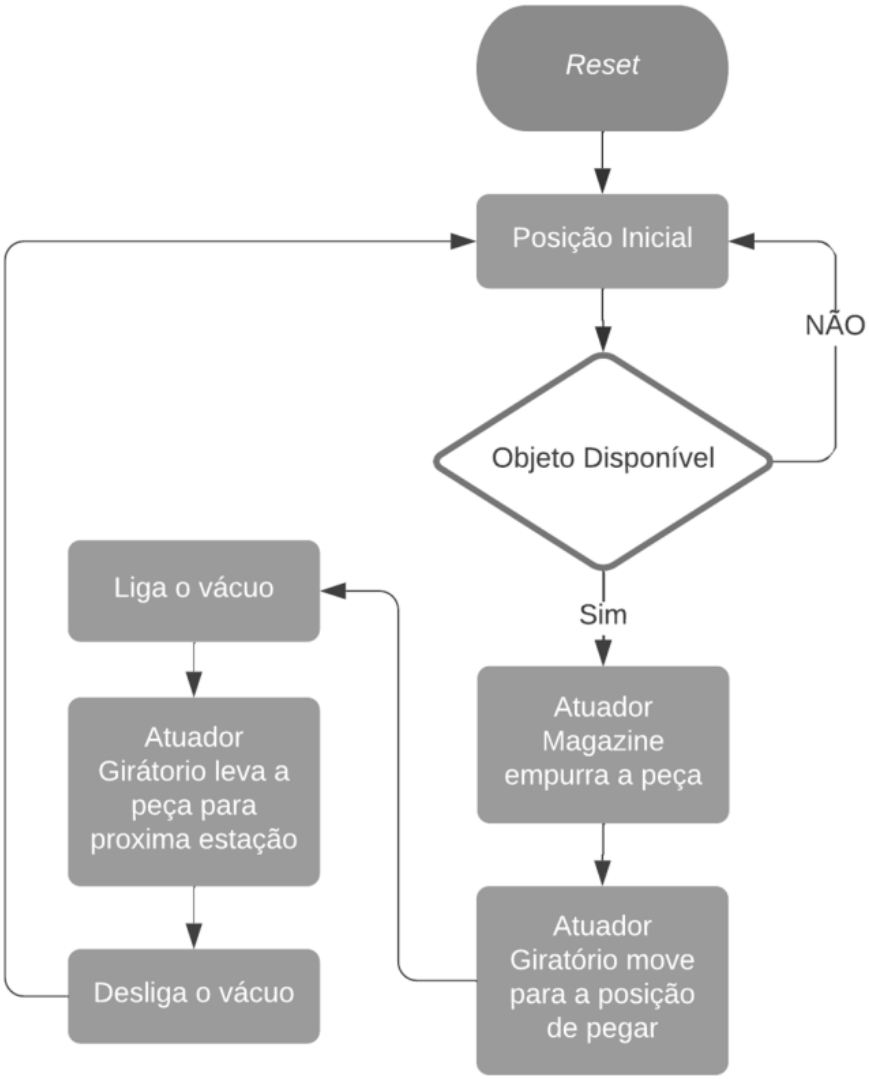
rico material disponibilizado de forma online pelo curso de verão: “Plan Synthesis” durante o *30th International Conference on Automated Planning and Scheduling* (ICAPS 2020). O treinamento é uma introdução à modelagem de problemas de planejamento usando a linguagem PDDL.

O planejador que avalia o domínio em busca de planos solução para os problemas é denominado “Solver”. O componente “Solver” do “Planning.Domains” é um planejador e validador automatizado na nuvem. Nele é possível invocar o software enviando links para os arquivos PDDL ou enviando conteúdo PDDL bruto no formato “JSON” diretamente para recuperar ou validar um plano. Atualmente o planejador está limitado a aproximadamente 500Mb de RAM e tem um limite de tempo de 10 segundos, mas o projeto é de código aberto e gratuito para que novos pesquisadores possam implantar seus próprios projetos usando diferentes limites de recursos.

4.1 Modelagem da estação recebimento

Antes da modelagem dos domínios de planejamento, elaborou-se um fluxograma do processo de manipulação das peças de trabalho para cada estação presente na bancada didática. Os referidos fluxogramas descrevem o funcionamento, de maneira facilitada, a realização das sequencias e transições dessa estação. A linguagem PDDL, para o correto funcionamento dos algoritmos de planejamento, aplica o conceito de requerimentos ou *requeriments*. Os requerimentos traduzem os requisitos que são necessários para avaliação do domínio de planejamento. O PDDL utiliza uma notação baseada em lógica de predicados com variáveis que em seguida serão instanciadas com objetos do problema. A Fig. 20 apresenta o fluxograma idealizado para a estação Recebimento, enquanto a Fig. 21 mostra a descrição em PDDL do domínio Estação Recebimento dado pelo conjunto de predicados e pelo conjunto de ações.

Figura 20. Fluxograma da estação Recebimento.



Fonte: Próprio autor (2021)

Figura 21. Descrição do domínio da estação Recebimento.

```
(define (domain changeover)
    (:requirements :strips :typing)

    (:types
            location atua_gir part - object
            storage input waiting output - location
    )

    (:predicates
        (connected ?from ?to - location)

        (in ?p - part ?l - location)
        (at ?a - atua_gir ?l - location)
        (landing ?a - atua_gir ?p - part)
    )

        (:action move
        :parameters (?a - atua_gir ?from - location ?to - location)
        :precondition (and
            (at ?a ?from)
        )
        :effect (and
            (not (at ?a ?from))
            (at ?a ?to)
        )
    )

    (:action push
        :parameters (?p - part ?e - storage ?i - input)
        :precondition (and
            (in ?p ?e)
        )
        :effect (and
            (not (in ?p ?e))
            (in ?p ?i)
        )
    )

    (:action catch
        :parameters (?a - atua_gir ?w - waiting ?i - input)
        :precondition (and
            (at ?a ?w)
        )
        :effect (and
            (not (at ?a ?w))
            (at ?a ?i)
        )
    )

    (:action landing
        :parameters (?a - atua_gir ?p - part ?i - input)
        :precondition (and
            (at ?a ?i)
            (in ?p ?i)
        )
        :effect (and
            (not (at ?a ?i))
            (not (in ?p ?i))
            (landing ?a ?p)
        )
    )

    (:action release
        :parameters (?a - atua_gir ?p - part ?w - waiting ?o - output)
        :precondition (and
            (landing ?a ?p)
        )
        :effect (and
            (not (landing ?a ?p))
            (in ?p ?o)
            (at ?a ?o)
        )
    )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Os predicados que detalham as características do ambiente são especificados em (:predicates), que são:

- (location ?l), predicado que indica que o objeto ?l é um local o qual ainda pode ser instanciado via herança como um local de input ou de output;
- (atuadorgir ?a), predicado que indica que o objeto ?a é um atuador giratório;
- (part ?p), predicado que indica que o objeto ?p é uma peça.

O predicado “object" é um tipo genérico que pode ser usado para qualquer objeto no domínio. Ele é usado para indicar que o “object" é um objeto em geral, sem uma classificação específica como “atuadorgir” ou “part”.

Cada ação do domínio é dada por meio de parâmetros, precondições e efeitos. No domínio da estação Recebimento observam-se as ações:

- Move: por meio da qual o atuador giratório está se movimentando;
- Push: onde o atuador linear do magazine empurra a peça quando a mesma está disponível no início da estação.
- Landing: indica que o atuador giratório está movimentando a peça para a próxima estação;
- Catch: mediante a qual o atuador giratório pega a peça no magazine;
- Release: na qual o atuador giratório deixa a peça na estação seguinte;

As precondições de uma ação (indicada na lista *:precondition*) são o conjunto de proposições que precisam ser verdade em um estado para que a ação possa ser executada nesse estado. Se uma ação pode ser executada em um determinado estado, diz que essa ação é aplicável a esse estado. Na estação Recebimento observa-se a precondições de cada ação:

- Ação Push: a precondição é que o objeto ?p esteja no local ?e de armazenamento indicado como “storage” de peças disponíveis;

- Ação Move: onde sua precondição é que o atuador giratório ?a esteja em algum local para que ele se movimente.

- Ação Catch: sua precondição nessa ação é que o atuador giratório ?a esteja no posição de espera ?w.

- Ação Landing: a precondição é que o atuador giratório ?a esteja no posição do magazine ?i e a peça ?p também esteja disponível no magazine ?i.

- Ação Release: a precondição para que esta ação ocorra é que o atuador giratório ?a esteja movimentando a peça ?p e o atuador giratório ?a esteja no local ?o que é a próxima estação.

Após o detalhamento das ações, são descritos os seus respectivos efeitos, ou proposições que serão modificadas após a aplicação da ação. Esses efeitos podem ser positivos, quando a proposição terá um valor verdadeiro no estado anterior à aplicação da ação, ou negativos, quando a proposição terá um valor falso no estado anterior à aplicação da ação.

Elaborado o modelo do domínio, é feita a descrição do problema de planejamento. A Fig. 22 descreve o problema de planejamento do domínio da estação Recebimento em PDDL.

Figura 22. Descrição do problema no domínio da estação Recebimento.

```
(define (problem problem_changeover)
(:domain changeover)
(:requirements :strips :typing)

    (:objects
        magazine - storage
        this_Station - input
        waiting_point - waiting
        next_Station - output
        Ag1 - atua_gir
        pack1 - part
    )

    (:init
        (at Ag1 waiting_point)
        (in pack1 magazine)

        ;; Ground Connections
        (connected magazine this_Station)
        (connected this_Station waiting_point)
        (connected waiting_point next_Station)
    )

    (:goal (and
        (in pack1 next_Station)
        (at Ag1 waiting_point)
    ))
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Foi utilizado o termo (:objects) para identificar os objetos que são instanciados no problema, o termo (:init) são proposições verdadeiras e os tipos de objetos no estado inicial, e o termo (:goal) são proposições para definir a meta do planejamento. É possível observar o estado inicial, onde:

- (at Ag1 waiting_point), indica que o atuador giratório deve estar na posição de espera;
- (in pack1 magazine), indica que o pacote deve estar no magazine;

A meta do planejamento é:

- (in pack1 next_Station), indica que o pacote estará na próxima estação;
- (at ag1 waiting_point), indica que o atuador giratório estará na posição de espera.

No problema de planejamento do domínio da estação Recebimento, após inserir os dois arquivos, o arquivo domínio e o problema, o editor online encontrou um plano solução, que é a sequência de ações apresentada na Fig. 23.

Figura 23. Plano solução do Domínio da estação Recebimento.

```
(push pack1 magazine this_station)
    (:action push
    :parameters (pack1 magazine this_station)
    :precondition
      (and
        (in pack1 magazine)
      )
    :effect
      (and
        (not
          (in pack1 magazine)
        )
        (in pack1 this_station)
      )
  )

(catch ag1 waiting_point this_station)
    (:action catch
    :parameters (ag1 waiting_point this_station)
    :precondition
      (and
        (at ag1 waiting_point)
      )
    :effect
      (and
        (not
          (at ag1 waiting_point)
        )
        (at ag1 this_station)
      )
  )

(landing ag1 pack1 this_station)
    (:action landing
    :parameters (ag1 pack1 this_station)
    :precondition
      (and
        (at ag1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (at ag1 this_station)
        )
        (not
          (in pack1 this_station)
        )
        (landing ag1 pack1)
      )
  )

(release ag1 pack1 waiting_point next_station)
    (:action release
    :parameters (ag1 pack1 waiting_point next_stat
    :precondition
      (and
        (landing ag1 pack1)
      )
    :effect
      (and
        (not
          (landing ag1 pack1)
        )
        (in pack1 next_station)
        (at ag1 next_station)
      )
  )

(move ag1 next_station waiting_point)
    (:action move
    :parameters (ag1 next_station waiting_point)
    :precondition
      (and
        (at ag1 next_station)
      )
    :effect
      (and
        (not
          (at ag1 next_station)
        )
        (at ag1 waiting_point)
      )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Segundo Ghallab *et al*. (2004), um plano é uma sequência de ações $\pi = (a1, ..., aj)$, onde $j \geq 0$. No plano solução gerado, foram encontradas cinco ações para domínio da estação Recebimento, que são:

- (push pack1 magazine this_station), onde a primeira ação é para que o atuador do magazine empurre a peça para a posição de peça disponível, sendo que a precondição para que esta ação ocorra é que haja peças no magazine;

- (catch ag1 waiting_point this_station), a segunda ação encontrada é para que o atuador giratório na posição de espera pegue a peça disponível na estação. A precondição dessa ação é que o atuador giratório esteja na posição de espera e tenha uma peça disponível;

- (landing ag1 pack1 this_station), terceira ação por meio da qual o atuador giratório move o objeto que estava na posição "magazine" para a posição "próxima estação", onde a precondição desta ação é que o atuador giratório esteja na posição do magazine;

- (release ag1 pack1 waiting_point next_station), na quarta ação o atuador giratório deixará o objeto disponível na próxima estação. A precondição da ação é que o atuador giratório esteja movimentando o objeto e chegue na posição "próxima estação";

- (move ag1 next_station waiting_point), a quinta e última ação encontrada é para que o atuador giratório retorne para a posição de espera, para que o ciclo possa recomeçar assim que o objeto estiver disponível novamente no início da estação, onde a precondição dessa ação é o atuador giratório estar na posição "próxima estação".

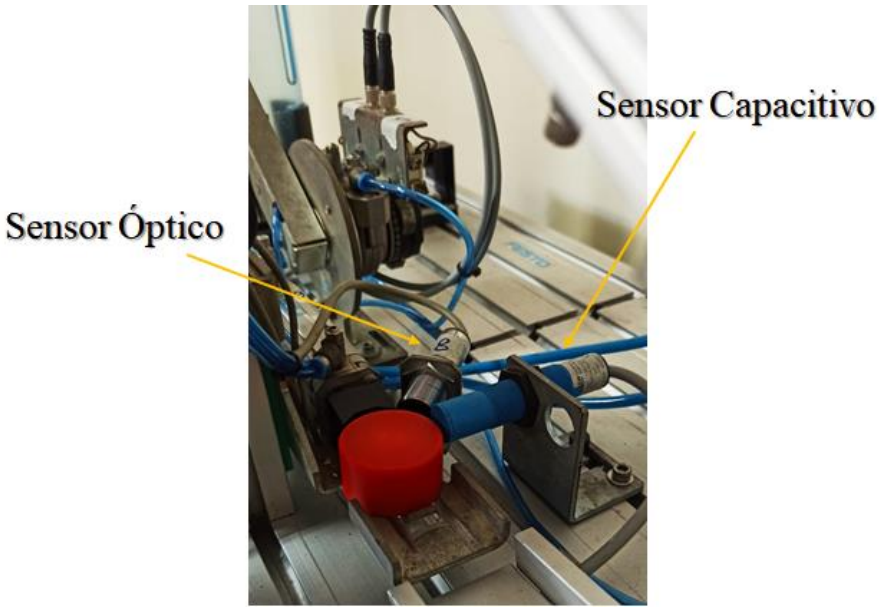
4.2 Modelagem da estação Processamento

Após a peça ser posicionada pela estação Recebimento, seu material e sua cor são determinados com a ajuda de dois sensores de proximidade: um capacitivo e um óptico, como mostrado na Fig. 24.

- Sensor óptico detecta as peças vermelhas e metálicas;

- Sensor capacitivo detecta a presença de qualquer peça.

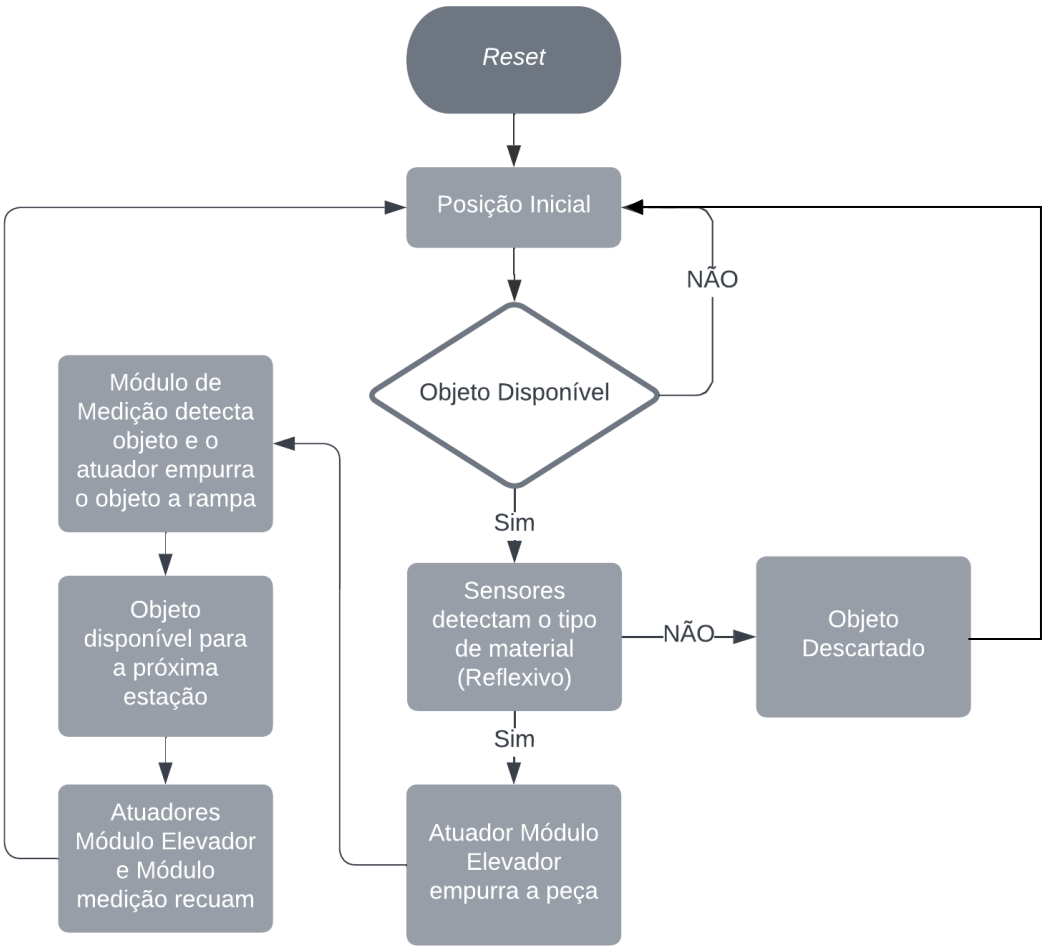
Figura 24. Sensores para identificação de peças.



Fonte: Próprio autor (2021)

A peça, após ser testada, é então elevada por um atuador pneumático, denominado elevador, responsável por movimentar a peça até o módulo de medição e direcionar a peça para a rampa até a próxima estação. Na Fig. 25 é apresentado o fluxograma da estação Processamento.

Figura 25. Fluxograma da estação Processamento.



Fonte: Próprio autor (2021)

Após a elaboração do fluxograma, elaborou-se o domínio da estação Processamento. A Fig. 26 mostra a descrição em PDDL do domínio estação Processamento dado pelo conjunto de predicados e pelo conjunto de ações.

Figura 26. Descrição do domínio da estação Processamento.

