UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO (EMC)
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA (PPGMEC)

HIAGO SILVA MOTTA

Integração on-line entre um sistema didático de manufatura controlado por CLP e um ambiente de planejamento automático hospedado em nuvem

GOIÂNIA
2026

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

Hiago Silva Motta

3. Título do trabalho

Integração on-line entre um sistema didático de manufatura controlado por CLP e um ambiente de planejamento automático hospedado em nuvem

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 20/02/2026, às 16:53, 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 **Hiago Silva Motta**, **Discente**, em 20/02/2026, às 17:28, 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 **5945385** e o código CRC **477FD0F1**.

Referência: Processo nº 23070.064916/2025-26 SEI nº 5945385

HIAGO SILVA MOTTA

Integração on-line entre um sistema didático de manufatura controlado por CLP e um ambiente de planejamento automático hospedado em nuvem

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Engenharia Mecânica, da Escola de Engenharia Elétrica, Mecânica e de Computação, da Universidade Federal de Goiás (UFG), como requisito para obtenção do título de Mestre em Engenharia Mecânica.
Área de concentração: Ciências Mecânicas
Linha de pesquisa: Materiais e Manufatura Avançada

Orientador: Prof. Dr. João Paulo da Silva Fonseca

GOIÂNIA

2026

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

Motta, Hiago Silva
Integração on-line entre um sistema didático de manufatura controlado por CLP e um ambiente de planejamento automático hospedado em nuvem [manuscrito] / Hiago Silva Motta. - 2026.
83 f.: 2026

Orientador: Prof. 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, 2026.
Inclui: siglas, tabelas, lista de figuras, lista de tabelas.

1. Indústria 4.0. 2. Indústria 5.0. 3. Planejamento Automático. 4. Ai-planning. 5. Integração de Sistemas PDDL, Python.

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 **01** da sessão de Defesa de Dissertação de **Hiago Silva Motta**, que confere o título de Mestre em **Engenharia Mecânica**, na área de concentração em **Ciências Mecânicas**.

Aos **vigésimo dia do mês de fevereiro de dois mil e vinte e seis**, a partir das 14**h20**, na sala 212 do Prédio da Engenharia Mecânica, realizou-se a sessão pública de Defesa de Dissertação intitulada **"**Integração on-line entre um sistema didático de manufatura controlado por CLP e um ambiente de planejamento automático hospedado em nuvem**"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **João Paulo da Silva Fonseca (UFG)** com a participação dos demais membros da Banca Examinadora: Professor Doutor **Sérgio Pires Pimentel (UFG)**, membro titular interno e Professor Doutor **José Jean-Paul Zanlucchi de Souza Tavares (UFU)** membro titular externo, cuja participaçao se deu por vídeo conferencia. 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 **vigésimo dia do mês de fevereiro de dois mil e vinte e seis**.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Joao Paulo Da Silva Fonseca**, **Professor do Magistério Superior**, em 20/02/2026, às 16:52, 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 **Sergio Pires Pimentel**, **Professor do Magistério Superior**, em 20/02/2026, às 16:52, 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 **José Jean Paul Zanlucchi de Souza Tavares**, **Usuário Externo**, em 20/02/2026, às 16:53, 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 **5906833** e o código CRC **CF25B1DE**.

Referência: Processo nº 23070.064916/2025-26 SEI nº 5906833

AGRADECIMENTOS

Inicialmente, agradeço a Deus, por conceder sabedoria, saúde e perseverança ao longo de toda a trajetória acadêmica, tornando possível a conclusão deste trabalho.

Agradeço ao meu orientador, Dr. João Paulo da Silva Fonseca, pela orientação criteriosa, disponibilidade e contribuições técnicas e científicas ao longo do desenvolvimento desta trajetória acadêmica, fundamentais para a consolidação de toda a pesquisa.

Ao Programa de Pós-Graduação em Engenharia Mecânica (PPGMEC), da Universidade Federal de Goiás (UFG), agradeço pela infraestrutura, apoio institucional e ambiente acadêmico favorável à realização deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001

Por fim, agradeço à minha noiva, Raianny Souza Lázaro, pela dedicação na revisão noturna do texto, pelo companheirismo, incentivo constante e apoio incondicional durante todas as etapas deste trabalho.

RESUMO

Os controladores lógicos programáveis (CLPs), embora fundamentais para as arquiteturas de automação tradicionais, demandam um tempo considerável para o desenvolvimento da sua programação. A abordagem proposta busca aumentar flexibilidade, modularidade e adaptabilidade, utilizando uma linguagem de programação de alto nível. O trabalho baseia-se no PlanPAS, solução desenvolvida que executa ações abstratas (obtidas de planejadores automáticos) diretamente no campo com o uso de CLPs, sensores, atuadores e redes de comunicação industrial. Tal abordagem é capaz de integrar engenharia baseada em conhecimento (modelagem de domínio, definição de problema e a obtenção de um plano de ações) com a automação industrial aplicada (sensores, atuadores, CLPs e protocolos de comunicação de dados em campo). Apesar do expressivo progresso técnico para a época, o PlanPAS opera de maneira *offline*, metodologia não intuitiva que limita a adaptação dos operadores ao sistema devido à necessidade de reconfiguração manual do plano. A metodologia proposta baseia-se na implementação de uma *Application Programming Interface* (API) desenvolvida em Python, capaz de obter do *planning.domains* (uma plataforma de *AI-Planning*) um arquivo solução que prevê precondições e descreve todas as ações necessárias para a solução de um determinado problema, o qual deve ser estruturado como representações de um cenário inicial e um cenário objetivo. O problema de planejamento é então formalizado com informações obtidas diretamente do campo: Os objetivos a serem alcançados pelo sistema são inseridos pelo operador em uma interface industrial, enquanto as condições iniciais do processo são construídas a partir do estado dos sensores no momento da requisição de uma nova solução. Dentro do sistema proposto, as informações relevantes são trafegadas em dois níveis distintos: da API para o CLP, e vice-versa, via rede Ethernet Industrial com protocolo Modbus TCP; da API para o framework de AI-Planning (PDDL Domains), e vice-versa, via protocolo HTTPS, via métodos POST e GET. A automação de processos físicos baseada em lógica simbólica de planejamento demonstrou uma integração: flexível, testada em mais de um modelo de CLP que possuía comunicação Modbus TCP; escalável, demonstrada através da expansão do domínio; e robusta, apresentando um recurso de contingência, de modo que, em casos críticos, o sistema se mantém operante mesmo com uma eventual perda de comunicação com a API. O trabalho permitiu a integração entre os processos decisórios de alto nível, realizados por um planejador

automático, e a execução de baixo nível, conduzida por meio de CLPs, características de aplicações industriais.

Palavras-chave: Indústria 4.0; Indústria 5.0; Planejamento Automático; AI-Planning; Integração de sistemas PDDL, Python.

ABSTRACT

Programmable Logic Controllers (PLCs), although fundamental to traditional automation architectures, require considerable time for their programming development. The proposed approach aims to increase flexibility, modularity, and adaptability by using a high-level programming language. The work is based on PlanPAS, a developed solution that executes abstract actions (obtained from automated planners) directly in the field through the use of PLCs, sensors, actuators, and industrial communication networks. This approach is capable of integrating knowledge-based engineering (domain modeling, problem definition, and the generation of an action plan) with applied industrial automation (sensors, actuators, PLCs, and field data communication protocols). Despite the significant technical progress achieved at the time, PlanPAS operates in an offline manner, a non-intuitive methodology that limits operator adaptation to the system due to the need for manual reconfiguration of the plan. The proposed methodology is based on the implementation of an Application Programming Interface (API) developed in Python, capable of obtaining from planning.domains (an AI-planning platform) a solution file that specifies preconditions and describes all actions required to solve a given problem, which must be structured as representations of an initial scenario and a goal scenario. The planning problem is then formalized using information obtained directly from the field: the objectives to be achieved by the system are entered by the operator through an industrial interface, while the initial conditions of the process are constructed from the state of the sensors at the moment a new solution is requested. Within the proposed system, relevant information is exchanged at two distinct levels: from the API to the PLC, and vice versa, via an Industrial Ethernet network using the Modbus TCP protocol; and from the API to the AI-planning framework (PDDL Domains), and vice versa, via the HTTPS protocol using POST and GET methods. Physical process automation based on symbolic planning logic demonstrated: a flexible integration, tested on more than one PLC model supporting Modbus TCP communication; scalable, as demonstrated by domain expansion; and robust, presenting a contingency mechanism such that, in critical cases, the system remains operational even in the event of a temporary loss of communication with the API. This work enabled the integration between high-level decision-making processes, performed by an automated planner, and low-level execution, carried out by means of PLCs, which is characteristic of industrial applications.

Keywords: Industry 4.0; Industry 5.0; Automated Planning; AI Planning; PDDL-Python System Integration

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CIP	*Common Industrial Protocol*
CLP	Controlador Lógico Programável
CPS	Sistemas Ciber-Físicos
ERP	*Enterprise Resource Planning*
ENHSP	*Expressive Numeric Heuristic Search Planner*
EtherCAT	*Ethernet for Control Automation Technology*
FBD	*Function Block Diagram*
FBD	*Function Block Diagram*
HTTP	*Hypertext Transfer Protoco*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
IHM	*Interface Homem-Máquina*
IL	*Instruction List*
IP	*Internet Protocol*
LAMA	*Landmark-Cut Heuristic with Anytime Search*
LAN	*Local Area Network*
LD	*Ladder Diagram*
IOT	*Internet of Things*
MA-PDDL	*Multi-Agent-Planning Domain Definition Language*
MES	*Manufacturing Execution Systems*
MQTT	*Message Queuing Telemetry Transport*
NIB	Nova Indústria Brasil
OPC	*Open Platform Communication*
OPC UA	*Open Platform Communications Unified Architecture*
PaaS	*Planning as a Service*
PDDL	*Planning Domain Definition Language*
PDDL +	*Planning Domain Definition Language +*
Plan PAS	*Plans Parser for Automated Systems*
PROFIBUS	*Process Field Bus*

PROFINET	*Process Field Network.*
PUB	*Publish*
RDDL	*Relational Dynamic Influence Diagram Language*
SCADA	*Supervisory Control and Data Acquisition*
SFC	*Sequential Function Chart*
SOA	*Service-Oriented Architecture*
ST	*Structured Text*
STRIPS	*Stanford Research Institute Problem Solver*
SUB	*Subscribe*
TCP	*Transmission Control Protocol*
TIA Portal	*Totally Integrated Automation Portal*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
USB	*Universal Serial Bus*
VAN	*Plan Validation System*

SUMÁRIO

1. INTRODUÇÃO

Por definição, qualquer atividade econômica que realiza processo em alguma matéria prima para obtenção de produtos ou subprodutos para consumo é chamada de indústria (IBGE, 2016). Assim, processos anteriores às revoluções industriais, como artesanato e manufatura, são também tipos de indústria (MARX, 1867). As revoluções industriais são baseadas em tecnologias que compõem os processos fabris de cada época, sendo a primeira revolução industrial marcada pelo processo de fabricação por maquinofatura, onde o ser humano deixou de ser a principal engrenagem e energia do processo de produção (HOBSBAWM, 1962; MARX, 1867).

A partir da segunda revolução industrial, a tecnologia permitia a produção em massa por meio de linhas de montagem e a adoção da energia elétrica nos processos industriais (SCHWAB, 2016). Já na terceira revolução industrial, os computadores e os princípios de automação foram introduzidos também no ambiente fabril (SCHWAB, 2016). Assim, os processos se tornaram cada vez mais específicos e modulares, demandando a conectividade de sistemas ciberfísicos, conceito da indústria 4.0 (SCHWAB, 2016, VYSKOČIL, 2023). A emergente revolução industrial (Indústria 5.0) busca trazer de volta o papel central do ser humano, com foco na sustentabilidade, resiliência e personalização dos produtos (NAHAVANDI, 2019).

No Brasil foi lançada em janeiro de 2024 uma política nomeada de Nova Indústria Brasil (NIB) que possui como objetivo impulsionar o desenvolvimento da indústria com seis missões principais: “cadeias agroindustriais sustentáveis e digitais”; “forte complexo econômico e industrial da saúde”; “infraestrutura, saneamento, moradia e mobilidade sustentáveis”; “transformação digital da indústria”; “bioeconomia, descarbonização e transição e segurança energéticas” e; “tecnologias de interesse para a soberania e a defesa nacionais” (BRASIL, 2024). Um dos principais eixos da NIB é fomentar tecnologias emergentes, digitalização e novos modelos de produção. Dentro da missão "Transformação digital da indústria", a NIB pretende digitalizar 90% das indústrias brasileiras e triplicar a participação da produção nacional no segmento de novas tecnologias." (BRASIL, 2024).

A pirâmide da automação tradicional, apresentada posteriormente no item 2.2.1 desse documento, é um modelo clássico que organiza a hierarquia tecnológica fabril em níveis distintos, partindo dos dispositivos de campo (sensores e atuadores) até os sistemas

de gestão empresarial (ERP). Com a modularidade, automação e integração de todos os níveis da pirâmide da automação, a Indústria 4.0, e a emergente Indústria 5.0, trouxeram novos paradigmas para os sistemas produtivos. Essa evolução tem como objetivo tornar os sistemas mais responsivos às atualizações cada vez mais rápidas do mercado, promovendo flexibilidade e eficiência operacional (HOEBERT, 2023). A crescente demanda de produtos personalizados a partir dos anos 2000 acarretou na redução do ciclo de vida dos produtos e aumento crescente dos negócios personalizáveis, surgindo demanda por ferramentas de planejamento e replanejamento automático para lidar com a dinâmica dos negócios sem comprometer custos e prazos (BURGGRAF *et al.*, 2021). Além disso, sistemas de manufatura que operam em larga escala precisam atender cada vez mais pedidos, cumprir cronogramas mais rígidos, possuir respostas rápidas e dinâmicas aos novos desafios (VYSKOČIL, 2023).

Segundo Marrella (2019), a evolução para a Indústria 4.0 — marcada por ambientes dinâmicos, imprevisíveis e pela crescente disponibilidade de dados — exige maior capacidade de automação, reatividade e flexibilidade na gestão de processos. A automação se destaca especialmente por meio de robôs e sistemas inteligentes, que possuem velocidade, precisão e confiabilidade por longos períodos produtivos (VYSKOČIL, 2023). Entretanto, as arquiteturas tradicionais amplamente difundidas enfrentam limitações significativas com relação a atualizações de produtos ou mudanças na linha de processo que requerem reprogramação manual de controladores lógicos programáveis (CLPs), demandando conhecimento técnico especializado e longo tempo de programação (WALLY *et al.*, 2019). A superação, possibilitando que os sistemas industriais se tornem mais autônomos e inteligentes, depende da adoção de modelos que desacoplem o conhecimento do domínio da lógica de execução (GHALLAB, 2016).

A evolução dos sistemas industriais reforça ainda mais essa tendência enfatizando não apenas a velocidade produtiva, mas também a colaboração entre máquinas e humanos, adaptabilidade, distribuição e dispersão da manufatura (NAHAVANDI, 2023). Assim, a capacidade de raciocínio automático é essencial. Ferramentas utilizando o *Planning Domain Definition Language* (PDDL), linguagem criada para especificar problemas de planejamento de forma declarativa, possuem abordagens promissora para resolução de problemas complexos de lógica de maneira automática (MCDERMOTT *et al.*, 1998). Com a PDDL, é possível descrever os elementos, estados, ações e restrições de um domínio industrial de maneira formal e abstrata (MCDERMOTT *et al.*, 1998;

GHALLAB; NAU; TRAVERSO, 2004), permitindo que planejadores automáticos gerem sequências de ações com base em metas e dispensando reprogramações frequentes em máquinas e dispositivos de automação (FONSECA, 2013).

Os planejadores automáticos realizam a separação entre o "o quê" – expresso pelo domínio de estado do "Sistema" e o objetivo a ser alcançado – e o "como" – a execução concreta, estratégia de busca e sequenciamento de ações realizada pelo "Planejador". Essa separação é fundamental para permitir uma automação orientada por requisições, onde o sistema físico permanece agnóstico à estratégia de solução até que o plano seja gerado (GHALLAB; NAU; TRAVERSO, 2004). Enquanto o planejamento automático ocorre em um *loop* aberto, utilizando raciocínio simbólico abstrato, a execução contínua, aplicada em CLPs industriais, operara em um loop fechado, garantindo robustez e confiabilidade operacional (FONSECA, 2016). Devido à natureza heterogênea dessas soluções lógicas, sua integração direta se torna complexa.

Para transpor essa barreira, as *Application Programming Interfaces* (APIs) são interfaces programadas para estabelecer contratos de comunicação, permitindo a integração entre dispositivos modulares. Graessler e Pöhler (2017) destacam a importância da adoção de sistemas de produção ciberfísicos dotados de capacidade de autocontrole, capazes de ampliar a colaboração homem–máquina sem comprometer a autonomia humana nos sistemas fabris. Assim, com o uso da API, a integração entre CLPs, que operam com linguagens de baixo nível, e sistemas baseados em PDDL, que modelam o conhecimento em alto nível, cria uma ponte entre os diferentes níveis da pirâmide de automação — da gestão ao campo — promovendo um sistema cognitivo, alinhado aos princípios de colaboração proposto pela Indústria 5.0. Essa abordagem viabiliza a reprogramação contínua e eficiente dos sistemas industriais, elevando sua capacidade de adaptação diante de mudanças de produto, demanda ou contexto.

1.1 Objetivos

Desenvolver e validar uma arquitetura integrada que conecte o processo decisório de alto nível hospedado em nuvem, baseado em planejamento automático com linguagem em PDDL, à execução de baixo nível em equipamentos industriais controlados por CLPs via protocolo Modbus TCP, garantindo modularidade, flexibilidade e operação *online*.

1.1.2 Objetivos Específicos

i. Projetar e implementar uma *Application Programming Interface* (API) em Python capaz de intermediar a comunicação entre o ambiente de planejamento automático hospedado em nuvem e dispositivos de controle industrial tradicional, em execução no campo.
ii. Criar um mecanismo de geração de arquivo solução em nuvem retornando todas as precondições e ações necessárias para atender solicitações em ambiente produtivo.
iii. Desenvolver um sistema ciberfísico capaz de sincronizar decisões lógicas e ações físicas *online*, considerando dados obtidos de sensores e atuadores.
iv. Implementar um sistema de monitoramento para validar precondições e efeitos das ações, garantindo segurança e confiabilidade em ambientes multitarefa.
v. Realizar testes experimentais em ambientes didáticos controlados, avaliando o desempenho, robustez e adaptabilidade da solução frente a diferentes demandas produtivas.

1.2 Justificativa

O aumento da complexidade dos sistemas produtivos, impulsionado pela Indústria 4.0 e diretrizes emergentes da Indústria 5.0, exige soluções capazes de integrar a lógica simbólicas de alto nível com o controle físico de máquinas e processos para alcançar como adaptabilidade, sustentabilidade e de sistemas cada vez mais colaborativos. Os ciclos de vida dos produtos cada vez mais curtos e as novas demandas produtivas necessitam de arquiteturas modulares, flexíveis e adaptáveis, capazes de responder às mudanças dinâmicas. A utilização do planejamento automático por meio da linguagem em PDDL permite modelar, antecipar ações e precondições necessárias para a execução de tarefas, mas sua aplicação prática requer uma integração robusta e segura com o hardware industrial. O protocolo Modbus TCP, amplamente difundido, oferece um meio padronizado de comunicação com CLPs, tornando viável a implementação em diferentes cenários industriais. Entende-se que a proposta deste trabalho não apenas contribui para o avanço da automação baseada em lógica simbólica, mas também oferece uma solução prática e de baixo custo para sincronizar decisões computacionais e ações físicas, potencializando a eficiência e a confiabilidade dos processos produtivos atuais.

1.3 Organização da dissertação

Capítulo 1 – Introdução: Apresenta o contexto e a motivação da pesquisa, o problema investigado, os objetivos gerais e específicos, a justificativa e a relevância do estudo.

Capítulo 2 – Referencial Teórico: Reúne e discute os conceitos fundamentais para a compreensão da proposta, abordando fundamentos de automação, planejamento automático, linguagem PDDL, integração de sistemas de alto e baixo nível e comunicação e programação industrial via CLP.

Capítulo 3 – Metodologia: Descreve detalhadamente as etapas de desenvolvimento da pesquisa, abrangendo a modelagem do domínio em PDDL, o projeto e implementação da API em Python, a integração com CLPs, a definição dos cenários de teste e abordagem para monitoramento.

Capítulo 4 – Resultados e Discussão: Apresenta os resultados obtidos com a aplicação da solução proposta, analisando o desempenho, a robustez, a adaptabilidade e a conformidade com os objetivos definidos, além de comparar os achados com trabalhos correlatos.

