UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Simulação de CLP em Navegadores Web: Uma Ferramenta para a Melhoria da Educação em Engenharia Elétrica.

Gabriel Vinícios Silva Maganha

Itajubá, 29 de janeiro de 2024

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Gabriel Vinícios Silva Maganha

Simulação de CLP em Navegadores Web: Uma Ferramenta para a Melhoria da Educação em Engenharia Elétrica.

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração: Sistemas Elétricos de Potência

Orientador: Prof. Dr. Ph.D Antônio Carlos Zambroni de Souza
Coorientador: Prof. Dr. Coorientador Ph.D Pedro Paulo Balestrassi

29 de janeiro de 2024
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Simulação de CLP em Navegadores Web: Uma Ferramenta para a Melhoria da Educação em Engenharia Elétrica.

Gabriel Vinícios Silva Maganha

Dissertação aprovada por banca examinadora em 19 de Dezembro de 2023, conferindo ao autor o título de **Mestre em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. Adriano Batista de Almeida
Prof. Dra. Claudia Eliane da Matta
Prof. Dr. Pedro Paulo Balestrassi
Prof. Dr. Antonio Carlos Zambroni de Souza

Itajubá
2023

Gabriel Vinícios Silva Maganha

Simulação de CLP em Navegadores Web: Uma Ferramenta para a Melhoria da Educação em Engenharia Elétrica/ Gabriel Vinícios Silva Maganha. – Itajubá, 29 de janeiro de 2024-

120 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Ph.D Antônio Carlos Zambroni de Souza

Dissertação (Mestrado)
Universidade Federal de Itajubá - UNIFEI
Programa de pós-graduação em engenharia elétrica, 29 de janeiro de 2024.

1. Tecnologias Web em Engenharia Elétrica. 2. Simulação Interativa. 3. Simulação de Sistemas de Controle. 4. Ferramentas Digitais para ensino. I. Antônio Carlos Zambroni de Souza. II. Universidade Federal de Itajubá. III. Mestrado em Engenharia Elétrica. IV. Título

CDU 07:181:009.3

Gabriel Vinícios Silva Maganha

Simulação de CLP em Navegadores Web: Uma Ferramenta para a Melhoria da Educação em Engenharia Elétrica

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Engenharia Elétrica.

Trabalho aprovado. Itajubá, 19 de Dezembro de 2023:

Prof. Dr. Ph.D Antônio Carlos Zambroni de Souza
Orientador

Prof. Dr. Coorientador Ph.D Pedro Paulo Balestrassi
Coorientador

Prof. Dr. Adriano Batista de Almeida

Prof. Dra. Claudia Eliane da Matta

Itajubá
29 de janeiro de 2024

Agradecimentos

Agradeço a Deus pela vida, pela família maravilhosa que me deu e por me ajudar a chegar até aqui com saúde e boas condições de vida.

Agradeço aos meus pais, Augusto Maganha e Marlene da Silva Maganha, por toda paciência e amor com que me criaram. Amo vocês!

Agradeço aos meus avós ainda vivos, vô Raimundo Minervino da Silva e vó Lindaura Acorsi Maganha, e aos já falecidos, vô Gildo Maganha e vó Raimunda Maria da Silva, por todo carinho com que sempre me trataram, pela paciência e por serem a base que construiu minha família.

Agradeço minha noiva, Fernanda Rabelo Gonçalves, que me inspira e me incentiva em todas as áreas da vida, todos os dias. Amo você minha linda!

Agradeço minha irmã, Maiara, a quem muito atormentei na infância e adolescência, mas que mesmo assim sempre me amou, sempre me tratou com todo o carinho e acreditou nos meus projetos.

Agradeço ao SENAI MG (Serviço Nacional de Aprendizagem Industrial), por ter me dado a oportunidade de trabalhar e lecionar lá, já há mais de 18 anos, o que me fez crescer tanto pessoalmente quanto profissionalmente.

Agradeço ao meu professor e orientador Antônio Carlos Zambroni de Souza, pela paciência, por todos os conselhos, análises, tempo gasto e carinho com que, desde o início do mestrado, sempre mostrou a mim e às minhas ideias.

Por fim, agradeço à UNIFEI pela oportunidade de realizar este Mestrado, que há muito tempo era um sonho para mim.

Pois onde estiver o teu tesouro, aí estará também o teu coração. - Mateus 6:21
Jesus Cristo

Resumo

Esta dissertação de mestrado explora o desenvolvimento e o impacto de um simulador de Controlador Lógico Programável (CLP) baseado na web, visando aprimorar a educação técnica e profissionalizante e promover a inclusão digital. Em um contexto onde a automação industrial está em rápida expansão, e os CLP desempenham um papel crucial no controle de máquinas e processos industriais, a complexidade da programação de CLP representa um obstáculo substancial, especialmente para iniciantes. O simulador de CLP desenvolvido, ao oferecer um ambiente virtual seguro e controlado para a programação e simulação, emerge como uma solução educacional inovadora, permitindo que os usuários experimentem e testem diversas estratégias de programação e controle. Sua principal vantagem reside no seu acesso baseado na web, possibilitando o uso em qualquer local com acesso à internet, o que confere flexibilidade aos estudantes e profissionais e elimina a necessidade de equipamentos físicos, reduzindo riscos de danos ou interrupções na produção. A relevância dos simuladores no processo de ensino e aprendizagem, reconhecida também na literatura acadêmica, é particularmente destacada neste estudo, dada a sua capacidade de proporcionar um ambiente seguro para a prática de habilidades complexas, crucial em campos de risco como engenharia elétrica e automação industrial. A abordagem da dissertação inclui desde a concepção até a implementação técnica do simulador, além de uma análise baseada em pesquisas com usuários de nível técnico, visando avaliar sua eficácia no processo de aprendizagem. Os resultados obtidos são importantes para o aperfeiçoamento contínuo da ferramenta e para a expansão de suas aplicações educacionais, enfatizando o potencial do simulador para aprimorar o ensino da programação de CLP e contribuir significativamente para a formação de profissionais mais qualificados e preparados para atender às exigências da indústria da automação.

Palavras-chaves: Tecnologias Web em Engenharia Elétrica. Simulação Interativa. Simulação de Sistemas de Controle. Ferramentas Digitais para ensino.

Abstract

This master's thesis explores the development and impact of a web-based Programmable Logic Controller (PLC) simulator, aiming to improve technical and professional education and promote digital inclusion. In a context where industrial automation is rapidly expanding, and PLCs play a crucial role in controlling industrial machines and processes, the complexity of PLC programming represents a substantial obstacle, especially for beginners. The developed PLC simulator, by offering a safe and controlled virtual environment for programming and simulation, emerges as an innovative educational solution, allowing users to experiment and test various programming and control strategies. Its main advantage lies in its web-based access, enabling use anywhere with internet access, which provides flexibility to students and professionals and eliminates the need for physical equipment, reducing the risk of damage or interruptions in production. The relevance of simulators in the teaching and learning process, also recognized in academic literature, is particularly highlighted in this study, given their ability to provide a safe environment for the practice of complex skills, crucial in risky fields such as electrical engineering and industrial automation. The dissertation approach includes everything from the conception to the technical implementation of the simulator, in addition to an analysis based on research with technical level users, aiming to evaluate its effectiveness in the learning process. The results obtained are important for the continuous improvement of the tool and for the expansion of its educational applications, emphasizing the simulator's potential to improve the teaching of PLC programming and contribute significantly to the training of more qualified professionals prepared to meet the demands of the automation industry.

Key-words: Web Technologies in Electrical Engineering. Interactive Simulation. Control Systems Simulation. Digital Tools for Teaching.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CLP	*Controlador Lógico Programável*	20
CPU	*Unidade Central de Processamento*	34
CSS	*Cascading Style Sheet*	74
HTML	*HyperText Markup Language*	74
IHM	*Interface Homem-Máquina*	34
OPC	*Open Platform Communications*	21
PLC	*Programmable Logic Controller*	31
PWM	*Pulse Width Modulation*	42
SCADA	*Supervisory Control And Data Acquisition*	74
SVG	*Scalable Vector Graphics*	30
TIC	*Tecnologias da Informação e Comunicação*	22
WebGL	*Web Graphics Library*	29

Sumário

1 Introdução

A automação industrial tem sido um campo de rápido crescimento e evolução, impulsionado pela necessidade de melhorar a eficiência, a produtividade e a qualidade dos processos de produção. No coração da automação industrial estão os CLP (*Controlador Lógico Programável*), dispositivos que controlam máquinas e processos industriais. A programação de CLP é uma habilidade essencial para engenheiros e técnicos no campo da automação industrial. No entanto, essa programação pode ser complexa e desafiadora, especialmente para aqueles que estão apenas começando a aprender sobre ela. Diante deste cenário, surge o simulador de CLP baseado em web como uma solução inovadora.

O simulador de CLP baseado em web é uma ferramenta que permite aos usuários programar e simular o funcionamento de um CLP em um ambiente virtual. Uma das principais vantagens é que o uso de um simulador permite que o aprendizado ocorra sem a necessidade de um laboratório ou de equipamentos caros. Além disso, no simulador os usuários podem aprender e praticar a programação de CLP sem o risco de danificar equipamentos caros ou causar interrupções na produção. Também permite aos usuários experimentar e testar diferentes estratégias de programação e controle em um ambiente seguro e controlado. Por fim, o simulador é uma ferramenta que pode ser usada em qualquer lugar, a qualquer hora, desde que haja uma conexão com a internet.

O simulador de CLP baseado em web descrito nesta dissertação é uma ferramenta interativa que permite a programação em linguagem Ladder, uma das linguagens de programação mais comumente usadas para CLP. A ferramenta oferece uma variedade de simulações interativas que os usuários podem escolher, incluindo simulações com botões e lâmpadas, controle de um tanque (controle de nível e temperatura), controle de semáforos em um cruzamento de trânsito, controle de cilindros (avanço e recuo de vários cilindros), e uma simulação que permite ao usuário montar o circuito de ligação do CLP livremente, ligando botões, sensores, lâmpadas, relés, etc.

1.1 Importância dos simuladores na educação

A importância dos simuladores no processo de ensino-aprendizagem é reconhecida na literatura acadêmica [22], como também será mais detalhado no decorrer deste trabalho. Simuladores oferecem uma oportunidade única para os alunos praticarem habilidades complexas em um ambiente seguro e controlado, o que é especialmente relevante em campos como a engenharia elétrica e de controle e automação, onde erros podem ter consequências perigosas ou indesejáveis [23].

1.2 O simulador desenvolvido neste trabalho

A presente dissertação de mestrado tem como objetivo principal apresentar e analisar o desenvolvimento de um simulador de CLP on-line, uma ferramenta de ensino que visa contribuir para a inclusão digital e o fortalecimento da educação técnica e profissionalizante.

O simulador desenvolvido neste trabalho opera de forma on-line, usando o próprio navegador do usuário,proporcionando um acesso mais fácil pelos usuários, independente da plataforma que esteja usando ou de qual sistema operacional o acesso está sendo feito. Pode ser acessado através do endereço <https://gvensino.com.br/sim/plc/>. Foi desenvolvido usando tecnologias web padrão, incluindo HTML, JavaScript e CSS. As simulações foram feitas no Canvas, usando uma biblioteca JavaScript chamada P5.js, mas também podem ser feitas usando SVG+JavaScript, Pixi.Js ou qualquer outro framework, inclusive 3D, como Babylon.js ou Three.js. Isso torna o simulador altamente flexível e adaptável, capaz de suportar uma ampla gama de simulações e cenários de programação.

A dissertação apresentará detalhes sobre o simulador de CLP desde sua concepção, passando pela implementação técnica e como ele pode ser utilizado em diferentes contextos educacionais. Além disso, serão exploradas as possibilidades de expansão e aprimoramentos futuros. Um exemplo seria implementar o acesso às simulações via protocolos de redes industriais Modbus ou OPC (*Open Platform Communications*) através de node.js, para que as simulações possam rodar em outros softwares de CLP também. Isso abriria ainda mais possibilidades para a aprendizagem e prática da programação de CLP, bem como para a simulação e teste de estratégias de controle, usando softwares realmente utilizados no contexto industrial.

1.3 Avaliando os impactos do simulador

Para avaliar o impacto do simulador desenvolvido, foram conduzidas pesquisas com alunos técnicos e não-técnicos, de modo a verificar a eficácia da ferramenta no processo de aprendizagem. Os resultados dessas pesquisas servirão como base para o aperfeiçoamento do simulador e sua aplicação em contextos educacionais diversificados.

Em resumo, o simulador de CLP baseado em web descrito nesta dissertação é uma ferramenta didática e flexível para a aprendizagem e prática da programação de CLP. Ele oferece uma variedade de simulações interativas que permitem aos usuários experimentar e testar diferentes estratégias de programação e controle. Espera-se que esta ferramenta seja de grande valor para estudantes, professores, engenheiros e técnicos no campo da automação industrial.

2 Tecnologias Digitais e Simuladores na Educação

Neste capítulo, será abordado o papel das Tecnologias Digitais da Informação e Comunicação (TDIC) na educação contemporânea, destacando especialmente a relevância dos simuladores como ferramentas pedagógicas. Exploraremos como estas tecnologias não apenas transformam os métodos de ensino e aprendizagem, mas também democratizam o acesso ao conhecimento, promovendo a inclusão digital e enriquecendo a experiência educacional. Além disso, será discutido como os simuladores fornecem uma maneira eficaz para uma compreensão mais profunda de conceitos complexos em diversas áreas do saber.

2.1 Tecnologias Digitais da Informação e Comunicação na Educação

A revolução tecnológica que o mundo presenciou no último século transformou a vida em sociedade de uma forma inédita na história. Carros, computadores, telefones móveis, aparelhos de televisão, internet e muitas outras inovações fazem com que nosso estilo de vida hoje seja muito diferente de duas ou três gerações atrás. Cada vez mais e mais rápido novos equipamentos surgem, com novas funcionalidades e recursos, fazendo com que os anteriores fiquem obsoletos em pouco tempo.

Com todo o avanço tecnológico das últimas décadas, a inclusão digital tem se tornado um tema cada vez mais relevante, e no contexto educacional não seria diferente. A crescente digitalização da sociedade [24] e o uso cada vez maior das TIC (*Tecnologias da Informação e Comunicação*) trouxe à tona a necessidade de preparar os alunos para um mundo cada vez mais digitalizado, tornando a inclusão digital uma prioridade para a educação [25].

2.1.1 Inclusão Digital

A inclusão digital é definida como o processo de garantir acesso e utilização eficaz das Tecnologias da Informação e Comunicação (TIC) para todas as pessoas. Este conceito vai além do simples acesso físico a dispositivos e internet, englobando a habilidade de usar efetivamente essas tecnologias para variados fins [25].

No Brasil, segundo a pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros – TIC Domicílios 2023, 95% dos domicílios tem um

telefone celular, 15% possuem um computador de mesa, 31% um notebook e 11% um tablet. Outros dados domiciliares de TIC no Brasil podem ser vistos na Tabela 1.

TIC	**percentual de domicílios**
Televisão	94%
Telefone celular	95%
Rádio	46%
Antena parabólica	19%
Telefone fixo	12%
TV por assinatura	24%
Computador de mesa	15%
Notebook	31%
Tablet	11%
Aparelho de videogame	13%

Tabela 1 – Percentual de equipamentos TIC nos domicílios brasileiros [19]

Além disso, a pesquisa também mostrou que 84% dos domicílios brasileiros já possuem acesso à internet no ano de 2023, com destaque para o grande crescimento das classes D/E, que de 2015 até 2023 saltaram de 16% para 67% [19].

Na área da educação também houve um crescimento no acesso à internet. Segundo a pesquisa TIC Educação de 2022, 94% das escolas de Ensino Fundamental e Médio possuíam acesso à internet — este número era de 82% em 2020 [26]. Esses dados mostram que a possibilidade de acesso à internet e às TIC digitais, como computadores e celulares, está aumentando gradualmente ano após ano no Brasil.

Entretanto, deve-se lembrar que a inclusão digital não se refere apenas ao acesso a dispositivos digitais e à internet, mas também à capacidade de usar efetivamente essas tecnologias para acessar, criar e compartilhar informações [27]. Dessa forma, a inclusão digital na educação envolve não apenas fornecer aos alunos acesso às tecnologias digitais, mas também ensiná-los a usar essas tecnologias de maneira eficaz e significativa.

Existem hoje diversos recursos digitais que podem exercer um papel relevante na educação, facilitando o processo de ensino-aprendizagem tanto do lado docente, quanto também do lado aluno, aumentando sua autonomia, engajamento e lhe fazendo adquirir competências de maneira mais rápida. Plataformas de aprendizado on-line, aplicativos educacionais, recursos de mídia audiovisuais, simuladores, ferramentas de colaboração on-line — estes são apenas alguns exemplos de ferramentas que, se bem empregadas, ajudam a melhorar a qualidade do ensino [28].

2.1.2 Simuladores

Simuladores são programas de computador que criam um ambiente virtual que imita o comportamento de um sistema físico real [29]. Podem ser utilizados para repro-

duzir situações que são perigosas, caras ou difíceis de replicar no ambiente em que se encontra. No contexto do ensino, os simuladores são usados para proporcionar aos estudantes experiências práticas de forma segura, sem o risco de danificar equipamentos ou machucar alguém [30].

Os simuladores podem variar amplamente em termos de complexidade e realismo, desde jogos de computador que simulam equipamentos e fenômenos físicos (como por exemplo o simulador de voo Microsoft Flight Simulator, exibido na Figura 1) até complexos sistemas de controle que replicam o comportamento de máquinas e processos industriais. Eles são usados em uma variedade de contextos educacionais, incluindo arte [31], engenharia, medicina, arquitetura, etc.

Figura 1 – Simulador de voo Microsoft Flight Simulator [1]

Um estudo realizado em 2020 por Chernikova et al. intitulado *"Simulation-Based Learning in Higher Education: A Meta-Analysis"* [30] concluiu que as simulações são uma das maneiras mais eficazes de facilitar a aprendizagem de habilidades complexas em vários domínios. Este estudo também destacou a importância do uso de tecnologia e estratégias de apoio à aprendizagem para melhorar a eficácia do aprendizado baseado em simulação.

Na área médica, por exemplo, bonecos e animais são utilizados há muitos séculos para simulação e treinamento de habilidades cirúrgicas [32]. Nas indústrias de aviação estima-se que houve uma redução de quase 50% nas taxas de acidentes aéreos relacionados à falha humana devido ao uso de simuladores [33].

Com o avanço da capacidade de processamento dos computadores tem se tornado possível a criação de simuladores cada vez mais realistas, tanto a nível visual, com simulações tridimensionais realistas, quanto a nível matemático, podendo representar com grande fidelidade sistemas complexos e dinâmicos.

Na área educacional o uso de simuladores também tem crescido e se mostrado cada vez mais importante na transmissão de conhecimentos, especialmente em áreas nas quais a resolução de problemas exige um raciocínio mais sistematizado e lógico [34].

Um dos sites mais conhecido de simuladores educacionais na internet é o site PhET, desenvolvido pela Universidade do Colorado em Boulder, que oferece uma vasta gama de

simulações interativas gratuitas centradas nas ciências e matemática. Uma de suas muitas simulações pode ser vista na Figura 2.

Os simuladores permitem que os estudantes possam aprender fazendo de forma parecida com que fariam numa prática real, com a vantagem de que podem testar livremente situações que na prática não seriam possíveis, fazer testes, comparativos rápidos, proporcionando uma experiência de aprendizado rica e envolvente, podendo analisar as consequências de suas decisões sem as consequências que eventuais erros teriam no mundo real.

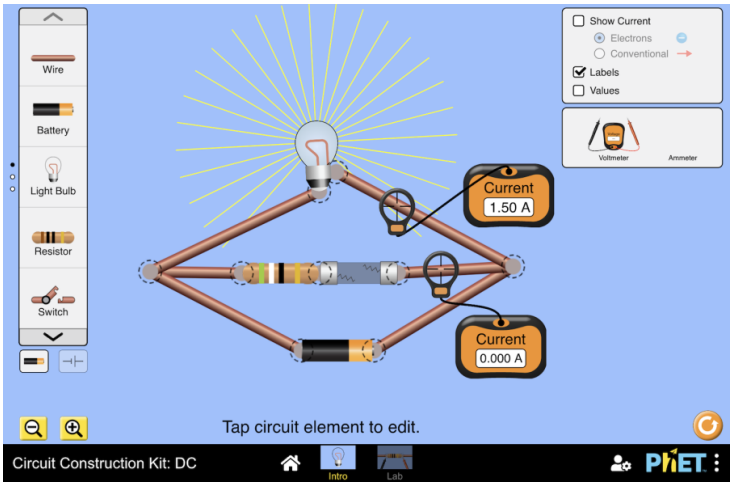

Figura 2 – Laboratório virtual de circuitos PHET [2] .

Existem diversos estudos [35] [30] que demonstram que o uso das tecnologias, em particular dos simuladores, mesmo quando usados de forma complementar, trazem contribuições significativas para o processo de ensino-aprendizagem, ajudando os alunos a desenvolver o pensamento crítico, a criatividade, a capacidade de resolução de problemas e até mesmo a motivação dos estudantes em aprender [36].

Além disso, simuladores que utilizam conceitos de gamificação, realidade aumentada e realidade virtual trazem ainda mais realismo e possibilidades nessa área.

2.1.2.1 O papel do docente no uso de simuladores

Apesar dos simuladores terem um impacto positivo na área educacional, é necessário que seu uso seja estratégico e contextualizado dentro das competências e conhecimentos que se deseja desenvolver nos estudantes. Nesse contexto, o papel do docente é crucial para garantir que o uso dos simuladores seja efetivo e alinhado com os objetivos pedagógicos.

Assim, o docente deve ter uma compreensão clara dos princípios fundamentais do assunto que quer ensinar através dos simuladores, para orientar os estudantes de forma eficaz durante as atividades de simulação. A habilidade do docente em integrar o uso do simulador com a teoria subjacente é vital para que os alunos não apenas aprendam a

operar o simulador, mas também compreendam os conceitos de automação e controle que estão sendo simulados.