```
(define (domain processing)
    (:requirements :strips :typing)

    (:types
            location atuadores part - object
            separation input measurement output - location
            atua_Ramp atua_Elev - atuadores
    )

    (:predicates
        (connected ?from ?to - location)

        (in ?p - part ?l - location)
        (at ?a - atuadores ?l - location)
    )

    (:action verification
        :parameters (?p - part ?s - separation ?i - input)
        :precondition (and
            (in ?p ?s)
        )
        :effect (and
            (not (in ?p ?s))
            (in ?p ?i)
        )
    )

    (:action move
        :parameters (?a - atuadores ?from ?to - location)
        :precondition (and
            (at ?a ?from)
        )
        :effect (and
            (not (at ?a ?from))
            (at ?a ?to)
        )
    )

    (:action raise
        :parameters (?e - atua_Elev ?p - part ?i - input ?m - measurement)
        :precondition (and
            (at ?e ?i)
            (in ?p ?i)
        )
        :effect (and
            (not (in ?p ?i))
            (not (at ?e ?i))
            (in ?p ?m)
            (at ?e ?m)
        )
    )

    (:action push
        :parameters (?r - atua_Ramp ?p - part ?o - output ?m - measurement)
        :precondition (and
            (at ?r ?m)
            (in ?p ?m)
        )
        :effect (and
            (not (at ?r ?m))
            (not (in ?p ?m))
            (at ?r ?o)
            (in ?p ?o)
        )
    )

    (:action return
        :parameters (?e - atua_Elev ?r - atua_Ramp ?o - output ?m - measurement ?i - input ?p - part)
        :precondition (and
            (at ?r ?o)
            (at ?e ?m)
            (in ?p ?o)
        )
        :effect (and
            (not (at ?r ?o))
            (not (at ?e ?m))
            (not (in ?p ?o))
            (at ?r ?m)
            (at ?e ?i)
        )
    )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Os predicados que detalham as características do domínio são:

- (location ?l), predicado que indica que o objeto ?l é um local o qual ainda pode ser instanciado via herança como um local de separation, input, measurement e output;
- (atuadores ?a), predicado que indica que o objeto ?a são atuadores o qual ainda pode ser instanciado via herança como um atuador da rampa e atuador elevatório;
- (part ?p), predicado que indica que o objeto ?p é uma peça.

No domínio da estação Processamento observam-se as ações:

- Verification: onde os sensores detectam a presença da peça e liberam para a próxima ação.
- Move: por meio da qual os atuadores dessa estação movimentam a peça quando esta estiver disponível em cada módulo;
- Raise: por meio da qual o atuador linear eleva a peça quando esta estiver disponível;
- Push: onde o atuador linear do módulo de medição empurra a peça para a rampa de deslizamento onde mesma estará disponível na próxima estação.
- Return: onde o atuador linear do módulo de medição e o atuador linear de elevação retornam à sua posição inicial

Na estação Processamento observa-se a precondições das ações:

- Ação Verification: a precondição é que a peça ?p esteja disponível no local ?s de separação;
- Ação Move: onde sua precondição é que o atuador giratório ?a esteja em algum local para que este se movimente.
- Ação Raise: a precondição para que esta ação ocorra é que o atuador elevatório ?e esteja no local de entrada ?i e a peça ?p também esteja nesse local ?i;

- Ação Push: a precondição para que esta ação ocorra é que o atuador da rampa ?r e a peça ?p estejam no local de verificação ?m.

- Ação Return: a precondição para que esta ação ocorra é que o atuador da rampa ?r esteja na posição avançado ?o e o atuador elevatório ?e esteja na posição de verificação ?m.

Feito o modelo do domínio da estação Processamento, realizou-se a descrição do problema de planejamento. A Fig. 27 descreve o problema de planejamento do domínio da estação Processamento em PDDL.

Figura 27. Descrição do problema no domínio da estação Processamento.

```
(define (problem problem_processing)
(:domain processing)
(:requirements :strips :typing)

    (:objects
        modulo_sep - separation
        modulo_med - measurement
        this_Station - input
        next_Station - output
        Ae1 - atua_Elev
        Ar1 - atua_Ramp
        pack1 - part
    )

    (:init
        (at Ae1 this_Station)
        (at Ar1 modulo_med)
        (in pack1 modulo_sep)

        ;; Ground Connections
        (connected modulo_sep this_Station)
        (connected this_Station modulo_med)
        (connected modulo_med next_Station)
    )

    (:goal (and
        (in pack1 next_Station)
        (at Ae1 this_Station)
        (at Ar1 modulo_med)
    ))
)
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No problema da estação Processamento é possível observar o estado inicial, onde:

- (in pack1 modulo_sep), indica que o pacote está no modulo de separação;
- (at Ae1 this_Station), indica que o atuador elevatório está na posição de entrada da estação;
- (at Ar1 mod_med), indica que o atuador da rampa está na posição recuado no módulo de medição;

A meta do planejamento para este problema é:

- (in pack1 next_Station), indica que o pacote estará na próxima estação;
- (at Ae1 this_Station), indica que o atuador elevatório estará de volta na posição de entrada da estação;
- (at Ar1 mod_med), indica que o atuador da rampa estará de volta na posição recuado no módulo de medição;

Após inserir os dois arquivos, o arquivo domínio e o problema da estação Processamento, o editor online encontrou um plano solução, que é a sequência de ações demonstrada na Fig. 28.

Figura 28. Plano solução do domínio da estação Processamento.

```
(verification pack1 modulo_sep this_station)
    (:action verification
    :parameters (pack1 modulo_sep this_station)
    :precondition
      (and
        (in pack1 modulo_sep)
      )
    :effect
      (and
        (not
          (in pack1 modulo_sep)
        )
        (in pack1 this_station)
      )
  )

(raise ae1 pack1 this_station modulo_med)
    (:action raise
    :parameters (ae1 pack1 this_station modulo_med)
    :precondition
      (and
        (at ae1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (not
          (at ae1 this_station)
        )
        (in pack1 modulo_med)
        (at ae1 modulo_med)
      )
  )

(push ar1 pack1 modulo_med next_station)
    (:action push
    :parameters (ar1 pack1 next_station modulo_med)
    :precondition
      (and
        (at ar1 modulo_med)
        (in pack1 modulo_med)
      )
    :effect
      (and
        (not
          (at ar1 modulo_med)
        )
        (not
          (in pack1 modulo_med)
        )
        (in pack1 next_station)
        (at ar1 next_station)
      )
  )

(move ar1 next_station modulo_med)
    (:action move
    :parameters (ar1 next_station modulo_med)
    :precondition
      (and
        (at ar1 next_station)
      )
    :effect
      (and
        (not
          (at ar1 next_station)
        )
        (at ar1 modulo_med)
      )
  )

(move ae1 modulo_med this_station)
    (:action move
    :parameters (ae1 modulo_med this_station)
    :precondition
      (and
        (at ae1 modulo_med)
      )
    :effect
      (and
        (not
          (at ae1 modulo_med)
        )
        (at ae1 this_station)
      )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No plano solução gerado, foram encontradas cinco ações para domínio da estação Processamento, que são:

- (verification pack1 modulo_sep this_station), onde a primeira ação é para que o objeto seja verificado e separado se necessário, sendo que a precondição para que esta ação ocorra é que o objeto esteja disponível no modulo de separação;

- (raise ae1 pack1 this_station), a segunda ação encontrada é para que o atuador linear eleve o objeto que está na entrada da estação. A precondição dessa ação é que o atuador elevatório esteja na posição de entrada da estação e tenha uma peça disponível;

- (push ar1 pack1 modulo_med next_station), a terceira ação encontrada é para que o atuador da rampa empurre o objeto que está no modulo de medição para a rampa onde ele estará disponível na próxima estação, onde o precondição dessa ação é o atuador da rampa e o objeto estejam na posição “módulo de medição”.

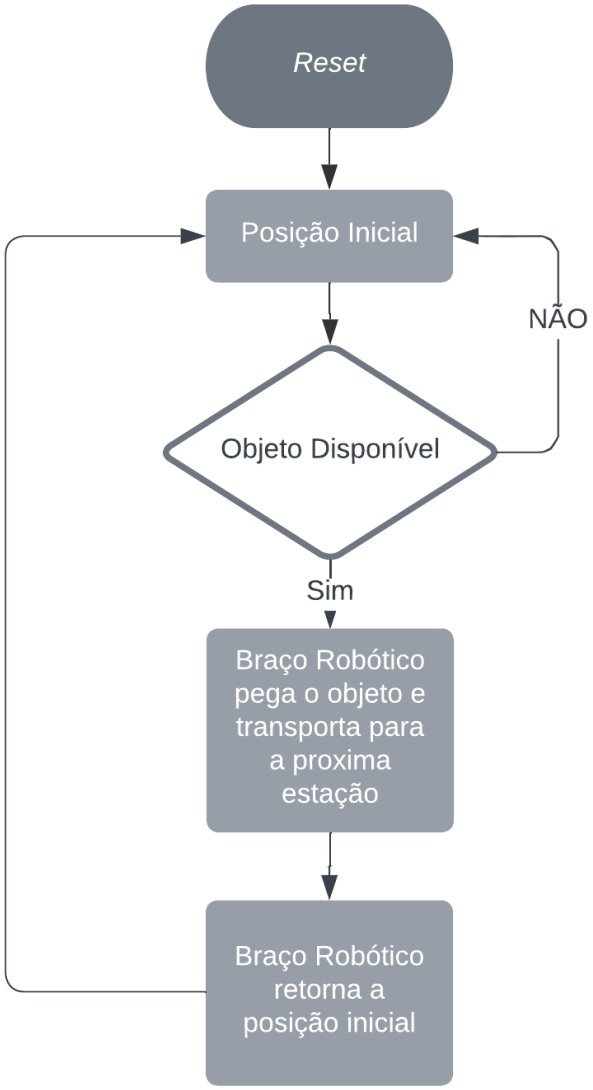
- (move ar1 next_station modulo_med), quarta ação por meio da qual o atuador de medição retorne para a posição “modulo de medição”, onde a precondição desta ação é que o atuador de medição esteja na posição “output”;

- (move ae1 modulo_med this_station), na quinta e última ação o atuador elevatório retornará a sua posição inicial. A precondição da ação é que o atuador elevatório esteja na posição “módulo de medição”;

4.3 Modelagem da estação Distribuição

A estação do manipulador robótico, neste trabalho realizará a manipulação e deslocamento da peça para estação seguinte. Nos processos de manufatura, os robôs industriais aprimoram as atividades de aplicações de solda, corte, movimentação de peças, entre muitas outras. Isso melhora a qualidade dos produtos e, ao mesmo tempo, reduz os custos de operação, principalmente a médio e longo prazo. Antes da modelagem, elaborou-se o fluxograma da estação Distribuição que é apresentado na Fig. 29.

Figura 29. Fluxograma da estação Distribuição.



Fonte: Próprio autor (2021)

Seguindo com a modelagem, após a elaboração do fluxograma do processo formalizou-se o domínio da estação Distribuição. A Fig. 30 mostra a descrição em PDDL do domínio da estação Distribuição com suas características.