Capítulo 5 – Conclusões e Trabalhos Futuros: Resume as contribuições do estudo, discute as limitações encontradas e indica perspectivas para aprimoramentos e pesquisas futuras.

2. REFERENCIAL TEÓRICO

Inicialmente, será demonstrada a relevância do tema, fundamentada com a análise bibliométrica e cienciométrica da literatura. Assim, selecionando os principais trabalhos e a validação do impacto associados ao tema. Posteriormente, será apresentado toda fundamentação teórica ao tema.

2.1. Método de revisão bibliográfica sistemática

Esse método tem como objetivo alcançar os artigos que possuem maior intersecção de assunto com o tema: "Integração entre um sistema de manufatura controlado por CLP e um ambiente de planejamento automático". Utilizando as bases de dados SCOPUS (ELSEVIER, 2024) e *Web of Science* (WoS) (CLARIVATE, 2024) e analisando os dados no "Google Colaboratory" como ambiente de desenvolvimento integrado (IDE) e a biblioteca PyBibX – uma biblioteca bibliométrica e cienciométrica alimentada por ferramentas de inteligência artificial – elaborada por Pereira (2024) – e simplificando o método de pesquisa, foram criados três grupos de palavras com objetivo principal de aumentar os resultados com a utilização dos seus sinônimos:

1° Grupo: "*AI Planning*" ∪ "*Artificial Intelligence Planning*" ∪ "*Automated Planning*";
2° Grupo: "*Industry*" ∪ "*Manufacturing*";
3° Grupo: "*Control**";

onde o operador "∪" representa união de palavras-chave sinônimas, enquanto o operador "*" aplica-se à busca por palavras cujo radical seja formado pelo termo "Control".

A partir desses grupos de palavras principais, procedeu-se com a intersecção entre elas com o objetivo de promover melhor aderência da busca ao tema de interesse. Foram testadas sete combinações distintas, conforme apresentado na Tabela 1.

Tabela 1 – Combinações de grupos de palavras.

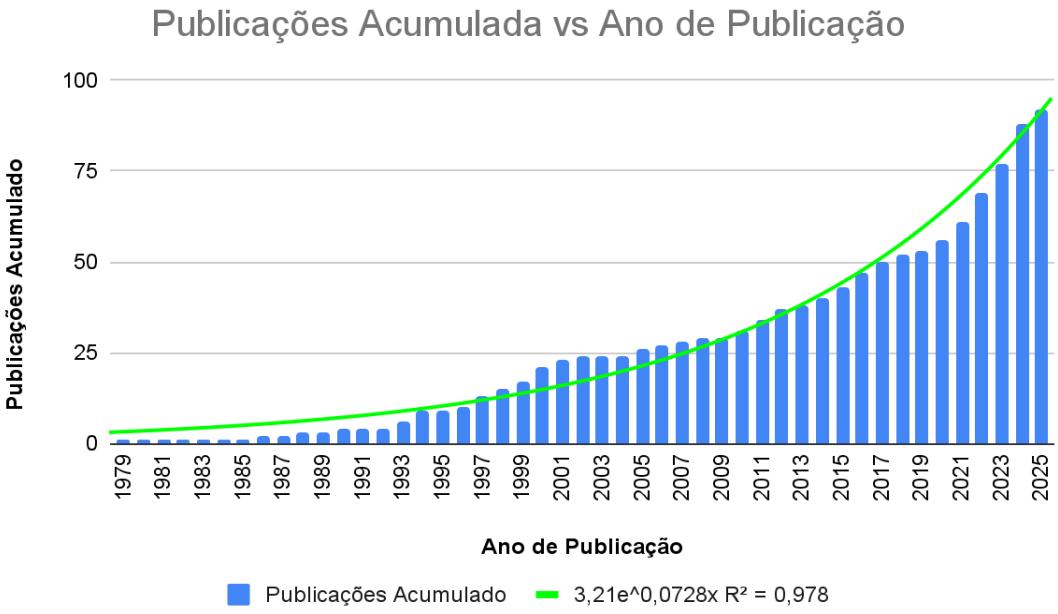
Combinação	Intervalos				
	1° Grupo	∩	2° Grupo	∩	3° Grupo
1°	X				
2°			X		
3°					X
4°	X	X	X		
5°	X			X	X
6°			X	X	X
7°	X	X	X	X	X

Fonte: Próprio autor.

Limitando os tipos de documentos para trabalhos de conferencias e artigos de ambos bancos de dados e realizando a intersecção dos grupos de palavras, foi ainda necessário retirar os trabalhos duplicados comuns aos bancos de dados Scopus e WoS utilizando a biblioteca PyBibX no Google Colaboratory que possibilitou afunilar 31.996.427 artigos em 92 artigos com maior aderência ao tema proposto.

Ainda com a utilização do PyBibX é possível extrair estudos de todos os artigos de maneira automatizada. A fim de determinar o potencial do tema, foi verificada a escala de produção de documentos e a Figura 1 apresenta o gráfico do número de publicações acumuladas que envolvem a temática pelo ano de publicação. Ainda pelo gráfico da Figura 1, é evidenciado o aquecimento na abordagem do tema até o momento, onde a regressão se aproxima de uma linha de tendência exponencial, que demonstra o aquecimento da abordagem da temática.

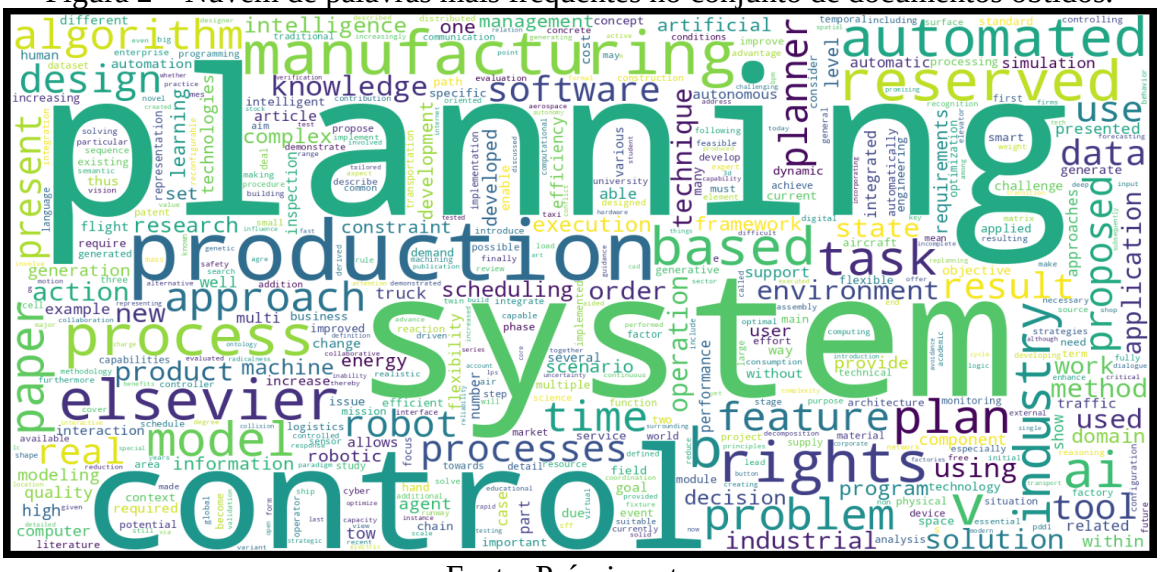
Figura 1 – Evolução do número de publicações ao longo do período considerado.



Fonte: Próprio autor.

Na análise bibliométrica, gerou-se uma lista de palavras mais frequentes, e uma equivalente nuvem de palavras, na qual o tamanho dos termos é proporcional à sua ocorrência nos documentos analisados, conforme a Figura 2. Nesta análise, as palavras são classificadas com graus de importância onde as cinco principais foram “planning”, “system”, “control”, “production” e “manufacturing”, respectivamente.

Figura 2 – Nuvem de palavras mais frequentes no conjunto de documentos obtidos.



Fonte: Próprio autor.

Contudo, apesar de grande aderência ao tema, foi necessário classificar a qualidade dos trabalhos, e para esse propósito, foi adequada a fórmula do *InOrdination*, aqui tratada como Fórmula de Priorização, de Pagani (2015) que abstrai o número de citações, ano de publicação, quantidade de citações por trabalho publicados nas revistas/jornais e o percentil dos artigos.

$$Fator\ de\ priorização = (Ci \times 0{,}35) + (An \times 0{,}25) + (Cs \times 0{,}2) + (Pe \times 0{,}2) \qquad (1)$$

Ci: Número de citações do trabalho
An: Ano de publicação
Cs: Quantidade de citações por trabalhos publicados pela revista/jornal
Pe: Percentil dos trabalhos

Após a classificação dos documentos, utilizando todos os dados normalizados, foram analisados os 15 melhores artigos com maior fator de priorização (*Fp*), apresentados na Tabela 2, obtendo assim uma lista com os trabalhos com maior aderência e qualidade acadêmica.

Tabela 2 – Os 15 trabalhos mais aderentes e relevantes ao tema.

Autor	Título	*Fp*
(GRAESSLER, 2017)	*Integration of a digital twin as human representation in a scheduling procedure of a cyber-physical production system*	0,76
(PATEL, 2025)	*Ai patent approvals in service firms, patent radicalness, and stock market reaction*	0,66
(CIORTEA, 2018)	*Repurposing manufacturing lines on the fly with multi-agent systems for the web of things*	0,65
(MARCHETTA, 2010)	*An artificial intelligence planning approach to manufacturing feature recognition*	0,64
(HOEBERT, 2023)	*Knowledge-driven framework for industrial robotic systems*	0,63
(MCCLUSKEY, 2017)	*Embedding automated planning within urban traffic management operations*	0,63
(LAZAROIU, 2024)	*Cognitive digital twin-based internet of robotic things, multi-sensory extended reality and simulation modeling technologies, and generative artificial intelligence and cyber-physical manufacturing systems in the immersive industrial metaverse*	0,59
(MAY, 2021)	*Product generation module: automated production planning for optimized workload and increased efficiency in matrix production systems*	0,59
(WALLY, 2021)	*Leveraging iterative plan refinement for reactive smart manufacturing systems*	0,57

(PALOMBARINI, 2012)	*Smartgantt - an intelligent system for real time rescheduling based on relational reinforcement learning*	0,57
(VYSKOČIL, 2023)	*A digital twin-based distributed manufacturing execution system for industry 4.0 with ai-powered on-the-fly replanning capabilities*	0,56
(AGRAWAL, 2024)	*Moving toward lean construction through automation of planning and control in last planner system: a systematic literature review*	0,54
(BINNARD, 2000)	*Design by composition for layered manufacturing*	0,52
(HINOSTROZA, 2024)	*Temporal mission planning for autonomous ships: design and integration with guidance, navigation and control*	0,51
(HIND, 2024)	*Cynical technical practice: from ai to apis*	0,50

Fonte: Próprio autor.

A relevância desse trabalho é sustentada pela analise dos trabalhos mais relevantes com maior aderência ao tema e alto impacto. Vyskočil (2023) e Wally (2021) reforçam a necessidade de sistemas com capacidade de replanejamento em tempo real, enquanto autores como Hind (2024) validam a arquitetura baseada em APIs como tendência moderna de integração. Além disso, como discutido por Lazaroiu (2024) e Graessler (2017), a inserção deste tema nos conceitos de Indústria 5.0 demostra que a automação estocástica é o caminho para a resiliência produtiva exigida atualmente em contraste com a degradação humana frente aos sistemas fabris existente.

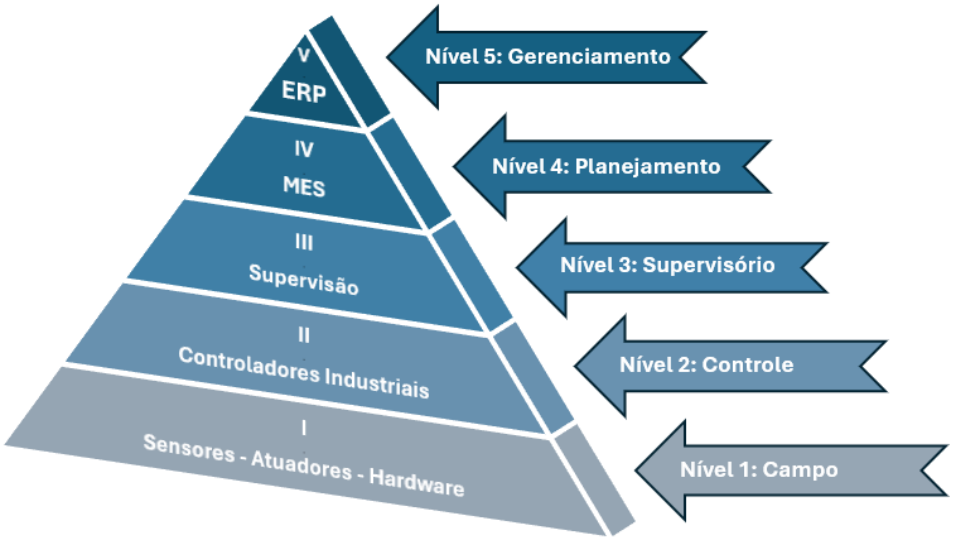
2.2. Automação Industrial

2.2.1. Pirâmide da automação

A ISA-95 é a norma que padroniza a integração entre sistemas de controle de manufatura e sistemas corporativos, conhecida como pirâmide de automação, para facilitar a comunicação de informações da produção à gestão empresarial. Dispositivos físicos até os sistemas corporativos são representados na pirâmide de automação industrial, ilustrada na Figura 3, que representa uma estrutura hierárquica que organiza os diferentes níveis funcionais de uma planta. Responsáveis por interagir diretamente com os processos físicos, na base da pirâmide estão os sensores e atuadores, chamado nível de campo (*Field Level*). Logo acima, no nível de controle (*Control Level*), encontram-se os dispositivos lógicos, como CLPs, controladores PID e computadores industriais, que processam os sinais do campo e executam decisões em tempo real. Mais acima se encontra o nível de supervisão (*Supervisory Level*), que permitem a visualização,

o monitoramento e o controle de diferentes áreas da planta por meio de interfaces gráficas e alarmes, que atuam os sistemas *Supervisory Control and Data Acquisition* (SCADA) (SMC, 2025). O penúltimo nível é o de planejamento (*Planning Level*), responsável por fazer a ponte entre a produção e a gestão, assim, sendo responsável por decisões como o sequenciamento da produção, o controle de qualidade e a manutenção preventiva e que são ocupados pelos sistemas *Manufacturing Execution Systems* (MES) (MONOSTORI, 2016). Por fim, no topo da pirâmide está o nível de gestão (*Management Level*), onde sistemas *Enterprise Resource Planning* (ERP) integram todas as áreas da empresa, e são utilizados para decisões estratégicas e corporativas, mas não comandam diretamente ações no nível de produção.

Figura 3 – Organização da pirâmide da automação.



Fonte: Próprio autor baseado em SMC, 2025.

Tradicionalmente, a programação e o controle dos sistemas de automação são realizados nos níveis mais baixos da pirâmide, principalmente nos níveis de campo e controle, utilizando linguagens de baixo nível e lógica sequencial implementadas diretamente nos CLPs. Essa abordagem, embora robusta, impõe limitações em termos de flexibilidade e integração com sistemas de tomada de decisão em níveis superiores (VOGEL-HEUSER, 2016). A pirâmide tradicional funciona como lote de dados, com os níveis desacoplados e o ERP não conversa com o CLP, existindo uma dependência humana ou de um *middleware* complexo para levar a necessidade do cliente até o chão de fábrica (ALMADA-LOBO, 2016). Em resumo, se o cliente alterar a requisição, a pirâmide clássica não consegue reconfigurar a produção instantaneamente, pois a lógica

“trava” em níveis inferiores. Alterações na lógica de produção ou na ordem das operações geralmente exigem reprogramação manual, o que demanda tempo e conhecimento técnico especializado.

A Indústria 4.0 e 5.0 necessitam de uma malha de dados, que representa o colapso da pirâmide onde o cliente é conectado diretamente com o chão de fábrica quebrando a rigidez hierárquica e tornando-se um ecossistema totalmente integrado (MONOSTORI, 2014; LEITAO *et al.*, 2016). Na proposta desse trabalho, os controladores industriais deixam de deter a “inteligência do processo” enquanto houver conexão com a rede e passam apenas a atuar como um executor de comandos enviados por um planejador.

2.2.2. Conceitos de Indústria 4.0 e Indústria 5.0

A Indústria 4.0 é marcada pela criação dos Sistemas Ciber-Físicos (CPS), onde o mundo físico e o digital são integrados, muito comumente associado a *Internet of Things* (IoT) (SCHWAB, 2016). Nesse contexto, a Indústria 4.0 traz como principais características a interoperabilidade, permitindo que as máquinas, sensores e pessoas se comuniquem via redes; virtualização, criando uma cópia virtual do mundo real (Gêmeos Digitais); descentralização, tornando os sistemas capazes de tomar decisões simples de forma autônoma; capacidade de processamento em tempo real, coletando e analisando dados instantaneamente para suporte à decisão e orientação a serviços, com a adoção de arquiteturas de software que oferecem aplicações complexas a partir de funcionalidades menores e independentes (ALMADA-LOBO, 2016).

Por outro lado, a emergente Indústria 5.0, foca diretamente no bem estar do fator humano de volta ao centro da produção. Logo, as principais tecnologias empregadas estão associadas a colaboração de humano-máquinas (como a adoção de robôs colaborativos), sustentabilidade, resiliência e, por consequência, customização em massa (EUROPEAN COMMISSION, 2021). Os sistemas da Indústria 5.0 devem contar com técnicas de automação estocásticas, que são capazes de prever, detectar e reagir a eventos imprevistos garantindo a segurança e contingenciamento dos sistemas produtivos em tempo real.

Enquanto a Indústria 4.0 estabeleceu por meio das tecnologias implementadas a conectividade necessária para a digitalização, a Industria 5.0 introduz a necessidade de sistemas resilientes e centrados no humano. Nesse cenário, o desenvolvimento de sistemas de contingenciados, diferente das abordagens clássicas, torna-se imprescindíveis

para operar sob incertezas não apenas para alcançar metas produtivas, mas também prever ações de recuperação diante de falhas, garantindo a continuidade operacional e a segurança (LENG *et al.*, 2022).

2.2.3. CLP

Um CLP é um controlador utilizado principalmente na indústria para automatizar processos, realizando o controle e monitoramento de máquinas e sistemas através de programas lógicos armazenados em sua memória (PETRUZELLA, 2014). Projetado para operar em ambientes industriais severos, oferecendo maior flexibilidade permitindo sua aplicação em projetos distintos, maior confiabilidade e facilidade de reprogramação, assim o CLP substituiu sistemas de controle baseados em relés (MAZUR, WEINDORF, 2021). O funcionamento se dá a partir da leitura de entradas digitais e analógicas, processamento lógico do programa definido, e a atuação nas saídas para controle dos dispositivos físicos, permitindo a automação dos variados processos industriais (BRINKSMA; MADER, 2006).

A popularização ocorreu nas décadas de 1970 e 1980 e os CLPs tornaram-se o padrão para controle industrial. Além da possibilidade de se integrar mais facilmente a sistemas supervisórios, proporcionou maior facilidade de adaptação, manutenção e integração dos sistemas elétricos industriais (PETRUZELLA, 2014; MAZUR, WEINDORF, 2021). Sua popularização deu-se de forma massiva em vários setores de manufatura, energia e até mesmo na automação predial, reforçando sua importância e a dependência estabelecida para garantir eficiência operacional na indústria (BOLTON, 2015).

A programação de CLPs, embora possa demandar bastante tempo de engenharia, é crucial para a automação industrial, exigindo a definição do mapa de endereços, linguagens de programação padronizada de acordo a IEC 61131-3 e a configuração/parametrização dos meios de comunicação de CLP em rede de comunicação industrial.

O mapa de endereços organiza a memória do CLP em áreas específicas para entradas, saídas, temporizadores, contadores e variáveis internas, e sua definição precisa é indispensável para garantir que o programa interaja corretamente com os componentes

físicos da planta, evitando erros e promovendo a rastreabilidade das variáveis (BOLTON, 2015).

A norma IEC 61131-3 é responsável por padronizar cinco linguagens de programação para CLPs, conferindo interoperabilidade entre fabricantes e flexibilidade metodológica. Dentre elas, a *Ladder Diagram* (LD) é amplamente utilizada em aplicações simples, devido à sua semelhança com esquemas elétricos e facilidade de compreensão visual. A *Structured Text* (ST), linguagem textual inspirada em Pascal e C, é adequada para algoritmos complexos, cálculos e manipulação de dados. O *Function Block Diagram* (FBD) utiliza lógica modular e reusável. Por fim, *Instruction List* (IL) e *Sequential Function Chart* (SFC), são aplicáveis em casos específicos como controle sequencial (SILVA, 2016).