Além disso, é importante que o docente utilize o simulador como uma ferramenta para promover o pensamento crítico e a resolução de problemas. Isso pode ser alcançado por meio do desenvolvimento de cenários de simulação que desafiam os alunos a aplicar o conhecimento teórico em situações práticas, incentivando a experimentação e a exploração de diferentes estratégias de programação e controle.

O docente também deve desempenhar um papel ativo no fornecimento de feedback aos alunos. Este feedback deve ser construtivo e orientado para o aprendizado, ajudando os estudantes a entenderem seus erros e a aprenderem com eles. Isso é particularmente importante em simulações onde os erros podem ter consequências significativas, como na programação de CLP em ambientes industriais.

Por fim, é fundamental que o docente esteja preparado para adaptar o uso do simulador às necessidades individuais dos alunos. Cada estudante terá um ritmo diferente de aprendizado e diferentes níveis de experiência prévia, o que requer uma abordagem flexível e personalizada. Isso pode incluir a adaptação das atividades de simulação para diferentes níveis de dificuldade ou o fornecimento de recursos adicionais para alunos que precisam de suporte extra.

Portanto, o papel do docente no uso de simuladores é multifacetado e essencial para maximizar o potencial educacional dessas ferramentas. Ao integrar os simuladores de maneira estratégica e contextualizada no currículo, os docentes podem aprimorar significativamente o processo de ensino-aprendizagem, especialmente em campos técnicos e científicos como a automação industrial.

2.1.2.2 Considerações sobre os simuladores educativos

Os simuladores educativos oferecem uma metodologia interativa e imersiva para o ensino, proporcionando uma experiência de aprendizado que vai além dos métodos tradicionais. No entanto, para maximizar sua eficácia, é crucial considerar vários aspectos-chave no design e implementação desses simuladores [37].

Integração com Métodos de Ensino Tradicionais: Simulações não devem ser vistas como substitutas, mas como complementos aos métodos de ensino tradicionais. A combinação de simulações com aulas expositivas, discussões em grupo e atividades práticas pode proporcionar uma experiência de aprendizado mais rica e diversificada. Essa abordagem integrada permite que os alunos apliquem teorias e conceitos em um ambiente controlado e prático.

Objetivos Claros e Definidos: Para serem eficazes, as simulações devem ter objetivos de aprendizado claros e bem definidos. Os alunos devem entender o que se espera

deles e quais competências e conhecimentos são visados com a atividade de simulação. Estabelecer objetivos claros ajuda os alunos a manter o foco e a direção durante a simulação, garantindo que o tempo gasto seja produtivo e alinhado com os objetivos educacionais.

Progressão da Complexidade: As simulações devem ser estruturadas de modo a começar com tarefas simples, aumentando gradualmente em complexidade. Esta progressão ajuda a construir confiança e competência nos alunos, permitindo que eles desenvolvam suas habilidades em um ritmo adequado. A complexidade crescente também ajuda a manter o desafio e o interesse dos alunos ao longo do tempo.

Engajamento dos Alunos: As simulações devem ser projetadas para serem envolventes e imersivas. Um simulador que captura o interesse dos alunos aumenta a motivação e o envolvimento, o que é crucial para uma aprendizagem efetiva. Aspectos como realismo, interatividade e relevância para a vida real dos alunos são fundamentais para criar um ambiente de simulação envolvente.

Feedback Contínuo e Construtivo: Para facilitar a aprendizagem, as simulações devem fornecer feedback regular e construtivo. Este feedback ajuda os alunos a entenderem onde estão acertando e onde precisam melhorar. Um feedback eficaz deve ser oportuno, específico e orientado para o crescimento e desenvolvimento dos alunos.

Ferramentas Explanatórias: As simulações devem incluir ferramentas explanatórias que ajudem os alunos a compreender os conceitos subjacentes. Estas podem incluir dicas, tutoriais, ou seções de teoria que explicam os princípios em jogo. Ferramentas explanatórias são essenciais para garantir que os alunos não apenas saibam como executar uma tarefa, mas também entendam o porquê e o contexto maior por trás dela.

Ao considerar esses elementos na criação e implementação de simuladores educativos, os educadores podem maximizar o potencial dessas ferramentas para enriquecer o processo de ensino-aprendizagem, tornando-o mais eficaz, envolvente e alinhado com os objetivos pedagógicos.

2.1.3 Tecnologias para construção de simuladores

O desenvolvimento de simuladores educativos pode ser feito através de ferramentas de programação de computadores. Existem empresas especializadas em criação de simuladores e jogos, mas com conhecimentos de programação é possível que uma pessoa sozinha ou um pequeno grupo possam desenvolver boas simulações.

Algumas das ferramentas de programação que são mais empregadas para construção de simuladores educativos podem ser vistas na tabela 2:

Unity: O Unity é uma plataforma de desenvolvimento de jogos que pode ser usada para criar simulações 2D e 3D, capazes de serem executadas tanto em computadores

	2D	3D	Web	Realidade Virtual / Aumentada
Unity	X	X	X	X
Unreal Engine	X	X	X	X
Godot Engine	X	X	X	X
Three.js / Babylon.js		X	X	X
Pixi.js	X		X	
P5.js / Canvas2D	X		X	
SVG + JavaScript	X		X	

Tabela 2 – Tabela com algumas das principais ferramentas de desenvolvimento de simuladores [20], [21].

quanto em smartphones, em ambiente local ou web. Tem complexidade de aprendizado de média para alta, já que requer conhecimentos não apenas de programação, mas também de modelagem 3D. O Unity é conhecido por sua flexibilidade, podendo ser usado gratuitamente para fins educacionais e não comerciais [38]. Um exemplo de aplicação no Unity é o simulador 3D desenvolvido pelo SENAI de uma planta de instrumentação, que pode ser vista na Figura 3.

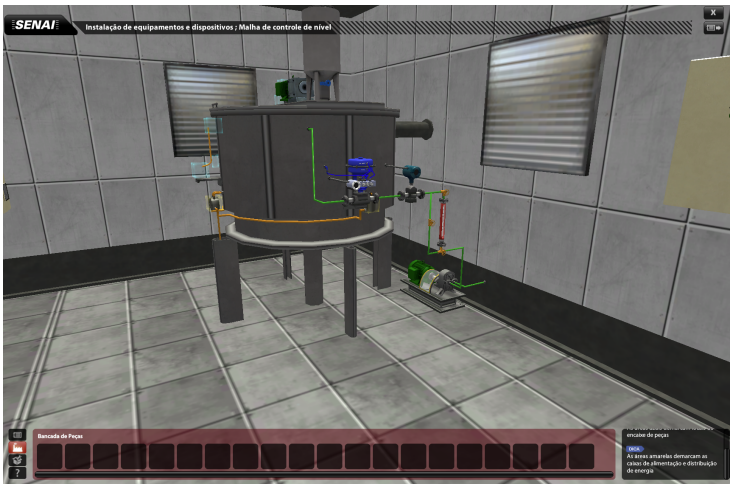

Figura 3 – Uma das simulações 3D desenvolvida em Unity usadas nos cursos técnicos EaD do SENAI. Fonte: o autor.

Unreal Engine: É outra plataforma de desenvolvimento de jogos que pode ser usada para criar simulações interativas 2D e 3D. Unreal Engine é conhecida por sua alta qualidade gráfica e poder de processamento. No entanto, como Unity, Unreal Engine é uma ferramenta mais complexa que requer mais tempo para aprender e usar. Além disso, embora Unreal Engine possa ser usado gratuitamente para fins educacionais e não comerciais, ele é uma ferramenta paga para uso comercial [39].

O Godot Engine é uma ferramenta gratuita e de código aberto para o desenvolvimento de jogos 2D e 3D, que também pode ser utilizada para a criação de simuladores educacionais. Ele possui um editor de cena visual poderoso e um editor de animação, além de um ambiente de desenvolvimento integrado. Funciona nos principais sistemas operaci-

onais (Windows, MacOS e Linux) e é conhecida por ser uma ferramenta leve em relação ao Unity e ao Unreal Engine [40].

Babylon.js e Three.js: São bibliotecas JavaScript (a linguagem de programação que roda nos navegadores de internet) para a criação de gráficos 3D na web. Ambas as bibliotecas são poderosas e flexíveis, permitindo a criação de uma ampla gama de simulações 3D. Babylon.js e Three.js são gratuitas e de código aberto. No entanto, a criação de gráficos 3D pode ser mais complexa e exigir mais recursos computacionais do que a criação de gráficos 2D [41].

Pixi.js: É uma biblioteca JavaScript para criação de gráficos 2D na web. Pixi.js é conhecida por sua velocidade e flexibilidade, tornando-a adequada para a criação de jogos e outras aplicações interativas como simuladores. Utiliza aceleração por hardware via WebGL (*Web Graphics Library*), o que faz ter desempenho superior a outras soluções, especialmente quando trabalha com centenas ou milhares de elementos desenhados na tela simultaneamente. O Pixi.js é gratuito e de código aberto e é a base para outras plataformas de desenvolvimentos de jogos 2D na Web, como o Phaser e o GDevelop [42].

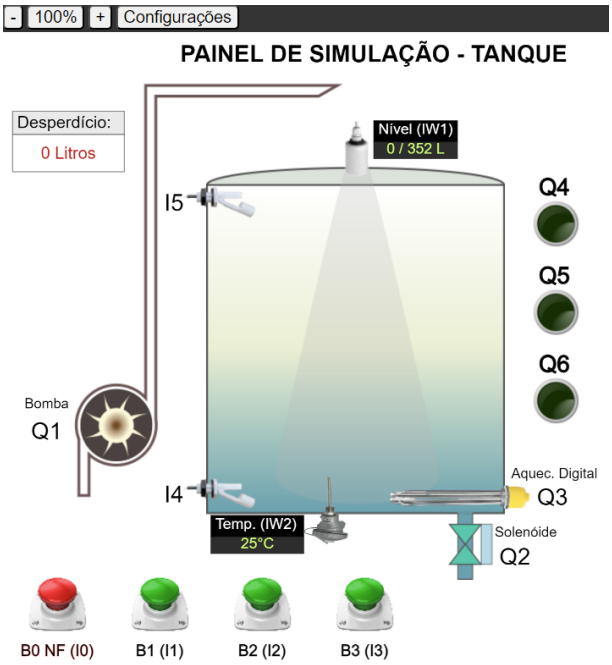

Figura 4 – Simulação de controle de nível e temperatura disponível no simulador de CLP desenvolvido neste trabalho, feito em P5.js. Fonte: o autor.

P5.js: É uma biblioteca JavaScript que simplifica a programação para artistas, designers, educadores e iniciantes. Utiliza o elemento Canvas, que é o elemento para desenhar gráficos na Web. O P5.js possui uma abordagem amigável para desenhar gráficos no Canvas, o que torna fácil para iniciantes começarem a programar. Apesar disso, o P5.js também tem a capacidade de lidar com projetos complexos e avançados. É uma ferramenta gratuita e de código aberto. No entanto, perde em desempenho em comparação com o Pixi.js, já que não utiliza a aceleração por hardware do WebGL [43]. O p5.js foi a

ferramenta escolhida para desenvolver a maior parte das simulações presentes no simulador de CLP que este trabalho, como na Figura 4 e também como será apresentado no decorrer deste trabalho.

SVG + JavaScript: SVG (*Scalable Vector Graphics*) é uma linguagem para criação de gráficos bidimensionais. O SVG é frequentemente usado em conjunto com JavaScript para criar gráficos interativos, animações e simulações, como por exemplo, o simulador de circuitos eletrônicos on-line Wokwi [44]. No entanto, SVG + JavaScript pode ser mais complexo para aprender e usar do que outras ferramentas, como P5.js.

3 Fundamentos do Controlador Lógico Programável

Este capítulo é dedicado a explorar os fundamentos do Controlador Lógico Programável (CLP). Iniciaremos com uma visão geral sobre o CLP, mostrando sua definição, história e aplicações. Em seguida, será mostrada a arquitetura interna dos CLPs, seus modos de operação e as características dos diferentes tipos existentes no mercado. O objetivo é fornecer uma base sólida de conhecimento que não apenas contextualize sua importância prática e teórica, mas também estabeleça uma fundação para compreender a relevância e o funcionamento dos simuladores de CLP discutidos anteriormente na dissertação. Por fim, será mostrado o funcionamento da linguagem de programação de CLPs Ladder.

3.1 Definição e História

O CLP, também conhecido como PLC (*Programmable Logic Controller*) em inglês, é um dispositivo eletrônico digital, uma espécie de computador especial feito para controlar vários tipos de máquinas ou processos, especialmente na indústria [45].

Como o foco do CLP é trabalhar na automação das indústrias, deve ser capaz de trabalhar em ambientes severos, com vibrações, variações de temperatura, poeira, distúrbios elétricos e outros problemas típicos de um ambiente hostil à equipamentos eletrônicos como é a indústria [46].

O CLP surgiu na indústria automobilística, na década de 1960, como uma alternativa aos sistemas de controle baseados em relés, que eram complexos e difíceis de alterar ou reconfigurar. A General Motors, uma das maiores fabricantes de automóveis do mundo, foi a primeira a identificar a necessidade de um dispositivo de controle mais flexível e reprogramável, que pudesse ser facilmente adaptado para controlar uma variedade de máquinas e processos em suas linhas de montagem [47].

O primeiro CLP foi o Modicon 084, que pode ser visto na Figura 5. Ele possuía entradas e saídas digitais e podia ser facilmente programado com uma linguagem de programação que lembrava muito a lógica a relés e que posteriormente seria conhecida como Linguagem Ladder.

Figura 5 – Primeiro modelo de CLP do mundo, o Modicon 084 [3].

3.2 Importância e Aplicações

Desde que surgiu, o uso de CLP no controle de processos industriais cresceu e adquiriu grande importância. Hoje é um equipamento amplamente utilizado em muitos setores industriais, incluindo manufatura, petroquímica, alimentos e bebidas, energia, água e tratamento de águas residuais, entre outros. O CLP é a espinha dorsal de muitos sistemas de controle industrial, permitindo o controle preciso e confiável de máquinas e processos, melhorando a eficiência, a segurança e a produtividade [48]. Um exemplo de CLP moderno pode ser visto na Figura 6.

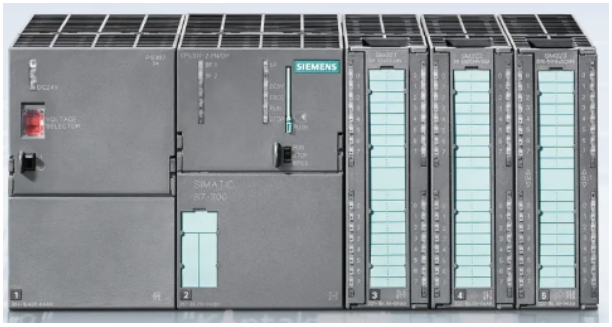

Figura 6 – CLP Siemens Simatic S7-300 [4].

Uma das principais vantagens do CLP é sua flexibilidade. Ele pode ser programado para controlar uma ampla gama de dispositivos e processos, desde simples máquinas até complexos sistemas de controle de processos. Além disso, o CLP pode ser facilmente reprogramado para alterar ou adaptar sua funcionalidade, tornando-o ideal para aplicações onde as necessidades de controle podem mudar ao longo do tempo [48].

Outra vantagem importante do CLP é sua robustez. Ele é projetado para operar

em ambientes industriais difíceis, onde pode ser exposto a condições adversas, como altas temperaturas, umidade, poeira, vibração e interferência eletromagnética. Além disso, alguns CLP possuem recursos de redundância e tolerância a falhas, para garantir a continuidade da operação em caso de falha de um componente [46].

No entanto, apesar de suas muitas vantagens, o CLP também tem algumas desvantagens. Por exemplo, a programação de um CLP pode ser complexa e requer um alto nível de conhecimento técnico. Além disso, o custo inicial de um CLP pode ser alto, embora este custo pode ser compensado ao longo do tempo pelos benefícios de eficiência, robustez e produtividade que ele pode proporcionar [49].

O fato dos CLP serem equipamentos caros é mais um motivo que favorece o uso de simuladores como uma estratégia pedagógica de ensino, uma vez que é possível aprender a usá-lo sem sequer ter um à disposição.

3.3 Arquitetura interna de um CLP

A arquitetura interna de um CLP pode ser dividida em cinco partes, conforme pode ser visto na Figura 7: 1. Fonte de Alimentação; 2. Unidade Central de Processamento (CPU); 3. Entradas (analógicas/digitais); 4. Saídas (analógicas/digitais); 5. Unidade de Comunicação [48].

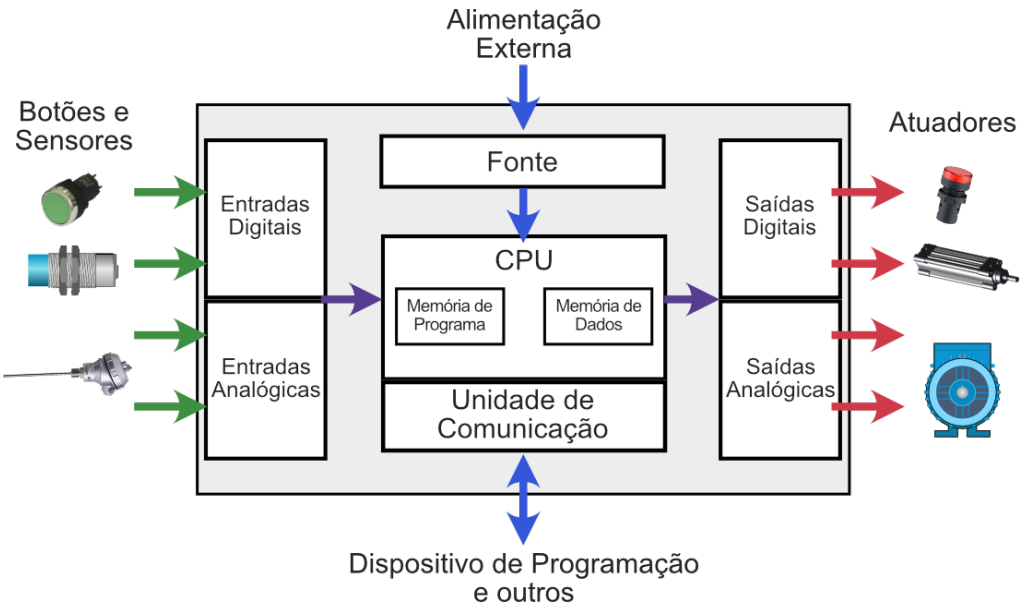

Figura 7 – Arquitetura básica de um CLP. Fonte: O autor.

Fonte de Alimentação: responsável por alimentar o CLP e todos os seus circuitos internos. A maioria dos CLP é alimentada ou com 24Vcc ou com 220Vac. A fonte também protege os circuitos internos do CLP contra ruídos advindos da rede elétrica e outras interferências.

CPU: a CPU (*Unidade Central de Processamento*) é o cérebro do CLP. Ela é quem processa todas as informações, na seguinte sequência: faz a leitura das entradas, faz o processamento da programação e então define os estados das saídas. Divide-se também a memória do CLP em duas partes: a memória de programa, que armazena a programação desenvolvida pelo usuário, e a memória de dados, que é utilizada pela CPU para armazenamento temporário de dados, como operações matemáticas e lógicas.

Entradas: também chamadas de *Inputs* (entradas em inglês), as entradas é o meio por onde o CLP recebe informações de fora. São nas entradas que são ligados botões e sensores. As entradas podem ser digitais, que são aquelas que recebem apenas dois níveis de sinal (ligado ou desligado) ou analógicas, que são aquelas que podem ler a intensidade do sensor.

Saídas: também chamadas de *Outputs* (saídas em inglês), as saídas é por onde o CLP envia informações para fora, acionando ou desacionando outras cargas, também chamadas de atuadores. Os atuadores atuam no ambiente, podendo produzir luz (lâmpadas), movimento (motores, cilindros), calor/frio (resistências), som (campainhas e buzzers), etc. As saídas também podem ser digitais ou analógicas: as digitais são para aqueles atuadores que apenas ligam ou desligam (lâmpadas, motores elétricos, e outros); já as analógicas são para os atuadores que possuem controle de intensidade (controle de velocidade de motores ou posicionamento de válvulas proporcionais, por exemplo).

Unidade de comunicação: é uma interface que permite ao CLP se comunicar com outros dispositivos, tais como um microcomputador para sua programação, outros CLP (para que trabalhem em rede), uma IHM (*Interface Homem-Máquina*), etc.

As entradas e saídas de um CLP são comumente chamadas de I/O (*inputs* e *outputs*), o que em português seria equivalente a E/S.

3.4 Modos de operação e Scan de um CLP

Um CLP normalmente possui dois modos de operação, chamados de Programação (ou PROG) e Execução (ou RUN). Esses modos podem ser acessados e alterados normalmente via algum botão/chave física, presente no corpo do CLP, mas também pode ser alterado diretamente via software.

No modo de execução (RUN) o CLP executa a programação que foi feita pelo usuário. O ciclo de varredura (também chamado de scan) de um CLP, que define quais tarefas são executadas pela CPU durante o modo RUN é composto por 3 etapas: 1. Leitura das Entradas; 2. Execução do Programa; 3. Atualização das saídas [48].

A Figura 8 mostra essas etapas em forma de fluxograma.

Já no modo de programação (PROG) o CLP não executa nenhuma lógica, ficando

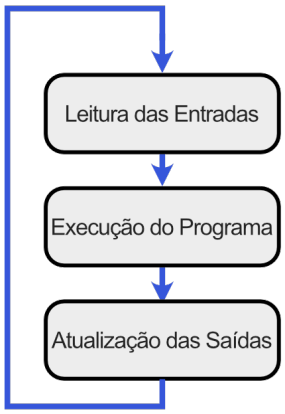

Figura 8 – Ciclo de varredura (scan) de um CLP. Fonte: O autor.

apenas aguardando o envio de uma nova programação ou a modificação do programa existente através da interface de comunicação.

O processo de envio do programa para o CLP é chamado de download. Já o processo contrário, de trazer de volta o programa que está no CLP para o computador é chamado de upload [50].

3.5 Tipos de CLP

Pode-se classificar os CLP em dois tipos gerais: CLP compactos e CLP modulares.