Figura 30. Descrição do domínio da estação Distribuição.

```
(define (domain distribution)
    (:requirements :strips :typing :negative-precondition

    (:types
            location robot part - object
            input wait output - location
    )

    (:predicates
        (connected ?from ?to - location)

        (in ?p - part ?l - location)
        (on ?p - part ?l - robot)
        (at ?r - robot ?l - location)
    )


    (:action move
        :parameters (?r - robot ?from ?to - location)
        :precondition (and
            (at ?r ?from)
        )
        :effect (and
            (not (at ?r ?from))
            (at ?r ?to)
        )
    )

    (:action catch
        :parameters (?r - robot ?p - part ?i - input)
        :precondition (and
            (not (on ?p ?r))
            (at ?r ?i)
            (in ?p ?i)
        )
        :effect (and
            (not (in ?p ?i))
            (on ?p ?r)
        )
    )

    (:action release
        :parameters (?r - robot ?p - part ?o - output)
        :precondition (and
            (at ?r ?o)
            (on ?p ?r)
        )
        :effect (and
            (in ?p ?o)
            (not (on ?p ?r))
        )
    )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Os predicados que detalham as características do domínio são:

- (location ?l), predicado que indica que o objeto ?l é um local o qual ainda pode ser instanciado via herança como um local de input, wait ou de output;
- (robot ?r), predicado que indica que o objeto ?r é um manipulador robótico;
- (part ?p), predicado que indica que o objeto ?p é uma peça.

No domínio da estação Distribuição observam-se seguintes as ações:

- Move: por meio da qual o manipulador robótico movimenta a peça quando esta estiver disponível;
- Catch: mediante a qual o manipulador robótico pega a peça na estrada da estação;
- Release: na qual o manipulador robótico deixa a peça na próxima estação;

Na estação Distribuição observa-se a precondições das ações:

- Ação move: onde sua precondição é que o manipulador robótico ?r esteja em algum local para que este se movimente de um local para outro.
- Ação catch: a precondição nessa ação é que o manipulador robótico ?r esteja no posição de entrada ?i e a peça ?p também esteja disponível na entrada ?i da estação;
- Ação release: sua precondição é que o manipulador robótico ?r esteja movimentando a peça ?p e o manipulador robótico ?r esteja na posição de saída ?o.

Modelado o domínio da estação Distribuição, a descrição do problema de planejamento foi elaborada. A Fig. 31 descreve o problema de planejamento do Domínio da estação Distribuição em PDDL.

Figura 31. Descrição do problema no domínio da estação Distribuição.

```
(define (problem problem_distribution)
(:domain distribution)
(:requirements :strips :typing)

    (:objects
        this_Station - input
        next_Station - output
        wait_point - wait
        Rb1 - robot
        pack1 - part
    )

    (:init
        (at Rb1 wait_point)
        (in pack1 this_Station)

        ;; Ground Connections
        (connected wait_point this_Station)
        (connected wait_point next_Station)
        (connected this_Station next_Station)
        (connected this_Station wait_point)
        (connected next_Station wait_point)
        (connected next_Station this_Station)
    )

    (:goal (and
        (in pack1 next_Station)
        (at Rb1 wait_point)
    ))
)
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No problema da estação Distribuição é possível observar o estado inicial, onde:

- (at Rb1 wait_point), indica que o manipulador está na posição de espera da estação;
- (in pack1 this_Station), indica que o pacote está na entrada da estação.

A meta do planejamento para este problema é:

- (in pack1 next_Station), indica que o pacote estará na próxima estação;
- (at Rb1 wait_point), indica que o manipulador estará na posição de espera da estação.

O plano solução do domínio da estação Distribuição foi encontrado pela sequência de ações demonstrada na Fig. 32.

Figura 32. Plano solução do domínio da estação Distribuição.

```
(move rb1 wait_point this_station)
    (:action move
    :parameters (rb1 wait_point this_station)
    :precondition
      (and
        (at rb1 wait_point)
      )
    :effect
      (and
        (not
          (at rb1 wait_point)
        )
        (at rb1 this_station)
      )
  )

(catch rb1 pack1 this_station)
    (:action catch
    :parameters (rb1 pack1 this_station)
    :precondition
      (and
        (not
          (on pack1 rb1)
        )
        (at rb1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (on pack1 rb1)
      )
  )

(move rb1 this_station next_station)
    (:action move
    :parameters (rb1 this_station next_station)
    :precondition
      (and
        (at rb1 this_station)
      )
    :effect
      (and
        (not
          (at rb1 this_station)
        )
        (at rb1 next_station)
      )
  )

(release rb1 pack1 next_station)
    (:action release
    :parameters (rb1 pack1 next_station)
    :precondition
      (and
        (at rb1 next_station)
        (on pack1 rb1)
      )
    :effect
      (and
        (in pack1 next_station)
        (not
          (on pack1 rb1)
        )
      )
  )

(move rb1 next_station wait_point)
    (:action move
    :parameters (rb1 next_station wait_point)
    :precondition
      (and
        (at rb1 next_station)
      )
    :effect
      (and
        (not
          (at rb1 next_station)
        )
        (at rb1 wait_point)
      )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No plano solução gerado, foram encontradas cinco ações para domínio da estação Distribuição, que são:

- (move rb1 wait_point this_station), onde a primeira ação é para que o manipulador robótico se movimente do ponto de espera para entrada da estação, sendo que a pre-condição para que esta ação ocorra seja que o manipulador robótico esteja no ponto de espera;

- (catch rb1 pack1 this_station), a segunda ação encontrada é para que o manipulador robótico pegue o objeto que está na entrada da estação. A precondição dessa ação é que o manipulador robótico esteja na posição de entrada da estação e tenha uma peça disponível;

- (move rb1 this_station next_station), terceira ação por meio da qual o manipulador robótico move o objeto que estava na entrada da estação para a saída da estação, onde a precondição desta ação é que o manipulador robótico esteja na posição de entrada;

- (release rb1 pack1 next_station), na quarta ação o manipulador robótico deixará o objeto na saída da estação. A precondição da ação é que o manipulador robótico esteja movimentando o objeto e chegue na posição de saída da estação;

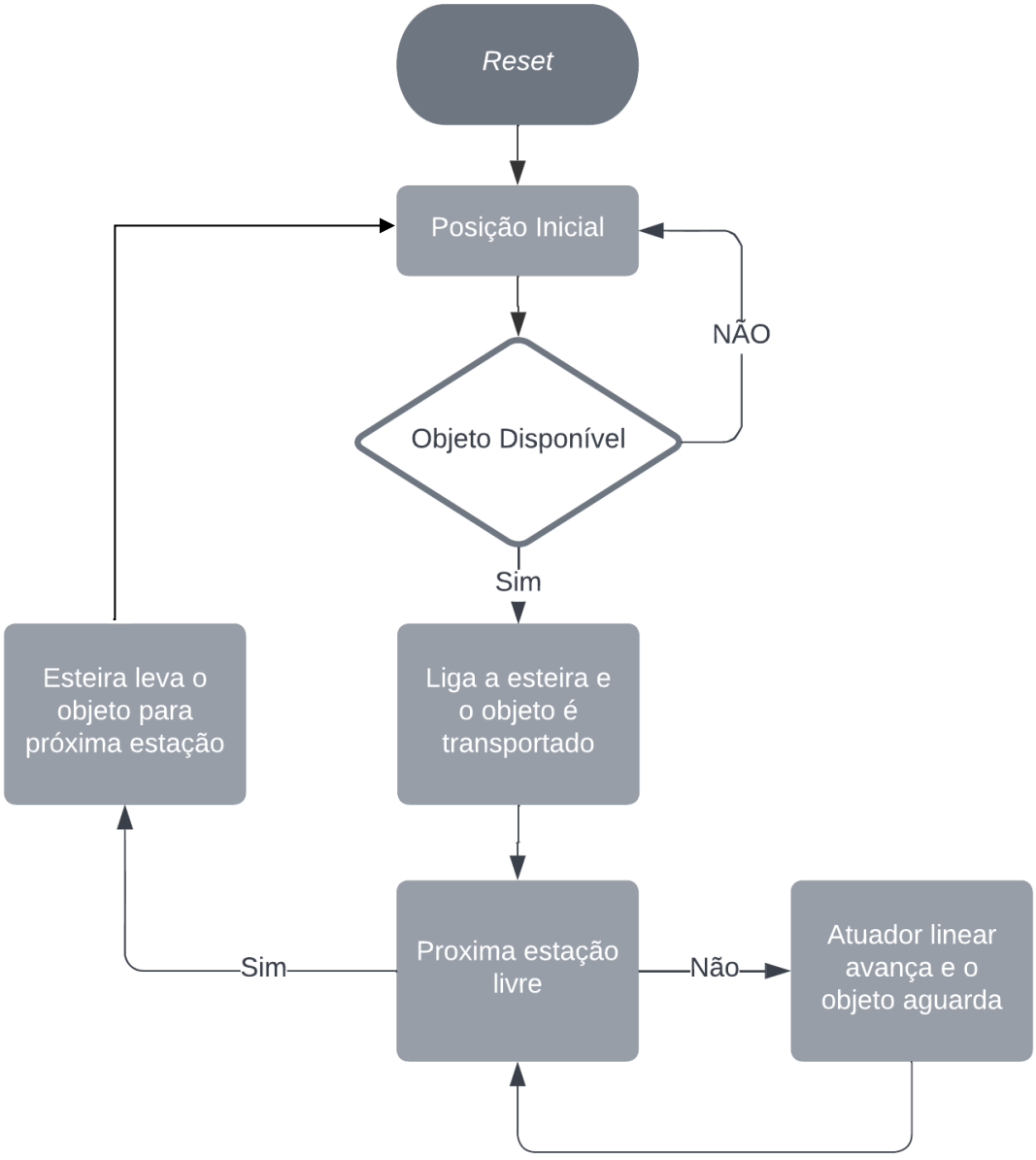
- (move rb1 next_station wait_point), a quinta e última ação encontrada é para que o manipulador robótico retorne para a posição de espera, para que o ciclo possa reco-meçar assim que o objeto estiver disponível novamente no início da estação, onde a precondição dessa ação é o manipulador robótico estar na posição "próxima esta-ção".

4.4 Modelagem da estação Espera

A estação de Espera realiza uma tarefa de transporte. Sua função é parar e segurar os objetos na esteira e liberar um por um quando a próxima estação estiver livre. A posição inicial do sistema é garantida com o motor desligado e o atuador que segura os objetos retraído. A esteira ativa ao receber o objeto no lado da entrada da estação.

Na Fig. 33 é apresentado o fluxograma da estação Espera. Neste caso, de forma bem simples e prática já que ele apenas descreve um processo de liberação das peças de trabalho, servindo como uma estação de organização dos objetos e como uma espécie de ponte para o último processo.

Figura 33. Fluxograma da estação Espera.



Fonte: Próprio autor (2021)

Elaborado o fluxograma, deu-se início na modelagem do domínio da estação Espera. A Fig. 34 mostra a descrição em PDDL do domínio da estação Espera.

Figura 34. Descrição do domínio da estação Espera.

```
(define (domain waiting)
    (:requirements :strips :typing :negative-preconditions)

    (:types
            location atua_mat part treadmill - object
            waiting_point input output - location
    )

    (:predicates
        (connected ?from ?to - location)

        (in ?p - part ?l - location)
        (advance ?a - atua_mat)
        (busy ?l - location)
        (on ?t - treadmill)
    )

    (:action move
        :parameters (?p - part ?from ?to - location)
        :precondition (and
            (in ?p ?from)
            (not (busy ?to))
        )
        :effect (and
            (not (in ?p ?from))
            (in ?p ?to)
            (busy ?to)
        )
    )

    (:action treadmill_on
        :parameters (?p - part ?i - input ?t - treadmill)
        :precondition (and
            (in ?p ?i)
            (not (on ?t))
        )
        :effect (and
            (on ?t)
        )
    )

    (:action push_atuador
        :parameters (?a - atua_mat ?p - part ?w - waiting_point ?o - output ?i - input)
        :precondition (and
            (busy ?o)
            (in ?p ?i)
            (not (advance ?a))
        )
        :effect (and
            (advance ?a)
            (in ?p ?w)
        )
    )

    (:action release
        :parameters (?a - atua_mat ?p - part ?w - waiting_point ?o - output)
        :precondition (and
            (advance ?a)
            (in ?p ?w)
            (not (busy ?o))
        )
        :effect (and
            (in ?p ?o)
            (not (advance ?a))
        )
    )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Os predicados que detalham as características do domínio são:

- (location ?l), predicado que indica que o objeto ?l é um local o qual ainda pode ser instanciado via herança como um local de waiting_point, input ou de output;
- (atua_mat ?a), predicado que indica que o objeto ?r é um atuador linear;
- (part ?p), predicado que indica que o objeto ?p é uma peça;
- (treadmill ?t), predicado que indica que o objeto ?t é uma esteira.

No domínio da estação Espera observam-se seguintes as ações:

- Move: por meio da qual a peça se movimenta entre a locais disponíveis;
- Treadmill_on: na qual a esteira é ligada para movimentar as peças;
- Push_atuador: mediante a qual o atuador linear da esteira avança para segurar a peça no ponto de espera;
- Release: na qual o atuador linear recua deixando a peça avançar para a próxima estação;

Na estação Espera observa-se a precondições das ações:

- Ação move: onde sua precondição é que a peça ?p esteja disponível para que esta se movimente de um local para outro e a próxima estação ?o não esteja ocupada.
- Ação treadmill_on: sua precondição é que a peça ?p esteja disponível no início da estação ?i.
- Ação push_atuador: a precondição nessa ação é que a próxima estação ?o esteja ocupada, a peça ?p esteja na entrada da estação e o atuador linear ?a não esteja avançado;
- Ação release: sua precondição é que o atuador linear ?a esteja avançado, a peça ?p esteja no ponto de espera ?w e a próxima estação ?o não esteja ocupada.

Após a elaboração do domínio da estação Espera, o problema de planejamento foi descrito. A Fig. 35 mostra o problema de planejamento do domínio da estação Espera em PDDL.

Figura 35. Descrição do problema no Domínio da estação Espera.

```
(define (problem problem_waiting)
(:domain waiting)
(:requirements :strips :typing :negative-preconditions)

    (:objects
        wait - waiting_point
        treadmill - treadmill
        this_Station - input
        next_Station - output
        Am1 - atua_mat
        pack1 - part
    )

    (:init
        (not (on treadmill))
        (in pack1 this_Station)
        (not (advance Am1))

        ;; Ground Connections
        (connected this_Station wait)
        (connected wait next_Station)
    )

    (:goal (and
        (on treadmill)
        (in pack1 next_Station)
        (not (advance Am1))
    ))
)
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No problema da estação Espera é possível observar o estado inicial, onde:

- (in pack1 this_Station), indica que o pacote está na entrada da estação;
- (not on treadmill), indica que a esteira está desligada;
- (not advance Am1), indica que o atuador linear da esteira está recuado.

No estado inicial do problema observa-se que os locais estão conectados entre si:

- (connected this_Station wait), indica que entrada da estação está conectada ao ponto de espera;

- (connected wait next_Station), indica que o ponto de espera está conectado a saída da estação.

A meta do planejamento para este problema é:

- (in pack1 next_Station), indica que o pacote estará na próxima estação;

- (not (advance Am1)), indica que o atuador linear da esteira estará recuado.

Inserindo no PDDL Editor o arquivo de domínio e problema da estação Espera, o Solver do Planning.Domains encontrou um plano solução, que é a sequência de ações demonstrada na Fig. 36.

Figura 36. Plano solução do domínio da estação Espera.

```
(treadmill_on pack1 this_station treadmill)
    (:action treadmill_on
    :parameters (pack1 this_station treadmill)
    :precondition
      (and
        (in pack1 this_station)
        (not
          (on treadmill)
        )
      )
    :effect
      (and
        (on treadmill)
      )
  )

(move pack1 this_station next_station)
    (:action move
    :parameters (pack1 this_station next_station)
    :precondition
      (and
        (in pack1 this_station)
        (not
          (busy next_station)
        )
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (in pack1 next_station)
        (busy next_station)
      )
  )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022).