A comunicação entre CLPs e demais sistemas é basilar para a descentralização da automação. Cada projeto de automação depende das exigências específicas da aplicação, fabricantes e a escolha do protocolo de comunicação – que considera fatores como tempo de latência, segurança do envio de dados, interoperabilidade entre dispositivos e a complexidade da rede. Protocolos como Modbus TCP, EtherNet/IP e PROFINET são usualmente adotados por suas características de interoperabilidade e desempenho. O Modbus TCP destaca-se por sua simplicidade e implementação por código aberto (*open-source*), com arquitetura cliente-servidor, facilitando a leitura e a escrita de registradores. Contudo, não é determinístico, o que limita seu uso em aplicações que exigem sincronismo estrito. O EtherNet/IP, baseado no *Common Industrial Protocol* (CIP), permite comunicação produtor-consumidor com suporte a tempo real e segurança. Por fim, o PROFINET, desenvolvido pela fabricante Siemens, é amplamente difundido na automação industrial e combina a flexibilidade do Ethernet padrão com mecanismos de comunicação em tempo real, como também é eficiente em aplicações em que a comunicação é primordial, mantendo uma alta disponibilidade, diagnósticos avançados e integração com sistemas complexos.

2.2.4. Redes de comunicação industrial

A arquitetura de comunicação consiste na estruturação e nos princípios que definem a forma como os dispositivos, sistemas e a rede trocam informações a partir da definição dos protocolos de comunicação, modelos de referência, camadas e métodos, para garantir

a troca de dados em ambientes diversos e distribuídos (TANENBAUM; WETHERALL, 2021). A arquitetura não é estabelecida apenas por canais físicos de conexão, mas também impõem as regras e os meios para possibilitar a integração e a hierarquia dos processos. Uma boa arquitetura de comunicação, deve atender a requisitos específicos do projeto e garantir baixa latência, alta robustez e compatibilidade entre os equipamentos dos diversos fabricantes de automação (SCHNEIDER, 2020).

A arquitetura não se resume aos protocolos utilizados, mas também em estabelecer conexão, possibilidade de controle, sincronização, garantir a segurança e gestão da rede. Isso possibilita uma automação bem distribuída e permite a supervisão remota de todo processo de diferentes subsistemas e níveis hierárquicos da indústria. A estrutura modular e escalável viabiliza atender às demandas com flexibilidade e confiabilidade nos processos industriais.

A arquitetura de comunicação em sistemas industriais se dá pela organização e definição do modo pelo qual os dispositivos, controladores e sistemas de supervisão trocam informações que, frequentemente, são compostas por diversos níveis que garantem interoperabilidade, velocidade na transmissão dos dados, segurança e robustez para os processos (TANENBAUM; WETHERALL, 2021). Com o intuito de padronizar as mensagens e o controle do fluxo de informações entre os dispositivos da rede industrial, atualmente há diversos protocolos de comunicação disponíveis no mercado como Modbus, Profibus, PROFINET e EthernetIP que podem ser aplicados em diversos níveis e camadas da arquitetura.

O Modbus por exemplo, destaca-se pela simplicidade e alta compatibilidade, operando também em diversos tipos de meios físicos, como RS-485 (ModbusRTU) e Ethernet (Modbus TCP); contudo, conta com limitações de desempenho e determinismo (GAMESS, 2020). O Profibus, por sua vez, atua com uma arquitetura de campo dedicada, que traz alta robustez e confiabilidade na comunicação dos controladores com os dispositivos de campo (FALKIEWICZ, 2024). O PROFINET é uma evolução, trazida pela Siemens, das arquiteturas tradicionais e opera como uma rede Ethernet industrial, compatibilizando melhor com mecanismos de comunicação em tempo real, diagnósticos avançados e versatilidade nas aplicações complexas da Indústria 4.0 (FELD , 2004).

Outros protocolos, como EtherNet/IP, EtherCAT, OPC UA e MQTT, podem fazer parte das arquiteturas de comunicação, cada um atuando em camadas específicas e com

funções distintas — desde o transporte rápido e determinístico de dados, até a integração segura e escalável com sistemas corporativos e em nuvem (HOELBERT *et al.*, 2023).

Nesse contexto de expansão da conectividade, a IoT surge como base que sustenta a dispersão da linha de fábrica. Diferente das arquiteturas tradicionais centralizadas, a IoT permite que unidades produtivas organizadas em células de trabalhos e ativos que estão geograficamente distribuídos possam operar de forma integrada e colaborativa (JESCHKE *et al.*, 2017). A dispersão exige que a comunicação transite do *Local Area Network* (LAN) para as arquiteturas baseadas em nuvem e borda, permitindo que o objetivo do cliente interaja mais diretamente com o chão de fábrica *online*, independente da sua localização física.

Para viabilização da comunicação de sistemas dispersos e com restrições de largura de banda, o protocolo MQTT (*Message Queuing Telemetry Transport*) tornou-se padrão para a IoT industrial. Os protocolos de campo tradicionais são baseados em *polling* (mestre-escravo) e, em contraste, o MQTT utiliza um modelo de *publish/subscribe* (Pub/Sub) que utiliza um *broker* (servidor central) que media a troca de mensagens entre os dispositivos *client*. Essa configuração permite que os dispositivos locais remotos ou com conexões instáveis enviem dados apenas quando houver mudanças de estados, ou seja, é uma comunicação baseada em eventos que garante alta escalabilidade e baixo consumo de recursos (HOU *et al.*, 2021; IBN-KAHLA, 2023). Assim, enquanto protocolos como OPC UA e EtherCAT garantem o determinismo no nível de controle local, o MQTT atua na integração vertical e horizontal da pirâmide de automação permitindo a aplicação de uma linha de fábrica mais dispersa, que seja monitorada e reconfigurada em tempo real por meio de APIs e sistemas de planejamento inteligente.

2.2.5. API

Uma API é um conjunto de códigos que padroniza as rotinas, protocolos e definições, permitindo a integração entre diferentes softwares que abstrai a complexidade específica e fornece uma interface clara para o uso de funcionalidades de um sistema ou serviço. Essa abordagem permite a integração de sistemas modulares e acúmulo de funções, sendo amplamente empregada em arquiteturas propostas pela indústria 4.0, sistemas baseados em serviços (*Service-Oriented Architecture*, SOA) e arquiteturas de microsserviços (FIELDING, 2000; PAPAZOGLOU; VAN DEN HEUVEL, 2007).

Acerca da integração de sistemas, as APIs são como contratos de comunicação que estabelecem os formatos dos pedidos e reações que permitirão a interoperabilidade de aplicações heterogêneas, sejam elas locais ou distribuídas.

O desenvolvimento de uma arquitetura integrada entre planejamento automático e controle físico de processos industriais requer uma API robusta, capaz de intermediar a comunicação entre o software de alto nível e os dispositivos de campo. A API tem que possuir capacidade para se comunicar com diversos serviços autônomos e, na maioria das vezes, possuir a capacidade de se comunicar com diversos protocolos.

A linguagem Python destaca-se nesse contexto por possuir uma grande gama de bibliotecas, simplicidade sintática e suportar múltiplos protocolos industriais, como Modbus, OPC UA, Ethernet/IP e MQTT. O Modbus se sobressai devido à baixa complexidade de implementação e interoperabilidade entre diferentes fabricantes de CLPs, características que se traduzem em modularidade.

Entre as bibliotecas mais relevantes para Modbus/TCP em Python, está o pymodbus e o pyModbusTCP, uma implementação pura em Python que não possui dependências externas e que oferece suporte à leitura e escrita em registradores por meio de interface IP, utilizando métodos como *read_holding_registers* e *write_multiple_registers* (LEFEBVRE, 2024). O pymodbus apresenta uma implementação completa do protocolo Modbus, incluindo clientes e servidores, APIs síncronas e assíncronas, além de utilitários para simulação, codificação/decodificação e suporte a múltiplos meios de comunicação como TCP, UDP, serial e TLS (COLLINS; CHO, 2025). A biblioteca pymodbus é bastante difundida em cenários de pesquisa e desenvolvimento por conta da sua grande cobertura de funcionalidades, flexibilidade de comunicação e capacidade de integração com diferentes meios físicos, o que a torna adequada para projetos de integração.

A interface IP é elementar para a integração com CLPs, permitindo que a API utilize a rede Ethernet para encapsular pacotes Modbus/TCP sobre IP, assegurando a transmissão de comandos e dados em tempo real para múltiplos dispositivos na rede. O Modbus estabelece a conexão e mantém a comunicação, identificando os dispositivos alvos pelo *unit identifier*, enquanto gerencia o tráfego de acordo com as especificações do protocolo.

A manipulação de dados no CLP pode ser realizada por meio de funções específicas para leitura e escrita, que acessam regiões de memória como *coils*, *discrete*

inputs, *holding registers* e *input registers*. Cada região possui endereços e formatos de dados distintos, e esses acessos exigem atenção especial em ambientes *multithreaded* para evitar a leitura simultânea em registradores por meio de duas ou mais *threads*, que pode levar a inconsistência dos dados.

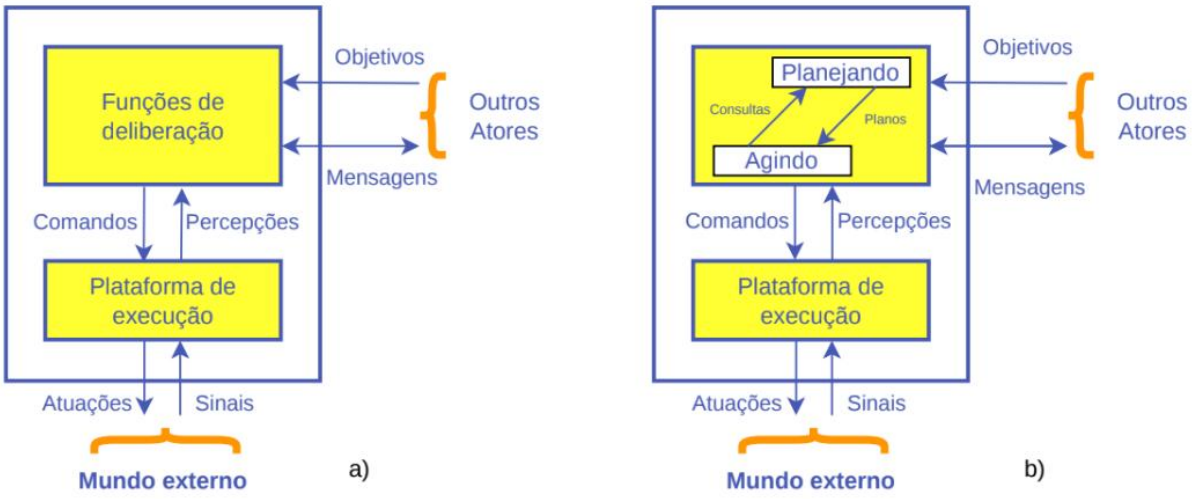
2.3. Planejamento automático

O planejamento automático, ou *AI Planning*, é um ramo da inteligência artificial que tem como propósito entregar uma sequência de ações capaz de conduzir um determinado sistema de um estado inicial até um estado objetivo, respeitando restrições, pré-condições e efeitos impostos a cada ação possível de ser executada naquele domínio. No planejamento clássico determinístico, totalmente observável e atemporal, o problema é modelado como um sistema de transição no qual um planejador busca por um plano válido, isto é, que alcance as metas que minimize os recursos e a duração (GHALLAB; NAU; TRAVERSO, 2004; RUSSELL; NORVIG, 2020). Esse modelo é estruturado em uma representação do domínio de planejamento e uma representação do problema que se deseja resolver, na forma de um plano de ações capaz de conduzir o sistema considerado de uma situação inicial até uma situação objetivo, respeitando restrições, otimizando políticas impostas e minimizando custos.

Para que o sistema de planejamento não seja apenas um gerador de sequências estática, a literatura propõe a transição do conceito de "planejador" para de "Ator Deliberativo". Segundo Ghallab, Nau e Travesso (2016), um ator é uma entidade que não apenas planeja, mas interage com o ambiente e outros agentes de forma dinâmica. Essa estrutura conceitual, apresentada na Figura 4, é importante para entender como a automação estocástica se diferencia da automação rígida tradicional. Na Figura 4 o ator é composto por dois módulos principais: as funções de deliberação ("*Deliberation functions*") e a plataforma de execução ("*Execution platform*"). A plataforma de execução atua como a integração com o mundo físico, transformando comandos ("*Commands*") em atuações ("*Actuations*") e sinais sensoriais ("*Signals*") em percepções ("*Percepts*"). Já no módulo de deliberação, ilustrado na Figura 4b, é onde decorre o ciclo de *Planning* e *Acting*. Nessa arquitetura, o planejamento ("*Planning*") deve definir o que fazer para atingir os objetivos, enquanto a atuação ("*Acting*") foca no como fazer em tempo real, que refina ações abstratas em comandos executáveis, monitorando o ambiente e reagindo

a eventos inesperados. A importância dessa estrutura reside na distinção entre as funções de planejamento e atuação. Enquanto o planejamento lida com a lógica de estados e metas, a atuação é responsável por monitorar a execução e reagir a incertezas do ambiente. A adaptabilidade de "fechar o ciclo" entre o plano abstrato e a realidade física que define o Ator Deliberativo e permite com que o sistema possa reagir a naturezas estocásticas ao invés de seguir uma sequência rígida de comando.

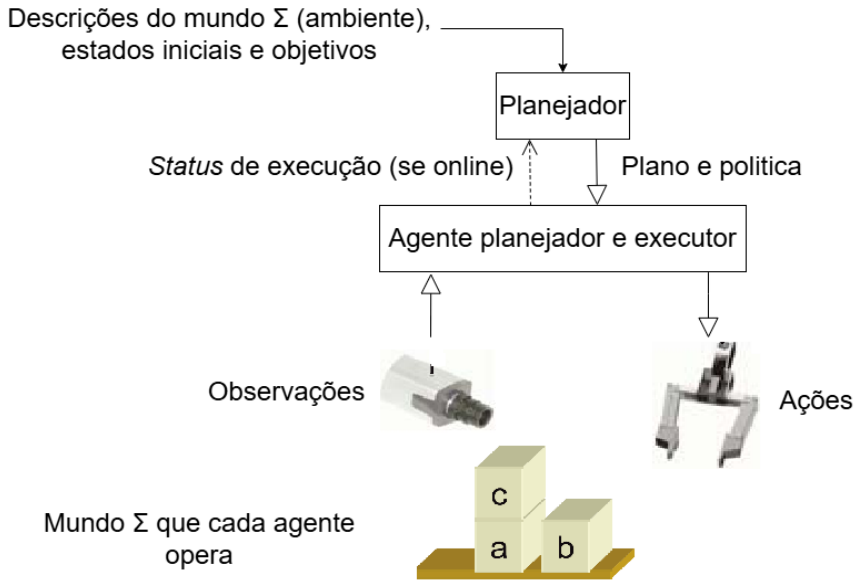
Figura 4 - Visão conceitual de um ator (a); sua restrição ao planejamento e atuação (b)



Fonte: Traduzido de Ghallab, Nau e Travesso (2016)

Conforme representado na Figura 5, a entrada do planejador é composta por: (I) a descrição formal do ambiente Σ; (II) o(s) estado(s) inicial(is) em que Σ pode se encontrar antes da execução de qualquer ação; e (III) o conjunto de metas ou estados desejados. Em cenários de planejamento *online* — nos quais o processo de planejamento ocorre em tempo de execução —, a entrada também incorpora informações de monitoramento, ou *feedback*, referentes ao estado atual da execução do plano ou da política. A saída do planejador pode assumir duas formas: (a) um plano determinístico, representado por uma sequência ordenada de ações; ou (b) uma política, definida como um conjunto de pares estado-ação, onde cada estado está mapeado para, no máximo, uma ação admissível (AU; KUTER; NAU, 2009).

Figura 5 – Modelo prático de AI Planning.



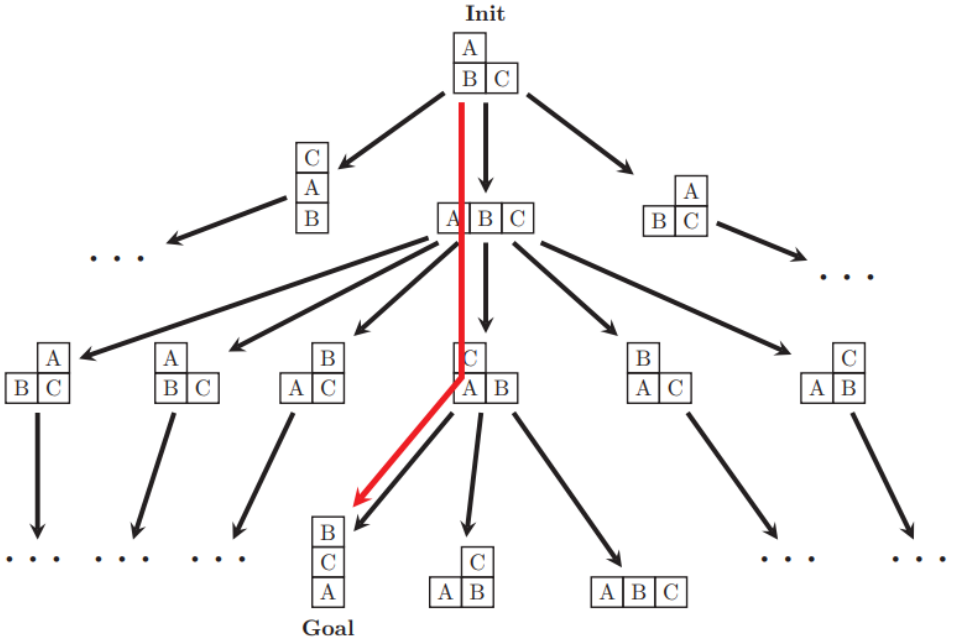
Fonte: Modificado de Au, Kuter e Nau (2009).

O Mundo de Blocos é um exemplo clássico da Inteligência Artificial Simbólica utilizado para comprovar a capacidade de resolução de problemas complexos a partir de um conjunto mínimo de regras e lógicas (NILSSON, 1980; RUSSELL; NORVIG, 2020). Linguagens de programação para planejamento automático dividem o problema do domínio para garantir que o sistema seja genérico, ou seja, o mesmo domínio é capaz de resolver/planejar qualquer problema (plano) descrito dentro do mesmo domínio (McDERMOTT *et al.*, 1998).

Em síntese, o domínio é onde se define como o sistema é caracterizado, indicando os predicados alcançáveis, ou seja, relações e estados que os objetos do domínio podem assumir, bem como as ações possíveis dentro daquele ambiente, indicando as condições necessárias para a execução de cada ação, bem como os efeitos alcançados após a realização daquela ação (GHALLAB; NAU; TRAVERSO, 2004). Independente da escala ou do número de instâncias do problema, essas definições não irão se alterar. Já o problema, é a definição do cenário específico, no qual é definido o estado inicial do sistema e o estado objetivo. Na Figura 6, é possível observar um exemplo de como um plano é traçado de maneira automatizada do Mundo de Blocos, com um mundo definido por predicados lógicos e a descrição de ações, por meio de pré-condições e efeitos. O algoritmo procura dentre as possibilidades, uma maneira a encontrar um caminho para o objetivo final respeitando as leis da física definidas no domínio (GEFFNER; BONET,

2013). Cabe destacar que, independentemente do número de blocos que o sistema possua, e desde que exista uma possibilidade, o algoritmo sempre vai achar uma resolução do problema, respeitando estritamente as leis físicas definidas no domínio.

Figura 6 – Exemplo da geração de um planejamento automático.



Fonte: Geffner, Bonet 2013.

Diferente das abordagens baseadas em Aprendizado de Máquina (*Machine Learning*) que demanda de grandes volumes de dados e sofrem com a falta de interpretabilidade (capacidade de ser interpretado pelo ser humano), o planejamento automático baseado em PDDL é realizado com uma estrutura declarativa onde a causa e o efeito são explicitamente mapeados (GEFFNER; BONET, 2013). A separação entre o modelo do domínio (física) e a instância do problema (estado inicial e meta) permite que o sistema seja flexível e as alterações na configuração do chão de fábrica não exijam um novo processo de 'treinamento', mas apenas a atualização dos predicados lógicos utilizados para descrever o novo estado do ambiente (domínio).

O modelo foi padronizado pela linguagem de programação PDDL, que separa o domínio – possuindo tipos, predicados e ações – do problema – com os objetos, estados iniciais e as metas. Com a evolução desse modelo, foram implementados aspectos temporais e numéricos (FOX; LONG, 2003).