3.5.1 CLP compactos

Os CLP compactos possuem todas as suas partes básicas (fonte, CPU, entradas e saídas) em um único corpo. Um exemplo de CLP compacto é a linha Micrologix, da Allen Bradley, que pode ser visto na Figura 9.

Figura 9 – CLP compacto Allen Bradley Micrologix [5].

3.5.2 CLP modulares

Os CLP modulares possuem uma estrutura modular, com cada módulo separado executando uma função distinta. Nos CLP modulares a CPU é um módulo separado e

destacável do resto do CLP, bem como os módulos contendo as entradas e saídas. As partes podem ser substituídas, trocadas ou expandidas, conforme a necessidade do projeto. Um exemplo de CLP modular é a linha SLC500, da Allen Bradley, conforme pode ser visto na Figura 10.

Figura 10 – CLP modular Allen Bradley SLC 500 [6].

Nos CLP modulares a estrutura onde os módulos são encaixados é chamada de rack. Um exemplo de rack pode ser visto na Figura 11.

Figura 11 – CLP modular Allen Bradley, com rack à mostra [7].

3.6 I/O de um CLP

As Entradas e Saídas de um CLP podem ser de diversos tipos.

3.6.1 Entradas Digitais ou Discretas

As entradas digitais (ou discretas) são aquelas que conseguem ler apenas dois estados: se a entrada está energizada (estado ativado ou 1) ou se a entrada não está energizada (estado desativado ou 0). Nos CLP mais modernos, por razões de segurança, as entradas digitais tipicamente trabalham com 24V.

Nas entradas digitais são conectados botoeiras (ou botão está pressionado ou está solto), chaves ou sensores digitais, como sensores capacitivos, indutivos, de presença, etc.

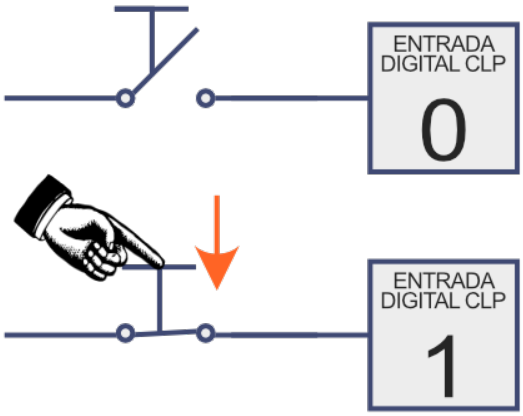

Figura 12 – Botão ligado a uma entrada digital. Fonte: O autor.

A Figura 12 mostra um exemplo de entrada digital sendo acionada por um botão.

Já a Figura 13 mostra uma entrada digital fazendo a leitura de um sensor (capacitivo ou indutivo).

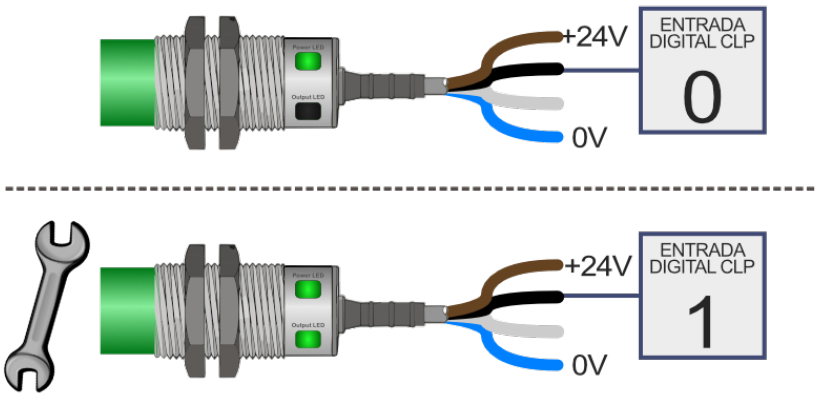

Figura 13 – Sensor Capacitivo/Indutivo ligado a uma entrada digital, ora detectando um objeto, ora não. Fonte: O autor.

As entradas digitais de um CLP podem ser do tipo *Sink* (dreno, também chamadas de NPN ou lógica positiva) ou do tipo *Source* (fonte, também chamadas de PNP ou lógica negativa) [51].

As entradas digitais do tipo *Sink* são acionadas com sinal positivo, logo, o comum das entradas deve ser ligado ao potencial negativo. A Figura 14 mostra como é a ligação de um botão a uma entrada *sink*.

Já as entradas digitais do tipo *Source* são acionadas com sinal negativo, logo, o comum das entradas deve ser ligado ao potencial positivo. A Figura 15 mostra como é a ligação de um botão a uma entrada *source*.

Os Sensores do tipo NPN, como emitem sinal negativo, devem ser ligados a entradas do tipo *Source*, conforme mostra a Figura 16. Já sensores do tipo PNP, como emitem sinais positivos, devem ser ligados a entradas do tipo *Sink*, como mostra a Figura 17.

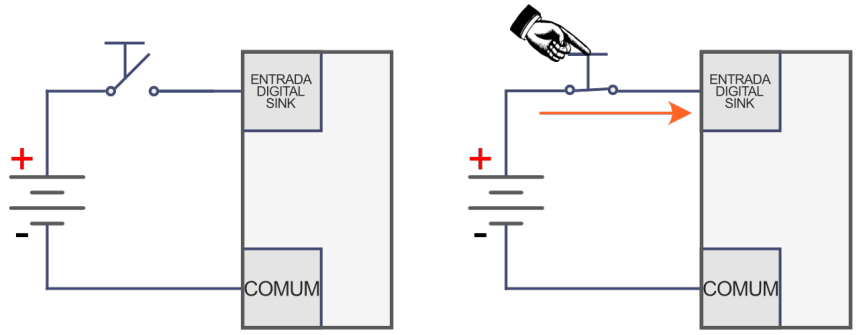

Figura 14 – Entradas digitais do tipo *Sink*. Fonte: O autor.

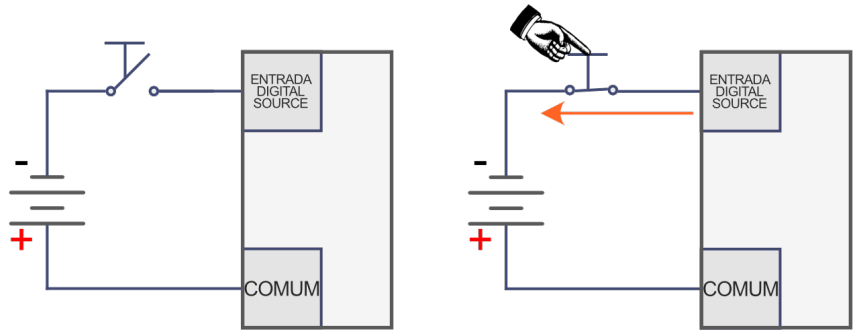

Figura 15 – Entradas digitais do tipo *Source*. Fonte: O autor.

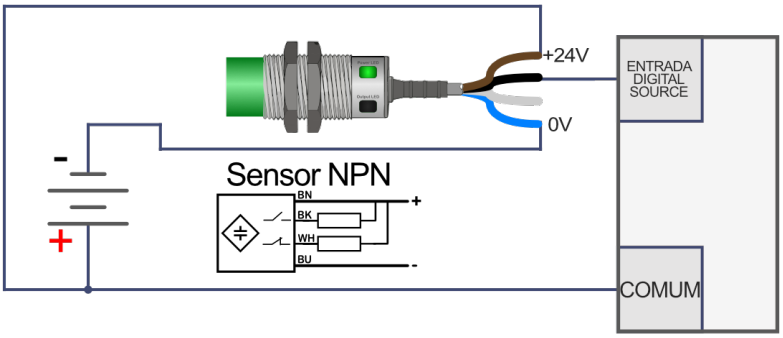

Figura 16 – Sensores NPN devem ser ligados a entradas *Source*. Fonte: O autor.

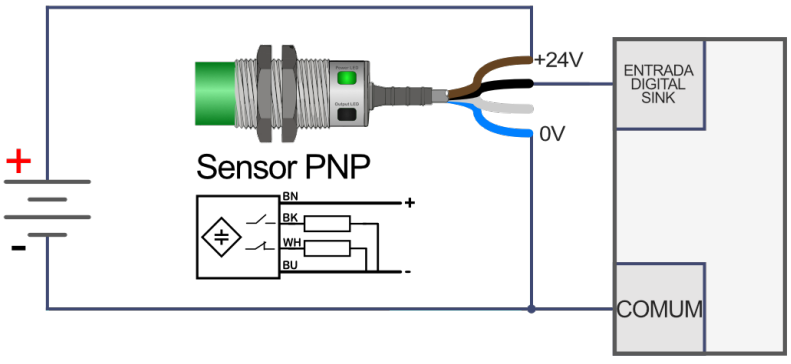

Figura 17 – Sensores PNP devem ser ligados a entradas *Sink*. Fonte: O autor.

Caso o tipo de sensor e o tipo de entrada sejam incompatíveis entre si (Sensor NPN e CLP com entradas *Sink* ou Sensor PNP e CLP com entradas *Source*) o CLP não

irá conseguir fazer a leitura do sinal do sensor. Nesse caso, um circuito elétrico auxiliar deverá ser usado para adequar o sinal que o sensor envia para o CLP. Uma das formas mais comuns de se fazer essa adaptação de sinais é um circuito com um relé, conforme as Figuras 18 e 19 demonstram.

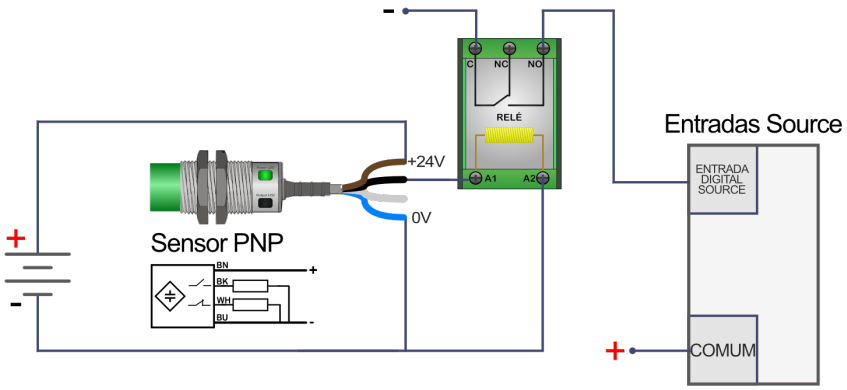

Figura 18 – Sensor PNP adaptado a entrada *Source* via Relé. Fonte: O autor.

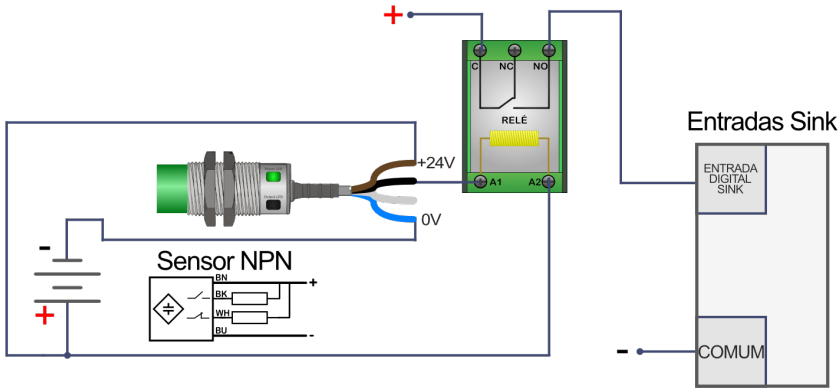

Figura 19 – Sensor NPN adaptado a entrada *Sink* via Relé. Fonte: O autor.

Alguns CLP, como o CLP compacto Siemens S7-200, possuem entradas híbridas: elas funcionam se ligadas tanto no modo *Sink* como no modo *Source*.

3.6.2 Entradas Analógicas

As entradas analógicas (também chamadas de contínuas) permitem que o CLP faça a leitura da intensidade do sinal que chega nelas. As grandezas analógicas que podem ser lidas varia de CLP para CLP, porém as mais comuns são por tensão (0-10V) e por corrente (4-20mA).

Na Figura 20 pode-se ver um sensor ultrassônico fazendo a leitura do nível de um tanque e enviando esta leitura para a entrada analógica do CLP.

Outros exemplos de sensores que emitem sinais analógicos são sensores de pressão, densidade, vazão, temperatura, etc.

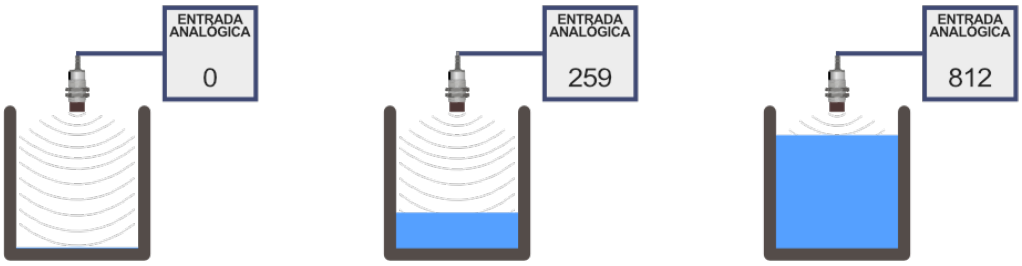

Figura 20 – Sensor Ultrasônico ligado a uma entrada analógica medindo o nível do tanque. Fonte: O autor.

Um aspecto importante das entradas analógicas é a sua resolução, que é medida em bits. Quanto mais bits uma entrada analógica tiver, mais sensível será a sua leitura — e também mais suscetível aos ruídos elétricos.

Se uma entrada analógica 0-10V tiver 10 bits, por exemplo, o CLP irá dividir essa faixa de tensão em 1024 valores distintos (2^{10}), o que vai permitir uma sensibilidade de 9,77mV (10/1024). Esse valor será representado dentro do programa como um número inteiro de 0 a 1023 (pois como são 1024 valores e o 0 é o primeiro valor, o último será o 1023).

3.6.3 Saídas Digitais

As saídas são os elementos por onde o CLP consegue acionar atuadores, tais como lâmpadas, motores, válvulas, aquecedores, sirenes, etc.

As saídas digitais podem apenas ligar ou desligar esses atuadores, pois possuem apenas dois estados possíveis. Não é possível fazer controle de intensidade nessas saídas. Na Figura 21, pode-se ver a saída digital de um CLP ligando e desligando uma lâmpada.

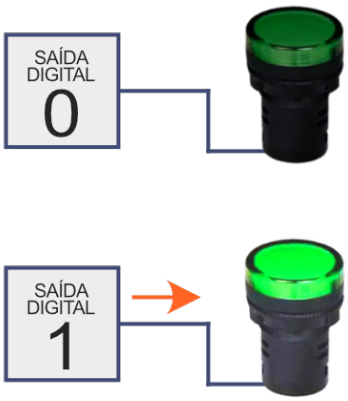

Figura 21 – Saída Digital de um CLP, ligando e desligando uma lâmpada. Fonte: O autor.

Existem três tipos de saídas digitais nos CLP:

Saída digital a Relé: Nesse tipo de saída, cada saída do CLP possui um relé próprio que é acionado toda vez que essa saída é ligada. Ao ligar, o sinal que está no

pino COMUM (COM) é chaveado por um contato seco direto para a saída que foi ligada, conforme pode-se ver na Figura 22. Como o chaveamento é feito direto por um contato seco, esse tipo de saída consegue acionar cargas alimentadas tanto por corrente contínua quanto por corrente alternada. Entretanto, como desvantagem, essa saída é mais lenta e pode se desgastar ao longo do tempo, já que o relé possuí partes mecânicas móveis.

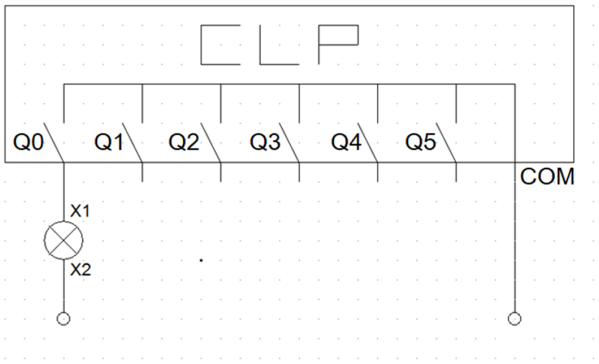

Figura 22 – Saída Digital a Relé. Fonte: O autor.

Saída digital a Transistor: Na saída digital a transistor, cada saída possui um transistor (que pode ser bipolar ou de efeito de campo), permitindo comutações das saídas em alta velocidade e com alta vida útil, já que transistores não possuem partes móveis. Entretanto, por se tratar de transistores, essas saídas só podem alimentar cargas de Corrente Contínua e possuem uma capacidade de corrente geralmente inferior ao das saídas a relé.

No caso dos CLP com saídas digitais a transistor existem dois tipos: as saídas do tipo *Sink* e as saídas do tipo *Source*, conforme pode-se ver na Figura 23.

As saídas do tipo *Sink* (também chamadas de tipo N ou NPN) são aquelas que, quando acionadas, levam o potencial negativo até a carga.

Já as saídas do tipo *Source* (também chamadas de tipo P ou PNP) são aquelas que, quando acionadas, levam o potencial positivo até a carga.

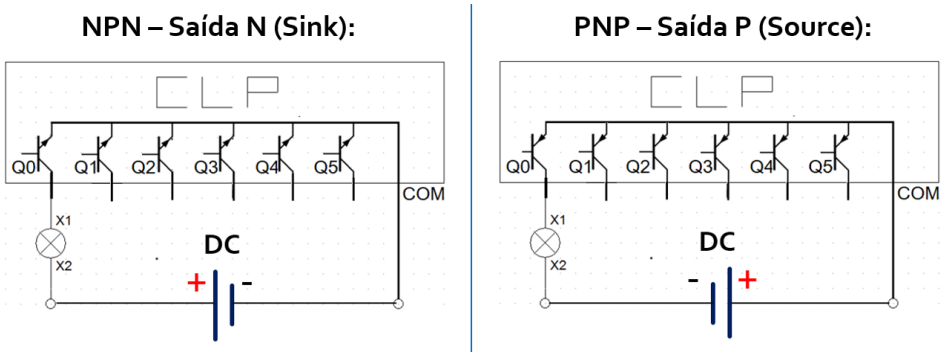

Figura 23 – 2 Tipos de Saídas Digitais a Transistor. Fonte: O autor.

Saída digital a Triac:

Na saída digital a Triac, cada saída é equipada com um Triac, o que permite o acionamento apenas de cargas de corrente alternada (AC). As saídas a Triac superam as saídas a relé devido à alta velocidade de comutação e à longa vida útil proporcionada pelos Triacs, já que assim como os transistores, são componentes de estado sólido. Um exemplo de circuito interno com a saída a Triac de CLP pode ser visto na Figura 24.

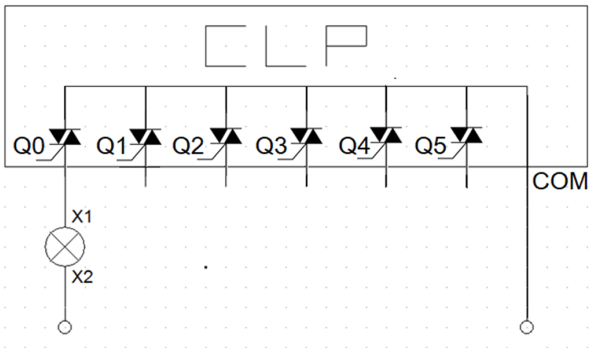

Figura 24 – Saída Digital a Triac. Fonte: O autor.

3.6.4 Saídas Analógicas

Ao contrário das saídas digitais, as saídas analógicas possuem muito mais do que apenas 2 estados. Elas podem trabalhar por tensão, emitindo qualquer tensão dentro de uma faixa específica (0-10V, por exemplo) ou por corrente (4mA a 20mA, por exemplo). Dessa forma, esse tipo de saída pode ser capaz não apenas de ligar ou desligar um determinado atuador, mas também de controlar a intensidade com que é acionado.

As saídas analógicas normalmente são utilizadas para controlar velocidades de motores (através de inversores de frequência), a posição de abertura de válvulas proporcionais (podendo controlar se a válvula estará 0% aberta, 10%, 45%, etc.), a intensidade de acionamento de resistências elétricas, dentre outros.

3.6.5 Outros tipos de entradas e saídas especiais

Entradas de contagem rápida: são uma entradas projetadas para lidar com sinais de alta frequência, como os gerados por *encoders* ou outros dispositivos de medição que operam em altas taxas de pulso [52].

Saída de alta velocidade (saída PWM): são saídas que conseguem ligar e desligar em um período de tempo muito menor que as saídas convencionais. São utilizadas para gerar sinais PWM (*Pulse Width Modulation*) (Modulação por largura de pulso), própria para controle de velocidade e posicionamento de motores [53].

Módulos de comunicação: são interfaces que permitem ao CLP se conectar com outros equipamentos de rede, como sensores inteligentes, inversores de frequência, sistemas supervisórios, etc.

Módulos de entradas especiais: permitem a conexão direta ao CLP de sensores especiais, tais como sensores de temperatura termopar e células de carga. Esses sensores trabalham com sinais padronizados, mas que são pequenos demais para serem utilizados diretamente nas entradas analógicas convencionais, já que possuem poucos mili-volts de magnitude. Esses módulos são específicos para que o CLP possa ler diretamente os sinais desses tipos de sensoress, sem a necessidade da intermediação de transmissores ou condicionadores de sinais [52].

3.7 Histórico do CLP e suas linguagens de programação

Até os anos 1960, a automação industrial era composta especialmente por grandes painéis controlados por lógica a relés. Dependendo das lógicas que estavam realizando, esses painéis tinham centenas de relés, milhares de fios e blocos de terminais. O grande problema na lógica estar nos circuitos a relé é que qualquer alteração demanda um enorme esforço de modificação física do circuito do painel. Além disso, os relés são lentos e por terem partes móveis internas, apresentam desgaste mecânico e tem uma vida útil reduzida.