No plano solução gerado, foram encontradas duas ações para domínio da estação Espera, que são:

- (treadmill_on pack1 this_station treadmill), onde a primeira ação é para que a esteira ligue, sendo que a precondição para que esta ação ocorra seja que peça esteja disponível na estação;

- (move pack1 this_station next_station), a segunda ação encontrada é para que o objeto se movimente do ponto inicial para a próxima estação. A precondição dessa ação é que a esteira esteja ligada e a próxima estação não esteja ocupada.

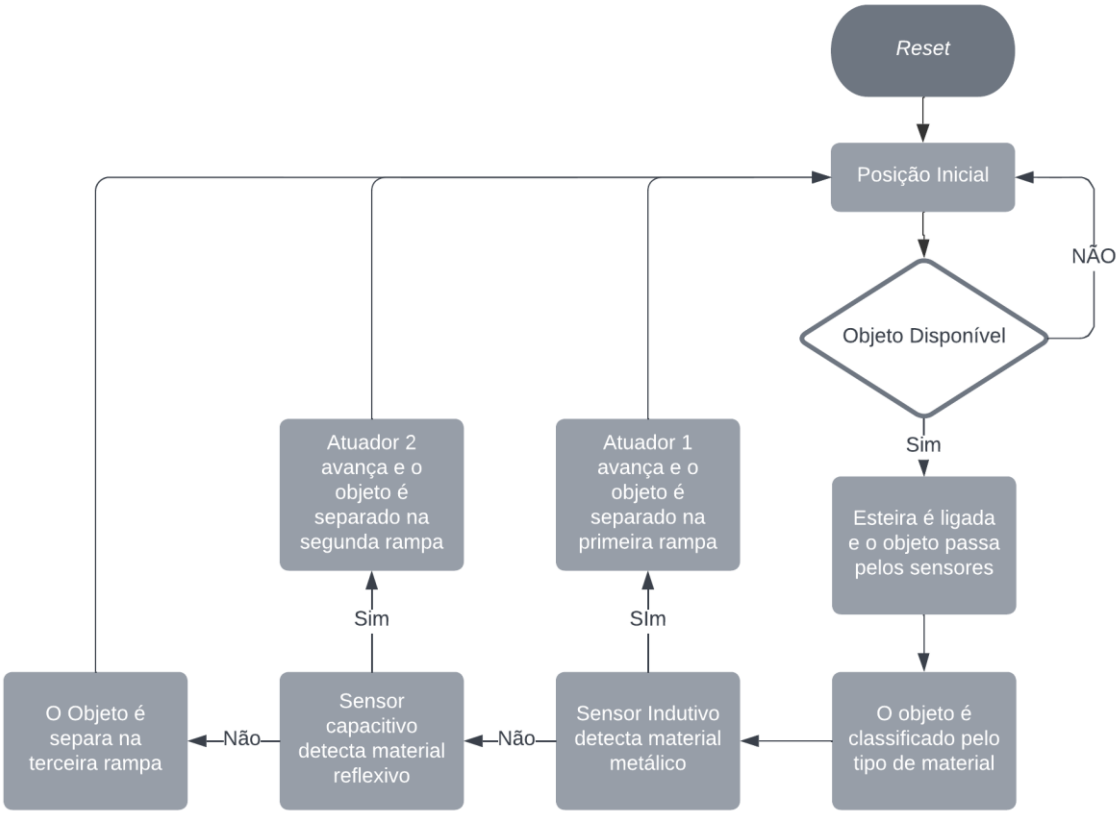
4.4 Modelagem da estação Separação

Este módulo é responsável por realizar o direcionamento das peças considerando suas características, cor e material. Existem três rampas para as quais cada peça será direcionada, de acordo com suas propriedades. Esta é a parte final da linha de produção considerada na bancada didática.

A estação Separação é ativada quando o primeiro objeto é detectado pelo sensor de entrada do transportador. E, assim, a esteira é ligada. É feita a verificação do produto de acordo com sua cor e material para definição final de qual será o destino do objeto, ou seja, em qual posição de separação ele será retirado da esteira e direcionado para a rampa de produto acabado. A bifurcação para o qual o objeto será levado é feita pela ativação de um atuador que move um braço semicircular e direciona o objeto para o local adequado.

A Fig. 37 mostra o fluxograma do processo elaborado para a estação Separação.

Figura 37. Fluxograma da estação Separação.



Fonte: Próprio autor (2021)

O domínio da estação Separação foi descrito após a elaboração do fluxograma do processo. A Fig. 38 mostra a descrição em PDDL do domínio estação Separação.

Figura 38. Descrição do domínio da estação Separação.

```
(define (domain separation)
    (:requirements :strips :typing)

    (:types
            location actuators part - object
            Input ramps sensors - location
            PartMetal PartRed PartBlack - part
            Atua_Metal Atua_Red - actuators
            Sens_Ind Sens_Cap Sens_Ramp - sensors
            Ramp_Metal Ramp_Red Ramp_Black - ramps
    )

    (:predicates
        (connected ?from ?to - location)

        (in ?p - part ?l - location)
        (reading ?s - sensors ?p - part)
        (forward ?a - actuators)
    )

    (:action move-part
        :parameters (?p - part ?from - Input ?to - Sens_Ind)
        :precondition (and
            (connected ?from ?to)
            (in ?p ?from)
        )
        :effect (and
            (not (in ?p ?from))
            (in ?p ?to)
        )
    )

    (:action keep-moving
        :parameters (?p - part ?i - Input ?from ?to - location)
        :precondition (and
            (connected ?from ?to)
            (in ?p ?from)
            (not (in ?p ?i))
        )
        :effect (and
            (not (in ?p ?from))
            (in ?p ?to)
        )
    )

    (:action push_part_metal
        :parameters (?pm - PartMetal ?si - Sens_Ind ?am - Atua_Metal ?sp - Sens_Ramp)
        :precondition (and
            (in ?pm ?si)
            (not (forward ?am))
        )
        :effect (and
            (in ?pm ?sp)
            (forward ?am)
        )
    )

    (:action push_part_red
        :parameters (?pr - PartRed ?sc - Sens_Cap ?ar - Atua_Red ?sp - Sens_Ramp)
        :precondition (and
            (in ?pr ?sc)
            (not (forward ?ar))
        )
        :effect (and
            (forward ?ar)
            (in ?pr ?sp)
        )
    )

    (:action return_actuator_red
        :parameters (?ar - Atua_Red ?pr - PartRed ?sp - Sens_Ramp ?rr - Ramp_Red)
        :precondition (and
            (forward ?ar)
            (in ?pr ?sp)
        )
        :effect (and
            (not (forward ?ar))
            (not (in ?pr ?sp))
            (in ?pr ?rr)
        )
    )

    (:action return_actuator_metal
        :parameters (?am - Atua_Metal ?pm - PartMetal ?sp - Sens_Ramp ?rm - Ramp_Metal)
        :precondition (and
            (forward ?am)
            (in ?pm ?sp)
        )
        :effect (and
            (not (forward ?am))
            (not (in ?pm ?sp))
            (in ?pm ?rm)
        )
    )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

Os predicados que detalham as características do domínio são:

- (location ?l), predicado que indica que o objeto ?l é um local o qual ainda pode ser instanciado via herança como um local de input, ramps e sensors;
- (actuators ?a), predicado que indica que o objeto ?a é um atuador o qual ainda pode ser instanciado via herança como um atua_Metal e atua_Red;
- (part ?p), predicado que indica que o objeto ?p é uma peça o qual ainda pode ser instanciado via herança como PartMetal, PartRed e PartBlack.

No domínio da estação Separação observam-se as ações:

- Keep-moving: por meio da qual a peça se movimenta entre a locais disponíveis;
- Move-part: por meio da qual a peça se movimenta entre o início da estação o ponto de leitura do sensor indutivo;
- Push_part_metal: por meio da qual o atuador linear avança e empurra a peça detectada no sensor indutivo para a rampa determinada para peças metálicas;
- Push_part_red: por meio da qual o atuador linear avança e empurra a peça detectada no sensor capacitivo para a rampa determinada para peças vermelhas;
- Return_actuator_metal: onde o atuador linear que empurra a peças metálicas recua após as peças serem detectadas na rampa de armazenamento.
- Return_actuator_red: onde o atuador linear que empurra a peças vermelhas recua após as peças serem detectadas na rampa de armazenamento.

Na estação Separação observa-se a precondições das ações:

- Ação move: onde sua precondição é que a peça ?p esteja em algum local para que esta se movimente.
- Ação push_part_metal: a precondição para que esta ação ocorra é que a peça metálica ?pm esteja no local de entrada ?i e também seja detectada pelo sensor indutivo ?si;
- Ação push_part_red: a precondição para que esta ação ocorra é que a peça vermelha ?pr esteja no local de entrada ?i e também seja detectada pelo sensor capacitivo ?sc.
- Ação return_actuator_metal: a precondição para que esta ação ocorra é que o atuador linear ?am esteja avançado e o sensor das rampas ?sp detecte a presença de uma peça ?p.
- Ação return_actuator_red: a precondição para que esta ação ocorra é que o atuador linear ?ar esteja avançado e o sensor das rampas ?sp detecte a presença de uma peça ?p.

Após da descrição do domínio da estação Separação foi elaborado o problema de planejamento. A Fig. 39 descreve o problema de planejamento do Domínio da estação Separação em PDDL.

Figura 39. Descrição do problema no Domínio da estação Separação.

```
(define (problem problem_separation)
(:domain separation)
(:requirements :strips :typing)

    (:objects
        this_station - input
        red_part - PartRed
        black_part - PartBlack
        metal_part - PartMetal
        sensor_cap - Sens_Cap
        sensor_ind - Sens_Ind
        sensor_rampa - Sens_Ramp
        Atuador_Red - Atua_Red
        Atuador_Metal - Atua_Metal
        ramp_red - Ramp_Red
        ramp_black - Ramp_Black
        ramp_metal - Ramp_Metal
    )

    (:init
        (in red_part this_station)
        (in black_part this_station)
        (in metal_part this_station)
        (not (forward Atuador_Red))
        (not (forward Atuador_Metal))

         ;; Ground Connections
        (connected this_station sensor_ind)
        (connected sensor_ind sensor_cap)
        (connected sensor_cap sensor_rampa)
        (connected sensor_rampa ramp_black)
    )

    (:goal (and
        (in red_part ramp_red)
        (in black_part ramp_black)
        (in metal_part ramp_metal)
        (not (forward Atuador Red))
        (not (forward Atuador_Metal))
    ))
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No problema da estação Separação é possível observar o estado inicial, onde:

- (in red_part this_station), indica que uma peça vermelha está na entrada da estação;
- (in black_part this_station), indica que uma peça preta está na entrada da estação;
- (in metal_part this_station), indica que uma peça metálica está na entrada da estação;
- (not (forward Atuador_Red)), indica que o atuador de peças vermelhas está recuado;
- (not (forward Atuador_Metal)), indica que o atuador de peças metálicas está recuado;

No estado inicial do problema observa-se também que os locais estão conectados entre si:

- (connected this_station sensor_ind), indica que entrada da estação está conectada ao ponto de leitura do sensor indutivo;
- (connected sensor_ind sensor_cap), indica que o sensor indutivo está conectado ao sensor capacitivo;
- (connected sensor_cap sensor_rampa), indica que o sensor capacitivo está conectado ao sensor das rampas de armazenamento;
- (connected sensor_rampa ramp_black), indica que o sensor das rampas de armazenamento está concetado a rampa de armazenamento de peças pretas.

A meta do planejamento para este problema é:

- (in red_part ramp_red), indica que a peça vermelha estará na rampa designada para peças vermelhas;
- (in black_part ramp_black), indica que a peça preta estará na rampa designada para peças pretas;
- (in metal_part ramp_metal), indica que a peça metálica estará na rampa designada para peças metálicas;
- (not (forward Atuador_Red)), indica que o atuador de que empurra peças vermelhas estará recuado ao final do ciclo.
- (not (forward Atuador_Metal)), indica que o atuador de que empurra peças metálicas estará recuado ao final do ciclo.

Na última estação, o plano solução do domínio da estação Separação foi encontrado pela sequência de ações demonstrada na Fig. 40.

Figura 40. Plano solução do domínio da estação Separação.

```
(move-part metal_part this_station sensor_ind)
    (:action move-part
    :parameters (metal_part this_station sensor_ind)
    :precondition
      (and
        (connected this_station sensor_ind)
        (in metal_part this_station)
      )
    :effect
      (and
        (not
          (in metal_part this_station)
        )
        (in metal_part sensor_ind)
      )
  )

(push_part_metal metal_part sensor_ind atuador_metal sensor_rampa)
    (:action push_part_metal
    :parameters (metal_part sensor_ind atuador_metal sensor_rampa)
    :precondition
      (and
        (in metal_part sensor_ind)
        (not
          (forward atuador_metal)
        )
      )
    :effect
      (and
        (in metal_part sensor_rampa)
        (forward atuador_metal)
      )
  )

(return_actuator_metal atuador_metal metal_part sensor_rampa ramp_metal)
    (:action return_actuator_metal
    :parameters (atuador_metal metal_part sensor_rampa ramp_metal)
    :precondition
      (and
        (forward atuador_metal)
        (in metal_part sensor_rampa)
      )
    :effect
      (and
        (not
          (forward atuador_metal)
        )
        (not
          (in metal_part sensor_rampa)
        )
        (in metal_part ramp_metal)
      )
  )

(move-part red_part this_station sensor_ind)
    (:action move-part
    :parameters (red_part this_station sensor_ind)
    :precondition
      (and
        (connected this_station sensor_ind)
        (in red_part this_station)
      )
    :effect
      (and
        (not
          (in red_part this_station)
        )
        (in red_part sensor_ind)
      )
  )

(keep-moving red_part this_station sensor_ind sensor_cap)
    (:action keep-moving
    :parameters (red_part this_station sensor_ind sensor_cap)
    :precondition
      (and
        (connected sensor_ind sensor_cap)
        (in red_part sensor_ind)
        (not
          (in red_part this_station)
        )
      )
    :effect
      (and
        (not
          (in red_part sensor_ind)
        )
        (in red_part sensor_cap)
      )
  )

(push_part_red red_part sensor_cap atuador_red sensor_rampa)
    (:action push_part_red
    :parameters (red_part sensor_cap atuador_red sensor_rampa)
    :precondition
      (and
        (in red_part sensor_cap)
        (not
          (forward atuador_red)
        )
      )
    :effect
      (and
        (forward atuador_red)
        (in red_part sensor_rampa)
      )
  )

(return_actuator_red atuador_red red_part sensor_rampa ramp_red)
    (:action return_actuator_red
    :parameters (atuador_red red_part sensor_rampa ramp_red)
    :precondition
      (and
        (forward atuador_red)
        (in red_part sensor_rampa)
      )
    :effect
      (and
        (not
          (forward atuador_red)
        )
        (not
          (in red_part sensor_rampa)
        )
        (in red_part ramp_red)
      )
  )

(move-part black_part this_station sensor_ind)
    (:action move-part
    :parameters (black_part this_station sensor_ind)
    :precondition
      (and
        (connected this_station sensor_ind)
        (in black_part this_station)
      )
    :effect
      (and
        (not
          (in black_part this_station)
        )
        (in black_part sensor_ind)
      )
  )

(keep-moving black_part this_station sensor_ind sensor_cap)
    (:action keep-moving
    :parameters (black_part this_station sensor_ind sensor_cap)
    :precondition
      (and
        (connected sensor_ind sensor_cap)
        (in black_part sensor_ind)
        (not
          (in black_part this_station)
        )
      )
    :effect
      (and
        (not
          (in black_part sensor_ind)
        )
        (in black_part sensor_cap)
      )
  )

(keep-moving black_part this_station sensor_cap sensor_rampa)
    (:action keep-moving
    :parameters (black_part this_station sensor_cap sensor_rampa)
    :precondition
      (and
        (connected sensor_cap sensor_rampa)
        (in black_part sensor_cap)
        (not
          (in black_part this_station)
        )
      )
    :effect
      (and
        (not
          (in black_part sensor_cap)
        )
        (in black_part sensor_rampa)
      )
  )

(keep-moving black_part this_station sensor_rampa ramp_black)
    (:action keep-moving
    :parameters (black_part this_station sensor_rampa ramp_black)
    :precondition
      (and
        (connected sensor_rampa ramp_black)
        (in black_part sensor_rampa)
        (not
          (in black_part this_station)
        )
      )
    :effect
      (and
        (not
          (in black_part sensor_rampa)
        )
        (in black_part ramp_black)
      )
```