2.3.1. *Planning Domain Definition Language* (PDDL)

PDDL é uma linguagem de programação orientada a objetos que surgiu para padronizar a representação de problemas de planejamento, amplamente utilizado em: competições internacionais, implementações industriais e robóticas. Sua estrutura é composta por dois componentes: o domínio, que descreve os tipos, predicados, as ações, suas precondições e efeitos; e o problema, que especifica os objetos, estados iniciais e as metas (GEREVINI; LONG, 2005).

A representação de conhecimento por meio de um domínio de planejamento requer uma modelagem rigorosa e formal do domínio. A modelagem vai definir quais os objetos relevantes ao ambiente, suas respectivas classes e as relações entre eles. A partir dessa estrutura, o domínio descreve formalmente os estados possíveis desses objetos e transições, permitindo com que o planejador busque por sequências válidas de ações para alcançar objetivos específicos. A herança entre classes, que promove a reutilização e abstração na modelagem, pode ser utilizada para propagar propriedades e comportamentos comuns.

O controle lógico das operações é estruturado por meio de arquivos no formato em PDDL, como o arquivo problema (*problem file*), que define o estado inicial, precondições, efeitos e objetivos, e o arquivo domínio (*domain file*), que especifica tipos de objetos, ações e restrições aplicáveis (GEREVINI; LONG, 2005; YOUNES; LITTMAN, 2004). Com domínio e o problema definidos, utiliza-se um *Solver* (um planejador automático) para tentar resolver o problema de planejamento, ou seja, buscar um plano capaz de atingir o objetivo definido. Entre os diversos planejadores compatíveis com PDDL, como *plan validation system* (VAN), *Metrie Fast-Forward 2.0* e *Expressive Numeric Heuristic Search Planner* (ENHSP), destaca-se o LAMA (*Landmark-based Anytime Multi-heuristic Agent*), que combina a identificação de *landmarks* — estados intermediários obrigatórios para alcançar a meta — com estratégias de busca informada, como o *Weighted A** — que prioriza caminhos mais promissores dando um “peso extra” à heurística —, permitindo geração de planos com maior qualidade em tempo reduzido (RICHTER; WESTPHAL, 2010).

Essa característica o torna particularmente adequado para sistemas que exigem replanejamento rápido, como aplicações industriais com variações frequentes no estado do processo. Em resumo, a configuração *LAMA-first* executa apenas a primeira fase do

algoritmo, priorizando o tempo de resposta e utilizando um peso fixo para balancear custo e velocidade na busca. (HELMERT, 2006).

Além disso, a integração de planejadores automáticos, neste contexto o *LAMA-first*, com sistemas de controle de chão de fábrica pode ser realizada por meio de camadas intermediárias de software que traduzem as ações do plano em comandos para CLPs, utilizando protocolos industriais como Modbus TCP, OPC UA ou Profinet. Essa abordagem viabiliza o uso de técnicas avançadas de IA em ambientes industriais tradicionais, conciliando eficiência de planejamento e confiabilidade operacional.
Portanto, essa abordagem possui potencial de tornar o processo da automação industrial mais dinâmico, baseado em metas, e permitir que planejamento seja alterado diretamente a partir do topo da pirâmide da automação conforme mudanças no ambiente produtivo, demandas do mercado ou restrições técnicas.

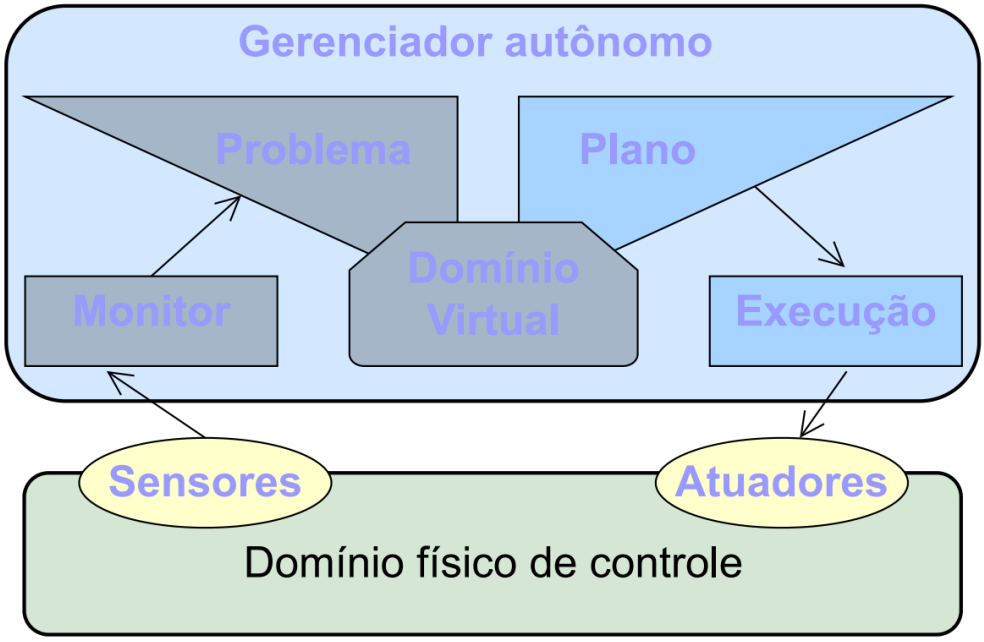
2.3.2. *Plan execution & Monitoring system*

O sistema de execução e verificação de planos (*plan execution & monitoring system*) consiste em um sistema de automação inteligente que recebe planos previamente gerados e os executa em um ambiente físico ou simulado, verificando continuamente se as pré-condições e os efeitos definidos para cada ação estão sendo satisfeitos. O sistema de monitoramento do domínio físico é realizado por meio de sensores e dados de processo, compara-se com o estado esperado e, caso ocorram desvios, o processo pode ser pausado, replanejado ou corrigido sua execução para garantir que o objetivo seja atingido (GHALLAB *et al.*, 2016). O foco do *plan execution & monitoring system* está na garantia de conformidade entre o plano e a realidade operacional, funcionando como um elo direto entre planejamento automático e o controle de campo, diferente de um gêmeo digital, que representa virtualmente um sistema físico de forma sincronizada e contínua (HOEBERT *et al.*, 2023).

A Figura 7 apresenta a arquitetura simplificada de um *plan execution & monitoring system*. Com um domínio virtual previamente definido, o módulo de monitoramento registra (ou “escreve”) formalmente um problema, o qual é utilizado para gerar um plano de execução. Esse plano é aplicado aos atuadores de um domínio físico de controle, que, por sua vez, é monitorado por sensores. As informações coletadas

realimentam o módulo de monitoramento, possibilitando a formulação de novos problemas, planos e execuções de forma contínua.

Figura 7 – Exemplo simplificado do *plan execution* & *monitoring system*.



Fonte: Modificado de De Paola (2014).

2.3.3. Visão sobre planejamento clássico e não clássico

O planejamento clássico é definido como a geração de sequencias de ações em ambientes que satisfazem as premissas restritivas definidas. Segundo Ghallab, Nau e Traverso (2004), esse paradigma assume que o sistema opera em um ambiente determinístico, estático e totalmente observável. Os primeiros sistemas baseados no planejamento clássico, como o sistema de planejamento e linguagem formal STRIPS (*Stanford Research Institute Problem Solver*), são utilizados para descrever domínios que operam com premissas rígidas que simplificam o sistema (FIKES; NILSSON, 1971). Entre essas premissas destaca-se o “Determinismo”, onde ações podem ter apenas um resultado de estado; a “Observabilidade Total”, onde o agente sabe exatamente em que estado o mundo está a qualquer tempo e; o “Ambiente Estático” que não muda a não ser que o agente execute a ação, mas que são limitados para lidar com incertezas do mundo real (GHALLAB; NAU; TRAVERSO, 2004).

O planejamento clássico trabalha com estados como conjuntos, assim um estado é apenas uma lista de fatos verdadeiros e tudo que não faz parte da lista é considerado

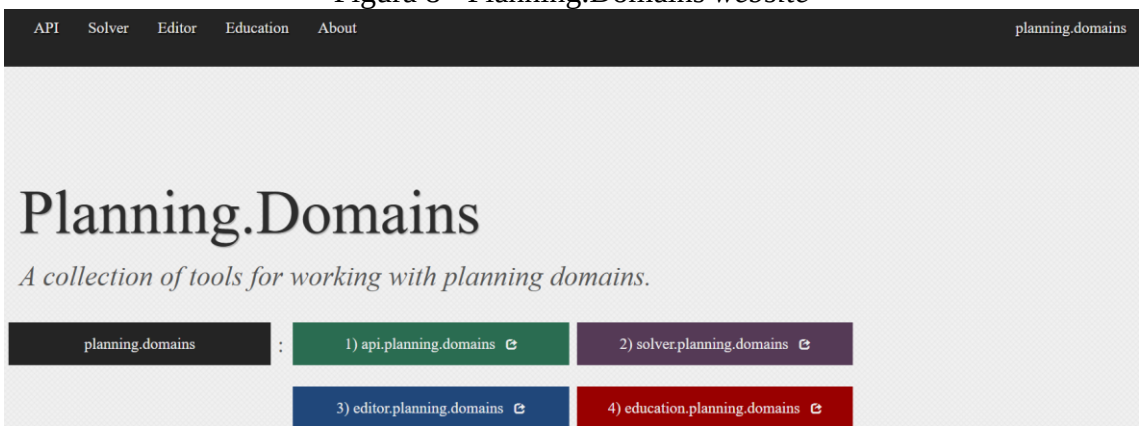
falso (suposição de mundos fechados) (REITER, 1978). Além disso, o planejamento clássico trabalha com ações determinísticas, ou seja, são ações definidas por pré-condições (o que precisa para poder agir), lista de adição (o que passa a ser verdade) e lista de exclusão (o que deixa de ser verdade), modelo que se tornou base para a evolução das linguagens de planejamento modernas (GHALLAB; NAU; TRAVERSO, 2004).

O planejamento moderno, também conhecido como estocástico, é marcado pela quebra das "amarras" do STRIPS focando na robustez e flexibilidade de ambientes dinâmicos que trabalha suavizando as restrições do planejamento clássico (GHALLAB; NAU; TRAVERSO, 2016). No planejamento moderno admite-se: Incertezas nos Efeitos, existe uma probabilidade da ação no estado 1 levar ao estado 2; Objetivos, ao invés de focar alcançar uma meta enrijecida de estado alvo; e busca-se maximizar a probabilidade de sucesso (política de decisão ótima) (PUTERMAN, 1994; BELLMAN, 1957).

2.3.4. *Planning.Domains*

O *Planning.Domains* é uma plataforma que junta ferramentas para trabalho em *planning domains*, com seu *website* ilustrado na Figura 8. Essa é uma coleção de domínios e problemas de *benchmark* descritos em PDDL que se consolida como infraestrutura central para pesquisa em planejamento automático. Essa infraestrutura é dividida em 4 ferramentas: *api.planning.domains*, *solver.planning.domains*, *editor.planning.domains* e *education.planning.domains*.

Figura 8 - Planning.Domains *website*

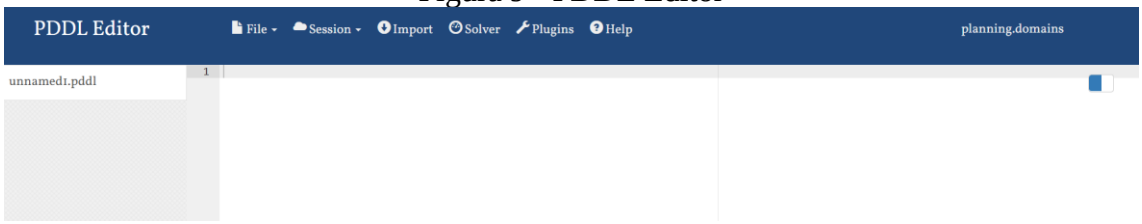


Fonte: Muise (2016).

A API do *planning.domains* consite em um sistema unificado de acesso a repositórios de problemas e domínios em PDDL, permitindo que pesquisadores consultem coleções de todas edições da *International Planning Competition* (IPC). Nesta interface, é possível obter estatísticas detalhadas sobre limite de custo e requisito de modelagem, além de permitir a integração por bibliotecas Python e Javascript.

O Solver do *planning.domains* é um *software open source* voltado para aplicações educacionais para solucionar problemas de planejamento por meio de um serviço web. Já o *Editor* do *planning.domains*, apresentado na Figura 9, é uma plataforma web que permite escrever domínios e problemas para solicitar um plano por diversos tipos de *Solvers*. Por fim, com objetivo de centralizar o recurso pedagógico voltado para o ensino de técnicas de planejamento automatizado e modelagem, o *Planning.Domains* possui o repositório *Education* que reúne materiais didáticos de diversas instituições internacionais, incluindo slides de aulas, grupos de pesquisas renomados, vídeos tutoriais e tarefas práticas.

Figura 9 - PDDL Editor



Fonte: Fonte: Muise (2016).

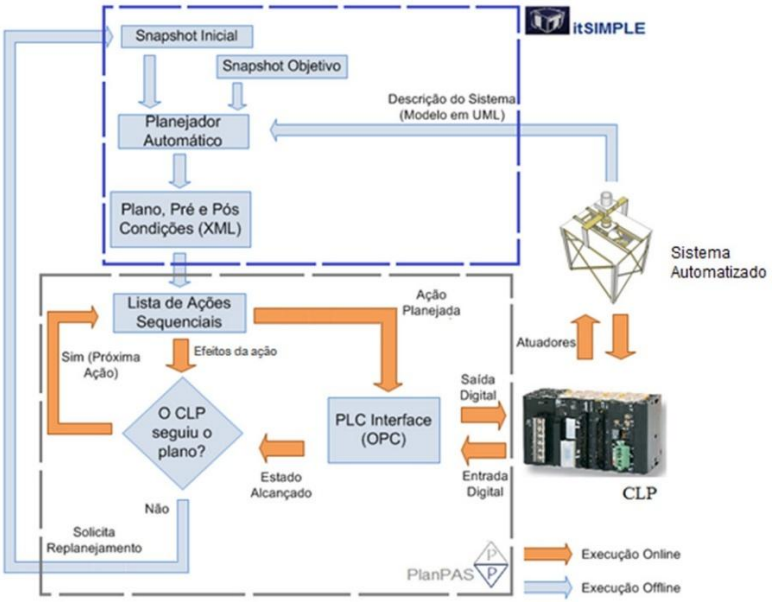
2.4. Trabalho anterior

O *Plans Parser for Automated Systems* (PlanPAS) é um software desenvolvido na Universidade Federal de Uberlândia que atua como ponte entre o nível tático (planejamento automático/ inteligência artificial) e o nível operacional (chão de fábrica controlado por CLPs). O PlanPAS atua lendo um plano-solução de uma sequência gerada por IA e executa essas ações diretamente no CLP sem a necessidade de um programador escrever a lógica (FONSECA, 2013).

O PlanPAS, conforme apresentado na Figura 10, opera com dois módulos principais que se comunica via TCP/IP, o *Parser* (comparador), responsável pela interpretação do arquivo plano e por gerenciar as ações, e o CLP Interface, responsável

pela comunicação com o CLP via tecnologia *Open Platform Communication* (OPC) (OPC Foundation, 2025). Nessa abordagem, o CLP opera como um driver do ambiente e o *Parser* como o cerne da lógica integrado ao plano-solução. O ambiente físico em estudo tem uma modelagem do domínio implementada no *software* itSIMPLE (VAQUERO, 2013), por meio de diagramas UML que modelam o domínio de planejamento e posteriormente traduz para PDDL, tornando o problema aptos para análise de planejadores automáticos e, caso a solução seja possível, retorna o arquivo XML com a lista de ações necessárias para atingir o objetivo considerando as restrições de domínio.

Figura 10 - Representação esquemática do projeto PlanPAS



Fonte: Fonseca (2013).

3. MÉTODO

O projeto foi desenvolvido em quatro etapas. Inicialmente modelou-se o ambiente de estudo como um domínio de planejamento utilizando a linguagem PDDL e, para validar a aplicação e testar a escalabilidade das soluções geradas pelo planejador automático, modelou-se, três problemas ampliando o número de objetos do ambiente e variação do cenário objetivo. Em seguida, desenvolveu-se uma API em Python, responsável pela integração entre o ambiente deliberativo de alto nível, aonde estão hospedados o planejador automático e o controlador do sistema físico, que no caso foi testado e validado tanto em um CLP Schneider M221, como em um CLP Siemens SIMATIC S7-1200. Posteriormente, configurou-se o mapa de endereços e a rede de comunicação do CLP, em consonância com as informações aplicadas na API. Por fim, uma interface gráfica do sistema físico foi implementada utilizando uma HMI industrial, modelo HMIFXU5512x, por meio da qual o usuário, além de conseguir monitorar graficamente o ambiente físico, pode interagir e controlar o sistema.

Esse sistema foi baseado no PlanPAS desenvolvido por Fonseca (2013), a geração dos planos era realizada por meio de planejadores embarcados no software de Modelagem itSIMPLE. A integração do ambiente de planejamento automático com o mundo físico se dava de forma *offline*, ou seja, o plano-solução obtido pelo software itSIMPLE era processado *offline* e implementado via servidor integrado ao CLP via comunicação OPC. Após a geração do plano, um módulo Parser era responsável por realizar a leitura das entradas do CLP e garantir que as pre-condições fossem satisfeitas, de modo a permitir a escrita das ações correspondentes nas saídas elétricas do controlador. A metodologia desse trabalho diferencia-se em função da aplicação *online* do serviço *planning.domains*, possibilitando a configuração e a reconfiguração do sistema no chão de fábrica de maneira automática, por meio de uma HMI comumente utilizada em ambientes industriais. Adicionalmente, foi incorporado um plano de contingência com a finalidade de manter o processo operando minimamente caso a API fique *offline*, indisponibilizando o planejamento automático. Por fim, enquanto o projeto do PlanPAS utilizava a comunicação OPC, a abordagem atual adota o protocolo Modbus, para comunicação com o CLP, e HTTP para comunicação com serviço *planning.domains*. Ambos são padrões de protocolos de comunicação industrial são abertos, contudo o Modbus é mais amplamente difundido em nível de máquina devido ao baixo custo e simplicidade, ao

passo que o OPC é voltado à interoperabilidade entre sistemas e à integração com camadas superiores. Essa função de mediação entre a máquina e o ambiente digital é desempenhada pela API desenvolvida.

3.1 Materiais

O sistema físico utilizado foi a bancada didática de separação Exsto XC243, ilustrada na Figura 11, composta por uma correia transportadora, caixas de separação de estoque, detectores de proximidade dos tipos indutivo, capacitivo, magnético, e fotoelétrico, detectores eletromecânicos do tipo chave fim de curso, atuadores pneumáticos, válvulas eletropneumáticas, filtro de ar comprimido com regulador de pressão, tubulações pneumáticas, cabos elétricos, cabos de rede e comunicação, conectores elétricos e de comunicação, um switch, o CLP, a HMI e um computador pessoal.

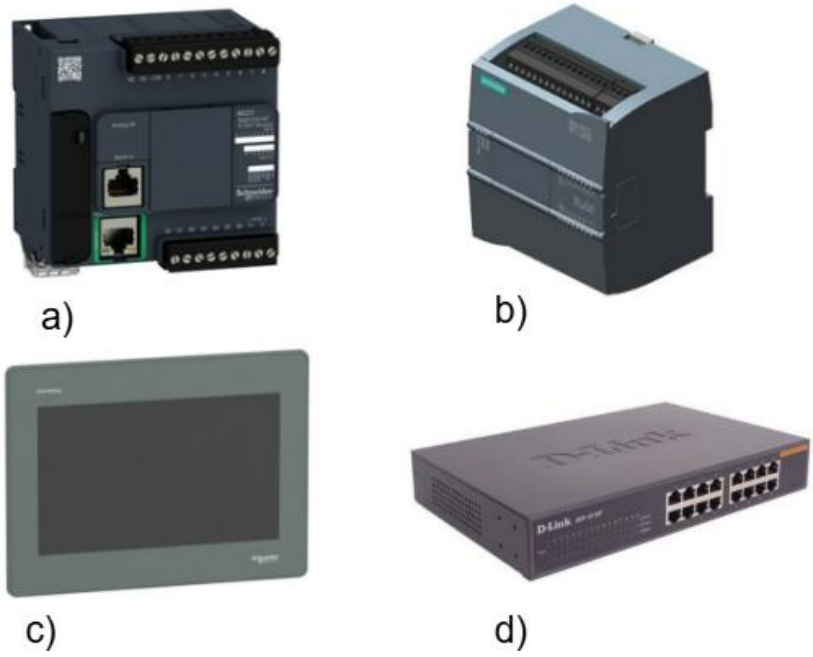
Figura 11 – Bancada didática de separação Exsto XC243

Fonte: Exsto Tecnologia (s.d.).