Em 1964, Richard E. "Dick"Morley e George Schwenk fundaram a Bedford Associates em Bedford, Massachusetts, uma empresa especializada em engenharia de sistemas de controle. Na época, grandes empresas industriais, especialmente na indústria automotiva, tinham suas automações fabris construídas com painéis a relés, que apesar de funcionarem bem, traziam consigo grandes desafios, seja na construção desses painéis, na manutenção ou na hora de realizar alterações na lógica do funcionamento, já que todo o funcionamento estava baseado em ligações elétricas, devido a enorme quantidade de relés.

Em 1968, Bill Stone, membro de uma equipe de engenheiros da General Motors Corporation, fez uma apresentação na Conferência Westinghouse [54] em conjunto com outros engenheiros da GM, onde demonstrou os diversos problemas com a automação a relés até então existentes e estabeleceu critérios de projeto para criar um novo "Controlador de Máquina Padrão". De acordo com os critérios formulados, a primeira versão desta máquina tinha como foco acabar com o desperdício financeiro associado ao descarte de relés durante as alterações de linha de montagem e também:

- Ampliar o uso de circuitos de estado sólido (que não tem partes móveis) para 90% das máquinas presentes na unidade (eliminando o uso de relés no processamento da lógica).
- Minimizar os períodos de parada das máquinas associados a falhas de controle, sendo de fácil manutenção e programação.
- Programação em conformidade com a lógica de relé (diagrama de contatos) previamente existente.

- Que seja modular, facilitando ampliações futuras, permitindo a substituição simples de componentes e adição de novos módulos.
- Ser compatível com ambiente industrial, enfrentando suas impurezas, umidade, influências eletromagnéticas e vibrações.
- Possuir funcionalidades lógicas[55].

Esses critérios foram apresentados a quatro empresas especializadas em controle da época:

- Allen-Bradley, através da empresa sediada em Michigan, Information Instruments, Inc.
- Digital Equipment Corporation (DEC).
- Century Detroit.
- Bedford Associates [55].

Para atender estes critérios, Morley e sua equipe da Bedford Associates buscaram uma forma de controle computacional que pudesse substituir os relés e oferecer um processo mais simples para realizar mudanças para reconfigurações de projeto. Foi quando em 1968, Morley e sua equipe criaram o que seria conhecido como o primeiro CLP da história, chamado à época apenas de Controlador Programável (ou PC, do inglês Programmable Controller), o Modicon 084, que recebeu esse nome por ser o 84º projeto da empresa [56]. Pode-se ver uma imagem deste primeiro CLP e seus principais criadores na Figura 25.

Figura 25 – Primeiro CLP comercial, Modicon 084. Da esquerda para direita: Richard Morley, Tom Bossevain, George Schwenk e Jonas Landau [8].

O Modicon 084 tinha 16 entradas digitais e 16 saídas digitais, além de 1KB de memória, o que apesar de hoje ser pouco, na época era o bastante para armazenar a lógica de pequenas e médias automações.

A grande característica dos Controladores Programáveis eram o fato de serem programáveis, ou seja, a sua lógica de funcionamento estava baseada em um programa

de computador e não em fiações e circuitos físicos. Isso permitiu que não apenas a lógica pudesse ser construída mais rapidamente, mas também as modificações e manutenções fossem muito mais ágeis.

Após conseguirem novos investimentos, Morley fundou a empresa Modicon (MODular DIgital CONtroller ou Controlador Digital Modular em português) para fabricação desses controladores [55].

Visando facilitar a programação desse novo dispositivo, foi criada uma forma de programação que utilizava as mesmas lógicas e símbolos que os circuitos a relé. Foi assim criada a linguagem de programação Ladder, que tem esse nome porque sua programação gráfica lembra muitas vezes os degraus de uma escada (Ladder em inglês), conforme pode-se ver na Figura 26.

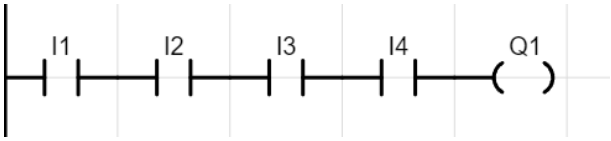

Figura 26 – A linguagem Ladder pode lembrar uma escada. Fonte: o autor.

3.7.1 Norma IEC 61131-3 e as Linguagens de programação de CLP

Nos anos 1990 a norma IEC 61131-3 foi criada, padronizando cinco linguagens de programação para CLP [57]. Instituída pela Comissão Eletrotécnica Internacional (IEC), esta norma visa promover a uniformidade e a interoperabilidade entre os sistemas de automação, facilitando assim a integração e a manutenção de sistemas de automação programáveis. A norma delineia cinco linguagens de programação, cada uma atendendo a diferentes necessidades e preferências dos programadores, permitindo uma abordagem mais flexível e adaptável ao desenvolvimento de sistemas de controle. As linguagens podem ser divididas entre textuais (que são escritas com texto) e gráficas (que utilizam símbolos gráficos).

Linguagens Textuais:

- LI - Lista de Instruções (*Instruction List*)
- ST - Texto Estruturado (*Strutured Text*)

Linguagens Gráficas:

- LD - Diagrama de Contatos (*Ladder*)
- FBD - Diagrama de Blocos (*Function Block Diagram*)
- SFC - Gráficos de Função sequencial (*Sequential Function Chart*)

Lista de Instruções (IL): É uma linguagem de baixo nível que utiliza uma série de instruções mnemônicas para representar operações. Embora seja menos intuitiva em comparação com as linguagens gráficas, oferece um controle mais granular sobre o processo de programação, sendo particularmente útil para aplicações que requerem otimização e eficiência no nível do código.

Texto Estruturado (ST): Como uma linguagem textual, o Texto Estruturado oferece uma abordagem mais versátil à programação de CLP. Utilizando sintaxes semelhantes às linguagens de programação de alto nível, como a Linguagem de computador C, permite a implementação de algoritmos complexos e estruturas de controle, facilitando a criação de programas mais sofisticados e personalizados.

Ladder (LD): Originada dos circuitos elétricos de relés, a linguagem Ladder é visualmente representada por elementos elétricos como contatos normalmente aberto, normalmente fechado, bobinas e outros elementos, facilitando a interpretação e a análise por engenheiros e técnicos da área da elétrica. A linguagem ladder é amplamente utilizada devido à sua simplicidade e eficácia na representação de lógicas de controle sequenciais.

Diagrama de Blocos (FBD): Esta linguagem gráfica permite a programação através do uso de blocos funcionais lógicos, como lógica E (AND), OU (OR), Inversora (NOT), dentre outras, que são interconectados por linhas representando os fluxos de dados. A FBD facilita a modelagem de sistemas complexos, permitindo uma visualização clara das interações entre diferentes componentes e promovendo uma abordagem modular ao desenvolvimento de sistemas de controle.

Gráficos de Função Sequencial (SFC): Esta linguagem gráfica é especializada na representação de processos sequenciais e paralelos, permitindo a visualização clara de diferentes etapas e transições em um processo de controle. A SFC facilita a organização e a estruturação de programas complexos, promovendo uma abordagem sistemática ao desenvolvimento de sistemas de controle.

Dentre as linguagens acima, a linguagem Ladder é a mais usada [58]. Além disso, existem estudos que mostram que a Linguagem Ladder é a de mais fácil entendimento dentre todas as linguagens acima [59].

3.8 Introdução à Linguagem Ladder

A linguagem de programação Ladder é uma linguagem gráfica que se assemelha a esquemas elétricos, amplamente utilizada na programação de controladores lógicos programáveis (PLCs). Vamos introduzir os conceitos básicos da linguagem Ladder, fornecendo exemplos com acesso direto ao simulador desenvolvido neste trabalho, para facilitar a compreensão dos principais elementos dessa linguagem de programação de CLP.

3.8.1 Linhas (Rungs)

As "rungs"ou linhas, são semelhantes às linhas de um diagrama elétrico, onde os componentes são organizados de forma lógica para realizar operações específicas.

3.8.2 Contatos e Bobinas

Os contatos podem ser de dois tipos: normal aberto (NA) e normal fechado (NF). Eles representam as condições lógicas básicas na programação Ladder. Já as bobinas representam as saídas do CLP, que podem ser ligadas ou desligadas, assim como as bobinas de um relé.

Um contato "normal aberto"(NA), representado graficamente como -| |-, funciona como uma espécie de interruptor que, em seu estado normal (ou seja, quando não está energizado), permanece aberto, não permitindo a passagem da energia. Quando este contato recebe um sinal (ou seja, é energizado), ele fecha, permitindo assim a passagem da energia.

Figura 27 – Contato Normal Aberto acionando uma bobina. Disponível em: <https://gvensino.com.br/sim01na>. Fonte: o autor.

No exemplo visto na Figura 27, quando a entrada I1 do CLP é energizada, o contato normalmente aberto na programação se fecha, enviando a energia para a saída Q1, acionando-a. Dessa forma, a saída Q1 do CLP será energizada, alimentando o que estiver ligado nela (uma lâmpada, um motor, uma sirene, dentre outros). A Figura 28 ilustra esse funcionamento.

Por outro lado, um contato "normal fechado"(NF), representado graficamente como -|/|-, é um interruptor que, em seu estado normal, está fechado, permitindo a passagem da energia. Quando este contato recebe um sinal (ou seja, é energizado), ele abre, interrompendo a passagem da energia.

No exemplo visto na Figura 29, a saída Q1 já começa energizada, pois o contato de I1 é normalmente fechado, permitindo que a energia chegue até a bobina. Quando a entrada I1 do CLP for acionada, o contato na programação irá se abrir, impedindo a energia de chegar até a bobina Q1, fazendo com que a saída Q1 se desligue. A Figura 30 ilustra esse funcionamento.

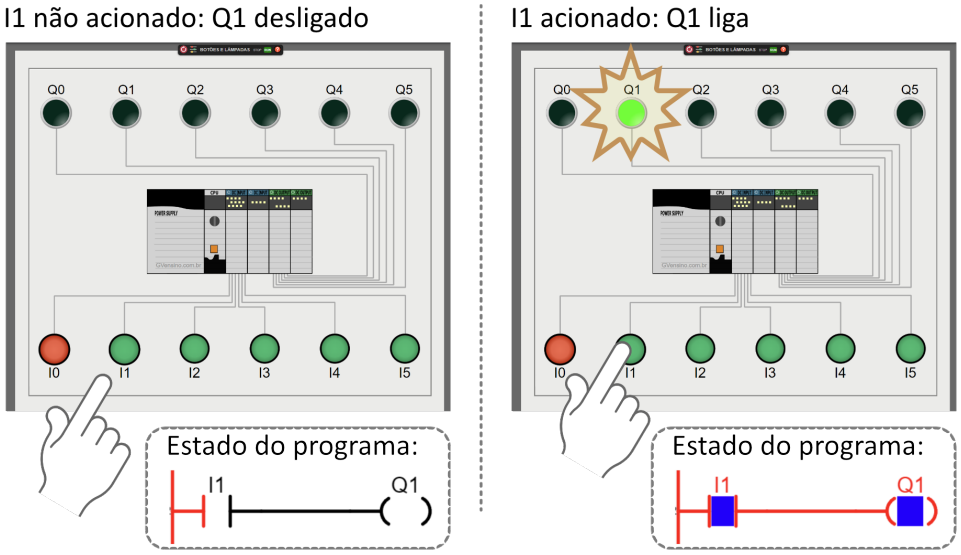

Figura 28 – Comportamento do contato NA no CLP. I1 ligará Q1. Fonte: o autor.

Figura 29 – Contato Normal Fechado acionando uma bobina. Disponível em: <https://gvensino.com.br/sim02NF>. Fonte: o autor.

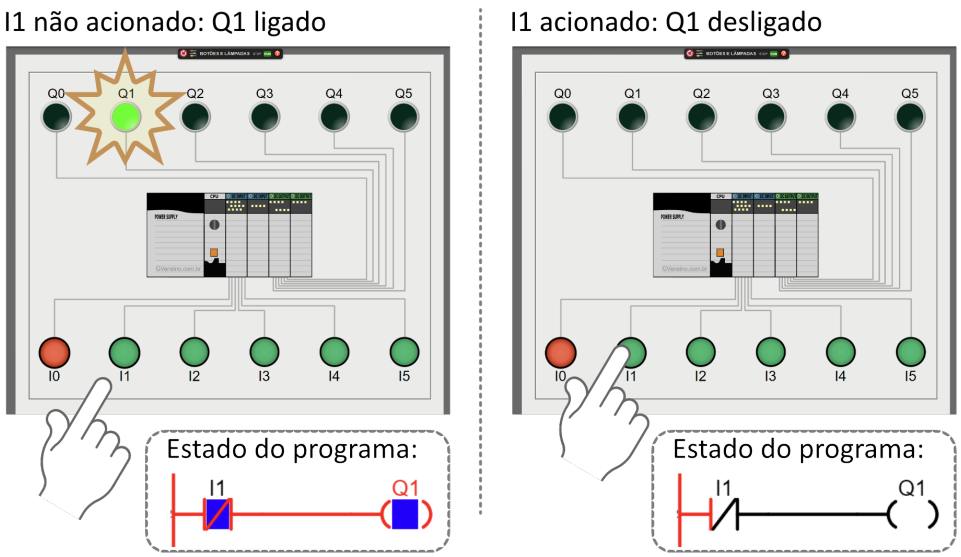

Figura 30 – Comportamento do contato NF no CLP. Fonte: o autor.

3.8.3 Lógica E/AND

Ao associar os contatos em série ou em paralelo, é possível criar lógicas mais complexas, como uma lógica E ou uma lógica OU, que envolvem mais de uma entrada apenas.

Na lógica E, a saída será ligada apenas se todas as entradas forem acionadas. Dito de outra forma: a saída será ligada se a primeira E a segunda entrada forem acionadas — daí o nome de lógica E. Já se qualquer entrada estiver desligada, a energia não conseguirá energizar a bobina da programação. Esse funcionamento pode ser visto na Tabela 3.

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3 – Tabela-Verdade da Lógica E

Para fazer a lógica E em linguagem Ladder, interliga-se os contatos normalmente abertos em série, conforme o programa visto na Figura 31.

Figura 31 – Lógica E - a saída Q1 só será acionada se as duas entradas forem energizadas. Disponível em: <https://gvensino.com.br/sim03and>. Fonte: o autor.

A Figura 32 mostra o funcionamento da lógica E na prática.

3.8.4 Lógica OU/OR

Já na lógica OU, a saída será ligada se qualquer uma das entradas for acionada. Dito de outra forma, a saída será acionada se a primeira entrada OU a segunda entrada for acionada. Daí o nome de lógica OU. Esse funcionamento pode ser visto na Tabela 4.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 4 – Tabela-Verdade da Lógica OU

A lógica OU é feita em linguagem Ladder interligando-se contatos N.A. em paralelo, conforme visto na Figura 33.

A Figura 34 mostra o funcionamento da lógica OU na prática.

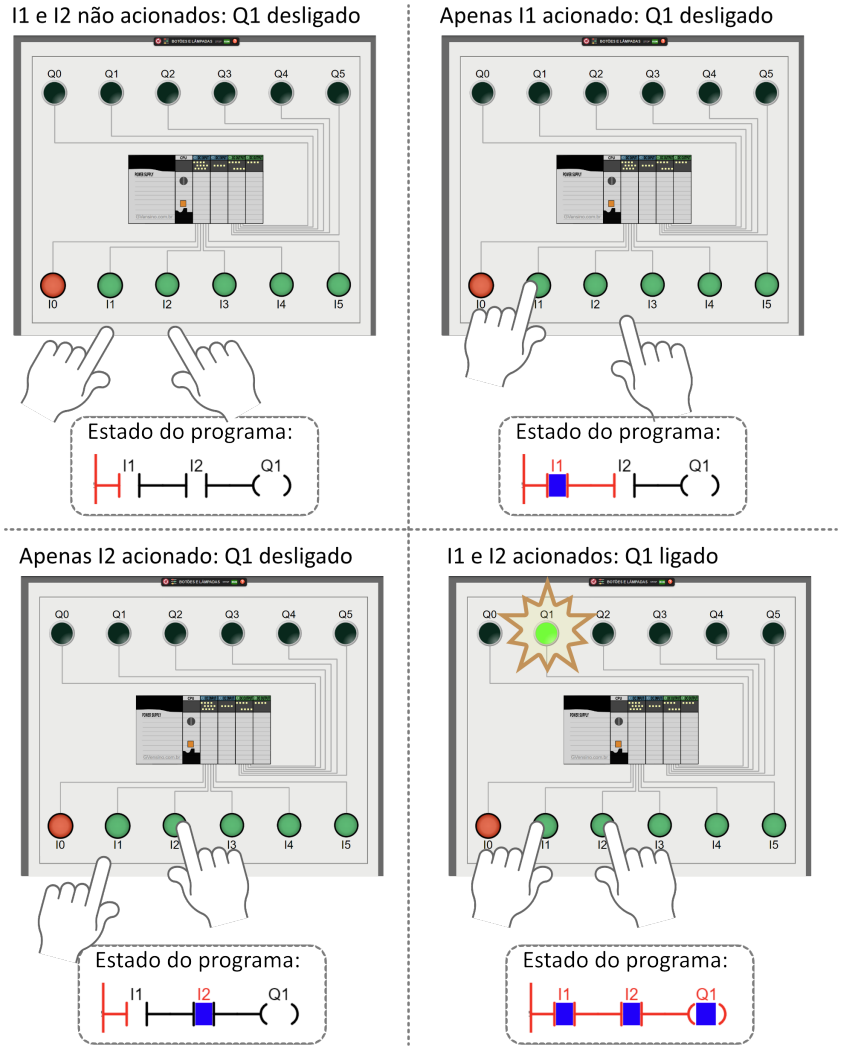

Figura 32 – Comportamento da lógica E no CLP. Fonte: o autor.

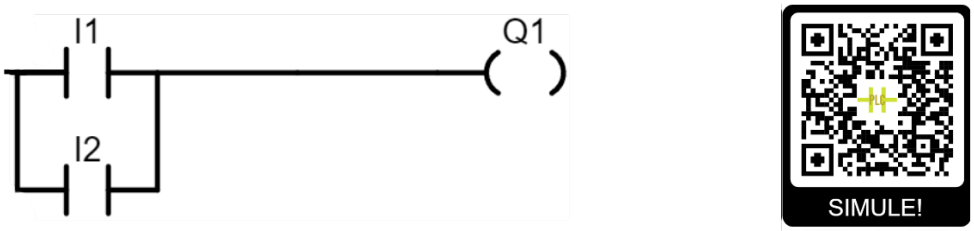

Figura 33 – Lógica OU em Ladder. Disponível em: <https://gvensino.com.br/sim04or>. Fonte: o autor.

3.8.5 Lógica de Selo

Em algumas situações deseja-se que uma saída continue ligada mesmo quando a entrada que a acionou for desativada. Em Ladder, uma das formas de se fazer isso é através da chamada lógica de selo.

A técnica do selo é empregada há muitos anos nos circuitos elétricos, com o objetivo de manter a bobina de um relé ou de um contator ligada, mesmo quando o sinal que a acionou for desativado. O circuito que ilustra essa técnica na elétrica pode ser visto na

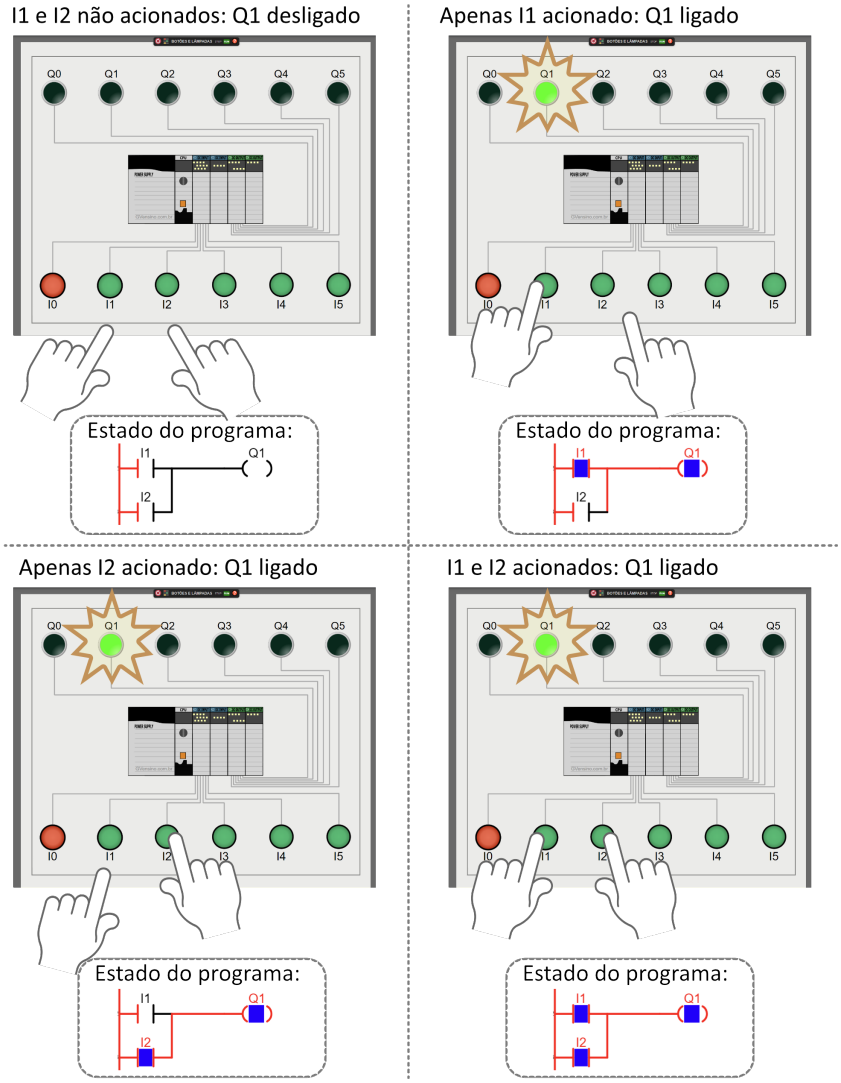

Figura 34 – Comportamento da lógica OU no CLP. Fonte: o autor.

Figura 35.