Fonte: Próprio autor. Editado no PDDL Editor (2022)

No plano solução gerado, foram encontradas onze ações para domínio da estação Separação, que são:

- (move-part metal_part this_station sensor_ind), onde a primeira ação é para que a peça metálica se movimente para o sensor de leitura indutivo, sendo que a precondição para que esta ação ocorra seja que a peça esteja disponível na entrada da estação;

- (push_part_metal metal_part sensor_ind atuador_metal sensor_rampa), a segunda ação encontrada é para que o atuador de peças metálicas avance para a peça seja levada ao sensor das rampas, sendo que a precondição para que esta ação ocorra seja que a peça tenha sido detectada pelo sensor indutivo e o atuador de peças metálicas esteja recuado;

- (return_actuator_metal atuador_metal metal_part sensor_rampa ramp_metal), a terceira ação encontrada é para que o atuador de peças metálicas recue para o posição inicial, sendo que a precondição para que esta ação ocorra seja que o atuador de peças metálicas esteja avançado e a peça tenha sido detectada pelo sensor das rampas;

- (move-part red_part this_station sensor_ind), a quarta ação é para que a peça vermelha se movimente para o sensor de leitura indutivo, sendo que a precondição para que esta ação ocorra seja que a peça esteja disponível na entrada da estação;

- (keep-moving red_part this_station sensor_ind sensor_cap), a quinta ação é para que a peça vermelha se movimente para o sensor de leitura capacitivo, sendo que a precondição para que esta ação ocorra seja que a peça já tenha passado pela leitura do sensor indutivo e não tenha sido detectada;

- (push_part_red red_part sensor_cap atuador_red sensor_rampa), a sexta ação encontrada é para que o atuador de peças vermelhas avance para que a peça seja levada ao sensor das rampas, sendo que a precondição para que esta ação ocorra seja que a peça tenha sido detectada pelo sensor capacitivo e o atuador de peças vermelhas esteja recuado;

- (return_actuator_red atuador_red red_part sensor_rampa ramp_red), a sétima ação encontrada é para que o atuador de peças vermelhas recue para o posição inicial, sendo que a precondição para que esta ação ocorra seja que o atuador de peças vermelhas esteja avançado e a peça tenha sido detectada pelo sensor das rampas;

- (move-part black_part this_station sensor_ind), a oitava ação é para que a peça preta se movimente para o sensor de leitura indutivo, sendo que a precondição para que esta ação ocorra seja que a peça esteja disponível na entrada da estação;

- (keep-moving black_part this_station sensor_ind sensor_cap), a nona ação é para que a peça preta se movimente para o sensor de leitura capacitivo, sendo que a precondição para que esta ação ocorra seja que a peça já tenha passado pela leitura do sensor indutivo e não tenha sido detectada;

- (keep-moving black_part this_station sensor_cap sensor_rampa), a decima ação é para que a peça preta se movimente para o sensor de leitura das rampas, sendo que a precondição para que esta ação ocorra seja que a peça já tenha passado pela leitura do sensor capacitivo e não tenha sido detectada;

- (keep-moving black_part this_station sensor_rampa ramp_black), a decima primeira e última ação é para que a peça preta se movimente para a rampa de armazenamento de peças pretas, sendo que a precondição para que esta ação ocorra seja que a peça já tenha passado pela leitura do sensor das rampas;

5. RESULTADOS

Neste capítulo, são apresentados os resultados obtidos por meio da execução do plano-solução na bancada didática de robótica. Para uma demonstração mais visual e abrangente, foi disponibilizado um vídeo dos resultados que pode ser acessado através do seguinte link: Vídeo de Demonstração dos Resultados[2]. Neste vídeo, é possível observar o funcionamento do sistema, a interação entre o CLP *Siemens S7-1500*, o desenvolvimento do processo indicado pela atualização de cada SFC segundo os planos gerados. Essa demonstração em vídeo oferece uma visão mais completa e detalhada dos resultados obtidos, permitindo uma análise mais precisa e uma compreensão mais clara dos benefícios proporcionados por essa integração tecnológica. Utilizou-se estes planos para elaborar uma programação utilizando a linguagem *SFC* dentro do CLP *Siemens S7-1500*, com o objetivo de simular um planejamento automático a partir dos modelos gerados pelo "PDDL Editor". A simulação foi realizada em um ambiente controlado, os resultados obtidos foram promissores, demonstrando que a integração dos sistemas pode ser uma excelente solução para a automação de processos em diversas áreas da indústria. A principal vantagem dessa abordagem (modelagem do domínio de planejamento/geração de planos solução e conversão dos planos solução para SFC) em relação a uma solução clássica (programação direta da lógica no CLP, em qualquer linguagem prevista na IEC 61131-3) está na automatização do processo de planejamento e na flexibilidade para ajustar a lógica de controle. A modelagem do domínio de planejamento permite definir as variáveis, estados e ações relevantes para o sistema, tornando a lógica mais concisa e focada nas tarefas específicas.

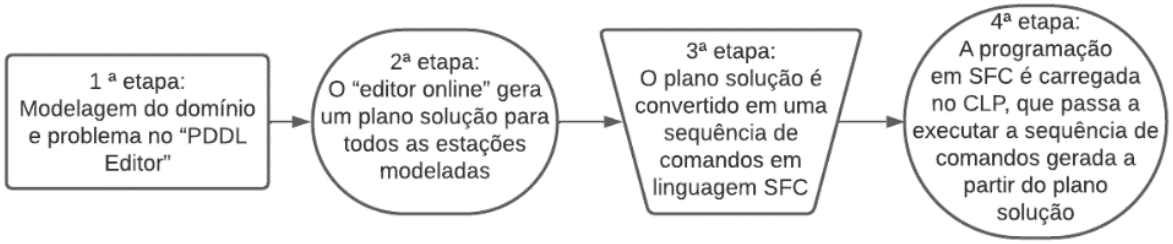
Para testar a viabilidade do projeto, foram realizados experimentos utilizando uma bancada educacional de fabricação robótica. O domínio da bancada foi representado por meio da linguagem PDDL, o que permitiu avaliar os problemas de planejamento usando um planejador automático online.

A integração de um plano solução gerado pelo "PDDL Editor" na construção de uma programação em SFC no CLP como proposta de planejamento automático envolveu algumas etapas importantes. A primeira etapa foi a modelagem do domínio e do problema no "PDDL

[2] https://sesigoias-my.sharepoint.com/:v:/g/personal/weslleysilva_senai_fieg_com_br/EZGsE5M2-DlGk1Lm5vr56SoBBhtfBHwi6M9dVymiGJmU1w?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0RpcmVjdCJ9fQ&e=fq4chb

Editor", que consiste em definir as variáveis, estados, ações e objetivos do sistema. Em seguida, o "editor online" gerou um plano solução para todos as estações modeladas. Na terceira etapa, o plano solução é convertido em uma sequência de comandos em linguagem SFC, que foi executada pelo CLP. Por fim, a programação em SFC é carregada no CLP, que passa a executar a sequência de comandos gerada a partir do plano solução. A Fig. 41 ilustra na forma de um diagrama essas etapas.

Figura 41. Diagrama método proposto da integração de um plano solução no CLP.

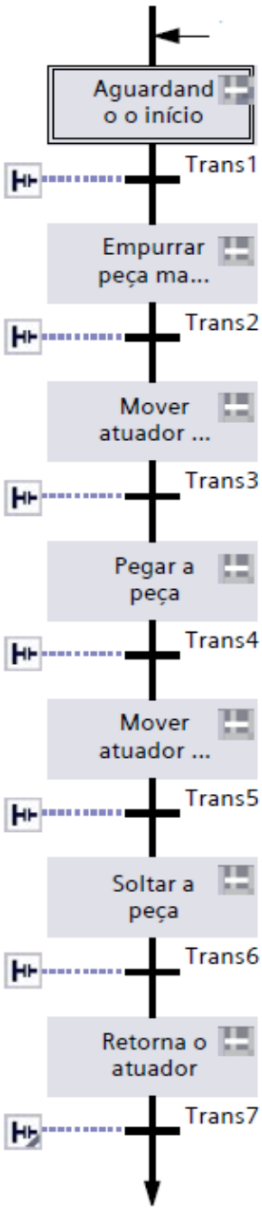


Fonte: Próprio autor (2022)

A Fig. 42 ilustra a relação entre os planos gerados automaticamente pelo planejador automático e os modelos em SFC criados a partir desses planos. Os planos são exibidos como uma sequência de etapas ou ações necessárias para atingir um objetivo específico. Essas etapas são mostradas em uma linha do tempo ou em um fluxograma, destacando a ordem e a interdependência das ações. Ao lado dos planos, os modelos em SFC são representados por diagramas que visualizam as etapas e a lógica de controle do sistema robótico.

Figura 42. Relação entre os planos gerados automaticamente e os modelos em SFC estação Recebimento.

```
(push pack1 magazine this_station)
    (:action push
    :parameters (pack1 magazine this_station)
    :precondition
      (and
        (in pack1 magazine)
      )
    :effect
      (and
        (not
          (in pack1 magazine)
        )
        (in pack1 this_station)
      )
  )

(catch ag1 waiting_point this_station)
    (:action catch
    :parameters (ag1 waiting_point this_station)
    :precondition
      (and
        (at ag1 waiting_point)
      )
    :effect
      (and
        (not
          (at ag1 waiting_point)
        )
        (at ag1 this_station)
      )
  )

(landing ag1 pack1 this_station)
    (:action landing
    :parameters (ag1 pack1 this_station)
    :precondition
      (and
        (at ag1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (at ag1 this_station)
        )
        (not
          (in pack1 this_station)
        )
        (landing ag1 pack1)
      )
  )

(release ag1 pack1 waiting_point next_station)
    (:action release
    :parameters (ag1 pack1 waiting_point next_stat
    :precondition
      (and
        (landing ag1 pack1)
      )
    :effect
      (and
        (not
          (landing ag1 pack1)
        )
        (in pack1 next_station)
        (at ag1 next_station)
      )
  )

(move ag1 next_station waiting_point)
    (:action move
    :parameters (ag1 next_station waiting_point)
    :precondition
      (and
        (at ag1 next_station)
      )
    :effect
      (and
        (not
          (at ag1 next_station)
        )
        (at ag1 waiting_point)
      )
```



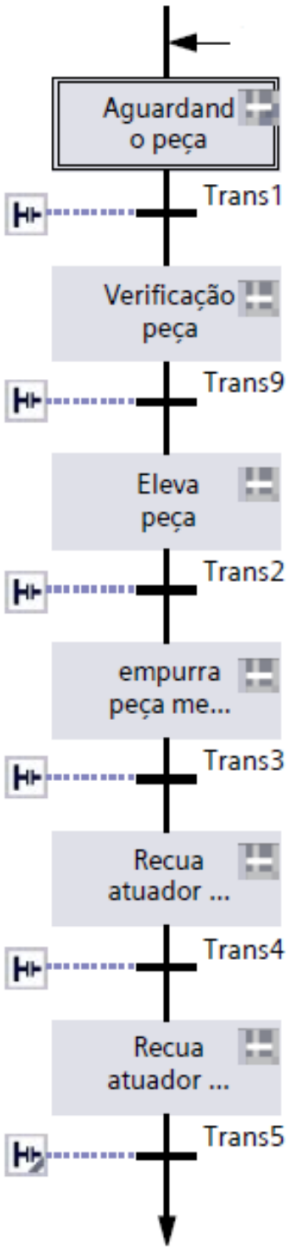
Fonte: Próprio autor (2022)

Cada etapa do plano é mapeada para um estado ou transição no modelo em SFC, demonstrando como as ações são traduzidas e incorporadas ao controle do CLP. Essa imagem proporciona uma compreensão visual clara da relação entre os planos gerados automaticamente e os modelos em SFC, destacando a aplicação prática do planejamento automático na simplificação do desenvolvimento e controle da bancada robótica.

A Fig. 43 ilustra a relação entre os planos gerados automaticamente pelo planejador automático e os modelos em SFC para a estação Processamento.

Figura 43. Relação entre os planos gerados automaticamente e os modelos em SFC estação Processamento.

```
(verification pack1 modulo_sep this_station)
    (:action verification
    :parameters (pack1 modulo_sep this_station)
    :precondition
      (and
        (in pack1 modulo_sep)
      )
    :effect
      (and
        (not
          (in pack1 modulo_sep)
        )
        (in pack1 this_station)
      )
  )

(raise ae1 pack1 this_station modulo_med)
    (:action raise
    :parameters (ae1 pack1 this_station modulo_med)
    :precondition
      (and
        (at ae1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (not
          (at ae1 this_station)
        )
        (in pack1 modulo_med)
        (at ae1 modulo_med)
      )
  )

(push ar1 pack1 modulo_med next_station)
    (:action push
    :parameters (ar1 pack1 next_station modulo_med)
    :precondition
      (and
        (at ar1 modulo_med)
        (in pack1 modulo_med)
      )
    :effect
      (and
        (not
          (at ar1 modulo_med)
        )
        (not
          (in pack1 modulo_med)
        )
        (in pack1 next_station)
        (at ar1 next_station)
      )
  )

(move ar1 next_station modulo_med)
    (:action move
    :parameters (ar1 next_station modulo_med)
    :precondition
      (and
        (at ar1 next_station)
      )
    :effect
      (and
        (not
          (at ar1 next_station)
        )
        (at ar1 modulo_med)
      )
  )

(move ae1 modulo_med this_station)
    (:action move
    :parameters (ae1 modulo_med this_station)
    :precondition
      (and
        (at ae1 modulo_med)
      )
    :effect
      (and
        (not
          (at ae1 modulo_med)
        )
        (at ae1 this_station)
      )
```



Fonte: Próprio autor (2022)

O plano referente à estação de Distribuição não foi convertido em SFC no CLP, uma vez que se trata da estação do manipulador robótico. Nesse contexto, o plano gerado foi diretamente convertido em programação para o manipulador, o qual emprega a linguagem “Melfa-Basic” para sua operação.

A Fig. 44 ilustra a relação entre os planos gerados automaticamente pelo planejador automático e a programação do manipulador robótico para a estação Distribuição.