A API foi testada em duas configurações de campo distintas. Na primeira configuração de campo, utilizou-se o dispositivo CLP da Schneider Electric, pertencente à família Modicon M221, modelo TM221CE16T apresentado na Figura 12 a), com suporte nativo do protocolo Modbus TCP/IP, porta serial e porta USB de programação.

Na segunda configuração de campo, utilizou-se o dispositivo CLP da Siemens, pertencente à família SIMATIC S7-1200, modelo CPU 1212C DC/DC/Relay referência 6ES7 212-1HE31-0XB0 apresentado na Figura 12 b), com suporte a protocolo PROFINET, S7 Communication, TCP/IP, UDP, HTTP / Web Server e Modbus TCP. Para facilitar a integração com os operadores do ramo industrial, utilizou-se também à uma HMI de 10,1 polegadas, fabricada também pela Schneider Electric, da família Magelis GXU / Harmony Easy GXU, modelo HMIFXU5512x apresentado na Figura 12 c), possuindo conexão via porta serial, USB e Ethernet. Por fim, toda comunicação foi intermediada pelo switch D-Link DES-1016D ilustrado na Figura 12 d).

Figura 12 – a) TM221CE16T, b) 6ES7 212-1HE31-0XB0, c) HMIFXU5512x e d) DES-1016D



Fonte: a) Schneider Electric (2026) b) Siemens (2026) c) Schneider Electric (2026) d) D-Link (2002)

A estação de desenvolvimento do trabalho consiste em um computador pessoal com capacidade para executar *scripts* em Python na versão 3.12.4, acesso a navegador para o acesso/uso do *editor.planning.domains*, suporte ao *software* da Schneider Electric *EcoStruxure Machine Expert – Basic* e *Vijeo Designer Basic 1.2.1* e ao *software* da Siemens *Totally Integrated Automation Portal* (TIA Portal) versão 16, com conexão cabeada à Ethernet e USB.

Toda essa configuração possibilita a simulação de um ambiente industrial de classificação e distribuição de peças com diferentes características solicitadas por um ou mais clientes. Neste caso, as caixas de separação de estoque são consideradas como docas da linha de distribuição dos produtos de acordo com a demanda dos clientes. A interconexão entre sensores, atuadores, CLP, HMI, switch e computador pessoal permite simular um ambiente de automação industrial totalmente robusta e integrada.

3.2 Modelagem do Problema em PDDL

Para a modelagem do domínio, utilizou-se a ferramenta *editor.planning.domains*, que permite a modelagem de domínios, problemas e ações, com suporte a geração de planos via *solver* baseado em IA. A linguagem é similar a programação orientada a objetos. A estrutura do domínio inclui a definição de tipos, predicados e ações, que representam os elementos e comportamentos do ambiente físico.

3.2.1.1 Domínio

O primeiro passo para iniciar a programação do código do domínio é definir os requisitos necessários. Nesse contexto, foi utilizado o STRIPS (*Stanford Research Institute Problem Solver*), que estrutura ações com pré-condições e efeitos, e o *typing*, que permite os objetos em classes (tipos), podendo existir categorias da então classe objeto, além disso, os objetos em PDDL passam a possuir hierarquia e semântica (significado dentro de um contexto).

Em seguida, definem-se os tipos de classes de objetos, que herdam as características de classes gerais como “objeto” e “localizacao”. No Quadro 1, estão representadas dentro dos*: types* (tipos) onde são definidas as classes “esteira”, “tamanho”, “atuador” e “peça”, que herdam características da classe geral “objeto”, bem como as classes “inicio” e “estoque”, que herdam características da classe geral “localizacao”.

Quadro 1– Definição do domínio da bancada (Objetos e Classes).

```
(define (domain bancada_ufg)
(:requirements :strips :typing)
   (:types
      esteira tamanho peca atuador - objeto
      inicio estoque - localizacao
   )
```

Fonte: Próprio autor.

Posteriormente, definem-se os predicados (*":predicates"*) das classes, ou seja, a formalização da representação dos objetos e de seus estados, que abrangem classes específicas e gerais do domínio da linha de separação automatizada. Os predicados, apresentados no Quadro 2, definem nesta aplicação o estoque de destino da peça, *status* de recebimento, status de funcionamento de esteiras e atuadores, qual posição das peças, e suas características, como o tamanho e material base (metálico ou não metálico).

Quadro 2 – Definição do domínio da bancada (Predicados).

```
(:predicates
        (link ?a - atuador ?estoque - estoque) ;;Informar para qual estoque o atuador
                                                direciona a peça

        (recebimento ?a - atuador) ;;Informa que o item foi recebido no estoque

        (on ?esteira - esteira) ;;Informa que a esteira está ligada

        (on ?a - atuador) ;;Informa que o atuador está acionado

        (lock_esteira ?esteira - esteira) ;;Mantém a esteira ligada durante o processo

        (at ?p - peca ?l - localizacao) ;;Descreve a posição

        (tam ?p - peca ?t - tamanho) ;;Descreve o tamanho da peça
)
```

Fonte: Próprio autor.

Por fim, pode-se definir as ações que os objetos de cada classe são capazes de realizar baseado nos “:parameters” (parâmetros) e “:precondition” (precondições) que aquela ação necessita, e os “:effect” (efeitos) que aquela ação provocará. O Quadro 3 define-se as ações da esteira e atuadores, que envolvem a interação entre objetos do domínio exigindo parâmetros como posição inicial, status da esteira, peça, tamanho, atuador e estoque de destino. As precondições incluem compatibilidade de tamanho, caminho disponível, status da esteira e status do atuador. O efeito é o movimento da peça para o estoque, garantindo que a esteira continue ativa até a retração do atuador. O retorno envolve apenas o status da esteira e do atuador para permitir sua liberação.

Quadro 3 – Definição do domínio da bancada (Ações).

```
;;Ação para ligar esteira
(:action liga_esteira
        :parameters (?esteira - esteira)
        :precondition (and (not (on ?esteira)))
        :effect (and (on ?esteira)))

;;Ação para desligar esteira
(:action desliga_esteira
        :parameters (?esteira - esteira)
        :precondition (and (on ?esteira)(not (lock_esteira ?esteira)))
        :effect (and (not (on ?esteira))))

;;Ação para solicitar peça e atuar o atuador que leva a peça para o estoque desejado
(:action solicita_peca_e_indica_atuador
        :parameters(?i - inicio ?esteira - esteira ?p - peca ?t - tamanho ?a - atuador
                        ?estoque - estoque)
        :precondition (and (tam ?p ?t)(link ?a ?estoque)(on ?esteira)(not (on ?a))
                            (not (lock_esteira ?esteira))(at ?p ?i))
        :effect (and (not (at ?p ?i))(at ?p ?estoque)(on ?a)
                    (recebimento ?a)(lock_esteira ?esteira)))

;;Ação para retornar atuador após a entrega da peça
```

```
(:action retorno_do_atuador
        :parameters(?esteira - esteira ?a - atuador)
        :precondition (and (recebimento ?a)(on ?a)(lock_esteira ?esteira))
        :effect (and (not (on ?a))(not (recebimento ?a))(not (lock_esteira ?esteira))))
)
```

Fonte: Próprio autor.

3.2.1.2 Problema

A modelagem do problema é como se definem os objetos, condições iniciais e cenário objetivo. Para a bancada de separação em estudo, são definidos os objetos ("*:objects*"), contendo todos os atuadores, todos os recipientes de estoque utilizados, o objeto esteira transportadora, todos os itens para separação – cada qual com o seu respectivo atributo de tamanho e posição admissível. Nas condições iniciais ("*:init*"), as características das peças, as posições no estado inicial, estado da esteira e associação de atuadores com posições de estoque são definidas. Por fim, nos objetivos ("*:goal*"), tratado neste estudo como a entrega de diferentes tipos de peças nas posições de estoque, de acordo com o solicitado pelo operador, são definidas as posições de interesse das peças e as condições finais de estados dos atuadores e esteira. O Quadro 4 mostra um código exemplo do arquivo problema em PDDL.

Quadro 4 – Exemplo de definição de um problema aplicado ao ambiente de estudo.

```
(define (problem bancada_ufg)
(:domain bancada_ufg)
        (:objects
                atuador1 atuador2 atuador3 - atuador
                estoq1 estoq2 estoq3 - estoque
                esteira - esteira
                pequena media grande - tamanho
                inicio - inicio
                peca1 peca2 peca3 peca4 – peca
        )
        (:init
                (tam peca1 pequena)
                (at peca1 inicio)
                (not (on esteira))
                (link atuador1 estoq1)
```

```
        ...)
    (:goal (and
        (at peca1 estoq1)
        ...
        (not (on esteira))
        (not (on atuador1))
        (not (on atuador2))
        (not (on atuador3)))))
```

Fonte: Próprio autor.

Nesta aplicação, a definição do problema, em suma, representa o estado atual do sistema, ou seja, contém informações que podem ser extraídas pelos sensores de monitoramento e pelas inserções de dados de um operado/cliente/demanda que dita o cenário objetivo. Com a união do arquivo domínio, que define “as regras do jogo” e o arquivo problema, que define o estado atual do sistema e quais são as “metas”, o planejador automático consegue, se existir, criar um plano que respeite as restrições do domínio para atingir o objetivo.

3.2.1.3 Plano

Dentre os planejadores disponíveis no *planning.domains*, adotou-se o *Solver* LAMA-first, que foi projetado para buscar soluções com maior velocidade sem levar em consideração custos e refinamentos. A Figura 13 mostra o exemplo de um documento problema que é gerado pela API a fim de se obter um plano solução. Uma vez definido o domínio para essa aplicação, é necessário somente definir a quantidade de peças nos objetos e informar quais as condições iniciais, como tipo das peças inseridas, a posição inicial e qual a meta (*goal*).

Figura 13 – Exemplo de arquivo problema gerado.

```
(define (problem bancada_ufg)
(:domain bancada_ufg)

    (:objects
        ;Definição dos objetos e classes
        atuador1 atuador2 atuador3 - atuador
        estoq1 estoq2 estoq3 - estoque
        esteira - esteira
        pequena media grande - tamanho
        inicio - inicio
        peca1 peca2 - peca
    )

    (:init
        ;;Informa o tipo de peças que serão inseridas na es
        (tam peca1 pequena)
        (tam peca2 media)

        ;;Infomra a posição inicial da peça
        (at peca1 inicio)
        (at peca2 inicio)


        ;;Informa que a esteira está desligada no início d
        (not (on esteira))

        ;;Informa quais atuadores levam para qual estoque
        (link atuador1 estoq1)
        (link atuador2 estoq2)
        (link atuador3 estoq3)
    )

    (:goal (and
        ;;Indica onde as peças estão sendo solicitadas
        (at peca1 estoq1)
        (at peca2 estoq3)

        ;;Indica a posição final de todos atuadores
        (not (on esteira))
        (not (on atuador1))
        (not (on atuador2))
        (not (on atuador3))
        )
    )
)
```

Fonte: Próprio autor.

Para o exemplo, é gerado um plano de acordo com a figura 14, detalhando a sequência de ações que devem ocorrer a fim de atingir o cenário objetivo. Essa solução obteve um tempo de processamento igual a 0,38 segundos para que o planejador retornasse um plano solução possível.

Figura 14 – Exemplo de plano solução gerado.

Found Plan (output)

(liga_esteira esteira)
(solicita_peca_e_indica_atuador inicio esteira peca1 pequena atuador1 estoq1)
(retorno_do_atuador esteira atuador1)
(solicita_peca_e_indica_atuador inicio esteira peca2 media atuador3 estoq3)
(retorno_do_atuador esteira atuador3)
(desliga_esteira esteira)

Fonte: Próprio autor.

Cada linha do plano corresponde a uma ação descrita no domínio, as quais são utilizadas para controlar o CLP. Nas Figuras 15, 16 e 17 ilustram-se as ações (liga_esteira esteira), (solicita_peca_e_indica_atuador início esteira peça 1 pequena atuador1 estoq1) e (retorno_do_atuador esteira atuador1). É possível notar que cada ação do plano possui

precondições e efeitos a serem respeitadas, ou seja, características do ambiente. As precondições serão os sinais que a API deve escutar do CLP durante o processo e, quando positivo, executar os efeitos daquela ação por meio da escrita de dados no CLP.

Figura 15 – Exemplo ação (liga_esteira esteira).

```
(:action liga_esteira
  :parameters (esteira)
  :precondition
    (and
      (not
        (on esteira)
      )
    )
  :effect
    (and
      (on esteira)
    )
)
```

Fonte: Próprio autor.

Figura 16 – Exemplo ação (solicita_peca_e_indica_atuador inicio esteira peca 1 pequena atuador1 estoq1).

```
(:action solicita_peca_e_indica_atuador
  :parameters (inicio esteira peca1 pequena atuador1 estoq1)
  :precondition
    (and
      (tam peca1 pequena)
      (link atuador1 estoq1)
      (on esteira)
      (not
        (on atuador1)
      )
      (not
        (lock_esteira esteira)
      )
      (at peca1 inicio)
    )
  :effect
    (and
      (not
        (at peca1 inicio)
      )
      (at peca1 estoq1)
      (on atuador1)
      (recebimento atuador1)
      (lock_esteira esteira)
    )
)
```

Fonte: Próprio autor.

Figura 17 – Exemplo ação (retorno_do_atuador esteira atuador1).

```
(:action retorno_do_atuador
  :parameters (esteira atuador1)
  :precondition
    (and
      (recebimento atuador1)
      (on atuador1)
      (lock_esteira esteira)
    )
  :effect
    (and
      (not
        (on atuador1)
      )
      (not
        (recebimento atuador1)
      )
      (not
        (lock_esteira esteira)
      )
    )
)
```

Fonte: Próprio autor.

De maneira geral, os planos gerados possuem baixo tempo de processamento com alto potencial de escalabilidade. Posteriormente, é possível converter esse plano para ler e escrever em entradas e saídas de um controlador da planta física, possibilitando a programação e reprogramação automática através das solicitações do cliente. Contudo, ainda é necessário quebrar a barreira da comunicação entre o planejador, cliente e CLP.

3.3 Desenvolvimento da API

Para amenizar as limitações da pirâmide da automação tradicional, a API desenvolvida, utilizando linguagens de alto nível, como a em PDDL, permite a automação orientada por metas inseridas diretamente pelo cliente. Essa API introduz uma camada de raciocínio automático sobre a estrutura tradicional, permitindo que decisões sejam tomadas nos níveis superiores e repassadas diretamente para os dispositivos de campo. O usuário pode definir metas e o planejador automático gera sequências de ações que satisfazem essas metas, com base no estado atual da planta (GHALLAB, 2016).

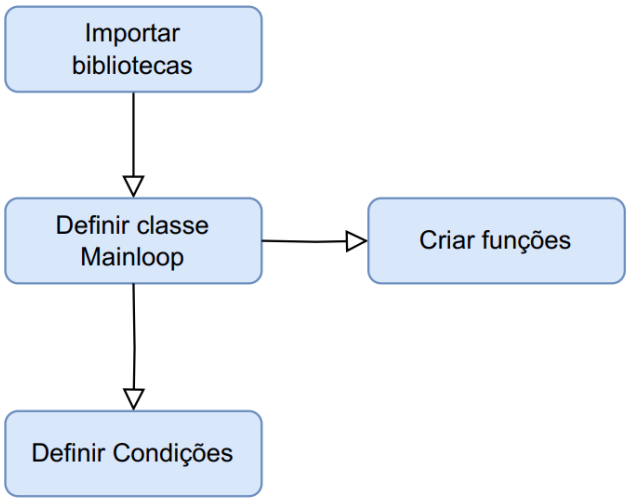
Para utilizar planejadores hospedados remotamente, a API deve enviar os arquivos em PDDL do domínio e do problema a ser solucionado para serviços tipo *Planning as a Servic*e (PaaS), que recebem os arquivos via requisições HTTP e retornam

um plano de execução. O plano geralmente conterá uma sequência ordenada de ações, o custo estimado e o *status* de cada etapa.

A execução do plano envolve traduzir ações simbólicas em comandos concretos enviados aos dispositivos industriais via Modbus/TCP, acionando atuadores e monitorando sensores. Essa integração assegura que decisões lógicas se convertam em ações físicas, que respeitem as condições iniciais e se adaptem às mudanças de ambiente.

A API foi desenvolvida em Python através de 2 arquivos principais. O "Main_program", ilustrado de forma simplificada na Figura 18, responsável pela parte de conexão com o *Solver* de problemas em PDDL e envio dos arquivos domínio e problema para obter um arquivo plano. Uma vez que o plano foi gerado, atualiza as entradas e saídas do CLP e aguarda até que as precondições das ações sejam obtidas para prosseguir até o final do plano.

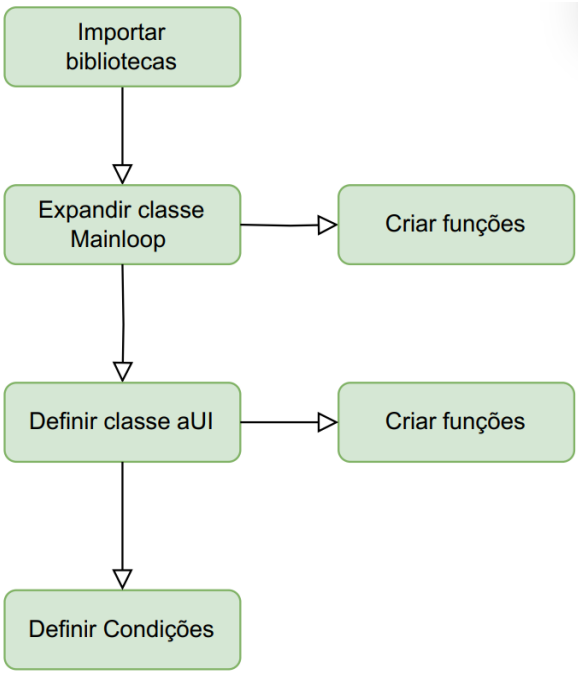
Figura 18 – Criação do Main_program.



Fonte: Próprio autor.

Já "Program_GUI", evidenciado de maneira também simplificada na Figura 19, é responsável pelos comandos do programa – monitorando as memórias da HMI, atualizações da interface e gera o arquivo problema em PDDL por meio de qualquer solicitação do cliente. Ambos os códigos rodam simultaneamente tornando possível parar a execução de um plano solução em caso de emergência ou nova solicitação do cliente.

Figura 19 – Criação do Program_GUI.



Fonte: Próprio autor.

3.4 Dispositivos e Arquitetura

O projeto utiliza um CLP industrial da Schneider Electric, modelo TM221CE16T, e um CLP industrial Siemens, modelo 6ES7 212-1HE31-0XB0, que, por meio do protocolo Modbus TCP, permite a comunicação direta com a API desenvolvida em Python. Essa API foi projetada para operar como intermediária entre um planejador simbólico de alto nível e os dispositivos físicos da planta industrial, monitorando e atualizando os registradores do CLP e, posteriormente, realizando ações. A API foi projetada para operar em qualquer computador que possua o Python 3.12.4 conectado a um *switch* com acesso aos dispositivos de campo e a acesso à Internet.

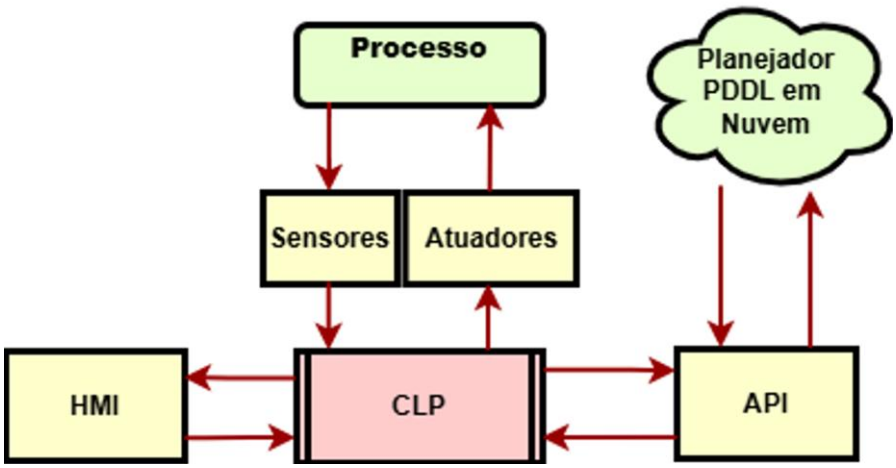
A API faz uso de um conjunto robusto de bibliotecas Python, sendo as principais: *pymodbus* – responsável pela codificação e decodificação dos dados Modbus, permitindo a leitura e escrita em registradores Modbus de forma estruturada com suporte a *payloads* binários (BinaryPayloadBuilder, BinaryPayloadDecoder) e controle de endianness (Endian) – responsável pela ordem de leitura dos blocos de *bytes*; *requests*, utilizada para envio de requisições HTTP entre módulos ou servidores externos, o que inclui comunicação com a plataforma *planning.domains* hospedada remotamente; e *tkinter*, aplicada para a criação da interface gráfica do sistema, permitindo o monitoramento e operação visual do processo industrial. Além disso, módulos como *Mainloop* e modbus

organizam a lógica principal do sistema, a abstração da comunicação com o CLP e as funções auxiliares.