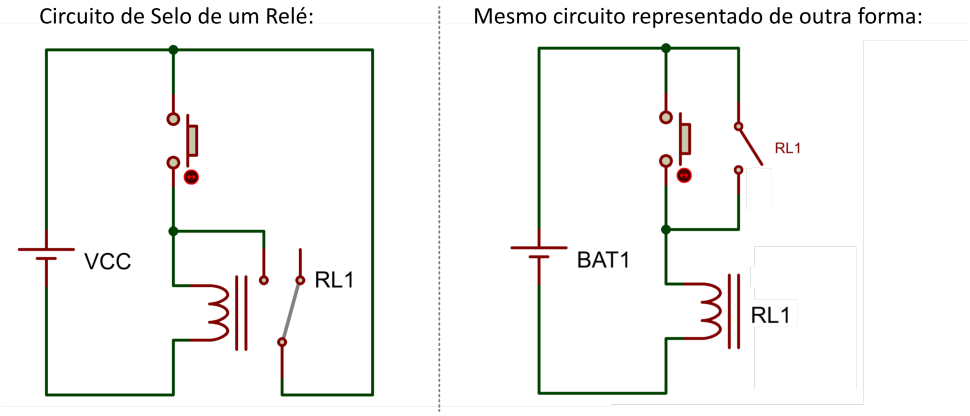

Figura 35 – Selo de um relé. Fonte: o autor.

A ideia por trás da técnica do selo é o próprio relé se mantendo energizado depois que recebeu energia a primeira vez. Quando o botão energiza a bobina do relé, o contato

N.A. do relé fecha, fazendo com que este contato é que passe a enviar energia para a bobina do relé, não necessitando mais da energia do botão que o acionou.

Entretanto, e se desejar desligar a bobina do relé? Para que isso seja possível, deve-se inserir algum elemento no circuito capaz de retirar a energia da bobina, desligando-a. Para isso, um botão normalmente fechado pode ser utilizado em série com a bobina, fazendo com que ao ser pressionado, desenergize a bobina, desligando-a. Esse circuito pode ser visto na Figura 36.

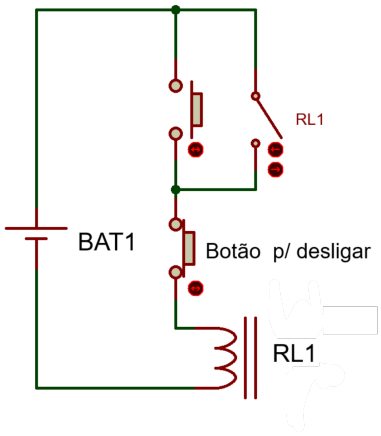

Figura 36 – Selo de um relé. Fonte: o autor.

Em linguagem Ladder também podemos fazer a técnica do selo exatamente da mesma maneira. Em ladder, o selo consiste de uma entrada acionando uma bobina e, em paralelo com esta entrada, um contato normalmente aberto com o mesmo endereço da saída, conforme pode ser visto na Figura 37. Dessa forma, quando a saída é acionada, o contato normalmente aberto desta saída se fecha, fazendo com que o próprio contato da saída a mantenha energizada, não precisando mais que o sinal de entrada fique ativo.

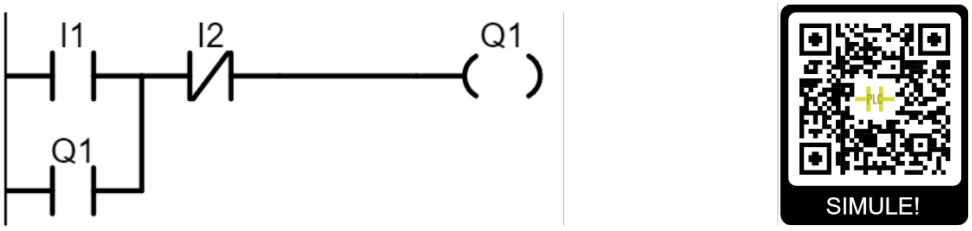

Figura 37 – Selo. Disponível em <https://gvensino.com.br/sim05selo>. Fonte: o autor.

Na lógica selo visto na Figura 37, ao acionar a entrada I1 do CLP a saída Q1 irá ligar e permanecerá ligada mesmo quando I1 for solto, mantida energizada pelo contato de Q1 de selo. O comportamento prático pode ser visto na Figura 38. A saída Q1 só será desligada ao energizar a entrada I2 do CLP, que então desligará a bobina Q1, desfazendo o selo. Outra forma de se manter uma saída ligada é utilizando as bobinas retentivas de Set e Reset.

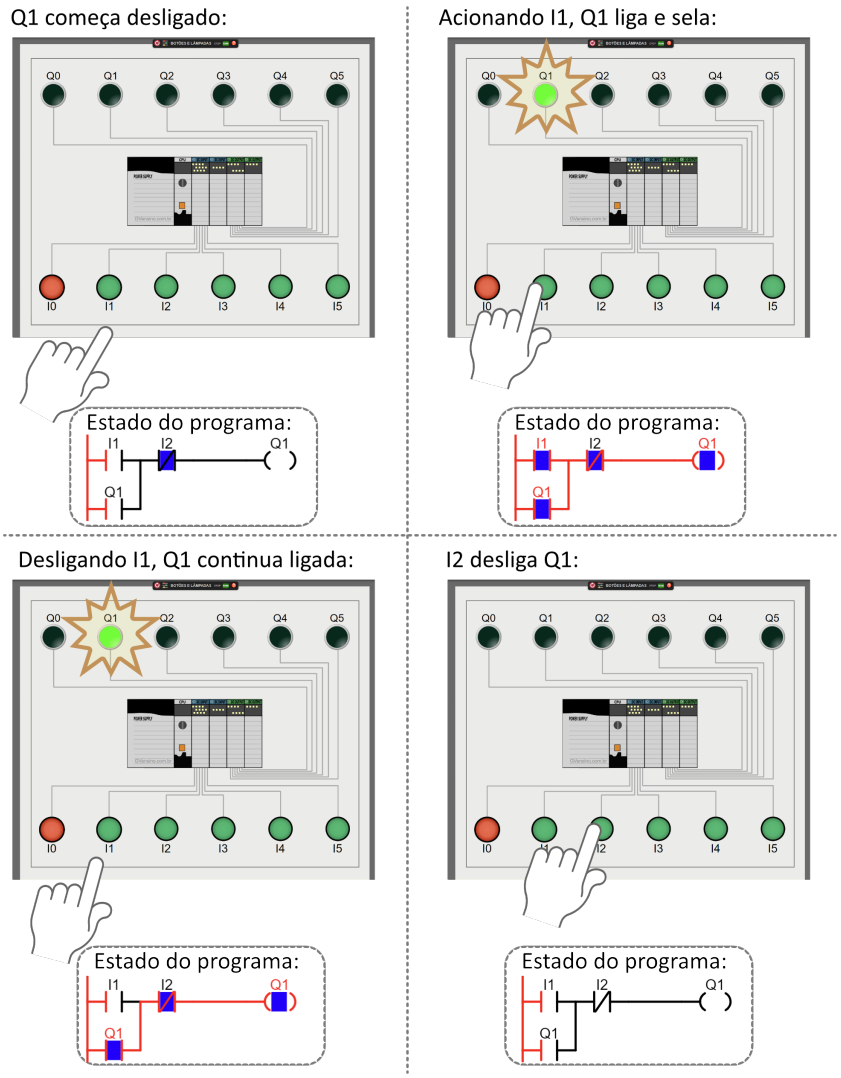

Figura 38 – Comportamento do selo. Fonte: o autor.

3.8.6 Bobinas retentivas de Set e Reset

Uma outra forma de se manter uma saída ligada, mesmo quando a entrada que a acionou é desligada é utilizando as bobinas retentivas de Set e Reset. A bobina de Set liga uma saída e a mantém ligada mesmo que a bobina pare de receber energia. Uma saída que foi Setada (isto é, ligada usando-se a bobina de Set) só será desligada quando a sua bobina de Reset for acionada. Portanto, a bobina de Set liga e mantém uma saída ligada; só será desligada quando o Reset dessa mesma saída for acionado.

Figura 39 – Bobinas retentivas de Set e Reset. Disponível em: <https://gvensino.com.br/sim06setreset>. Fonte: o autor.

No exemplo mostrado na Figura 39, usando as bobinas retentivas de Set e Reset, I1 seta a saída Q1, fazendo-a ligar e permanecer ligada. A saída Q1 só será desligada quando a sua bobina de Reset for acionada, o que acontecerá quando a entrada I2 do CLP for acionada. Repare que este exemplo usando as bobinas de Set e Reset apresenta exatamente o mesmo funcionamento do exemplo anterior, que utiliza a técnica do selo.

3.8.7 Temporizadores TON, TOF, TP e RTO

Em diversas aplicações é necessário que determinados processos aconteçam durante um certo período de tempo, ou após um período de tempo. Só pra citar um exemplo, imagine como seria se não conseguíssemos definir tempos específicos para o funcionamento dos semáforos que controlam o trânsito nas cidades: o tráfego de veículos seria ainda mais caótico do que já é! Para que se possa trabalhar com tempos, existem os temporizadores (também chamados de Timers), que nos permitem trabalhar com tempos na programação Ladder.

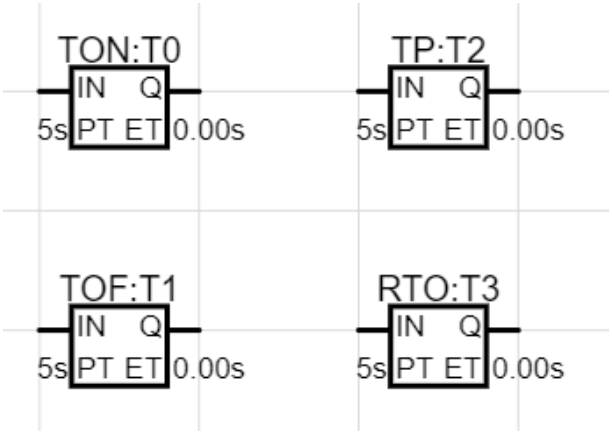

Figura 40 – Temporizadores. Fonte: o autor.

Existem 4 temporizadores principais dentro do universo da programação Ladder: TON, TOF, TP e RTO (ver Figura 40. Os temporizadores TON e TOF são os principais e presentes em praticamente todos os programas de CLP. Já os temporizadores TP e RTO são implementados apenas por alguns fabricantes.

Os temporizadores são blocos de programação que em geral possuem os seguintes elementos:

- Entrada IN - energiza o temporizador;
- Entrada PT - Preset Time (Ajuste de Tempo): valor de tempo definido pelo programador (em ms ou s);
- Saída Q - saída do temporizador (funcionará dependendo de cada tipo de temporizador);

- Saída ET - Elapsed Time (Tempo Decorrido): mostra a contagem de tempo acontecendo pelo temporizador.

Cada um dos temporizadores terá um funcionamento diferente. Vamos estudá-los separadamente.

Temporizador TON (Timer ON Delay): O temporizador TON é utilizado para criar um atraso antes de ativar uma saída. Por isso é também chamado de Temporizador de Atraso à Energização.

Funcionamento: ao energizar a entrada IN, o temporizador começa a contar o tempo em ET. Quando o tempo atinge o valor ajustado em PT, a saída do temporizador é energizada. Em outras palavras: a saída Q só liga após um certo tempo depois da entrada IN ser acionada. Se em qualquer momento durante a contagem a entrada IN do temporizador for desligada, o temporizador automaticamente para a contagem de tempo e nada mais acontece.

Na Figura 41, pode-se ver o funcionamento do temporizador TON explicado de forma gráfica.

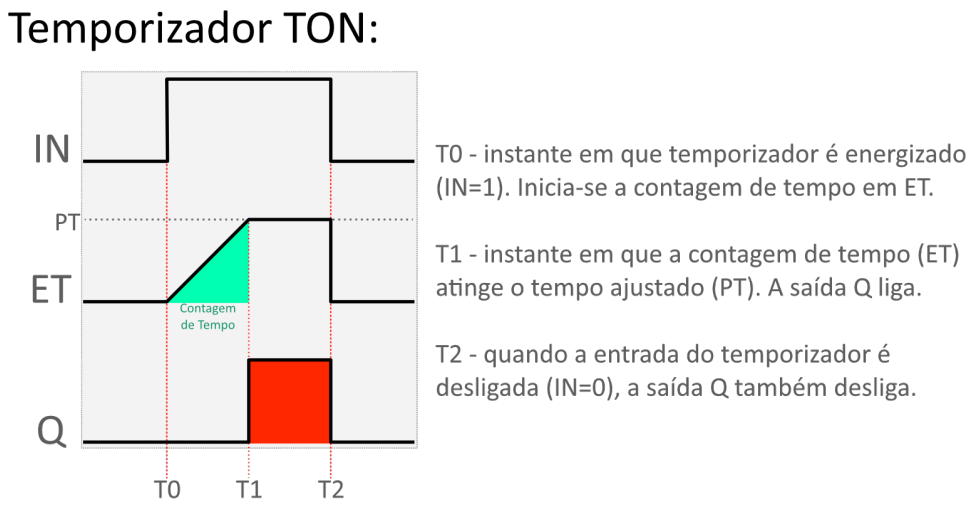

Figura 41 – Funcionamento do Temporizador TON. Fonte: o autor.

Na Figura 42, pode-se ver um exemplo de programação simples utilizando o temporizador TON.

Figura 42 – Temporizador TON. Disponível em <https://gvensino.com.br/sim07timerton>. Fonte: o autor.

No programa da Figura 42, ao acionar a entrada I1 do CLP o temporizador é energizado e começa a temporização. Enquanto a entrada I1 continua energizando o temporizador, ele continua contando o tempo. Após decorrido o tempo em PT (que é de 5 segundos no exemplo acima), a saída Q do temporizador á acionada, ligando a saída Q1 do CLP. Repare que, para que a saída Q1 chegue a ligar, é necessário que a entrada I1 permaneça acionada durante toda a contagem de tempo, caso contrário, nada acontecerá.

A partir do funcionamento básico do temporizador TON é possível criar diversos outros comportamentos. Por exemplo, imagine que, ao pressionar um botão, deseja-se que uma lâmpada ligue por 10s e então apague. Podemos ver essa programação feita na Figura 43

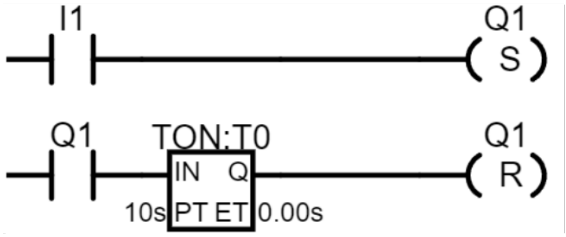

Figura 43 – Ligando Q1 ao acionar I1 e desligando automaticamente após 10s. Disponível em: <https://gvensino.com.br/sim07timerton_2>. Fonte: o autor.

Repare que ao acionar a entrada I1, a saída Q1 é setada, ligando e permanecendo ligada. Então é o contato da saída Q1 quem aciona o temporizador, que contará o tempo de 10 segundos. Após decorrido o tempo, o temporizador irá resetar a saída Q1, desligando-a.

Temporizador TOF (Timer OFF Delay): O temporizador TOF funciona de maneira oposta ao TON. Quando a entrada de controle (IN) é desenergizada, o temporizador começa a contar. A saída (Q) só é desativada após o tempo predefinido (PT) ser alcançado, mantendo-se ativa mesmo que a entrada seja desenergizada. Esse funcionamento pode ser visto na Figura 44.

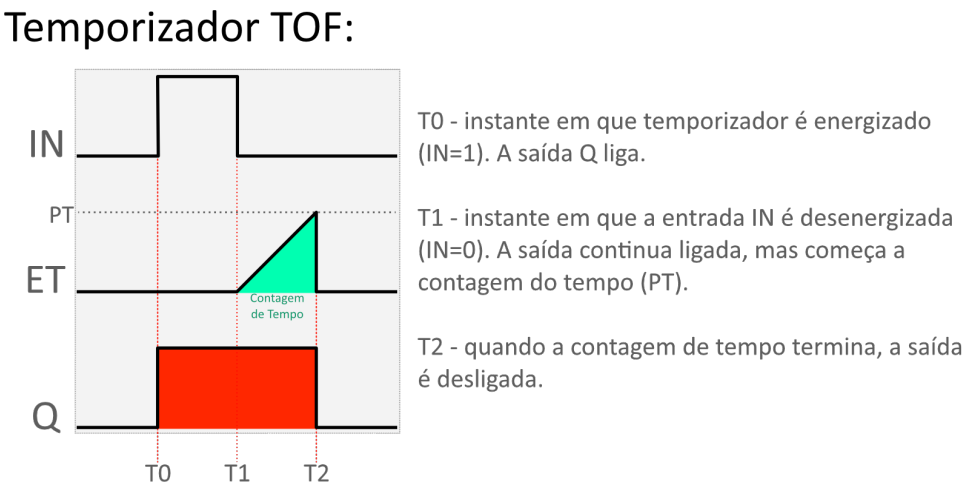

Figura 44 – Funcionamento do Temporizador TOF. Fonte: o autor.

O programa na Figura 45 mostra o funcionamento básico do temporizador TOF.

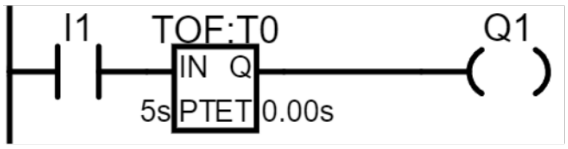

Figura 45 – Ligando Q1 ao acionar I1 e desligando automaticamente após 10s. Disponível em: <https://gvensino.com.br/sim08timertof>. Fonte: o autor.

Um exemplo de aplicação do temporizador TOF é em sensores de presença para iluminação, como em corredores de prédios e empresas. Uma vez que o sensor detecta a presença de alguém, a iluminação é acionada e permanece acionada enquanto uma presença estiver sendo detectada. Quando nenhuma presença é identificada, o temporizador mantém a iluminação ligada por um breve intervalo de tempo e então a desliga.

Temporizador TP (Pulse Timer): O temporizador TP não existe em todos os softwares de programação de CLP, por isso, dependendo do fabricante do software ele pode não estar disponível.

O temporizador TP é utilizado quando se deseja acionar uma saída durante um intervalo específico de tempo, após o qual a saída deve desligar. Quando a entrada de controle (IN) é energizada, o temporizador ativa a saída (Q) pelo tempo predefinido (PT). Passado esse tempo, a saída é desativada, mesmo que a entrada continue energizada. Esse funcionamento pode ser visto na Figura 46.

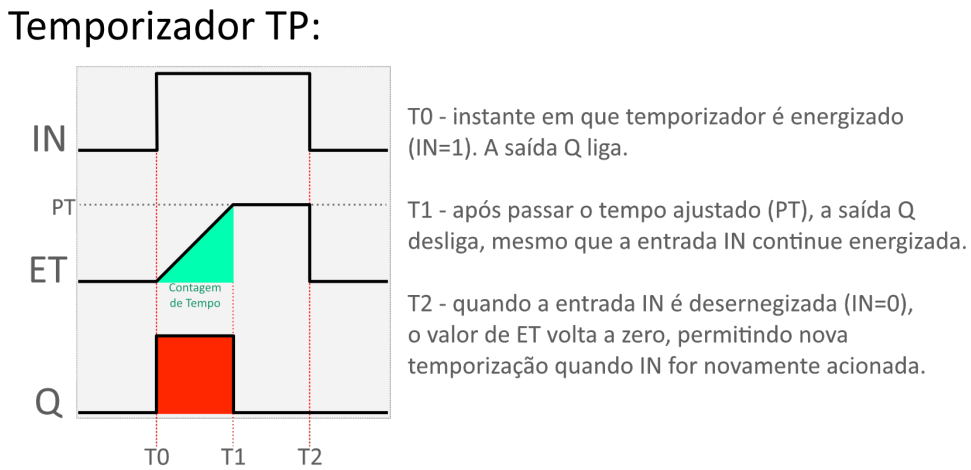

Figura 46 – Funcionamento do Temporizador TP. Fonte: o autor.

Na Figura 47, pode-se ver um exemplo de programação onde uma saída permanece ligada por 5s e então apaga, usando o temporizador TP.

Temporizador RTO (Retentive Timer On): O temporizador RTO, também conhecido como temporizador retentivo, tem o funcionamento parecido ao temporizador

Figura 47 – Temporizador TP ligando Q1 por 5s. Disponível em <https://gvensino.com.br/sim09timertp>. Fonte: o autor.

TON, com uma diferença: quando desernegizado no meio da contagem de tempo, o valor do tempo não volta a zero, mas sim fica armazenado no temporizador (em ET). A contagem é retomada do ponto onde parou quando é reenergizado. A saída (Q) é ativada quando o tempo acumulado atinge o tempo predefinido (PT). Para zerar o valor da temporização acumulada em ET, é necessário resetar este temporizador usando uma bobina de Reset. O funcionamento do temporizador RTO pode ser visto na Figura 49.

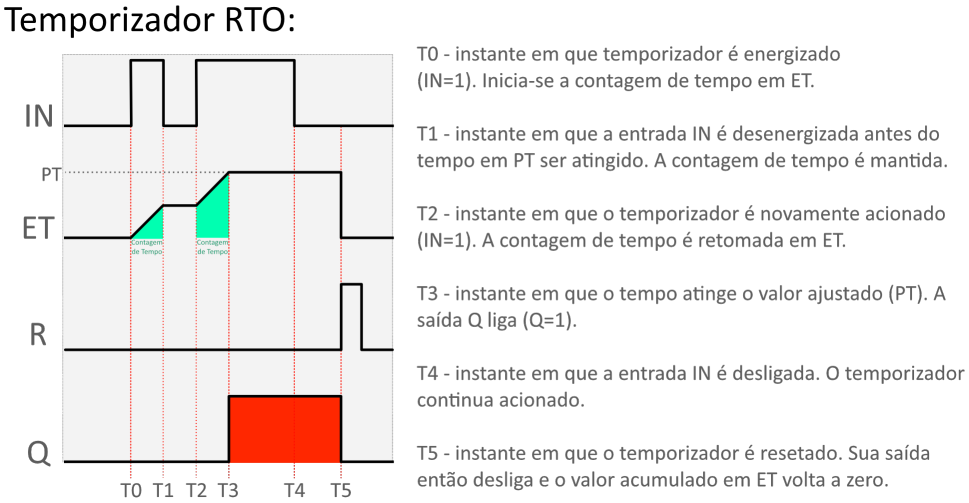

Figura 48 – Funcionamento do Temporizador RTO. Fonte: o autor.