Figura 44. Relação entre os planos gerados automaticamente e a programação da estação Distribuição.

```
(move rb1 wait_point this_station)
    (:action move
    :parameters (rb1 wait_point this_station)
    :precondition
      (and
        (at rb1 wait_point)
      )
    :effect
      (and
        (not
          (at rb1 wait_point)
        )
        (at rb1 this_station)
      )
  )

(catch rb1 pack1 this_station)
    (:action catch
    :parameters (rb1 pack1 this_station)
    :precondition
      (and
        (not
          (on pack1 rb1)
        )
        (at rb1 this_station)
        (in pack1 this_station)
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (on pack1 rb1)
      )
  )

(move rb1 this_station next_station)
    (:action move
    :parameters (rb1 this_station next_station)
    :precondition
      (and
        (at rb1 this_station)
      )
    :effect
      (and
        (not
          (at rb1 this_station)
        )
        (at rb1 next_station)
      )
  )

(release rb1 pack1 next_station)
    (:action release
    :parameters (rb1 pack1 next_station)
    :precondition
      (and
        (at rb1 next_station)
        (on pack1 rb1)
      )
    :effect
      (and
        (in pack1 next_station)
        (not
          (on pack1 rb1)
        )
      )
  )

(move rb1 next_station wait_point)
    (:action move
    :parameters (rb1 next_station wait_point)
    :precondition
      (and
        (at rb1 next_station)
      )
    :effect
      (and
        (not
          (at rb1 next_station)
        )
        (at rb1 wait_point)
      )
```

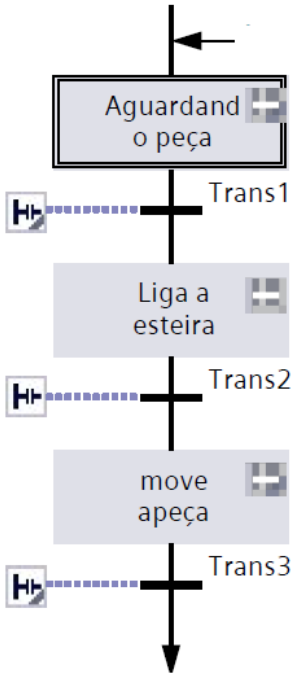
```
10 MOV P1
20 WAIT M_IN(8)=1
20 MVS P2,-50
30 HOPEN 1
40 MVS P2
50 HCLOSE 1
60 MVS P2,-50
70 MOV P4,-50
80 MVS P4
90 HOPEN 1
100 MVS P4,-50
110 HCLOSE
120 MOV P1
```

Fonte: Próprio autor (2022)

A Fig. 45 ilustra a relação entre os planos gerados automaticamente pelo planejador automático e os modelos em *SFC* para a estação Espera.

Figura 45. Relação entre os planos gerados automaticamente e os modelos em SFC estação Espera.

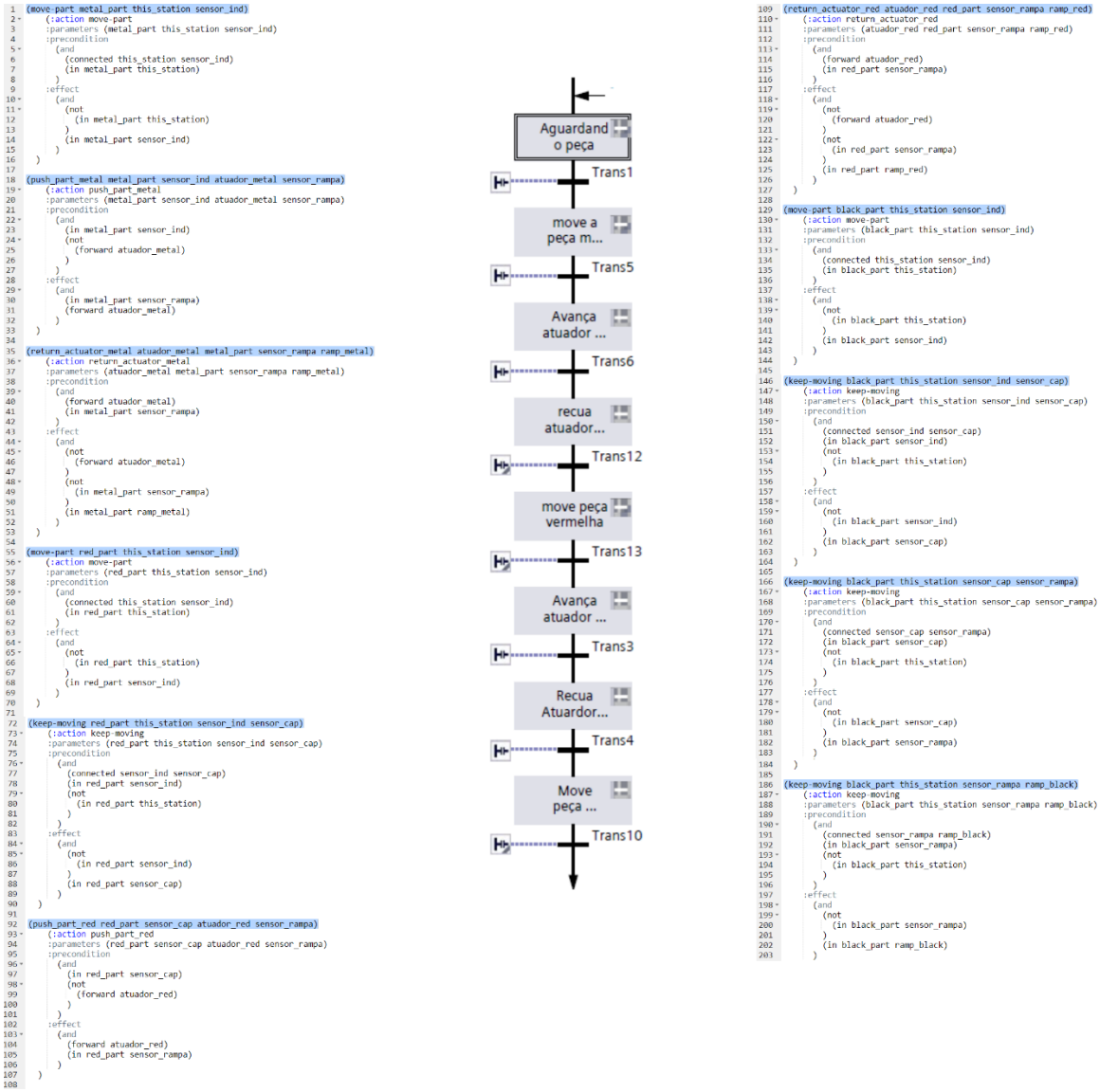
```
(treadmill_on pack1 this_station treadmill)
    (:action treadmill_on
    :parameters (pack1 this_station treadmill)
    :precondition
      (and
        (in pack1 this_station)
        (not
          (on treadmill)
        )
      )
    :effect
      (and
        (on treadmill)
      )
  )

(move pack1 this_station next_station)
    (:action move
    :parameters (pack1 this_station next_station)
    :precondition
      (and
        (in pack1 this_station)
        (not
          (busy next_station)
        )
      )
    :effect
      (and
        (not
          (in pack1 this_station)
        )
        (in pack1 next_station)
        (busy next_station)
      )
  )
```



Fonte: Próprio autor (2022)

A Fig. 46 ilustra a relação entre os planos gerados automaticamente pelo planejador automático e os modelos em *SFC* para a estação Separação.

Figura 46. Relação entre os planos gerados automaticamente e os modelos em SFC estação Separação.



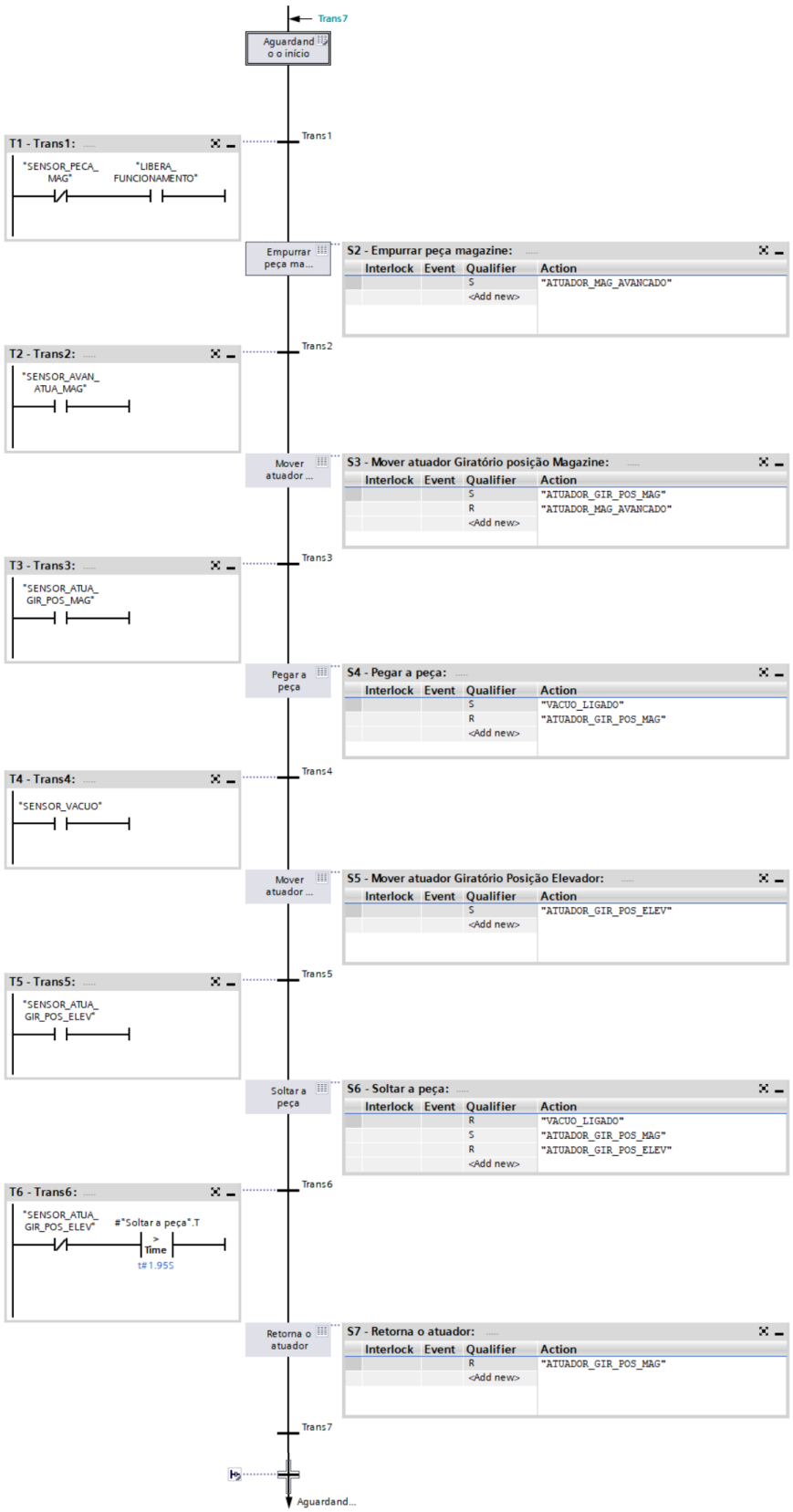
Fonte: TIA Portal V13 (2022)

A construção de uma programação em SFC é um processo importante para automatizar processos industriais e garantir a eficiência do sistema. É uma linguagem gráfica utilizada para descrever o comportamento de um sistema automatizado, e neste caso, a programação foi criada com base nos planos solução gerados. Isso significa que a programação foi projetada para seguir a sequência de ações definida nos planos solução. Primeiramente, é necessário definir os objetivos do processo a ser automatizado e modelar a sequência de etapas que serão executadas pelo sistema. Em seguida, a programação é criada, utilizando símbolos gráficos para representar as

etapas, eventos e ações que o sistema deve realizar. Essa programação é dividida em etapas, que representam as diferentes fases do processo, e cada etapa é composta por uma ou mais ações a serem executadas. Para garantir a segurança do sistema, a programação também inclui condições de transição, que indicam quando o sistema deve avançar para a próxima etapa. No caso da estação Recebimento cada condição de transição está diretamente ligada aos sensores e as ações estão ligadas aos atuadores.

A Fig. 47 mostra a programação expandida da estação Recebimento baseado no plano solução gerado pelo “editor online”. As programações em SFC para as estações Processamento, Espera e Separação estão detalhadas no Apêndice A.

Figura 47. Programação em SFC da estação Recebimento.



Fonte: TIA Portal V13 (2022)

As duas abordagens propostas na dissertação apresentam diferentes perspectivas na modelagem de uma bancada didática robótica controlada por CLP. A primeira abordagem consistiu na programação da bancada com o CLP utilizando o método mais comum nos ambientes industriais que é a linguagem de programação *Ladder* que é uma ferramenta gráfica usada para desenvolver programas ou softwares para CLPs. Isso permite uma maior compatibilidade e integração com os sistemas já existentes, facilitando a comunicação e o controle entre a bancada robótica e outros dispositivos industriais. Essa abordagem se destaca pela sua aplicabilidade prática e pela utilização de padrões já estabelecidos.

Por outro lado, a segunda abordagem propôs a modelagem da bancada didática robótica na integração dos planos solução gerados pelo "PDDL Editor" na construção de uma programação em SFC no CLP. Essa abordagem visa utilizar técnicas de planejamento automático para aprimorar o processo de programação e controle da bancada robótica. Ao gerar automaticamente um plano de solução e traduzi-lo para uma programação em SFC no CLP, buscou-se mostrar uma abordagem diferente baseada em sistemas automáticos para o desenvolvimento da bancada robótica.

A principal contribuição desse trabalho é mostrar, que o planejamento automático pode ser aplicado na prática, mesmo em casos onde soluções convencionais seriam mais complexas e demoradas de serem implementadas. A integração do "PDDL Editor" com o CLP possibilitou a geração automática de planos solução e sua tradução para a programação em SFC, tornando a modelagem da bancada didática de manufatura robótica mais acessível. A pesquisa reforça o potencial do planejamento automático como uma ferramenta para a automação de processos industriais. Com o avanço contínuo nessa área, acredita-se que essa abordagem tem o potencial de revolucionar a indústria, tornando-a mais produtiva e adaptável às demandas do cenário atual de manufatura inteligente.

No intuito de proporcionar um acesso direto aos detalhes da programação desenvolvida para a bancada didática robótica, todos os arquivos relevantes, foram disponibilizados no seguinte repositório “GitHub”: link do repositório[3]. Esta iniciativa visa promover a transparência, colaboração e a possibilidade de compartilhamento de conhecimento, permitindo que outros pesquisadores e interessados possam examinar, utilizar e construir sobre o trabalho realizado neste estudo.

[3] https://github.com/silvawesllley/Bancada_Rob-tica.git

6. CONCLUSÃO

Neste trabalho, apresentou-se a modelagem e implementação de uma planta didática de manufatura robótica, utilizando planejamento automático como base para a automação de processos. A integração CLP e a ferramenta online "PDDL Editor" se mostrou essencial para alcançar resultados promissores na execução dos planos gerados. Ao longo deste estudo, foi demonstrado que a utilização do "PDDL Editor" para a geração de planos solução permite uma maior flexibilidade no desenvolvimento de estratégias de automação.