Essas bibliotecas trabalham em um ambiente *multithread*, com suporte a filas de execução (*queue*) e sincronização, permitindo que a API monitore continuamente os estados dos sensores e atuadores conectados ao CLP, atualize a descrição do estado do sistema (Problema) em linguagem PDDL e solicite soluções a um planejador automático (como o utilizado LAMA-first). O sistema é capaz de processar o problema de planejamento a partir de entradas, responder com um plano de ação e convertê-lo automaticamente em comandos Modbus, garantindo a execução adaptável às tarefas industriais.

A Figura 20 demonstra a arquitetura do sistema, que centraliza os dados operacionais no CLP, mas transfere a lógica de decisão para a API, permitindo um fluxo bidirecional contínuo entre os níveis de campo, controle, supervisão e planejamento da pirâmide de automação. Nessa arquitetura, o CLP fica responsável pela leitura direta dos dispositivos de campo como, entradas do operador/cliente na tela de configuração da HMI, leitura de sensores e comando de atuadores. Essa abordagem possibilita a integração das linguagens de programação de baixo nível, como aquelas empregadas em CLPs, com linguagens formais de alto nível, como em PDDL, promovendo um sistema industrial autônomo, adaptável e alinhado com os princípios da Indústria 4.0 e, emergente, 5.0.

Figura 20 – Esquema simplificado ilustrando a arquitetura de comunicação de dados entre os diversos atores do sistema ciberfísico implementado.



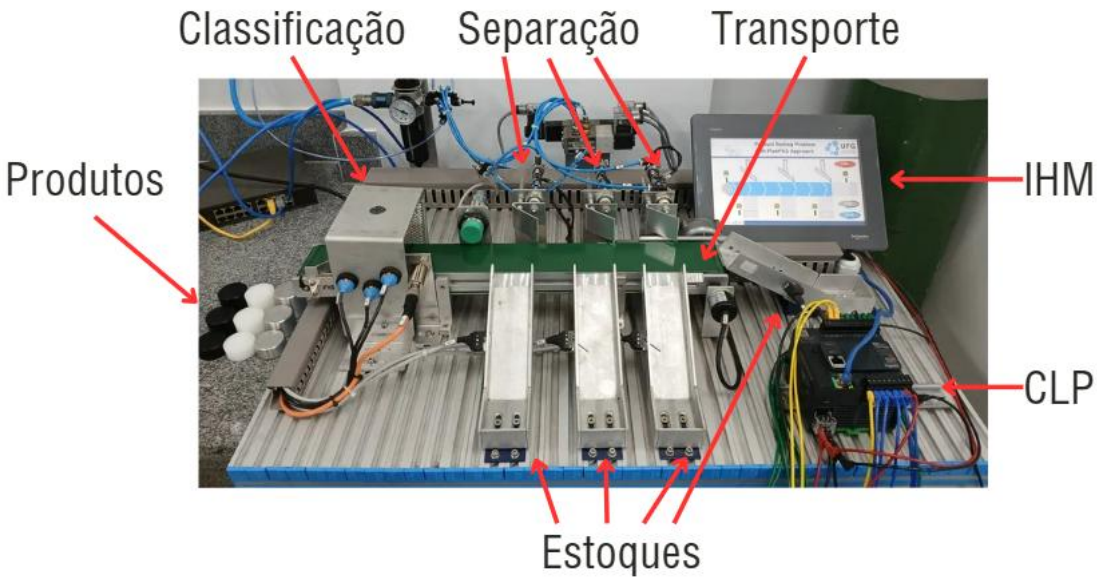
Fonte: Próprio autor.

3.5 Planta Didática de Separação

A Figura 21 apresenta a bancada didática de separação de peças da Universidade Federal de Goiás, que foi utilizada como referência para a modelagem do domínio em PDDL e validação do sistema. O sistema é controlado por um CLP Schneider Electric, da família Modicon M221, modelo TM221CE16T. A estrutura da bancada pode ser dividida em quatro setores principais: (I) classificação da peça, dada por sensores posicionados na entrada da esteira; (II) transporte da peça, executado pela esteira; (III) separação da peça, realizado por atuadores pneumáticos; e (IV) armazenamento, por compartimentos de estoque.

Esses setores são representados formalmente na modelagem em PDDL, por meio dos arquivos de domínio e problema. O setor de classificação foi modelado no arquivo de problema, por corresponder às condições iniciais do sistema; já os setores de transporte e separação são descritos no domínio, uma vez que são ações com pré-condições e efeitos que compõem o processo de execução do plano. Por fim, o setor de armazenamento é representado como o efeito final das ações de separação, caracterizando o objetivo do sistema automático.

Figura 21 – Bancada didática Exsto XC243 referência para modelagem.

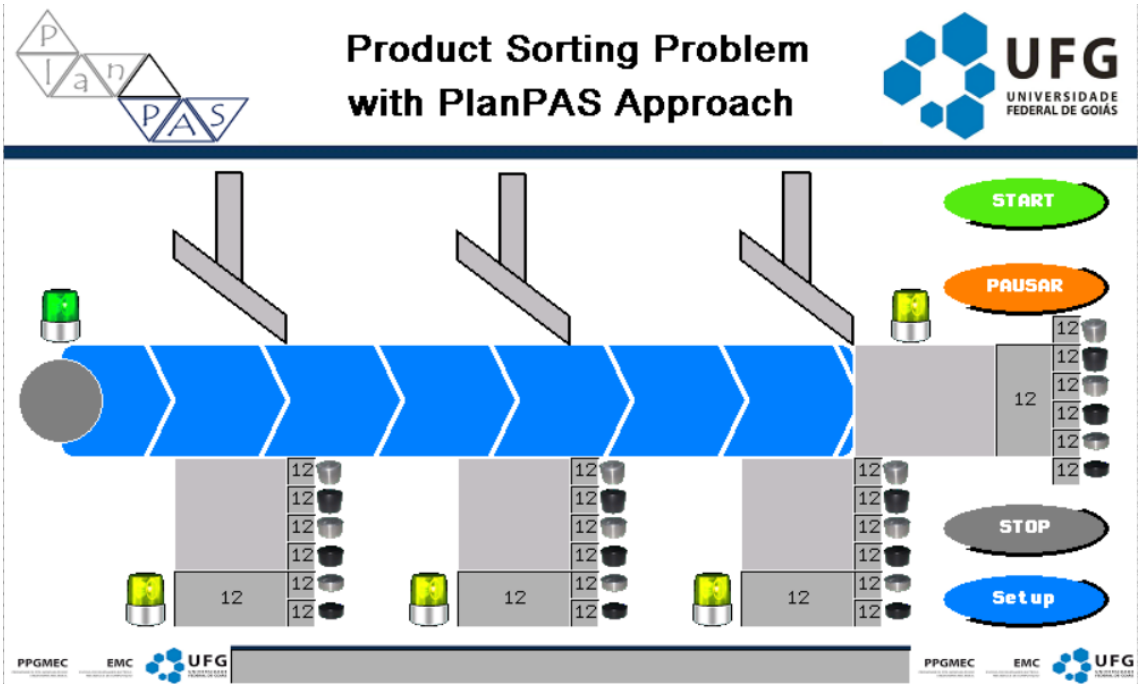


Fonte: Próprio autor.

3.6 Desenvolvimento da HMI

A interface homem máquina foi desenvolvida no *software Vijeo Designer Basic 1.2.1*, da fabricante Schneider Electric, que é destinado ao desenvolvimento de HMIs comerciais de aplicação industrial. A interface disponibiliza ao cliente/operador a visualização do processo *online*, além de uma tela dedicada à configuração e reconfiguração das solicitações que serão processadas, assim que for iniciado o processo. A Figura 22 ilustra a representação gráfica da bancada de separação, com botões de comando disponíveis para o operador. O botão "Start" inicia o processo de separação, o botão "Pausar" permite parar a execução do plano e todos os atuadores momentaneamente, o botão "Stop" para todo o processo de separação mantendo a integridade física da aplicação em casos de emergência e botão "Setup" acessa a tela de configuração da solicitação do cliente.

Figura 22 – Interface industrial: Página principal.

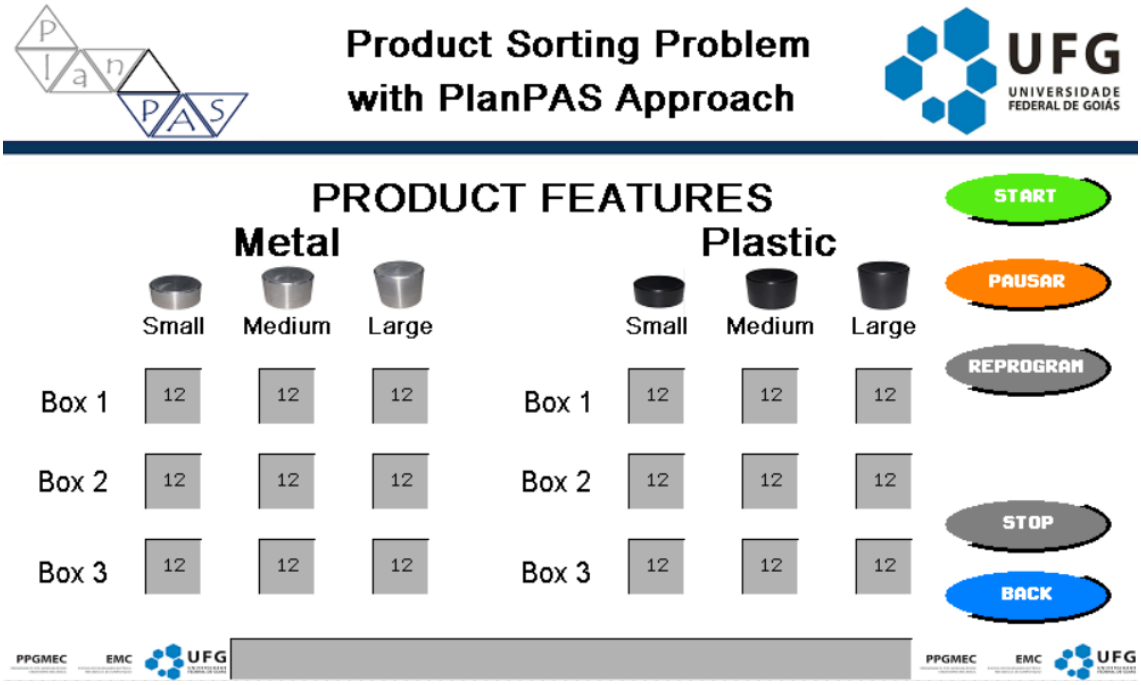


Fonte: Próprio autor.

Na Figura 23 é apresentada a tela de configuração, na qual o cliente pode inserir suas solicitações, nesse caso descrever as quantidades e características dos produtos desejados nos devidos destinos. Nesta tela estão habilitados os botões "Start", "Pausar",

"Stop", "Reprogram", que permite parar a execução e personalizar novamente as requisições, e "Back", que retorna à tela ilustrada na Figura 22.

Figura 23 – Interface industrial: Página de setup do problema.



Fonte: Próprio autor.

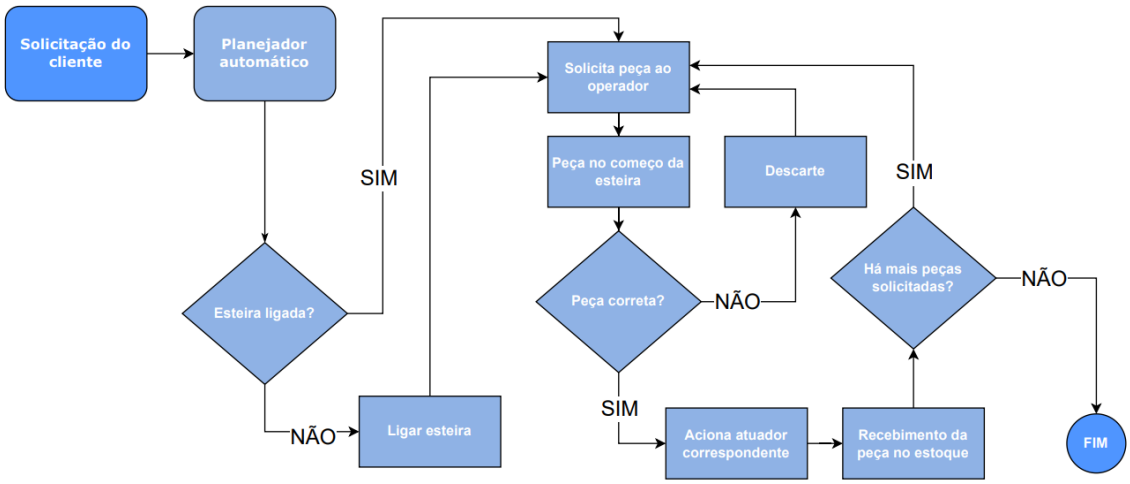
Vale destacar que a HMI utilizada é capaz de se conectar tanto ao CLP Siemens S7 1200 quanto ao CLP Schneider M221, com a devida configuração do driver de comunicação para cada CLP em específico o que gerou a necessidade de dois arquivos específicos para aplicar à HMI, cada qual dedicado ao CLP em execução.

3.7 Fluxograma do Processo

Com base na disposição física do sistema, foi elaborado um fluxograma representativo do processo de separação de peças, conforme ilustrado na Figura 24. Esse fluxograma possibilita a implementação da solução de planejamento automático, cuja programação ocorre mediante solicitações oriundas do cliente. A lógica de execução do plano prevê, de forma sistemática, a verificação e separação das peças previamente posicionadas pelo operador no início da esteira, mediante solicitação de inserção da próxima peça, indicada ao operador em um campo dedicado para mensagens na HMI. O processo assegura um fluxo contínuo e organizado ao longo de toda a execução da tarefa

de separação por meio de avisos na HMI e, caso o operador insira uma peça divergente da solicitação, descarta os produtos não solicitados. A solução proposta implementou essa lógica na API desenvolvida.

Figura 24 – Fluxograma detalhado de processos da bancada de teste.



Fonte: Próprio autor.

4 RESULTADOS E DISCUSSÕES

4.1 Descrição Geral do Sistema Proposto

O produto final foi projetado para monitorar e registrar todas as entradas, saídas e áreas de memória de CLPs (relevantes ao processo de separação) por meio do protocolo ModbusTCP. A HMI está igualmente conectada à rede, também via ModbusTCP quando utilizado o CLP da fabricante Schneider e via SIMATIC S7 Ethernet quando utilizando o CLP da fabricante Siemens, centralizando todas as informações inseridas pelo operador no CLP que está sob monitoramento da API. Após a inserção das requisições dos clientes na HMI, a API traduz essas informações em um problema descrito em linguagem PDDL, associa-o ao domínio da planta, solicita um plano utilizando um planejador automático hospedado em nuvem (nesse caso, a plataforma Planning Domains com o Solver LAMA-first) e inicia a execução do plano obtido.

Durante a execução do plano, a API monitora continuamente o CLP, exibe na HMI o produto que deve ser colocado na esteira e verifica o atendimento das precondições necessárias para a realização sequencial das ações prescritas integralmente pelo planejador automático. Em caso de falha na sequência de execução, a API, na versão atual, descarta o produto para a caixa localizada ao final da esteira transportadora e solicita novamente o produto correto, a fim de retomar a execução do plano.

O método de configuração da API para uso, toda a programação desenvolvida para a API, o CLP e a HMI estão disponíveis no GitHub[1] e a validação do produto final na bancada de teste, apresentada através de vídeos, encontram-se disponíveis no Google Drive[2] por prazo indeterminado.

Durante os testes, constatou-se que tudo se integra com fluidez, o CLP se comporta como um *driver* e não mais o cerne da operação lógica, assumindo uma função principal de ser a ponte entre o mundo físico, coletando todas as informações de campo, ao mundo deliberativo do planejamento, por meio da API quando *online*. Os comandos do operador na HMI e os estados dos sensores da planta, se comportam como entradas do domínio físico e das solicitações do cliente.

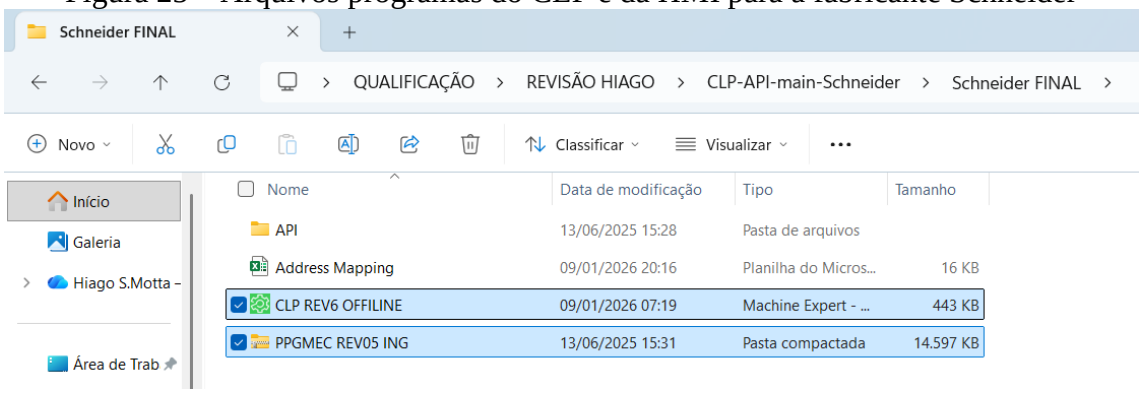
[1] https://github.com/HiagoSilvaMotta/PlanPAS-2.0

[2] https://drive.google.com/drive/folders/1FXq5e6yw_dWX9sZfC_j-L-cXZwh39-el

4.2 Rotina de configuração

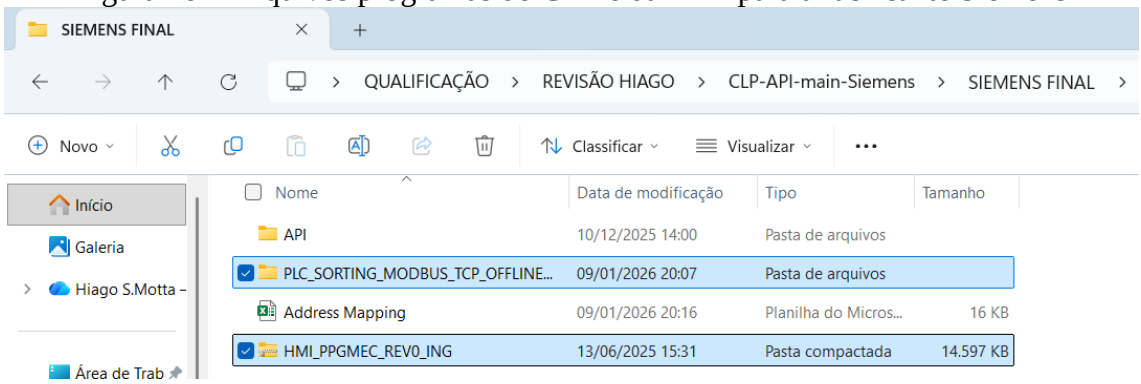
Uma vez que baixado os arquivos disponíveis no GitHub, as configurações de programação do CLP e da HMI, evidenciados na Figura 25 para o CLP da Schneider e na Figura 26 para o CLP da Siemens. Esses devem ser baixados nos respectivos dispositivos, a única configuração necessária posteriormente será a definição dos IPs.

Figura 25 – Arquivos programas do CLP e da HMI para a fabricante Schneider



Fonte: Próprio autor.