No programa presente na Figura 49, ao acoinar I1 o temporizador começa a contar. Se I1 ficar pressionada durante 5s inteiros, a saída Q1 ligará. Mas se no meio da temporização, I1 desligar, o temporizador irá reter o valor da temporização, aguardando que I1 seja novamente acionado. Quando isso acontecer, a contagem de tempo continua de onde parou, até atingir os 5s, quando então a saída Q1 é acionada.

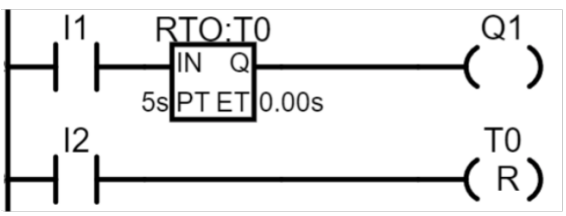

Figura 49 – Temporizador RTO acionando Q1 após I1 ter sido pressionada por 5s. Disponível em <https://gvensino.com.br/sim10timerrto>. Fonte: o autor.

O temporizador só será desligado quando for resetado através da bobina de Reset, com endereço do temporizador.

3.8.8 Contadores Crescente e Decrescente

Os Contadores são blocos de programação capazes de contar eventos (por exemplo, quantas peças passaram por uma esteira). Em linguagem Ladder existem dois tipos de Contadores: os contadores crescentes (CTU) e decrescentes (CTD), que podem ser vistos na Figura 50.

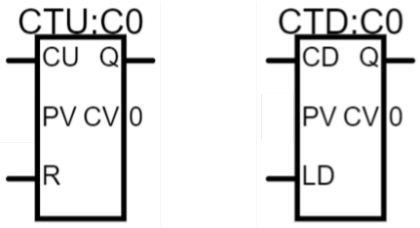

Figura 50 – Contadores CTU e CTD em Ladder. Fonte: o autor.

Contador Crescente (CTU - Count Up): Este é o contador mais comum e utilizado. O Contador CTU conta seu valor (somando +1) toda vez que sua entrada é energizada, armazenando esse valor contado na saída CV (Count Value - Valor da Contagem). Quando o valor de contagem (CV) atinge o valor ajustado pelo programador em PV (Preset Value - Valor de Ajuste), saída Q do contador liga.

Para desligar a saída do contador e zerar o valor de sua contagem, é necessário energizar a entrada de Reset (R) do Contador.

Na Figura 51 pode-se ver o funcionamento do contador CTU graficamente. Cada pulso na entrada IN do contador incrementa o valor de CV. Quando CV = PV, a saída do contador liga e permanece ligada até que a entrada de Reset (R) do Contador seja acionada.

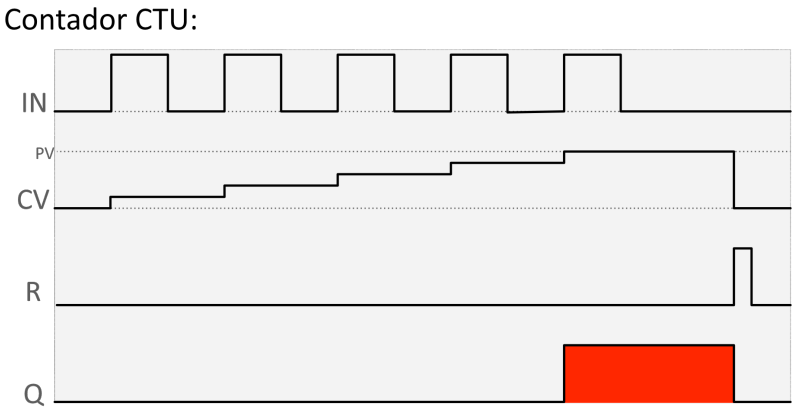

Figura 51 – Funcionamento do Contador CTU. Fonte: o autor.

Já na Figura 52, a saída Q1 do CLP ligará quando a entrada I1 for acionada 5 vezes. Ao acionar a entrada I2, a saída Q1 irá desligar e o contador irá reiniciar seu valor de contagem.

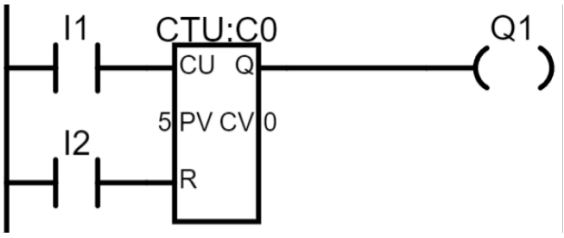

Figura 52 – Exemplo Contador CTU. Ao pressionar I1 5x, Q1 liga. I2 desliga. Disponível em <https://gvensino.com.br/sim11ctu>. Fonte: o autor.

Contador Decrescente (CTD - Count Down): O contador decrescente funciona de maneira inversa ao contador crescente. Ele diminui seu valor (subtraindo -1) cada vez que sua entrada é energizada e a sua saída Q é ativada quando o valor acumulado atinge zero. O contador possui uma entrada LD (Load - Carregar) que carrega o valor ajustado (PV) no valor de contagem (CV), para que o contador comece já com um valor específico. Esse funcionamento do contador CTD pode ser visto na Figura 53.

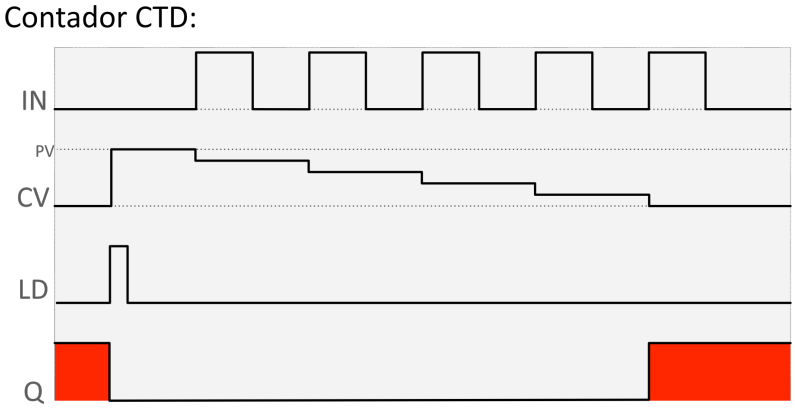

Figura 53 – Funcionamento do Contador CTD. Fonte: o autor.

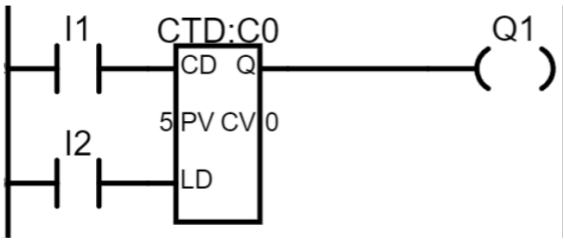

Figura 54 – Exemplo Contador CTD. Disponível em <https://gvensino.com.br/sim12ctd>. Fonte: o autor.

No exemplo da Figura 54, I2 deve ser pressionado para carregar o valor de ajuste (PV=5) na contagem. Cada vez que a entrada I1 é acionada, o valor é subtraído em -1. Quando o valor de contagem (CV) chega em zero, a saída do contador é acionada, ligando

Q1. Ao acionar LD novamente, a saída desliga e o contador volta ao valor inicialmente ajustado (PV=5).

Combinando CTU e CTD: caso se deseje realizar uma contagem que possa ser tanto crescente quanto decrescente, é possível combinar o uso dos contadores CTU e CTD. Para isso, ambos devem receber o mesmo endereço.

Na Figura 55, I1 aciona o contador crescente, incrementando o valor da contagem. I2 aciona o contador decrescente, decrementando o valor da contagem. Quando o valor da contagem chega em 5, a saída Q1 é acionada.

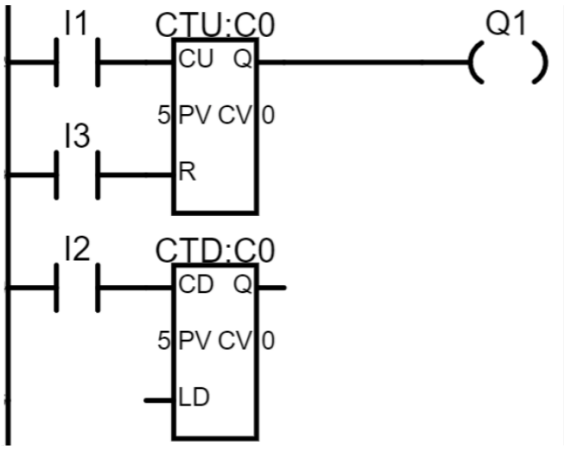

Figura 55 – Contadores CTU e CTD juntos. Disponível em <https://gvensino.com.br/sim13ctuctd>. Fonte: o autor.

3.8.9 Blocos Comparadores

Os blocos comparadores (Figura 56) permite que se compare valores dentro da programação, para definir se um valor é maior, menor, igual ou diferente de outro valor. Podem ser comparados valores de tempos, de contagens ou mesmo o valor de entradas analógicas, vindas de sensores como sensores de temperatura, pressão, etc.

Os blocos comparadores sempre comparam dois valores; caso a comparação seja verdadeira, a saída Q do bloco é acionada.

Existem seis tipos de blocos comparadores:

- EQ (Equal - Igualdade): compara se dois valores são iguais;
- NE (Not Equal - Diferente): compara se dois valores são diferentes;
- GT (Greater than - Maior): compara se o primeiro valor é maior que o segundo;
- GE (Greater or Equal than - Maior ou igual): compara se o primeiro valor é maior ou igual ao segundo;

- LT (Less than - Menor): compara se o primeiro valor é menor que o segundo;
- LE (Less or Equal than - Menor ou igual): compara se o primeiro valor é menor ou igual ao segundo;

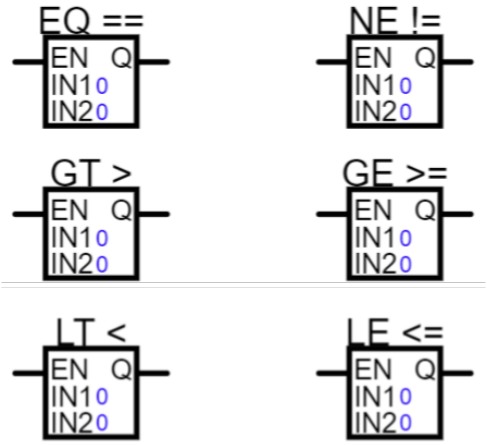

Figura 56 – Tipos de Comparadores. Fonte: o autor.

No exemplo da Figura 57, quando o valor da entrada analógica IW1 do CLP ultrapassa 500, o comparador aciona sua saída, ligando a saída Q1 do CLP.

Uma demonstração do funcionamento do programa da Figura 57 pode ser visto na Figura 58.

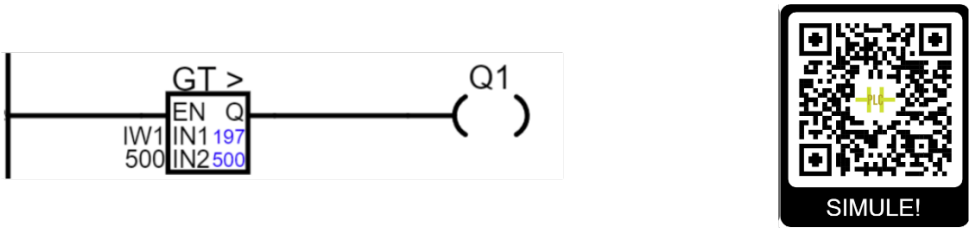

Figura 57 – Acionando Q1 se IW1 for maior que 500. Disponível em <https://gvensino.com.br/sim14comp>. Fonte: o autor.

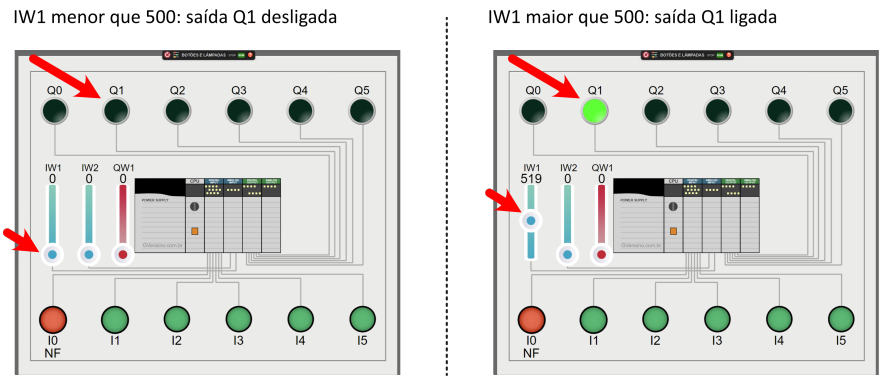

Figura 58 – Demonstração do programa anterior. Fonte: o autor.

3.8.10 Blocos Matemáticos / Aritméticos

Os blocos aritméticos, também conhecidos como blocos matemáticos, permitem a realização de operações matemáticas dentro da programação em Ladder. Estes blocos são utilizados quando se deseja efetuar cálculos, manipular valores de contadores, temporizadores ou mesmo dados provenientes de sensores (entradas analógicas). Uma imagem ilustrando os principais blocos aritméticos pode ser vista na Figura 59.

Os blocos aritméticos realizam operações entre dois valores, e o resultado pode ser armazenado em uma variável ou utilizado para controle direto de outros componentes do sistema.

Abaixo estão descritos os principais tipos de blocos aritméticos utilizados na programação Ladder:

- MOVE (Movimentação): este bloco é mais simples, pegando o valor de sua entrada e movendo para a saída.
- ADD (Adição): Este bloco soma dois valores e envia o resultado para a saída do bloco. O resultado da soma pode ser usado imediatamente para controle ou armazenado para uso posterior.
- SUB (Subtração): Realiza a subtração do segundo valor pelo primeiro.
- MUL (Multiplicação): Multiplica dois valores e envia o resultado para a saída do bloco. Esse bloco é essencial quando se necessita escalar ou amplificar um valor.
- DIV (Divisão): Divide o primeiro valor pelo segundo e envia o resultado para a saída do bloco.
- MOD (Módulo): Calcula o resto da divisão do primeiro valor pelo segundo. Útil para operações que envolvem ciclos ou periodicidade.
- ABS (Valor Absoluto): Retorna o valor absoluto do número inserido, isto é, retorna sempre o valor positivo do número. É um bloco útil para situações em que a direção do valor não é relevante, apenas a magnitude.
- SQRT (Raiz Quadrada): Calcula a raiz quadrada do valor fornecido, amplamente utilizada em cálculos matemáticos e de engenharia.

Repare que tanto o bloco MOVE, quanto o ABS e o SQRT possuem apenas uma entrada de valor (IN1), já que essas operações matemáticas precisam apenas de um valor.

O emprego correto desses blocos aritméticos possibilita a criação de programas Ladder capazes de realizar desde operações matemáticas básicas até cálculos complexos, essenciais para a tomada de decisões e controle preciso em sistemas automatizados.

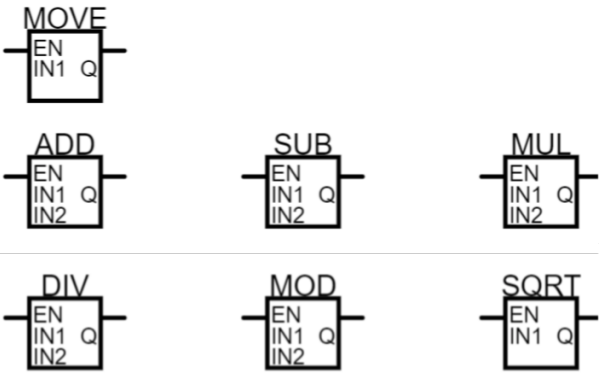

Figura 59 – Principais Blocos Aritméticos

Na programação de exemplo da Figura 60, a saída analógica QW1 recebe a soma das entradas analógicas IW1 e IW2.

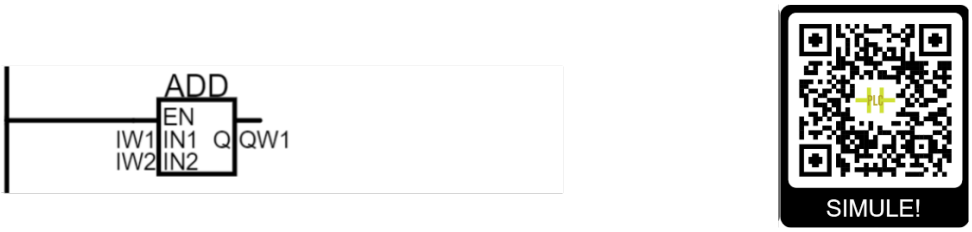

Figura 60 – QW1 recebe a soma de IW1 e IW2. Disponível em <https://gvensino.com.br/sim15arit>. Fonte: o autor.

4 Simuladores de CLP e Lógica Ladder existentes

Os Controladores Lógicos Programáveis (CLP) representam uma parcela significativa do investimento em infraestrutura em muitos laboratórios e ambientes industriais. Para os estudantes que aspiram a aprofundar-se na operacionalização e programação desses dispositivos, é importante que existam meios acessíveis e eficazes para facilitar o aprendizado sem incorrer em custos proibitivos.

Atualmente, a tecnologia oferece uma gama diversificada de ferramentas que possibilitam a simulação do funcionamento dos CLP, permitindo que professores e alunos possam trabalhar competências práticas sem a necessidade de interagir com o equipamento físico. Neste trabalho, categorizamos os simuladores de CLP em dois grupos distintos: simuladores WEB e simuladores instaláveis.

Os simuladores WEB, acessíveis através de navegadores de internet, apresentam a vantagem de serem facilmente acessíveis, sem a necessidade de instalação de software adicional. Estes simuladores, muitas vezes, oferecem uma interface amigável e intuitiva, facilitando a iniciação dos estudantes no mundo da programação de CLP. Além disso, a natureza on-line dessas ferramentas permite atualizações constantes, garantindo que os usuários tenham acesso às funcionalidades mais recentes e às melhores práticas da indústria. Alguns desses simuladores podem ser acessados também por smartphones, melhorando ainda mais a questão da acessibilidade.

Por outro lado, os simuladores instaláveis requerem um processo de instalação em um computador, algo que nem sempre é possível de ser feito em escolas e empresas. Em contrapartida, por terem a disposição todos os recursos de processamento da máquina, podem oferecer uma experiência mais robusta e aprofundada.

É vital que os educadores e os estudantes reconheçam as potencialidades e limitações de cada tipo de simulador, de modo a escolher a ferramenta que melhor atenda às suas necessidades específicas. A integração dessas ferramentas inovadoras no currículo acadêmico pode facilitar uma aprendizagem mais engajada e prática, preparando os estudantes de maneira mais eficaz para as demandas do mercado de trabalho na era digital.

4.1 Simuladores Web de CLP

PLC Fiddle (Figura 61): Este é um exemplo de um simulador WEB gratuito, que permite aos usuários criar, simular e compartilhar diagramas em linguagem Ladder dire-

tamente através de um navegador web. O PLC Fiddle é conhecido por sua interface gráfica intuitiva e recursos que facilitam a aprendizagem colaborativa. O PLC Fiddle possui todos os recursos básicos de programação Ladder: contatos NA e NF, bobinas, bobinas retentivas de Set e Reset (Latch e Unlatch), contadores crescente e decrescente, temporizadores TON, TOF e RTO, blocos de comparação e blocos aritméticos. Os programas podem ser feitos, simulados on-line e compartilhados através de um link gerado pelo próprio site. O PLC Fiddle não suporta dispositivos móveis, como celulares ou tablets, devendo ser usado apenas em computadores com mouse. Possui apenas simulação de botões e lâmpadas. O PLC Fiddle pode ser acessado sem a necessidade de cadastro através do endereço https://www.plcfiddle.com.

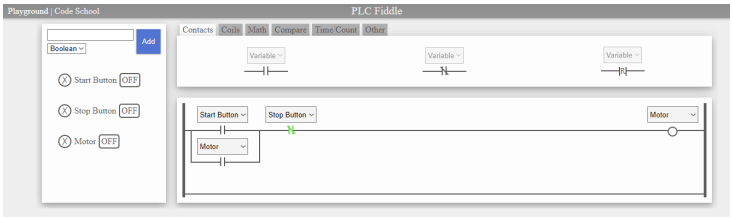

Figura 61 – Simulador on-line PLC Fiddle [9].

PLC Simulator on-line (Figura 62): Este é outro simulador de CLP em Linguagem Ladder gratuito disponível on-line, mantido pela comunidade e de código aberto. Assim como o PLC Fiddle, o PLC Simulator on-line possui todos os recursos básicos de programação Ladder: Contatos NA e NF, Bobinas, Bobinas retentivas de Set e Reset, Contadores crescente e decrescente, Temporizadores TON, TOF e RTO, Blocos de comparação e aritméticos. Apesar de todos os recursos, o simulador não apresenta a mesma acessibilidade no uso com dispositivos de toque mobile. Possui simulação de botões e lâmpadas e também de um sistema de esteira interativo. O PLC Simulator on-line pode ser acessado sem a necessidade de cadastro através do endereço: https://plcsimulator.on-line.

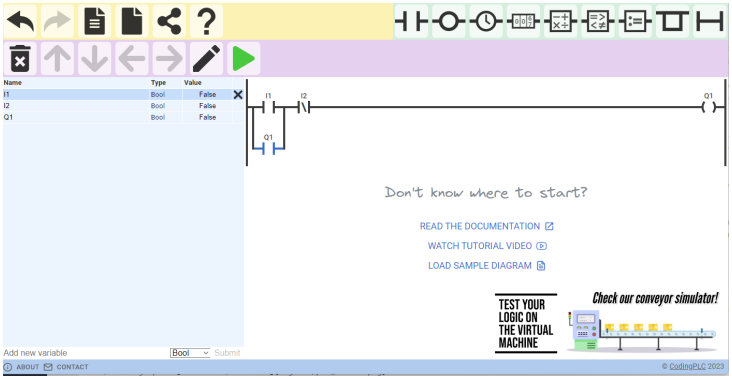

Figura 62 – Simulador PLC Simulator on-line [10].