O estudo se concentrou na integração do CLP com a ferramenta online "PDDL Editor" para a geração de planos solução. Através da linguagem PDDL, foi possível representar o domínio da planta de forma parcial, permitindo o planejamento automático de tarefas de montagem robótica. A construção de uma programação em SFC no CLP, a partir dos planos gerados pelo "PDDL Editor", viabilizou a execução das tarefas, resultando em um modelo de aplicação de planejamento automático a um modelo de processo industrial. A dissertação explorou, portanto, a viabilidade dessa abordagem, abrindo caminho para futuros aprimoramentos e aplicações no campo da automação industrial.

O exemplo prático neste trabalho mostra o potencial da linguagem, e principalmente do ambiente, no processo de modelagem e análise dos modelos de domínios de planejamento. A utilização de um plano solução gerado pelo “PDDL Editor” na construção de uma programação em SFC no CLP, como proposta de planejamento automático, foi realizada manualmente, portanto, isso pode ser um processo demorado e propenso a erros. Embora a integração manual possa simplificar o processo em alguns casos, o ideal seria que essa integração fosse realizada de forma automática, utilizando ferramentas de integração que possam conectar o “PDDL Editor” diretamente ao CLP. Dessa forma, o plano solução gerado pelo planejador seria convertido automaticamente em comandos na linguagem desejada, em conformidade com a norma IEC 61131-3, o que economizaria tempo e reduziria a chance de erros na integração.

6.1 Trabalhos Futuros

Acredita-se que os trabalhos a serem desenvolvidos no futuro possuem um grande potencial para agregar valor ao trabalho apresentado e, por consequência, fortalecer a área de pesquisa em planejamento automático, tanto em âmbito nacional quanto internacional. Com avanços contínuos na integração de tecnologias e algoritmos mais avançados, espera-se que a planta didática de manufatura robótica inspire novas soluções e aplicações em automação industrial, impulsionando o progresso do setor e fomentando o desenvolvimento de sistemas mais eficientes, confiáveis e adaptáveis. Ademais, a colaboração com pesquisadores de outras instituições e a disseminação dos resultados por meio de eventos e publicações científicas contribuirão para o enriquecimento do conhecimento e a projeção da relevância do trabalho no contexto global de pesquisa em automação robótica.

Este trabalho abre caminho para diversas ideias de trabalhos futuros. Uma possível direção é a otimização do processo de integração entre o "PDDL Editor" e o CLP, buscando automatizar a conversão dos planos gerados em comandos SFC, de modo a reduzir ainda mais a intervenção manual e, consequentemente, os possíveis erros. Além disso, a investigação de outras ferramentas de planejamento automático e a comparação com o "PDDL Editor" poderia revelar opções alternativas para a geração de planos solução. Novas abordagens e algoritmos podem ser explorados para ampliar o escopo da aplicação, buscando soluções mais eficazes e adaptáveis a diferentes cenários de manufatura robótica.

Outra possibilidade interessante seria a expansão da planta didática para simular ambientes mais complexos e realistas, incorporando outros elementos de automação e controle, como sensores, atuadores e sistemas de visão computacional. Isso permitiria explorar a interação entre a manufatura robótica e sistemas inteligentes, aproximando a planta didática das aplicações encontradas em ambientes industriais reais.

Em resumo, a modelagem e implementação desta planta didática de manufatura robótica inspirada em planejamento automático trouxe resultados significativos e promissores, evidenciando o potencial dessa abordagem para melhorar processos de automação industrial. As perspectivas de trabalhos futuros indicam um vasto campo de oportunidades para aprimorar e expandir ainda mais esse sistema, contribuindo para o avanço da indústria e da pesquisa em automação robótica.

REFERÊNCIAS BIBLIOGRÁFICAS

AMICE, 1993. "**CIMOSA: Open System Arquitecture for CIM, 2nd. Revised and extended version**". Springer-Verlag, Berlin.

IEC 61131-3:2013. "**International Electrotechnical Commission, Programmable controllers - Part 3: Programming languages**". 207p.

BAUR, C.; WEE, D., 2015. "**Manufacturing's next act**". In McKinsey quarterly, v. 1, n. 5.

BAZZO, W. A.; PEREIRA, L. T. V., 2000. "**Introdução à engenharia**". Florianópolis: UFSC.

BEHANDISH, M.; NELATURI, S.; DE KLEER, J., 2018. "**Automated process planning for hybrid manufacturing**". In Computer-Aided Design, 102, 115-127.

BELHOT, R. V.; FIGUEIREDO, R. S.; MALAVÉ, C. O., 2001. "**O uso da simulação no ensino de engenharia**". In Congresso Brasileiro de Ensino de Engenharia, XXIX COBENGE (pp. 445-451).

BORGO, S., CESTA, A., ORLANDINI, A., UMBRICO, A., 2016. "**A planning-based architecture for a reconfigurable manufacturing system**". In Proceedings of the International Conference on Automated Planning and Scheduling (Vol. 26, pp. 358-366).

BYLANDER, T., 1994. "**The computational complexity of propositional strips planning Artificial Intelligence**", 69, pp. 165-204.

CANTONI, L. F. A., 2010. "**Avaliação do uso da linguagem PDDL no planejamento de missões para robôs aéreos**". Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Brazil, 154p.

CAVALCANTI, L. L.; NOGUEIRA, M. S., 2017. "**Futurismo, Inovação e Logística 4.0: desafios e oportunidades**". VII Congresso Brasileiro de Engenharia de Produção.

CARDOSO, R. N., 2014. "**Contribuição ao estudo de planejamento automático aplicado a sistema de movimentação de materiais**". Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica, 244f.

CASILLO, D., 2018. "**Automação e controle – Aula 03 - Grafcet**". Disponível em: <http://www2.ufersa.edu.br/portal/view/uploads/setores/166/arquivos/Automacao%20e%20Controle%202011_1/Aula%2003%20-%20Grafcet.pdf>. Acesso em: ago. 2022.

CASTILLO, L.A.; ARMENGOL, E.; ONAINDIA, E.; SEBASTIA, L.; GONZÁLEZ-BOTICARIO, J.; RODRGUEZ, A.; FERNÁNDEZ, S.; ARIAS, J. D.; BORRAJO, D., 2008. "**Samap: An user-oriented adaptive system for planning tourist visits**". In Expert Systems with Applications, pp. 1318-1332.

DIAS, G. M., 2014. "**O que é Indústria 4.0?**". Doutor IoT, São Paulo. Disponível em: <https://www.doutoriot.com.br/negocios/industria-40/o-que-e/>. Acesso em: ago. de 2022.

DJEZAIRI, S.; RAZIKA, B. Z.; AKLI, I.; BOUZOUIA, B.; ABDELLAH, P. K., 2019. "**Automated Planning for Mobile Manipulators using passive RFID tags**". In 2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID) (Vol. 1, pp. 1-6). IEEE.

EDELKAMP, S.; HOFFMANN, J., 2004. "**PDDL2.2: The language for the classical part of the 4th international planning competition**". Technical Report 195.

FDEZ-OLIVARES, J.; ONAINDIA, E.; CASTILLO, L. A.; JORDÁN, J.; CÓZAR, J. A., 2019. "**Personalized conciation of clinical guidelines for comorbid patients through multi-agent planning Artificial**". In Intelligence in Medicine, 96, pp. 167-186.

FERREIRA, M., 2009. "**Modelagem de domínios temporais de planejamento com UML**".P. 120 f. Dissertação (Mestrado em Engenharia Elétrica) - Centro Universitário da Fei, São Bernardo do Campo.

FESTO. **MPSa PA Compact Workstation Manual**. Denkendorf: Festo Didactic GmbH Co. KG, 2003.

FIKES, R. E.; NILSSON, N. J., 1972. "**Strips: A new approach to the application of theorem proving to problem solving**". Artificial Intelligence, pp. 189-208.

FONSECA, J. P. da Silva, 2013. "**Analisador de planos para sistemas automatizados baseados em CLPs**". Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica, 180f.

FONSECA, J. P.; DE SOUSA, A. R.; FERREIRA, M. V. M.; DE SOUZA TAVARES, J. J. P. Z., 2016. "**Planpas: Plc and automated planning integration**". In International Journal of Computer Integrated Manufacturing, Vol. 29, n.11, pp. 1200-1217, doi: 10.1080/0951192X.2015.1067909

FOX, M.; LONG, D., 2003. "**PDDL 2. 1: An extension to pddl for expressing temporal planning domains**". In J. Artificial Intelligence Res. (JAIR), pp. 61-124.

FRIEDRICH, C.; AKOS, C.; ARMIN, L.; ALEXANDER, V., 2018. "**Efficient Task and Path Planning for Maintenance Automation Using a Robot System**." IEEE Transactions on Automation Science and Engineering 15.3: 1205-215.

GEREVINI, A.; LONG, D., 2005. "**Plan constraints and preferences in PDDL3 - the language of the fifth international planning competition**". Technical report.

GHALLAB, M.; HOWE, A. E.; KNOBLOCK, C. A.; MCDERMOTT, D.; RAM, A.; VELOSO, M. M.; WELD, D. S.; WILKINS, D., 1998. "**PDDL—the planning domain definition language**". Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CT.

GHALLAB, M.; NAU, D. S.; TRAVERSO, P., 2004. "**Automated Planning—Theory and Practice**". In Elsevier, Amsterdam, Netherlands, 635 p.

GHALLAB, M., NAU, D.S., TRAVERSO, P., 2016. "**Automated planning and acting**". Cambridge University Press.

GÖBELBECKER, M.; KELLER, T.; EYERICH, P.; BRENNER, M.; NEBEL, B., 2010. "**Coming up with good excuses: What to do when no plan can be found**". In Twentieth International Conference on Automated Planning and Scheduling, Toronto, Canada, AAAI Press.

HAND, A., 2018. "**Programming Sequential Function Charts (SFCs) for PLCs: Step-by-Step Guide**." Automation World. Disponível em: <https://www.automationworld.com/factory/programming/article/13319018/programming-sequential-function-charts-sfcs-for-plcs-stepbystep-guide>. Acesso em 11 de março de 2023.

IVANOV, D.; DOLGUI, A.; SOKOLOV, B.; WERNER, F.; IVANOVA, M., 2016. "**A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory Industry 4.0**". In International Journal of Production Research, Vol. 54 No. 2, pp. 386-402.

JI, S.; LEE, S.; YOO, S.; SUH, I.; KWON, I.; PARK, F. C.; LEE, S.; KIM, H., 2021. "**Learning-Based Automation of Robotic Assembly for Smart Manufacturing**". In IEEE, Vol. 109, n. 4, pp. 423-440, doi: 10.1109 / JPROC.2021.3063154.

KABIR, A. M.; KAIPA, K. N.; MARVEL, J.; GUPTA, S. K., 2017. "**Automated planning for robotic cleaning using multiple setups and oscillatory tool motions**". In IEEE Transactions on Automation Science and Engineering, 14(3), 1364-1377.

KAGERMANN, H.; WAHLSTER, W.; HELBIG, J., 2013. "**Recommendations for Implementing the Strategic Initiative Industrie 4.0 Working Group**". Disponível em: <http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf.>. Acesso em 11 de março de 2023.

KANG, H.; LEE, J.; CHOI, S.; KIM, H.; PARK, J.; SON, J.; KIM, B.; NOH, S., 2016. "**Smart manufacturing: past research, present findings, and future directions**". In International Journal of Precision Engineering and Manufacturing – Green Technology, Vol. 3 No. 1, pp. 111-128.

KAUTZ, H.A.; SELMAN, B., 1992. "**Planning as satisfiability**". In: ECAI, Vol. 92, pp. 359–363.

KOU, X.; CAO, Y.; WANG, Q.; QIAO, H.; 2020. "**Sub-assembly recognition algorithm and performance analysis in assembly sequence planning**". In The International Journal of Advanced Manufacturing Technology, 107, 971-981.

KRAKHECHE, I. A., 2007. "**Automação industrial - automação de pequenos processos com CLP**". Centro tecnológico de mecatrônica. Senai - RS.

LEE, J.; LAPIRA, E.; BAGHERI, B.; KAO, H. A., 2013. "**Recent advances and trends in predictive manufacturing systems in big data environment**". In Manufacturing letters, pp. 38-41.

LEE, E.A.; SESHIA, S.A.; SHA, L., 2018. "**Cyber-physical systems: A comprehensive overview**". Proceedings of the IEEE, 100(1), pp.13-28.

LEGAT, C.; VOGEL-HEUSER, B., 2017. "**A configurable partial-order planning appro-**

ach for field level operation strategies of PLC-based industry 4.0 automated manufacturing systems", Engineering Applications of Artificial Intelligence, Volume 66, Pages 128-144, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2017.06.014.

LOM, M.; PRIBYL, O.; SVITEK, M.; 2016. "**Industry 4.0 as a part of smart cities**". In Smart Cities Symposium Prague (SCSP), pp. 1-6.

LU, Y.; XU, J.; LENCHNER, J.C.; LEE, N.S., 2017. "**Smart Factory: Past, Present and Future**".

MAHATO, B.; MAITY, T.; ANTONY, J., 2015. "**Embedded Web PLC: A New Advances in Industrial Control and Automation**". In Second International Conference on Advances in Computing and Communication Engineering, pp. 156-160, doi: 10.1109/ICACCE.2015.141

MALHAN, R. K.; SHEMBEKAR, A. V.; KABIR, A. M.; BHATT, P. M.; SHAH, B.; ZANIO, S.; GUPTA, S. K., 2021. "**Automated planning for robotic layup of composite prepreg**". In Robotics and computer-integrated manufacturing, 67, 102020.

MCCLUSKEY, T. L.; VAQUERO, T. S.; VALLATI, M., 2017. "**Engineering knowledge for automated planning: Towards a notion of quality**". In Proceedings of the Knowledge Capture Conference (pp. 1-8).

MCDERMOTT, D., 1998. "**The PDDL Planning Domain Definition Language**". In AIPS-98 Planning Competition Committee, Pittsburgh Pennsylvania, USA.

MICHNIEWICZ, J.; REINHART, G., 2016. "**Cyber-Physical-Robotics–Modelling of modular robot cells for automated planning and execution of assembly tasks**". In Mechatronics, Vol. 34, pp.170-180.

MUISE, C., 2016. "**Planning.Domains.**" MIT CSAIL, USA.

NEWELL, A.; SIMON, H. A., 1961. "**GPS, a Program that Simulates Human Thought**". In Defense Technical Information Center.

OSTROUKH, A.; NATALIYA, S.; OLEG, V.; VALERY C.; ALEXANDER, B., 2018. "**Automated Process Control System of Mobile Crushing and Screening Plant**." Istrazivanja I Projektovanja Za Privredu 16.3: 343-48.

PLCH, T.; CHOMUT, M.; BROM, C.; BART´AK, R., 2012. "**Inspect, edit and debug PDDL documents: Simply and efficiently with PDDL studio**". In System Demonstrations and Exhibits at ICAPS 15–18.

ROGALLA, A.; FAY, A.; NIGGEMANN, O., 2018. "**Improved domain modeling for realistic automated planning and scheduling in discrete manufacturing**". In 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA) (Vol. 1, pp. 464-471).

SIEMENS Support. Disponível em: <https://www.support.industry.siemens.com/cs/pd/131271?pdti=pi&dl=en&lc=pt-BR>. Acesso em: agosto, 2021.

SIERLA, S.; VILLE, K.; PEKKA, A.; VALERIY, V., 2018. "**Automatic Assembly Planning Based on Digital Product Descriptions**." Computers in Industry 97 (2018): 34-46. Web.