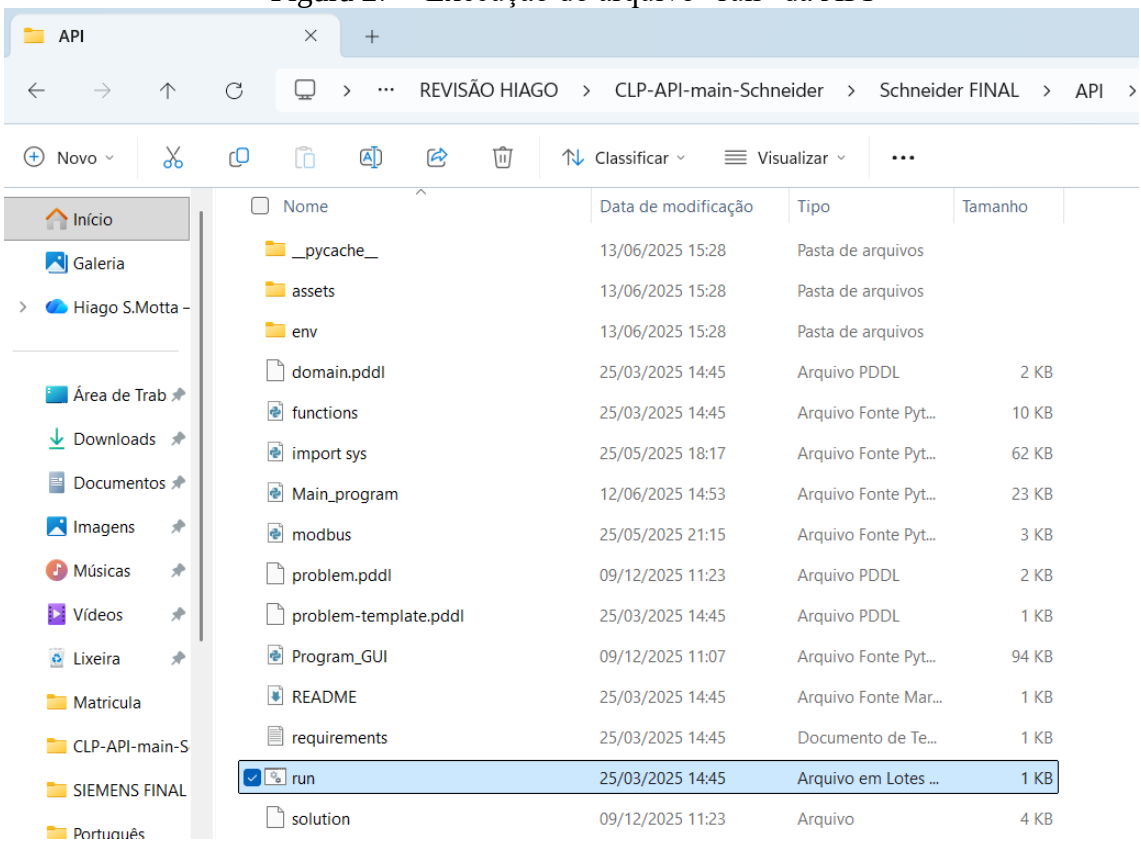
Figura 26 – Arquivos programas do CLP e da HMI para a fabricante Siemens



Fonte: Próprio autor.

Dentro da pasta “Schneider FINAL” ou “Siemens FINAL” existe a pasta “API”. Essa pasta tem todos os arquivos da API e é inicializada através da execução do arquivo “run”, mostrado na Figura 27, que instala todos os requerimentos necessários para execução previamente.

Figura 27 – Execução do arquivo “run” da API

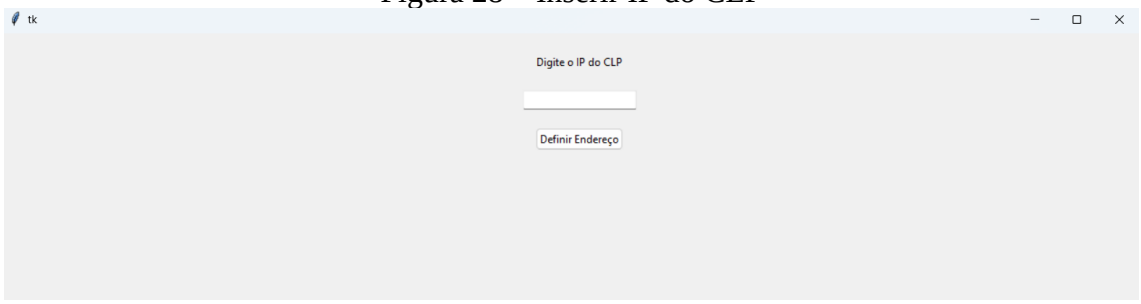


Fonte: Próprio autor.

Em seguida abrirá a tela para inserção do IP conforme a Figura 28. Deve-se realizar a inserção do IP 192.168.0.2, para programa desenvolvido no CLP da fabricante Schneider, ou do IP 192.168.0.111, para o programa desenvolvido no CLP da fabricante Siemens.

Observação: A máquina que irá executar a API deve possuir um IP fixo 192.168.0.12 (ou qualquer IP com final diferente do CLP e da HMI) nas configurações de rede do computador.

Figura 28 – Inserir IP do CLP



Fonte: Próprio autor.

Terminado essa configuração, a API já está conectada ao CLP e aguarda por solicitações do operador via HMI.

4.3 Compatibilidade da API com CLPs de diferentes fabricantes

Como descrito anteriormente, foi utilizado um método de comunicação via protocolo Modbus TCP. Contudo, o protocolo apresentou instabilidades quando comunicava com o CLP S7 1200 por não possuir o protocolo Modbus TCP nativo. A versão mais recente do arquivo da API se apresentou compatível com ambos os CLPs após implementar a lógica de requisição de comunicação condicionada a última solicitação. Ou seja, foi otimizado a quantidade de solicitações que a API realiza para acessar os dados do CLP reestabelecendo a conexão somente quando necessário. Assim a solução foi validada e testada em diferentes fabricantes com a mesma API.

Ambos programas do CLP possuem o mesmo mapa de endereço dentro da API, não sendo necessário nenhum tipo de adequação. Contudo, vale lembrar que cada um dos fabricantes vai mobilizar diferentes tipos de memória para a comunicação Modbus. Assim no CLP M221 da Schneider utilizamos as %M para sinais digitais e %MW para sinais dos tipos inteiros enquanto no CLP S7 1200 da Siemens utilizou-se %Q para os sinais digitais e os *Data Blocks* para sinais dos tipos inteiros. O mesmo ocorre com a HMI, como são dois tipos de comunicação diferentes com a interface, necessita atualização dos tipos de memorias.

4.4 Funcionalidades Relevantes

4.4.1 Gerador de Problemas e Condições Iniciais

No caso específico da bancada utilizada no estudo, os objetos definidos foram: três atuadores, três posições de estoque, uma esteira, 6 tipos de peças/produtos, uma posição inicial, as condições de cada objeto e o cenário objetivo. A API desenvolvida gera, automaticamente, o arquivo problema em PDDL, a partir dos dados de sensores e das entradas do operador na HMI, acessados pela API em posições de memória do CLP. O problema em PDDL, gerado automaticamente a partir da condição observada na planta, contempla a definição de objetos como peças, atuadores, estoques, esteiras e posições

dentro do processo, além de parâmetros lógicos que descrevem a configuração da planta e uma condição que descreve o cenário objetivo. O usuário pode definir, por exemplo, quantas peças existem no sistema, quais são seus tipos, quantos atuadores e estoques existem, qual a posição inicial de cada peça e quais devem ser os destinos finais. Essa entrada pode ser feita pelo operador ou pelo monitoramento dos sensores.

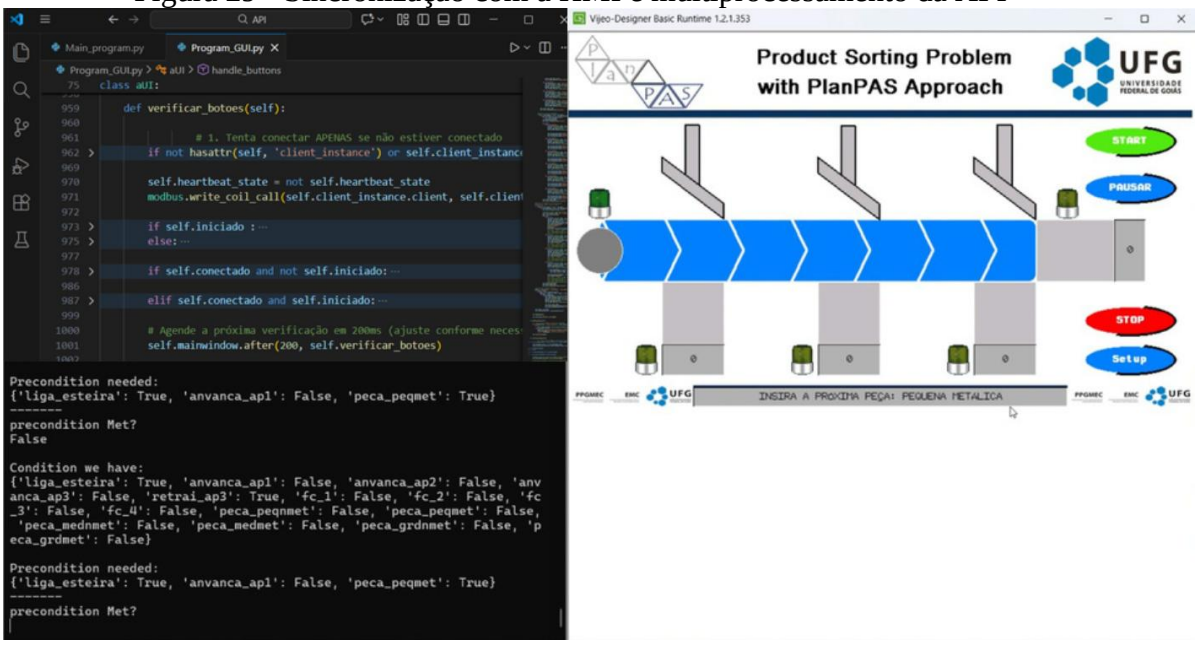
Esse arquivo problema descreve objetos, condições iniciais e metas desejadas para um determinado processo, permitindo a aplicação imediata de planejadores automáticos. Combinando o arquivo problema, gerado da forma descrita, com um arquivo domínio previamente modelado e avaliado no *planning.domains*, e devidamente armazenado no servidor local, a API acessa o serviço do *planning.domains* via protocolo HTTP. Entrega os arquivos domínio e problema e solicita um arquivo com o plano-solução. Um planejador busca por um plano solução e, caso a busca seja bem sucedida, o mesmo é entregue à API, que o interpreta e inicializa sua execução na planta.

Esse plano, composto por ações sequenciadas e associadas a precondições específicas, é continuamente monitorado pela API. A execução ocorre: à medida que as precondições de cada ação são satisfeitas, com base nos sinais captados do ambiente físico, as instruções são enviadas diretamente ao CLP via protocolo Modbus TCP. Esse ciclo de monitoramento e execução permite uma sincronização entre a lógica simbólica de planejamento e a realidade operacional da planta, promovendo autonomia e flexibilidade no controle.

4.4.2 Sincronização e *Multithreading*

O *Multithreading* (capacidade de realizar várias tarefas em um único processo) foi implementado à lógica da API para permitir que o processo completo fosse executado por diversas tarefas em paralelo dividindo partes de execução e a parte de sincronização do CLP. Em outras palavras, a API é capaz de monitorar todos os sensores, atuadores, comandos da HMI, realizar solicitações de planejamento automático e executar ações em tarefas paralelas, e ainda trocar informações entre as múltiplas tarefas. Isso permitiu que, enquanto o processo de solicitação de plano e execução dos comandos estavam rodando em uma tarefa, a sincronização com botões de controle/comando fossem atualizadas permitindo a pausa, parada ou reprogramação em qualquer momento do processo conforme ilustrado na Figura 29.

Figura 29 - Sincronização com a HMI e multiprocessamento da API



Fonte: Autor.

4.4.3 Funcionalidade de replanejamento

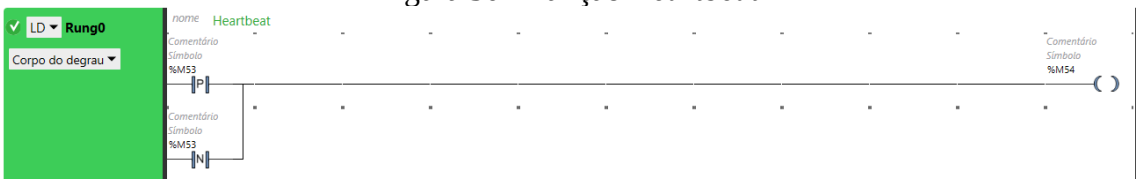
Durante a operação em modo de planejamento automático a API desenvolvida tem a possibilidade de replanejamento manual em casos de necessidade ou adaptação da demanda do cliente. Nesse modo a API para temporariamente a rotina do último plano e aguarda com que o operado adeque a nova solicitação. Assim que finalizado o operador deve pressionar o botão "*Start*" para que a API possa iniciar uma nova solicitação de plano ao *planning.domains* baseado nas condições do estado atual da planta. O plano obtido considera as peças que haviam sidos já separadas e religa esteira para continuar o processo de separação das demandas remanescentes e reprogramadas.

4.4.4 Modo de contingência

A API possui capacidade de mostrar ao CLP quando está *online*, por meio de uma funcionalidade chamada *Heartbeat* mostrada na Figura 30, para não executar nenhum plano de contingência se solicitado pelo operador. Caso o sinal não seja interpretado pelo CLP, ou seja, API estiver *offline*, conforme a Figura 31, o CLP é capaz ainda de operar parcialmente por meio de uma lógica básica. Essa funcionalidade implementou significativamente o nível de robustez da solução mantendo a planta operacional mesmo em situações de instabilidade da conexão à internet, que inviabilizava o acesso ao

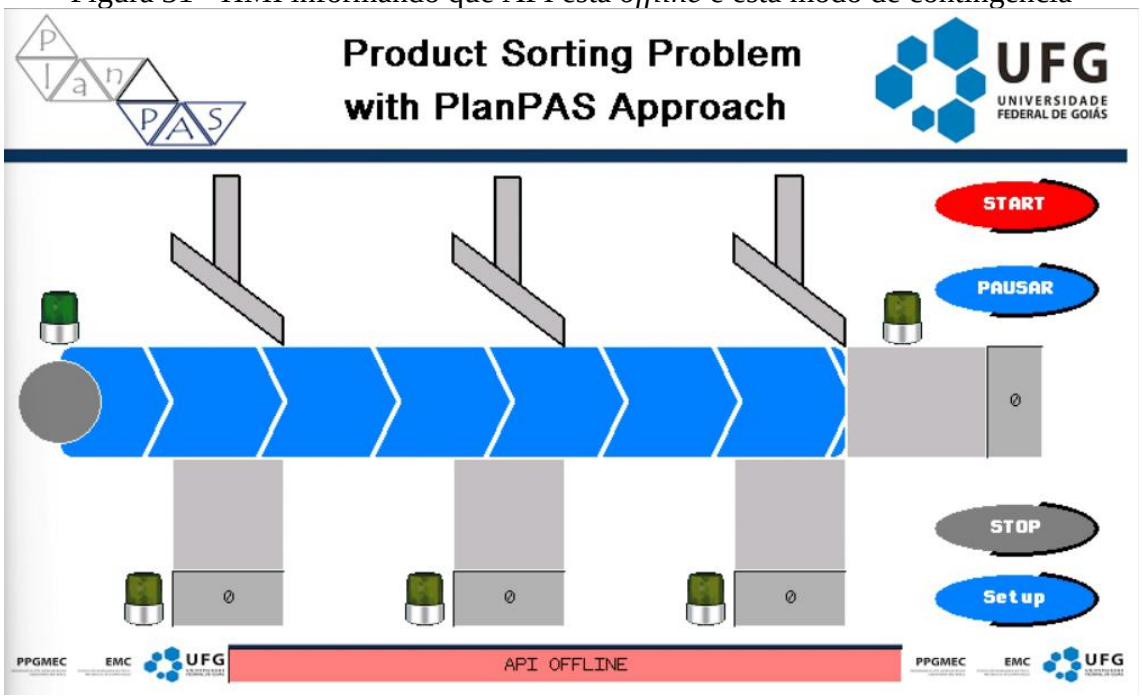
planning.domains, ou servidor onde está localizada a API. A implementação de planos de contingência ainda requer bastante desenvolvimento na linguagem de baixo nível do CLP.

Figura 30 - Função *Heartbeat*



Fonte: Autor.

Figura 31 - HMI informando que API está *offline* e está modo de contingência



Fonte: Autor.

4.5 Avaliação de Escalabilidade do Sistema Proposto

Para avaliar a escalabilidade da proposta, a partir da modelagem do domínio de planejamento, foram definidos cinco problemas de planejamento com variação dos parâmetros:

Problema 1: Solicitação de 2 peças em um caso que possui 3 estoques possíveis, 6 tipos de peças e 3 atuadores.

Problema 2: Solicitação de 4 peças em um caso que possui 3 estoques possíveis, 6 tipos de peças e 3 atuadores.

Problema 3: Solicitação de 216 peças em um caso que possui 3 estoques possíveis, 6 tamanhos de peças e 3 atuadores.

Problema 4: Solicitação de 216 peças em um caso que possui 4 estoques possíveis, 6 tipos de peças e 4 atuadores;

Problema 5: Solicitação de 216 peças em um caso que possui 6 estoques possíveis, 6 tipos de peças e 6 atuadores;

A estrutura do problema em PDDL mostrou-se facilmente adaptável. Cada problema de planejamento foi submetido cinco vezes ao *Solver* LAMA-first, que forneceu planos-solução em tempos médios de 0,39 ± 0,01 segundos, 0,39 ± 0,01 segundos, 2,84 ± 0,20 segundos, 3,25 ± 0,04 segundos e 4,39 ± 0,30 segundos, em que as incertezas padrão estão apresentadas na forma expandida, para um fator de abrangência igual a 2,776, obtido da distribuição *t* com 95% de confiança e 4 graus de liberdade conforme observado na Tabela 3.

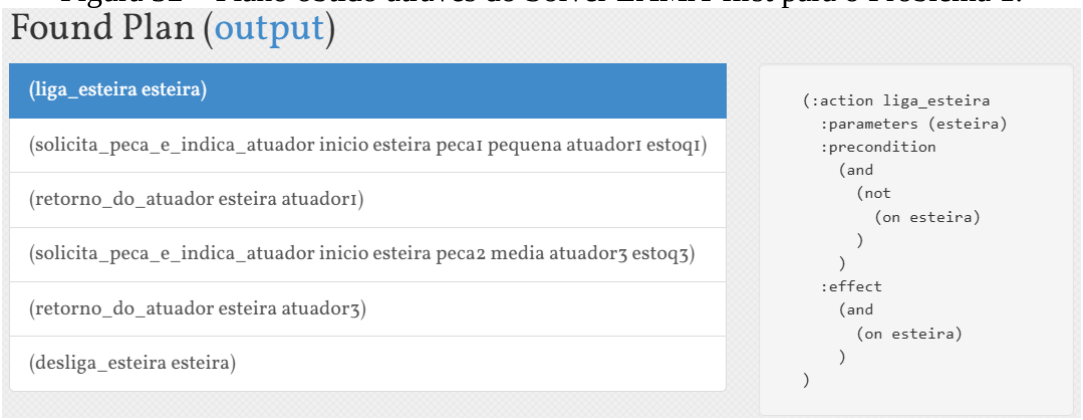
Tabela 3 – Teste de escalabilidade do planejador

Problema Proposto	Quantidades				Tempo Médio de Planejamento em segundos
	Peças	Tipos de peças	Estoques	Atuadores	
1	2	6	3	3	0,39 ± 0,01
2	6	6	3	3	0,39 ± 0,01
3	216	6	3	3	2,84 ± 0,20
4	216	6	4	4	3,25 ± 0,04
5	216	6	6	6	4,39 ± 0,30

Fonte: Autor.

Os resultados demonstram baixo tempo de processamento e capacidade de adaptação a diferentes configurações. Os planos gerados são integrados aos controladores da planta física por meio da API, viabilizando a programação e reprogramação automáticas com base nas solicitações do cliente. Para exemplificar o arquivo solução gerado, o plano do problema 1 é ilustrado na Figura 32.

Figura 32 – Plano obtido através do Solver LAMA-first para o Problema 1.



Fonte: Próprio autor.

Percebe-se, portanto, que o sistema proposto é mais adaptável, escalável e inteligente do que as soluções tradicionais, que centraliza a lógica de operação no controlador, incorporando conceitos de convergência à Indústria 4.0 e Indústria 5.0.

Para grande parte da indústria atualmente difundida, entende-se que haverá somente um custo de implementação em nível de programação, não exigindo grandes modificações físicas da instalação ou o custo de adquirir novos CLPs com tecnologia de IA embarcada.

5. CONCLUSÕES

A evolução da Indústria, acompanhada do surgimento de termos como Indústria 4.0 e Indústria 5.0, tem demandado transformações nos sistemas de manufatura atuais, exigindo maior integração dos sistemas modulares e maior reconfigurabilidade nos diferentes níveis da pirâmide de automação. Os avanços respondem à crescente necessidade de adaptação das linhas de produção com ciclos de vida mais curtos e demandas de produtos personalizáveis, o que pressiona tanto os custos quanto os prazos. Embora os sistemas automatizados atuais possuam velocidade e precisão, o grande desafio continua sendo a reprogramação frequente, exigindo mão de obra especializada e interferindo na flexibilidade operacional.