PLC-Simulator Virtual Labs (Figura 63): Este é mais um simulador de CLP on-line gratuito para programação com Linguagem Ladder, mantido pela CoEP Virtual labs. De uso menos intuitivo que os anteriores, possui apenas alguns dos recursos básicos de

programação e simulação de linguagem Ladder: Contatos NA e NF, bobinas, contadores e temporizadores. Não possui blocos de comparação nem blocos aritméticos. Embora seu uso é possível em dispositivos móveis, a tela fica muito pequena e os comandos não são fáceis nem intuitivos. Mesmo em um computador com mouse, a sua usabilidade é baixa se comparado às opções anteriores, já que mesmo ações simples como inserção de um contato exige diversas etapas. Possui apenas simulação de liga e desliga de contatos e saídas. O simulador pode ser acessado sem a necessidade de cadastro através do endereço: https://plc-coep.vlabs.ac.in/exp/up-down-counter/simulation/.

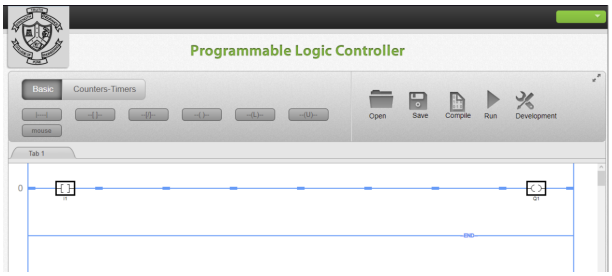

Figura 63 – Simulador PLC Simulator online [11].

PLC Simulator.NET (Figura 64): Este é outro simulador de CLP on-line gratuito para programação em linguagem Ladder, mantido por Phil Melore. Possui todos os recursos básicos de programação em linguagem ladder, tais como contatos, bobinas, temporizadores, contadores, blocos de comparação e aritméticos. Possui apenas simulação de botões e lâmpadas. Para acessar o simulador é necessário antes a criação de um cadastro. O acesso pode ser feito pelo endereço http://www.plcsimulator.net.

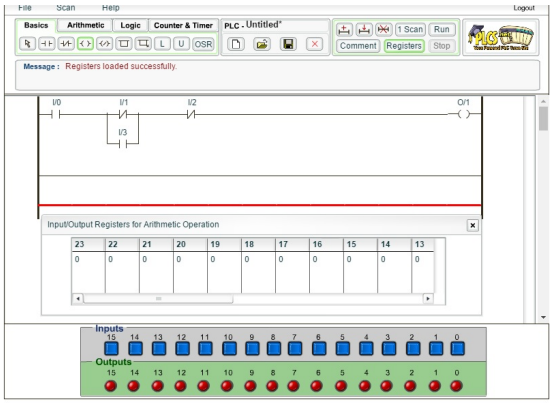

Figura 64 – Simulador PLC Simulator online [12].

4.2 Simuladores instaláveis

Existem diversos simuladores de CLP instaláveis, tanto gratuitos quanto pagos, que podem ser usados para o ensino de programação. Citaremos apenas alguns dos principais a seguir:

CODESYS (Figura 65): é um software que permite a programação de CLP e simulação de forma gratuita e em conformidade com o padrão IEC 61131-3. Este software, além de oferecer uma ampla gama de funcionalidades, permite que se utilize qualquer uma das cinco linguagens de programação padronizadas pela norma IEC. O CodeSys roda nativamente em sistema operacional Microsoft Windows para PCs, mas existe uma versão adaptada para rodar em Linux através do emulador Wine. Através do CodeSys é possível criar simulações simples, usando botões e lâmpadas usando uma ferramente de criação de visualizações interna. O CodeSys pode ser baixado e instalado após um cadastro através do endereço: https://www.codesys.com [51].

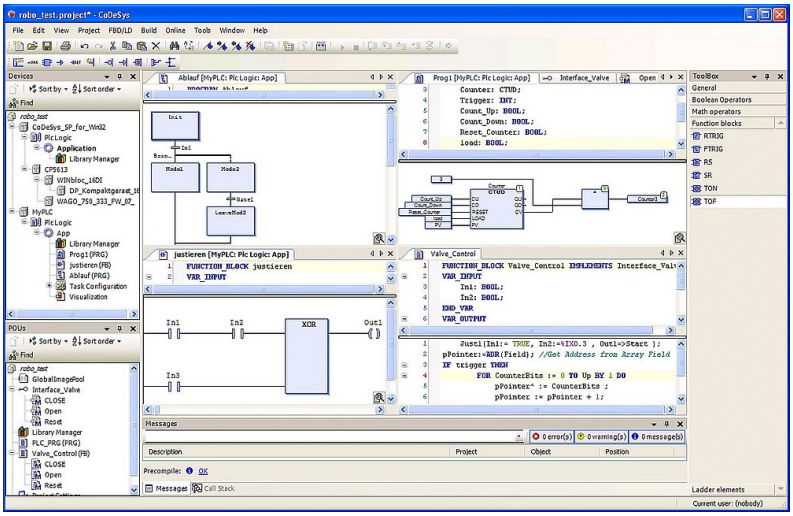

Figura 65 – Simulador CodeSys [13].

Do-More Designer (Figura 66): outro software gratuito para simulação de CLP em linguagem ladder. É bastante completo e com interface amigável, possuindo os principais recursos da linguagem ladder (contatos, temporizadores, contadores, blocos aritméticos e comparadores). Roda apenas em sistema operacional Microsoft Windows. Pode ser baixado e instalado através do site: https://www.automationdirect.com/do-more/h2/software.

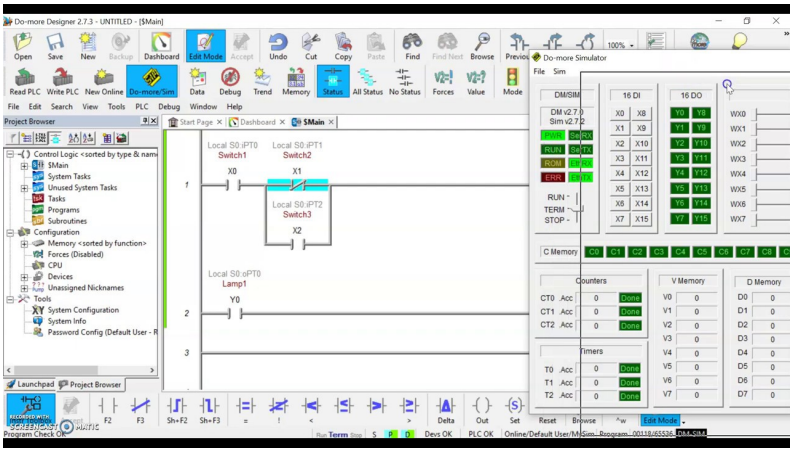

Figura 66 – Simulador Do-More Designer. Fonte: o autor

ISPSoft (Figura 67): é o software utilizado para programação dos CLP da fabricante Delta e que também permite a simulação sem a presença de um CLP físico. O simulador é gratuito e roda em ambiente Microsoft Windows, tendo à disposição todos os principais recursos de programação da Linguagem Ladder. O ISPSoft pode ser baixado e instalado através do site: https://www.deltaww.com/en-us/products/PLC-Programmable-Logic-Controllers/3598

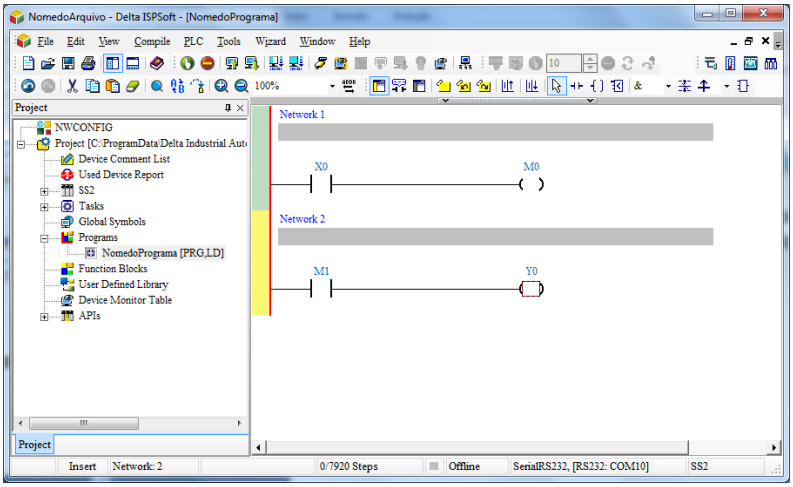

Figura 67 – Simulador Delta ISPSoft. Fonte: o autor

WTE PLC Simulator (Figura 68): outra ferramenta grátis para simulação de CLP em Linguagem Ladder disponível apenas em ambiente Microsoft Windows. Possui um visual moderno e seu uso é intuitivo. Permite a simulação apenas de botões e lâmpadas, ou seja, de entradas e saídas básicas. O WTE PLC Simulator pode ser baixado e instalado através do site: https://www.wte.co.nz/plc-simulator.html.

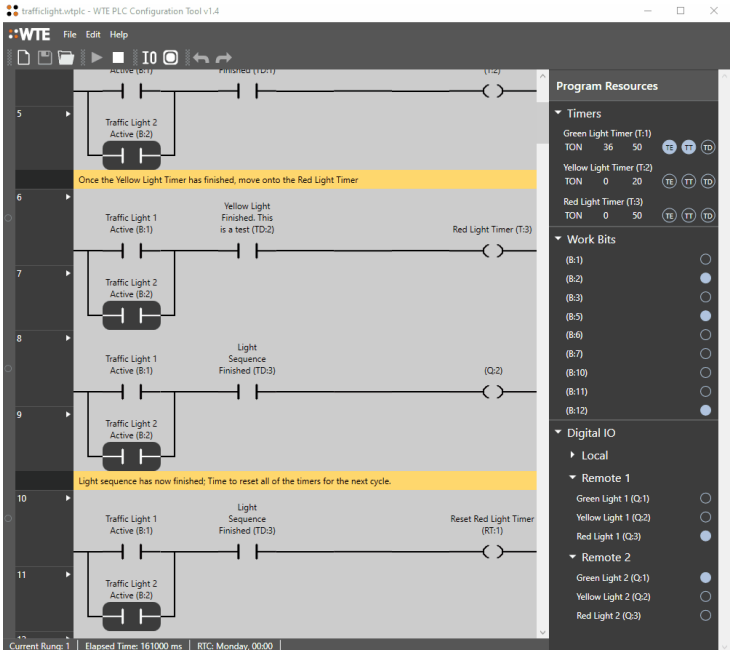

Figura 68 – Simulador WTE PLC [14].

Open PLC Editor (Figura 69): é um criador de lógica ladder que permite exportar o código feito para outras plataformas (como microcontroladores ou Arduíno) e também a simulação da lógica feita direto no software. Possui todos os principais recursos de programação da linguagem ladder e também permite a programação usando qualquer uma das cinco linguagens de programação da norma IEC 61131-3. Seu uso não é tão intuitivo quanto outros, mas é uma ferramenta completa e depois que se aprende, torna-se fácil trabalhar com ele. É capaz de rodar tanto em sistema operacional Microsoft Windows, como também em Linux e MacOS. O OpenPLC pode ser baixado e instalado através do endereço: https://autonomylogic.com/download/

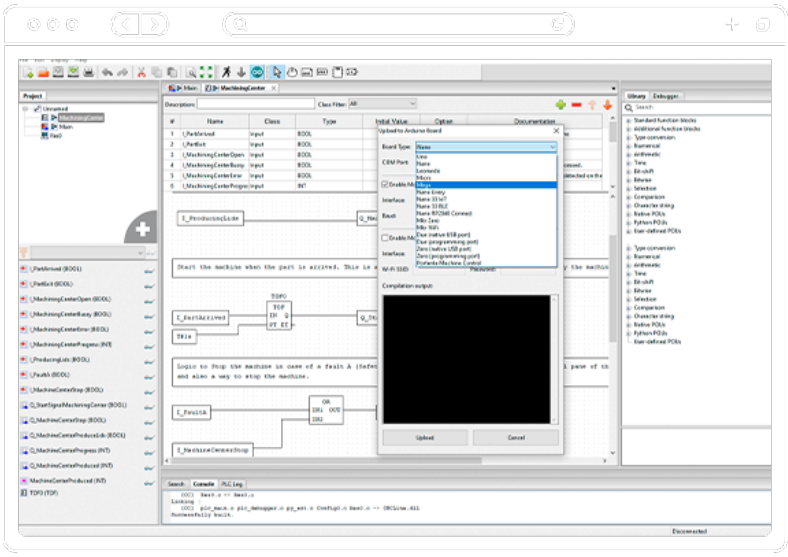

Figura 69 – Simulador OpenPLC Editor [15].

WinProLadder (Figura 70): é um software de programação dos CLP Fatek e que também serve para simulação em linguagem Ladder, para que se possa testar as lógicas sem ter nenhum CLP. Assim como a maioria dos simuladores, o WinProLadder possui todos os recursos básicos de programação da linguagem ladder. Está disponível apenas para ambiente Microsoft Windows e pode ser baixado e instalado após feito um cadastro através do endereço: https://www.fatek.com/en/download.php

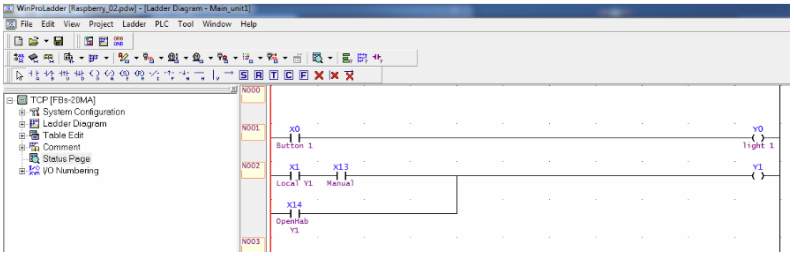

Figura 70 – Simulador WinProLadder [16].

PLC Ladder Simulator 2 (Figura 71): é um aplicativo para programação e simulação de CLP em linguagem Ladder disponível tanto para smartphones Android como

também para sistema Windows (embora neste último o uso não seja tão intuitivo quanto no celular). Perfeitamente adaptado a dispositivos de toque, permite a simulação de botões e lâmpadas com facilidade. Alguns blocos de programação estão bloqueados na versão gratuita e só podem ser acessados após feito o pagamento dentro do aplicativo em celular Android, o que limita bastante seu uso não apenas pela questão do custo, mas também para pessoas que utilizam sistema iOS. Os links para baixar a versão Android e Windows podem ser acessados através do site: https://plcladdersimulator2.weebly.com.

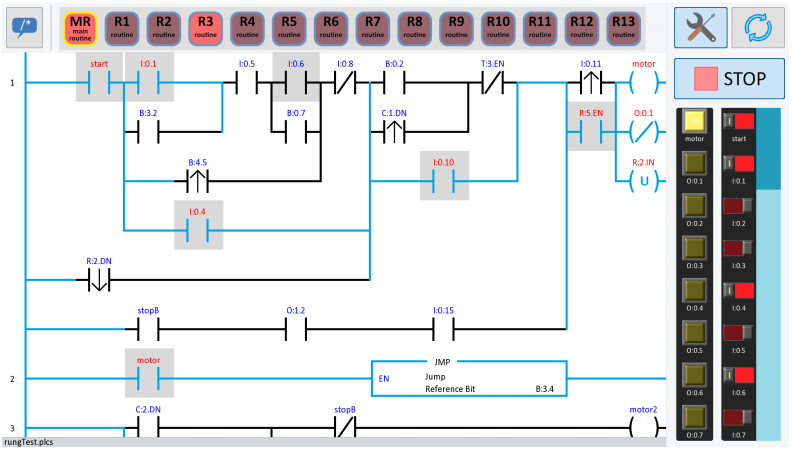

Figura 71 – Simulador PLC Ladder Simulator 2 [17].

LogixPro 500 (Figura 72): é um simulador de CLP instalável em computadores Windows e que permite a simulação de linguagem Ladder, baseado na interface do software Logix500, da Allen Bradley. Sua principal característica é a existência de diversas simulações, permitindo trabalhar não apenas com botões e lâmpadas, mas também com controle de portões, enchimento de tanques, elevadores, etc. Não é um software gratuito, mas é bastante popular devido a variedade de simulações nele presentes. A versão demo pode ser baixada, instalada e testada por 15 dias através do site: https://canadu.com/lp/logixpro.html.

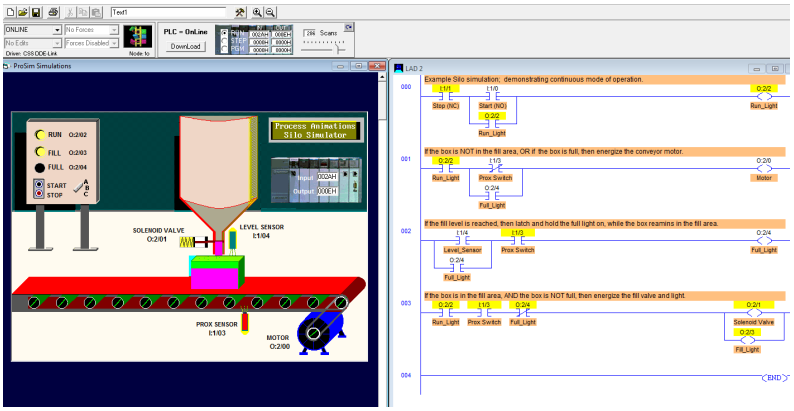

Figura 72 – Simulador RS Logix Pro. Simulação de silo. Fonte: o autor.

Factory I/O (Figura 73): é um simulador de CLP que se destaca por ter uma interface gráfica 3D realista e intuitiva, que permite aos usuários criar e simular ambientes

industriais complexos com uma variedade de equipamentos e maquinários. Dentro do próprio Factory I/O há uma interface de programação em linguagem de blocos (FBD), mas é possível comunicá-lo via protocolo Modbus, OPC, dentre outros, a outros softwares de programação de CLP. O Factory I/O é um software pago e funciona apenas em ambiente Windows. Uma versão demo de 30 dias pode ser baixada e instalada através do site: https://factoryio.com.

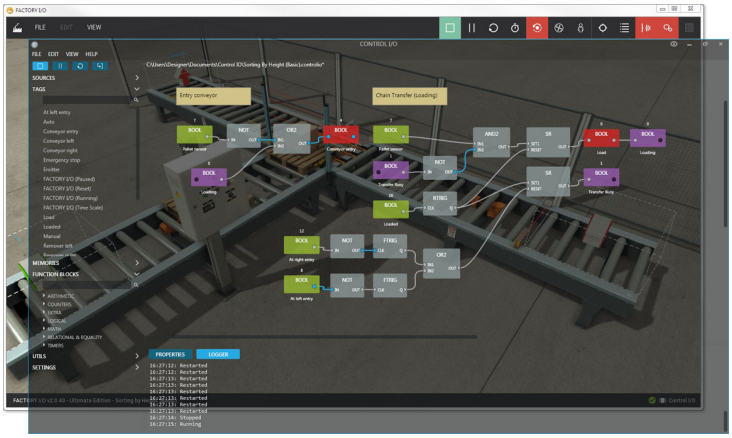

Figura 73 – Simulador Factory I/O. Programação na frente, simulação atrás [18].

5 O Simulador desenvolvido

5.1 Inspiração

A gênese do simulador de CLP desenvolvido neste trabalho encontra-se na intersecção de tecnologia avançada e a necessidade premente de ferramentas educacionais interativas e integradas. Inspirado pela evolução contínua das tecnologias de automação industrial, nota-se uma lacuna significativa no domínio da educação em engenharia: a falta de uma plataforma de simulação de CLP que fosse tanto intuitiva quanto abrangente, capaz de oferecer uma experiência prática e realista aos aprendizes.

A primeira centelha de inspiração veio da observação das limitações presentes nas ferramentas educacionais existentes. Muitas delas não conseguem replicar com precisão as complexidades e nuances dos sistemas de automação modernos, deixando os estudantes menos preparados para os desafios do mundo real. Como exemplo desse fato, a maioria dos simuladores existentes permite apenas a simulação de botões e lâmpadas, o que embora sirva bem para ensinar os rudimentos básicos de programação, não podem simular interações complexas de sensores e atuadores existentes em processos reais. Além disso, a maioria dessas ferramentas não oferece a flexibilidade e a adaptabilidade necessárias para atender às necessidades variadas dos usuários, já que alguns usuários usam computadores antigos, outros tem apenas smartphones e alguns sequer sabem instalar um programa no computador.

Diante dessa constatação surgiu a oportunidade de criar uma ferramenta que não apenas preenchesse essas lacunas, mas também elevasse o ensino de programação de CLP a um novo patamar. Toda essa jornada começou com a integração de simulações diversas e interativas, que vão além dos tradicionais botões e lâmpadas, incorporando elementos como tanques, cilindros, semáforos, dentre outros, proporcionando uma experiência mais rica e engajadora. Nesse ponto, o simulador LogixPro, já relatado anteriormente, serviu como inspiração inicial.

A inclusão de uma interface amigável de programação em linguagem ladder foi um passo crucial nesta jornada. Ao oferecer blocos básicos da linguagem, como contatos, bobinas, bobinas retentivas, temporizadores, contadores, comparadores e blocos aritméticos, o simulador facilita a compreensão e a aplicação dos conceitos de programação de CLP, tornando o aprendizado mais acessível e eficaz.

Além disso, a capacidade de criar novas simulações através de linguagens de programação modernas, como JavaScript, e a utilização de tecnologias como Canvas e SVG, permite uma maior flexibilidade e personalização, promovendo a inovação e a experimen-

tação.

A visão futura para este projeto é grandiosa, com planos de integrar funcionalidades como comunicação via OPC ou Modbus, facilitando a interação com outros softwares de programação e sistemas supervisórios SCADA (*Supervisory Control And Data Acquisition*). Além disso, a possibilidade de integrar a plataforma a tutoriais interativos on-line pode oferecer uma experiência de aprendizado mais dinâmica e colaborativa.