SELMAN, B.; HENRY, A.; KAUTZ COHAN, B., 1994. "**Noise strategies for improving local search**". In Proceedings of the Twelfth National Conference on Artificial intelligence, Vol. 94, pp. 337–343.

SILVA, J. M.; SILVA, J. R., 2019. "**A new hierarchical approach to requirement analysis of problems in automated planning**". In Engineering Applications of Artificial Intelligence, 81, 373-386.

SILVA, L. R. B.; ENDO, W.; LISBÔA, A. R. B. S., 2011. "**Expectativas da utilização de uma planta didática industrial como objeto de aprendizagem em um curso de graduação em engenharia**". In: XXXIX CONGRESSO BRASILERO DE EDUCAÇÃO EM ENGENHARIA, Anais. Blumenau: FURB.

SIOTO, F.; SCHIAVETO, V., 2018. "**Automação e supervisão de planta didática virtual**". In Trabalho de Conclusão de Curso – Engenharia de Controle e Automação. Universidade Tecnológica Federal do Paraná. Curitiba, Brasil, 99 f.

STEFANI, I. M., 2012. "**Uma Solução de Projeto de Automação da Montanha-Russa ZYKLON 40**". Universidade Federal de Viçosa, Centro de Ciências Exatas e Tecnológicas, Departamento de Engenharia Elétrica, Viçosa, Minas Gerais, Brasil.

STROBEL, V., 2015. "**myPDDL - Knowledge Engineering for PDDL**". http://pold87.github.io/myPDDL/. Acesso em: abril de 2021.

THOBEN, K.-D.; WIESNER, S.; WUEST, T., 2017. "**Industrie 4.0' and smart manufacturing-a review of research issues and application examples**". In International Journal of Automation Technology, Vol. 11 No. 1, pp. 4-16.

VAQUERO, T. S.; SILVA, J. R.; FERREIRA, M.; TONIDANDEL, F.; BECK, J. C., 2009. "**From Requirements and Analysis to PDDL in itSIMPLE3.0. In Proceedings**". In 3rd International Competition on Knowledge Engineering for Planning and Scheduling, Thessaloniki, Greece.

WALLY, B.; VYSKOČIL, J.; NOVÁK, P.; HUEMER, C.; ŠINDELAR, R.; KADERA, P.; MAZAK, A.; WIMMER, M., 2019. "**Flexible Production Systems: Automated Generation of Operations Plans Based on ISA-95 and PDDL**". In IEEE Robotics and Automation Letters, Vol. 4, pp. 4062-4069, doi: 10.1109/LRA.2019.2929991.

WALLY, B.; VYSKOČIL, J.; NOVÁK, P.; HUEMER, C.; ŠINDELÁŘ, R.; KADERA, P.; WIMMER, M. 2020. "**Leveraging iterative plan refinement for reactive smart manufacturing systems**". In IEEE transactions on automation science and engineering, 18(1), 230-243.

WHITEHEAD, E.; RUDOLF, F.; KALTENBACH, H. M.; STELLING, J., 2018. "**Automated planning enables complex protocols on liquid-handling robots**". In ACS synthetic biology, 7(3), 922-932.

WOOD, D.; LANE, L.; GILLESPIE, W.; BAKER, S.; ROWBOTTOM, C., 2014. "**The implementation of a PDR 3d-guided gynaecological brachytherapy service in a UK centre**", Journal of Radiotherapy in Practice, Vol. 13 No. 3, pp. 322-331.

WOLLSCHLAEGER, M.; SAUTER, T.; JASPERNEITE, J., 2017. "**The future of industrial**

communication: automation networks in the era of the Internet of Things and Industry 4.0". In IEEE Industrial Electronics Magazine, Vol. 11 No. 1, pp. 17-27.

YOO, J. J.; GÜNAY, E. E.; PARK, K.; TAHAMTAN, S.; OKUDAN KREMER G. E., 2021. "**An Intelligent Learning Framework for Industry 4.0 through Automated Planning**". In Computer Applications in Engineering Education 29.3, 624-40.

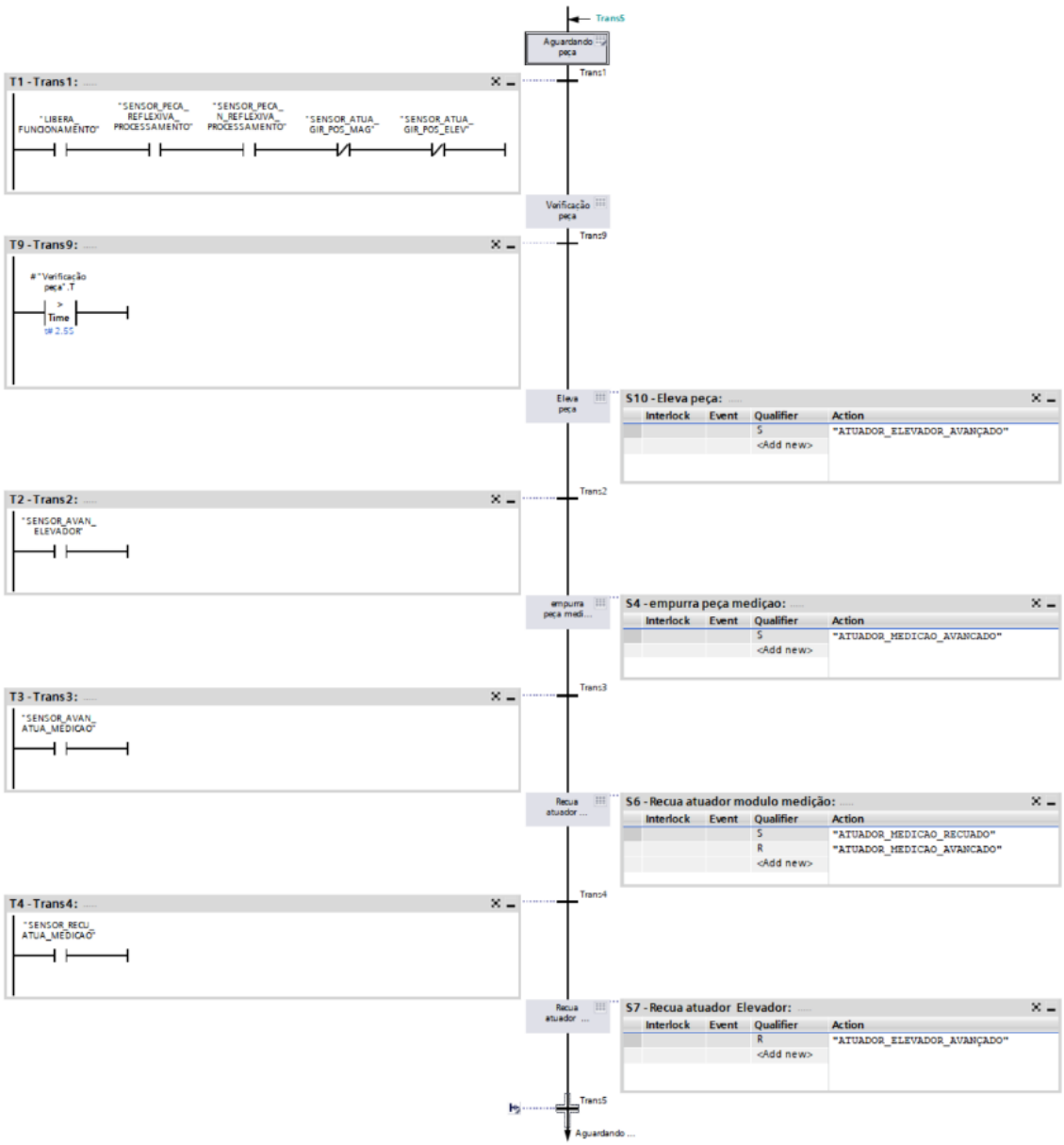
YOUNES, H. L.; LITTMAN, M. L., 2004. "**PPDDL1. 0: The language for the probabilistic part of ipc-4**". In Proc. International Planning Competition.

ZHOU, K., LIU, T. AND ZHOU, L., 2015. "**Industry 4.0: towards future industrial opportunities and challenges**". In 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, August, pp. 2147-2152.

APÊNDICE A

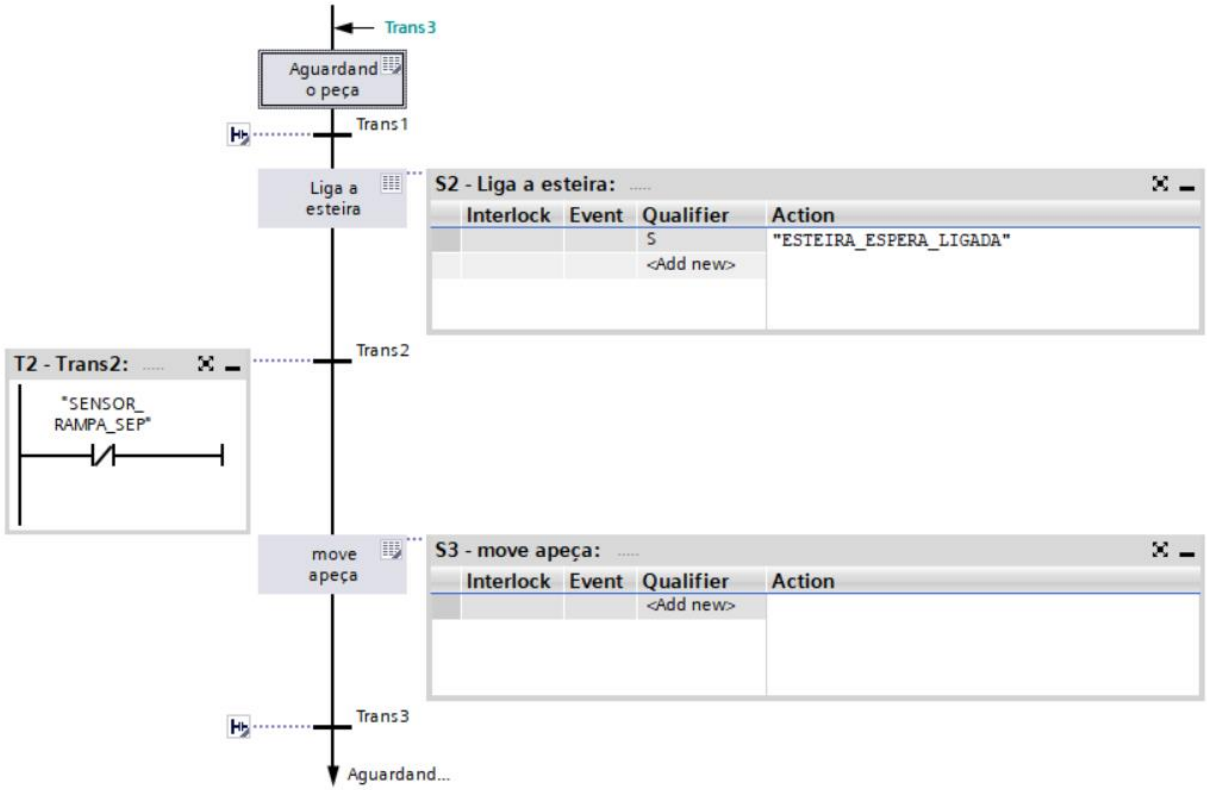
As Figuras 48, 49 e 50 representam as programações SFC das estações Processamento, Espera e Separação respectivamente.

Figura 48. Programação em SFC da estação Processamento.



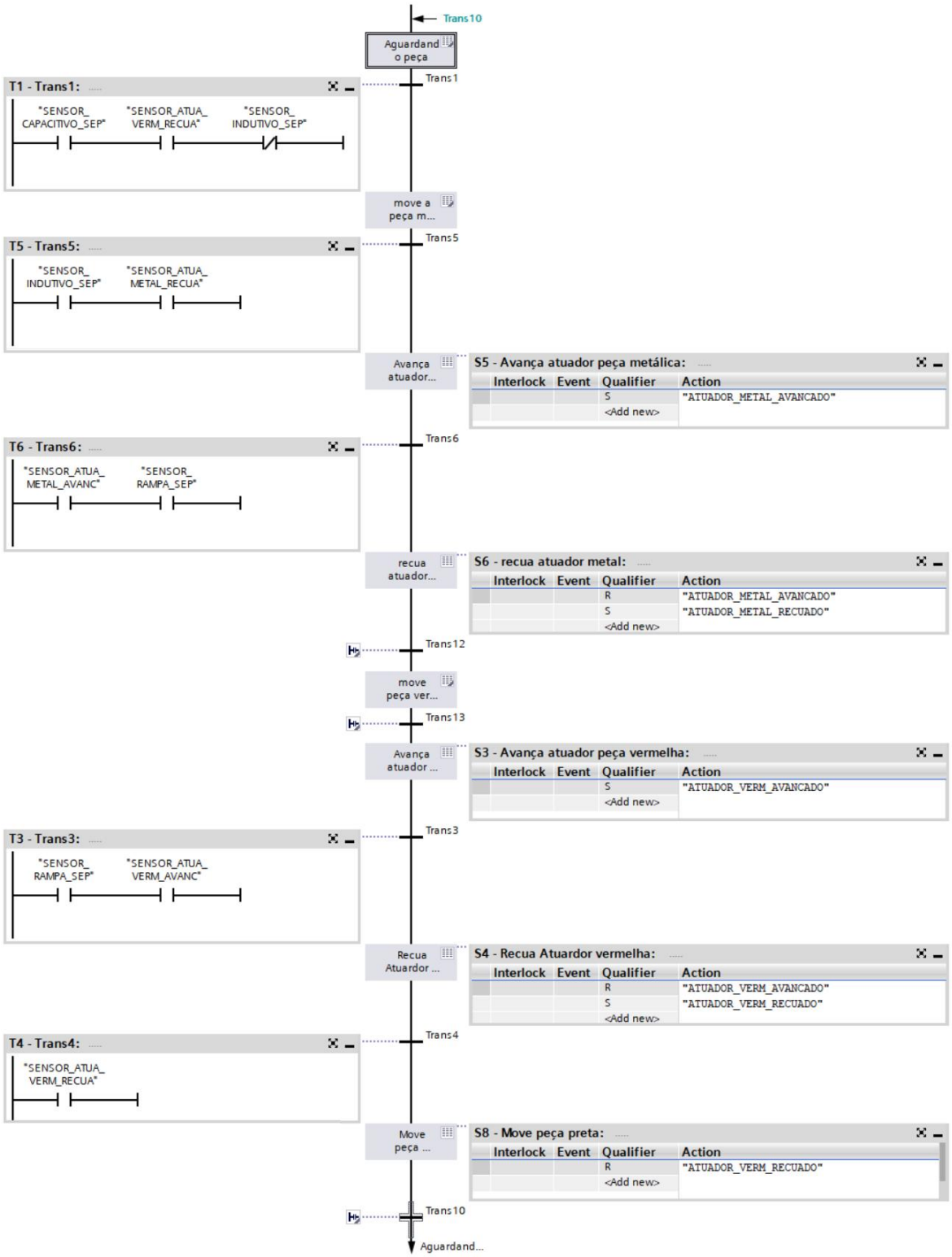
Fonte: TIA Portal V13 (2022)

Figura 49. Programação em SFC da estação Espera.



Fonte: TIA Portal V13 (2022)

Figura 50. Programação em SFC da estação Separação.



Fonte: TIA Portal V13 (2022)