Neste cenário, a separação entre o conhecimento do domínio e a lógica de execução viabiliza a geração automática de planos com base na solicitação do cliente e restrições definidas do domínio, sem a necessidade de modificar manualmente a lógica operacional dos dispositivos de controle, nesse caso, o CLP. Isso pode reduzir significativamente o tempo e o esforço necessários para reconfigurar sistemas industriais diante de mudanças no processo produtivo. Além disso, o uso da API desenvolvida possibilitou que o raciocínio de alto nível fosse capaz de executar tarefas atribuídas à linguagem de programação de baixo nível, podendo ampliar a autonomia dos sistemas industriais.

A API desenvolvida representa uma solução para a integração da lógica de planejamento automático com a execução física de processos industriais já existentes. Operando como integrador entre o planejador automático e os dispositivos de campo, a API interpreta planos em linguagem PDDL e traduz as ações planejadas para comandos Modbus TCP, lendo e escrevendo diretamente nas entradas e saídas dos dispositivos de campo. Dessa forma, é possível conectar, sem grandes investimentos, um ambiente deliberativo de alto nível com os ciclos de controle de baixo nível de CLPs já inseridos, aceitos e típicos de ambientes industriais.

Ao eliminar a dependência de reprogramação manual a cada *setup* da aplicação e permitir o replanejamento em qualquer momento, a abordagem proposta consolida os princípios da Indústria 4.0 — aumentando a flexibilidade, a customização e a colaboração humano-máquina nos processos, bem como da Indústria 5.0 — ao viabilizar uma melhoria nas condições de trabalho dos programadores, responsáveis por tarefas

repetitivas de *setup* e reprogramação de máquinas para atender demandas altamente variáveis devido a personalização de pedido ou um produto.

A implementação do modo de contingência conferiu robustez ao sistema, atendendo níveis de segurança característicos de ambientes industriais. Em paralelo, a solução proporcionou flexibilidade frente a novas demandas produtivas, permitindo a interrupção controlada de planos em execução, a reedição dos requisitos de clientes e a solicitação de replanejamentos *online*. Assim, capaz de se adaptar dinamicamente às mudanças de produto e demanda com alto nível de segurança, sem comprometer eficiência ou a qualidade, essa integração representa um passo importante para o desenvolvimento de fábricas inteligentes.

Por fim, os objetivos deste trabalho foram estruturados para viabilizar a integração efetiva entre o planejamento automático em alto nível de programação, baseado em linguagem PDDL, e a execução em baixo nível em ambientes controlados por CLPs via protocolo Modbus TCP. A arquitetura assegurou a modularidade, flexibilidade e operação *online*, que permitiu a adaptação mais dinâmica do sistema às demandas do processo produtivo. A implementação da API possibilitou a intermediação entre os domínios decisórios e operacional, enquanto o mecanismo de geração de arquivos solução garante a correta execução dos planos. Além disso, a estratégia desenvolvida de sincronização entre a decisão lógica e ações físicas implementou confiabilidade, reforçando a viabilidade da abordagem em cenários industriais reais.

TRABALHOS FUTUROS

Este trabalho abre caminho para novas investigações, sendo possível aprofundar a análise por meio de estudos que envolvam:

1. Validar o projeto em outros modelos de CLPs, de fabricantes diferentes, para validar a ainda mais a flexibilidade da API utilizando o protocolo ModbusTCP;
2. Acrescentar a necessidade de melhoria para usar protocolos mais aplicados à Indústria 4.0, como MQTT e OPC-UA;
3. Validar o uso da API em um ambiente fabril controlado;
4. Aperfeiçoamento de servidor para geração de planos fora da nuvem, ou seja, diminuir a dependência de acesso ao servidor web onde está hospedado o *solver*;
5. Desenvolver módulos reutilizáveis para diversos atuadores e sensores usuais para facilitar a montagem dos domínios em PDDL e;
6. Desenvolver módulos para a API, assim abstrair o desenvolvimento da mesma e reduzir a complexibilidade de programação.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMADA-LOBO, F. The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). Journal of Innovation Management, v. 3, n. 4, p. 16-21, 2016.

AU, T. C.; KUTER, U.; NAU, D. Planning for interactions among autonomous agents. In: HINDRIKS, K. V. et al. (org.). Programming multi-agent systems. ProMAS 2008. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009. v. 5442. Disponível em: https://doi.org/10.1007/978-3-642-03278-3_1. Acesso em: 8 ago. 2025.

BELLMAN, R. Dynamic Programming. Princeton, NJ: Princeton University Press, 1957.

BOLTON, W. Programmable Logic Controllers. 6. ed. Oxford: Newnes, 2015.

BRASIL. Ministério da Fazenda. Nova Indústria Brasil (NIB). Brasília, 2024. Disponível em: https://www.gov.br/fazenda/pt-br/acesso-a-informacao/acoes-e-programas/transformacao-ecologica/programas-em-destaque/nova-industria-brasil. Acesso em: 21 fev. 2026.

BRINKSMA, E.; MADER, A. Verification and Optimization of a PLC Control Schedule. In: Proceedings of the 7th International SPIN Workshop on SPIN Model Checking and Software Verification, 7., 2006, Auckland. Lecture Notes in Computer Science, v. 4144, p. 61–75, dez. 2006. DOI: 10.1007/10722468_5.

BURGGRÄF, P.; WANNLÖFFEL, T.; DANKABA, A.; MAYER, J. B.; ADLON, T.; WEISS, B. Achieving parametric transparency in model-based factory planning. Production Engineering Research and Development, v. 15, p. 57–67, 2021. Disponível em: https://doi.org/10.1007/s11740-020-01010-6. Acesso em: 8 ago. 2025.

CLARIVATE, 2024. Web of science. <https://www.webofscience.com/wos/author/search>.

COLLINS, J.; CHO, H. Pymodbus: Modbus protocol implementation for Python. Versão 3.6.3. 2025. Disponível em: https://pypi.org/project/pymodbus/. Acesso em: 11 ago. 2025.

D-LINK. DES-1016D: 16-Port Fast Ethernet Switch (Datasheet). Eschborn, Alemanha: D-Link GmbH, 1 dez. 2002. Disponível em: https://gzhls.at/blob/ldb/4/e/5/2/227d6e8cb7748cb264f8c559af23e66d729a.pdf. Acesso em: 19 jan. 2026.

DE PAOLA, A. An Ontology-Based Autonomic System for Ambient Intelligence Scenarios. In: CORCHADO, Juan M. et al. (org.). Advances in Intelligent Systems and Computing. Cham: Springer, 2014. p. 3-13. DOI: https://doi.org/10.1007/978-3-319-03992-3_1.

ELSEVIER, B., 2024. Scopus. <www.scopus.com>.

EUROPEAN COMMISSION. Industry 5.0: towards a sustainable, human-centric and resilient European industry. Policy Brief. Luxembourg: Publications Office of the European Union, 2021.

EXSTO TECNOLOGIA. XC243: Banco de Ensaios para Processo de Manufatura com Dispositivos de Comando. Santa Rita do Sapucaí: Exsto Tecnologia, [s.d.]. Catálogo técnico. Disponível em: https://exsto.com.br/Catalogos/SENAIPB/XC243.pdf. Acesso em: 10 jan. 2026.

FALKIEWICZ, K. Profibus DP: Communication Protocol, PLC Integration and Application. International Journal of Innovative Technology and Exploring Engineering, v. 13, n. 9, p. 35-37, ago. 2024. DOI: 10.35940/ijitee.I9938.13090824.

FELD, J. PROFINET – Scalable factory communication for all applications. In: Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems (WFCS), 2004, p. … DOI: 10.1109/WFCS.2004.1377673.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. 2000. Tese (Doutorado em Ciências da Computação) – University of California, Irvine.

FIKES, R. E.; NILSSON, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. 1971. *Artificial Intelligence*, 2(3-4), 189-208.

FONSECA, J. P. S. Analisador de planos para sistemas automatizados baseados em CLPs. 2013. 180 f. Dissertação (Mestrado em Engenharia Mecânica) – Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, 2013.

FONSECA, J. P. S.; SOUSA, A. R.; FERREIRA, M. V. M.; TAVARES, J. J. P. Z. S. PlanPAS: PLC and automated planning integration. International Journal of Computer Integrated Manufacturing, v. 29, n. 11, p. 1200–1217, 2016. Disponível em: https://doi.org/10.1080/0951192X.2015.1067909.

FOX, M.; LONG, D. PDDL+: Modeling Continuous Time-dependent Effects. Journal of Artificial Intelligence Research, v. 26, p. 411-472, 2006.

FOX, M.; LONG, D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Problems. Journal of Artificial Intelligence Research, v. 20, p. 61-124, 2003.

GAMESS, E.; SMITH, B.; FRANCIA III, G. Performance Evaluation of Modbus TCP in Normal Operation and Under A Distributed Denial of Service Attack. International Journal of Computer Networks and Communications, v. 12, n. 2, p. 1–21, mar. 2020.

GEFFNER, H.; BONET, B. A Concise Introduction to Models and Methods for Automated Planning. 2. ed. San Rafael: Morgan & Claypool, 2013.

GEREVINI, A.; LONG, D. Plan constraints and preferences in PDDL3. In: Proceedings of the ICAPS 2005 Workshop on Planning Competition. Monterey, CA: AAAI Press, 2005.

GEREVINI, A.; LONG, D. Plan constraints and preferences in PDDL3. Technical Report, Department of Electronics for Automation, University of Brescia, 2005. Disponível em: http://icaps05.icaps-conference.org/. Acesso em: 11 ago. 2025.

GHALLAB, M.; NAU, D.; TRAVERSO, P. Automated planning and acting. Cambridge: Cambridge University Press, 2016.

GHALLAB, M.; NAU, D.; TRAVERSO, P. Automated Planning: Theory and Practice. San Francisco: Morgan Kaufmann, 2004.

GRAESSLER, I.; PÖHLER, A. Integration of a digital twin as human representation in a scheduling procedure of a cyber-physical production system. In: 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 Dez. 2017. IEEEXplore, 2017. p. 289–293. doi:10.1109/IEEM.2017.8289898.

HELMERT, M. The Fast Downward planning system. Journal of Artificial Intelligence Research, v. 26, p. 191–246, 2006.

HIND, S.; SEITZ, T. Cynical technical practice: From AI to APIs. Convergence: The International Journal of Research into New Media Technologies, v. 30, n. 1, p. 29–48, 2022. doi:10.1177/13548565221133248.

HOBSBAWM, E. J. The age of revolution: Europe 1789-1848. London: Weidenfeld & Nicolson, 1962.

HOEBERT, T.; SOROKINA, J.; KARASTOYANOV, D.; GALAMBOS, P.; PANEV, S. Knowledge-driven framework for industrial robotic systems. Journal of Intelligent Manufacturing, v. 34, p. 771–788, 2023. Disponível em: https://doi.org/10.1007/s10845-021-01826-8. Acesso em: 8 ago. 2025.

HOU, L.; WU, S.; ZHANG, G.; TAN, Y.; WEN, H. Internet of Things (IoT) and Its Applications in Manufacturing: A Review. Journal of Manufacturing Systems, 2021.

IBGE. CNAE 2.0: Notas explicativas. Rio de Janeiro: IBGE, 2024. Disponível em: https://concla.ibge.gov.br. Acesso em: 17 jan. 2026.

IBN-KAHLA, M. Wireless Sensor Networks: Applications and Protocols. CRC Press, 2023. (Ideal para referenciar o MQTT em ambientes industriais dispersos).

JESCHKE, S.; BRECHER, C.; MEISEN, T.; ÖZDEMIR, D.; ESCHERT, T. Industrial Internet of Things: Cybermanufacturing Systems. Cham: Springer, 2017. ISBN 978-3-319-42559-7.

KAELBLING, L. P.; LITTMAN, M. L.; CASSANDRAS, C. G. Planning and acting in partially observable stochastic domains. Artificial Intelligence, v. 101, n. 1-2, p. 99-134, 1998.

KALATEC. IEC 61131-3 e a programação CLP: qual sua importância? s.d. Disponível em: https://blog.kalatec.com.br/iec-61131-3/. Acesso em: 9 ago. 2025.

KOMENDA, A.; ŠTOLBA, M.; KOVÁCS, D. L.; BORDINI, R. H.; VIZZARI, G.; PĚCHOUČEK, M.; LOGAN, B. The International Competition of Distributed and Multiagent Planners (CoDMAP). AI Magazine, v. 37, n. 3, p. 109–115, 2016.

LAZAROIU, G.; VALASKAKOVA, K.; MACHOVA, V.; KYASAKOVA, L. Cognitive digital twin-based Internet of Robotic Things, multi-sensory extended reality and simulation modelling technologies, and generative artificial intelligence and cyber-physical manufacturing systems in the immersive industrial metaverse. Equilibrium: Quarterly Journal of Economics and Economic Policy, v. 19, n. 3, p. 719–748, 2024. doi:10.24136/eq.3131.

LEFEBVRE, L. pyModbusTCP: a simple Modbus/TCP client library for Python. 2024. Disponível em: https://pypi.org/project/pyModbusTCP/. Acesso em: 9 ago. 2025.

LEFEBVRE, S. pyModbusTCP: A simple pure Python Modbus/TCP client. Versão 0.3.0. 2024. Disponível em: https://pypi.org/project/pyModbusTCP/. Acesso em: 11 ago. 2025.

LEITÃO, P.; COLOMBO, A. W.; KARNOUSKOS, S. Industrial automation based on cyber-physical systems technologies: prototype implementations and challenges. Computers in Industry, v. 81, p. 11–25, 2016. Disponível em: https://doi.org/10.1016/j.compind.2015.08.004. Acesso em: 8 ago. 2025.

LENG, J.; SHAO, W.; WANG, B.; ZHAO, J.; LIU, Q.; JIN, X.; WEI, L.; ZHUANG, Z.; CHEN, X. Industry 5.0: Prospect and retrospect. Journal of Manufacturing Systems, v. 65, p. 279-295, 2022.

MARX, K. Das Kapital: Kritik der politischen Oekonomie. Hamburg: Verlag von Otto Meissner, 1867. Bd. 1.

MARRELLA, A., Automated Planning for Business Process Management, 2019. J Data Semant 8, 79–98, <https://doi.org/10.1007/s13740-018-0096-0>.

MAZUR, G. A.; WEINDORF, W. J. Programmable Logic Controllers: Principles and Applications. 3. ed. Orland Park, Illinois: American Technical Publishers, 2021. ISBN 978-0-8269-1396-8.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. PDDL - The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

MONOSTORI, L. Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP, v. 17, p. 9-13, 2014.

MONOSTORI, L.; KÁDÁR, B.; BAUERNHANSL, T.; KONDOH, S.; KUMARA, S.; REINHART, G.; SAUER, O.; SCHUH, G.; SIHN, W.; UEDA, K. Cyber-physical systems in manufacturing. CIRP Annals, v. 65, n. 2, p. 621–641, 2016. Disponível em: https://doi.org/10.1016/j.cirp.2016.06.005. Acesso em: 8 ago. 2025.

NAHAVANDI, S. Industry 5.0—A human-centric solution. Sustainability, v. 11, n. 16, p. 4371, 2019. Disponível em: https://doi.org/10.3390/su11164371. Acesso em: 8 ago. 2025.

NILSSON, N. J. Principles of Artificial Intelligence. Palo Alto: Tioga Publishing Company, 1980.

PAGANI, R. N.; KOVALESKI, J. L.; RESENDE, L. M. Methodi Ordinatio: a proposed methodology to select and rank relevant scientific papers encompassing the main classic and contemporary factors. Scientometrics, v. 105, n. 3, p. 2109–2135, 2015. DOI: 10.1007/s11192-015-1744-x.

PAPAZOGLOU, M. P.; VAN DEN HEUVEL, W. Service-oriented architectures: approaches, technologies and research issues. The VLDB Journal, v. 16, n. 3, p. 389–415, 2007. DOI: 10.1007/s00778-007-0044-3.

PEREIRA, V.; BASILIO, M. P.; SANTOS, C. H. T. PyBibX: A Python Library for Bibliometric and Scientometric Analysis Powered with Artificial Intelligence Tools. Data Technologies and Applications, v. 59, n. 2, p. 302-337, 2025. Disponível em: https://pypi.org/project/pybibx/. Acesso em: 13 out. 2025..

PETRUZELLA, F. Controladores Lógicos Programáveis. 4. ed. Porto Alegre: AMGH, 2014.

PUTERMAN, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York: John Wiley & Sons, 1994.

REITER, R. On Closed World Data Bases. In: GALLAIRE, H.; MINKER, J. (Eds.). Logic and Data Bases. Boston, MA: Springer US, 1978. p. 55–76.

RICHTER, S.; WESTPHAL, M. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research, v. 39, p. 127–177, 2010.

RUSSELL, S.; NORVIG, P. Inteligência Artificial: uma abordagem moderna. 4. ed. Rio de Janeiro: Elsevier, 2020.

SANNER, S. Relational Dynamic Influence Diagram Language (RDDL): Language Description. Unpublished ms. Australian National University, 2010.

SCHNEIDER ELECTRIC. HMIGXU5512: Interface homem-máquina colorida - 10,1 polegadas - TFT WVGA - portas seriais, Ethernet e USB. São Paulo: Schneider Electric Brasil, [2026]. Disponível em: https://www.se.com/br/pt/product/HMIGXU5512/. Acesso em: 29 jan. 2026.

SCHNEIDER ELECTRIC. Modicon M221 logic controller – TM221CE16T. Disponível em: https://www.se.com/. Acesso em: 8 ago. 2025.

SCHNEIDER ELECTRIC. TM221CE16T: Controlador lógico programável - 9 entradas e 7 saídas digitais - saídas source PNP - Modbus TCP - Ethernet/IP Slave - 24

VDC. São Paulo: Schneider Electric Brasil, [2026]. Disponível em: https://www.se.com.br/pt/product/TM221CE16T/. Acesso em: 19 jan. 2026.

SCHNEIDER, M. Redes industriais e protocolos de comunicação. São Paulo: Editora Atlas, 2020.

SCHWAB, K. The Fourth Industrial Revolution. Geneva: World Economic Forum, 2016.

SIEMENS. SIMATIC S7-1200, CPU 212C, Compact CPU, DC/DC/Relay (6ES7212-1HE31-0XB0). Munich, Alemanha: Siemens AG, [2026]. Disponível em: https://sieportal.siemens.com/en-ww/products-services/detail/6ES7212-1HE31-0XB0. Acesso em: 19 jan. 2026.

SILVA, E. A. da. Introdução às linguagens de programação para CLP. São Paulo: Blucher, 2016.

SMC INTERNATIONAL TRAINING. Industry 4.0 didactic equipment. Disponível em: https://www.smctraining.com/en/webpage/indexpage/312/. Acesso em: 30 maio 2025.

TANENBAUM, A. S.; WETHERALL, D. J. Redes de computadores. 6. ed. Rio de Janeiro: Elsevier, 2021.

VAQUERO, T. S.; SILVA, J. R.; TONIDANDEL, F.; BECK, J. C. itSIMPLE: towards an integrated design system for real planning applications. The Knowledge Engineering Review, v. 28, n. 2, p. 215–230, 2013.

VOGEL-HEUSER, B.; HESS, D. Guest editorial: Industry 4.0—prerequisites and visions. IEEE Transactions on Automation Science and Engineering, v. 13, n. 2, p. 411–413, 2016. Disponível em: https://doi.org/10.1109/TASE.2016.2523639. Acesso em: 8 ago. 2025.

VYSKOČIL, J.; NOVÁK, P.; ONDERKOVÁ, B.; KADERA, P. A digital twin-based distributed manufacturing execution system for Industry 4.0 with AI-powered on-the-fly replanning capabilities. Sustainability, v. 15, n. 7, p. 6251, 2023. Disponível em: https://doi.org/10.3390/su15076251. Acesso em: 8 ago. 2025.

YOUNES, H. L. S.; LITTMAN, M. L. PPDDL1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, 2004.

YOUNES, H. L. S.; LITTMAN, M. L.; WEISSMAN, D.; ASMUTH, J. The First Probabilistic Track of the International Planning Competition. Journal of Artificial Intelligence Research, v. 24, p. 851–887, 2005.

WALLY, B., VYSKOČIL, J., NOVAK, P., HUEMER, C., SINDELAR, R., KADERA, P., MAZAK, A., & WIMMER, M. Production planning with iec 62264 and pddl. In 2019 IEEE 17th international conference on industrial informatics (INDIN) (Vol, 1, pp. 492–499). https://doi.org/10.1109/INDIN41052.2019.8972050

WALLY, B.; HUEMER, C.; MAZAK, A.; WIMMER, M. Leveraging iterative plan refinement for reactive smart manufacturing systems. IEEE Transactions on Automation Science and Engineering, v. 18, n. 1, p. 230–243, 2021. doi:10.1109/TASE.2020.3018402.