Em suma, a inspiração para a criação deste simulador de CLP nasceu da vontade de superar as limitações das ferramentas educacionais existentes e de oferecer uma solução mais integrada, interativa e inovadora. Através da combinação de tecnologia de ponta e uma abordagem centrada no usuário, este projeto promete não apenas transformar o ensino de programação de CLP, mas também contribuir significativamente para a formação de profissionais mais competentes e preparados para os desafios do século XXI.

5.2 Como foi feito: implementação técnica

5.2.1 Estrutura geral do simulador

A implementação técnica deste simulador é caracterizada pelo uso integrado de tecnologias web modernas, que não apenas facilitam uma experiência de aprendizado interativo, mas também promovem uma abordagem pedagógica mais inclusiva e adaptável.

A espinha dorsal da implementação técnica reside em uma tríade de tecnologias web fundamentais: HTML, CSS e JavaScript. O HTML (*HyperText Markup Language*) serve como a estrutura básica, proporcionando a estruturação dos elementos da página web, enquanto o CSS (*Cascading Style Sheet*) é empregado para estilizar e apresentar o conteúdo de maneira visualmente agradável e funcional. O JavaScript, uma linguagem de programação de alto nível, desempenha um papel crucial, facilitando a criação de simulações interativas e dinâmicas, que são vitais para uma experiência de aprendizado engajadora.

Desde o início a ideia do simulador era dividir a tela em duas áreas, conforme pode ser visto na Figura 74: a área da programação e a área da simulação. Na área de programação o usuário montaria o programa com toda a liberdade, inserindo os elementos da linguagem ladder e conectando-os para fazer a lógica desejada. Já na área de simulação seria visualizado os efeitos da programação em um cenário simulado: botões e lâmpadas, tanques, cilindros, portões, etc. Várias simulações podem existir na área de simulação.

Para fazer a área de programação contou-se com o auxílio de uma biblioteca JavaScript chamada P5.js. Esta biblioteca, conhecida por sua acessibilidade e flexibilidade, permite a criação de simulações gráficas interativas de maneira relativamente simples, utilizando o elemento Canvas do HTML. Com o Canvas é possível não apenas desenhar

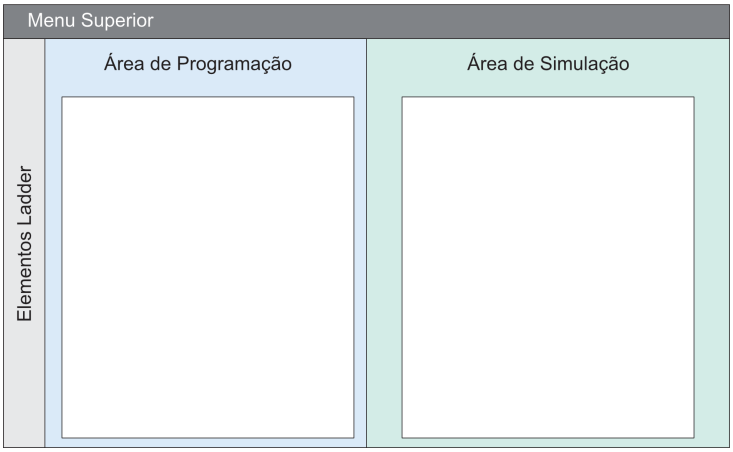

Figura 74 – Estrutura idealizada do simulador. Fonte: O autor.

elementos na tela, mas também interagir com eles pelo teclado, mouse ou toque em dispositivos móveis. Portanto, a área de programação foi feita no Canvas do HTML com o auxílio da biblioteca P5.js.

Já na área de simulação, as simulações podem ser feitas de diversas maneiras: usando Canvas do HTML, usando SVGs (Scalable Vector Graphics) animados/interativos com Javascript, etc. A área da simulação existe dentro de um iframe, isolada da área de programação, mas comunicando-se com ela várias vezes por segundo através de uma função de comunicação no JavaScript. Assim, quando um botão é pressionado na área de simulação, por exemplo, a página da programação recebe essa informação, que a processa e insere dentro da lógica do programa criado pelo usuário.

A tela inicial do simulador em computadores pode ser visto na Figura 75, enquanto que para celulares pode ser vista na Figura 76.

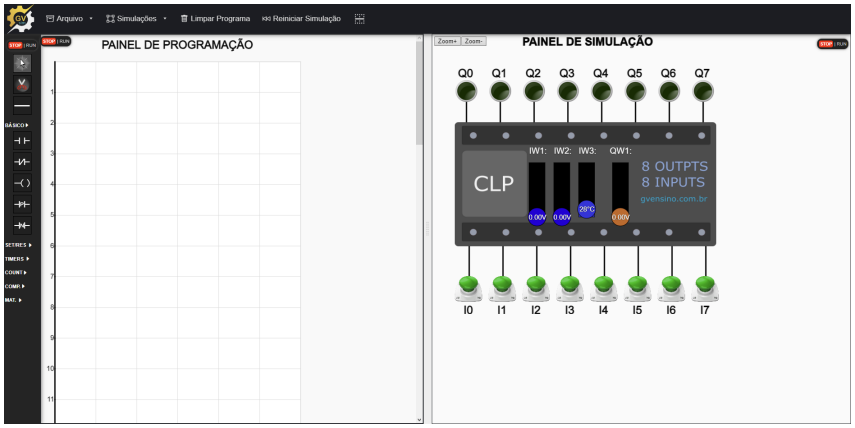

Figura 75 – Tela inicial do simulador em computadores. Fonte: O autor.

A estrutura geral da página foi feita utilizando o modelo de layout CSS chamado FlexBox, onde foi separado o site em três partes gerais:

- Topo (com menu superior);

Figura 76 – Tela inicial do simulador em smartphones. Fonte: O autor.

- Painel de programação (à esquerda ou acima);
- Painel de Simulação (à direita ou abaixo).

5.2.1.1 Divisão entre os painéis

A divisão entre o Painel de Programação e o Painel de Simulação é uma área redimensionável, que pode ser ajustada livremente pelo usuário tanto com mouse quanto com dispositivos de toque (em celulares ou tablets) e pode ser vista na Figura 77.

Figura 77 – Divisão de tamanho ajustável entre os painéis. Fonte: O autor.

Para criação desses painéis ajustáveis foi utilizada uma biblioteca JavaScript chamada Split.js[60], que já trás diversos recursos prontos para a realização dessa função tanto em computadores de mesa quanto em dispositivos de toque.

Através de um botão no menu, os usuários podem alternar a organização dos painéis entre direita/esquerda e cima/baixo, conforme for mais conveniente para eles no momento da programação, como pode ser visto na Figura 78.

Figura 78 – Orientações dos painéis possíveis: direita/esquerda ou cima/baixo. Fonte: O autor.

Além disso, graças ao fato dos painéis serem redimensionáveis, usuários que estão trabalhando com telas pequenas podem facilmente mover a divisão dos painéis toda para esquerda ou toda para direita, deixando visível apenas um dos painéis de acordo com o que estiver sendo feito no momento. Essa possibilidade é especialmente útil em dispositivos móveis de tela pequena, como os celulares.

5.2.1.2 Menu superior

O menu superior foi feito em HTML, estilizado em CSS e seu funcionamento de abrir e fechar programado em JavaScript, sem o uso de nenhuma biblioteca auxiliar. O menu pode ser visto na Figura 79.

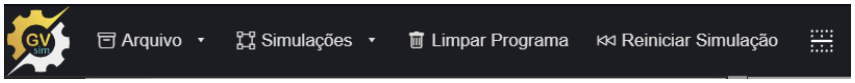

Figura 79 – Menu Superior do simulador. Fonte: O autor.

O Menu Arquivo contém as opções de Abrir um projeto salvo no computador do usuário, salvar o projeto existente no computador ou sobre, onde é possível saber mais sobre o simulador.

Já o menu Simulações contém as diversas simulações existentes, tais como Botões e Lâmpadas, Tanque, Cilindros, Casa Elétricas, Ligações, etc.

A opção Limpar Programa limpa todo o painel de programação, excluindo a programação feita pelo usuário. Já a opção Reiniciar Simulação limpa todo o painel de simulação.

Por fim, o botão com o ícone de uma divisão é para alternar a orientação dos painéis entre direita/esquerda e cima/baixo.

5.2.1.3 Painel de Programação

O painel de programação é onde o usuário irá montar livremente a programação que desejar. Esse painel é dividido basicamente em duas partes: à esquerda, existem os botões com os elementos de programação; à direita, o grid, onde os elementos são posicionados e interligados para se montar o programa. Uma imagem mostrando essa organização pode ser vista na Figura 80.

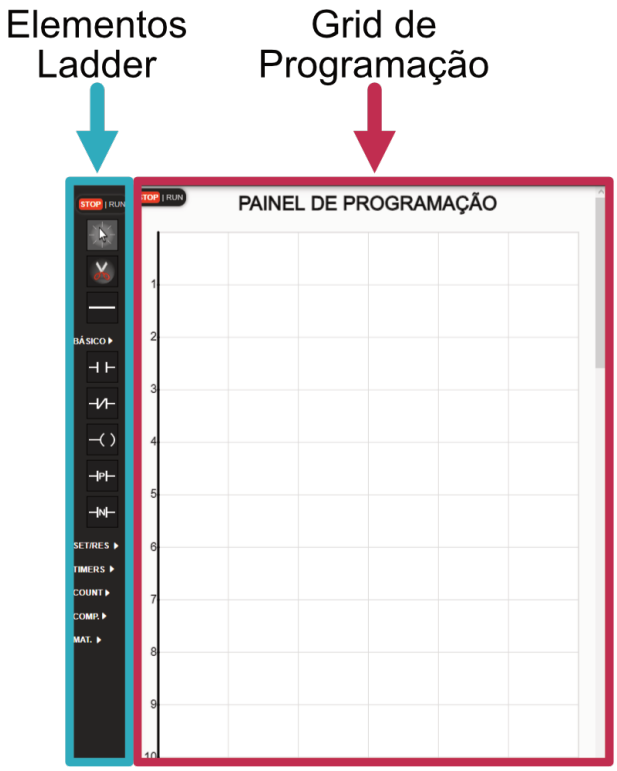

Figura 80 – Divisões do Painel de Programação. Fonte: O autor.

Na área de elementos Ladder existem os seguintes itens:

- Mouse - para desabilitar qualquer outra função;
- Tesoura - para excluir elementos do grid;
- Traço - para fazer ligações no grid.

Dentro do sub-menu Básico é possível inserir os seguintes itens no Grid:

- Contato Normalmente Aberto;
- Contato Normalmente Fechado;
- Bobina;
- Contato de Transição Positiva;

- Contato de Transição Negativa;

Dentro do sub-menu Set/Res é possível inserir os seguintes itens no Grid:

- Bobina Retentiva Set;
- Bobina Retentiva Reset.

Dentro do sub-menu Timers é possível inserir os seguintes itens no Grid:

- Temporizador TON;
- Temporizador TOF;
- Temporizador TP;
- Temporizador RTO.

Dentro do sub-menu Counters é possível inserir os seguintes itens no Grid:

- Contador Crescente CTU;
- Contador Decrescente CTD.

Dentro do sub-menu Comp. (Comparadores) é possível inserir os seguintes itens no Grid:

- Comparador de Igualdade EQ;
- Comparador de Diferença NEQ;
- Comparador de Maior GT;
- Comparador de Maior ou Igual GE;
- Comparador de Menor LT;
- Comparador de Menor ou Igual GE;

Dentro do sub-menu Mat. (Operadores Matemáticos) é possível inserir os seguintes itens no Grid:

- Bloco de movimentação MOVE;
- Bloco de Soma ADD;
- Bloco de Subtração SUB;

- Bloco de Multiplicação MUL;
- Bloco de Divisão DIV;
- Bloco de Resto de Divisão MOD;
- Bloco de Módulo de número / Número Absoluto;
- Bloco de Raíz Quadrada SQRT;
- Bloco de Limite LIMIT (em processo de implementação);
- Bloco de Escala de valores SCALE (em processo de implementação);

Para inserir qualquer um dos elementos acima no grid, basta clicar no botão relativo ao elemento, ir até a posição onde se deseja inserir o elemento no grid e clicar novamente. Pronto, automaticamente o elemento selecionado será inserido na posição desejada, conforme mostra a Figura 81.

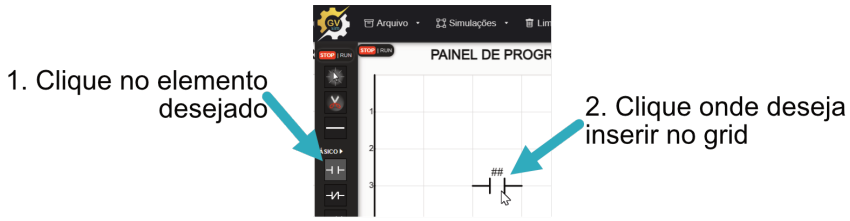

Figura 81 – Inserção de elementos no grid. Fonte: O autor.

Além disso, no painel de programação existe um botão para colocar o CLP em RUN (rodando a simulação) ou STOP (sem rodar a simulação). Em modo RUN a simulação é executada e não é possível realizar alterações na programação.

Quanto ao grid, inicialmente ele vem com 44 linhas e 6 colunas. O número de linhas pode ser aumentado, mas o número de colunas não. No futuro pretende-se implementar uma opção onde o usuário pode alterar o número de colunas através de uma tela de configuração.

5.2.1.4 Painel de Simulação

Já o painel de simulação é variado, já que depende da simulação aberta poderão haver mais ou menos opções. No painel de simulação também é possível dar RUN ou STOP no PLC. Algumas simulações permitem aumentar ou diminuir o zoom com o scroll do mouse ou botões de zoom. As simulações existentes no simulador serão discutidas na próxima seção.

5.2.2 Estrutura da programação

Todo o simulador foi feito utilizando tecnologias WEB, ou seja, HTML, CSS e JavaScript. A estrutura geral do simulador pode ser dividida em duas partes: o núcleo do simulador, que onde fica o painel de programação e onde a lógica da programação é executada; e o painel de simulação, que é onde ficam as simulações e cada simulação tem sua implementação própria.

A estrutura do núcleo do simulador e sua explicação pode ser visto no Apêndice B.1. Já a estrutura geral das simulações e sua explicação pode ser vista no Apêndice B.2

5.3 Simulações existentes dentro do simulador

As simulações existentes no simulador podem variar de acordo com a versão e o momento em que for feito o acesso. No momento da escrita dessa dissertação existem as seguintes simulações disponíveis e funcionais:

5.3.1 Simulação Básica de Botões e Lâmpadas

A simulação básica de botões e lâmpadas é a simulação mais básica e geral do simulador. Ela conta com a imagem de um CLP com 8 botões interligados em 8 entradas (I0 a I7), 8 lâmpadas ligadas a 8 saídas (Q0 a Q7), além de 3 entradas analógicas controladas por controles deslizantes (IW1, IW2 e IW3) e uma saída analógica 0-10V (QW1), conforme a Figura 82.

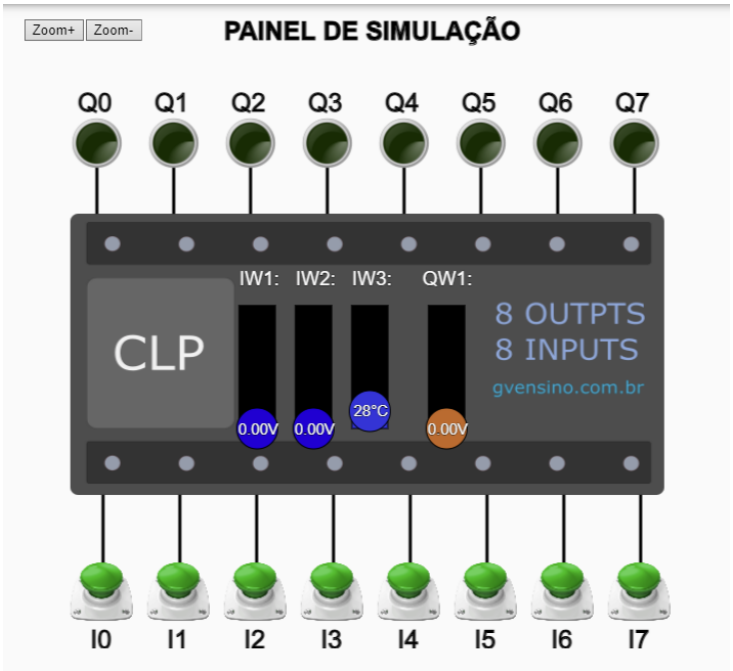

Figura 82 – Simulação Básica de Botões e Lâmpadas. Fonte: O autor.

Nessa simulação é possível pressionar os botões de diversas formas:

- O pressionamento simples com o botão do mouse faz o botão ser acionado; ao soltar o mouse, o botão volta e é solto;
- Ao pressionar o botão segurando Shift, o botão permanece pressionado mesmo ao soltar o mouse: ótimo para simular o pressionamento simultâneo de vários botões;
- Ao pressionar o botão com o botão direito do mouse, o botão troca de estado e vira um botão Normalmente Fechado no campo. Agora, esse botão funcionará de maneira invertida: ao clicar nele, irá retirar o sinal. Essa função também pode servir para manter o botão pressionado, substituindo o pressionamento da tecla Shift vista no item acima.

Já as lâmpadas podem ter suas cores trocadas entre Verde, Amarelo, Vermelho e Azul, conforme clica-se com o botão direito sobre elas, como é exibido na Figura 83.

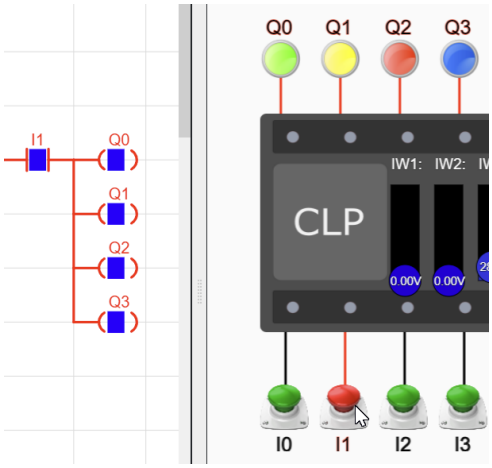

Figura 83 – Lógica simples mostrando cores das lâmpadas. Fonte: O autor.

5.3.2 Segunda Simulação Básica de Botões e Lâmpadas

O simulador também conta com uma segunda simulação de botões e lâmpadas (Figura 84), dessa vez totalmente adaptada para uso em dispositivos móveis, similar à simulação já mostrada. Ela também funciona perfeitamente em computadores de mesa.

Essa simulação conta com a imagem de um CLP com 6 botões interligados em 6 entradas digitais (I0 a I5), 6 lâmpadas ligadas a 6 saídas digitais (Q0 a Q5), além de 2 entradas analógicas controladas por controles deslizantes (IW1 e IW2) e uma saída analógica 0-10V (QW1).

Nessa simulação é possível configurar, através do botão de Configurações na parte superior, qual será o comportamento dos botões, conforme se vê na Figura 85.

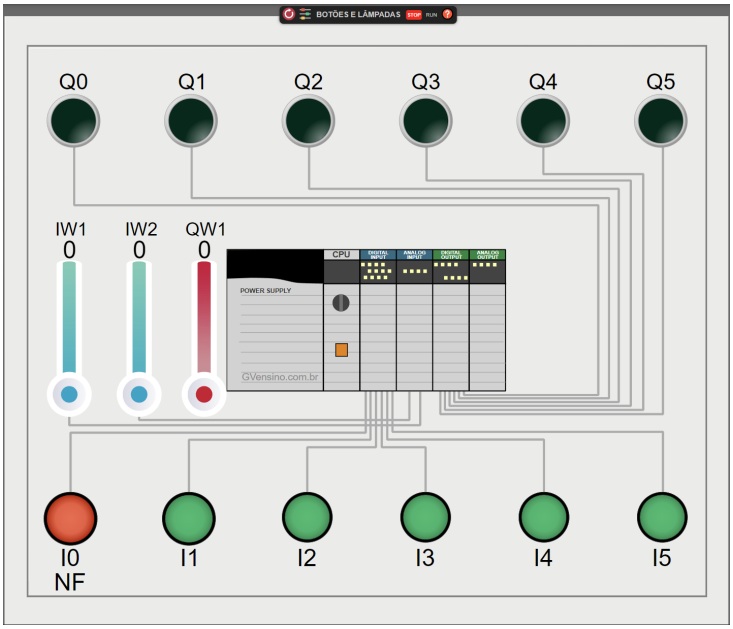

Figura 84 – Segunda simulação de Botões e Lâmpadas

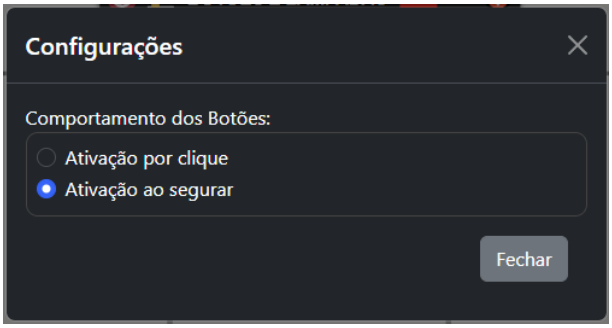

Figura 85 – Configurando comportamento dos botões. Fonte: o autor.

Um exemplo de programação de um pisca-pisca nessa simulação pode ser visto e acessado através do Apêndice A.1.

5.3.3 Simulação do Portão

Esta simulação (exibida na Figura 86) é a primeira que contém elementos animados e que simula uma aplicação real dos CLP. O estudante deve fazer o controle de abertura e fechamento de um portão automático, tendo de parar o portão quando chega em seus limites (totalmente aberto ou fechado) e evitando que o portão colida com um carro, caso esteja parado abaixo dele.

Caso a programação seja feita sem considerar os sensores fim de curso I4 e I5, que indicam que o portão já está aberto ou fechado, o motor do portão irá pegar fogo, indicando que há um erro crítico na programação. A Figura 87 ilustra essa situação.

Na Figura 88 pode-se ver e simular um exemplo de uma das possíveis programações para o portão.

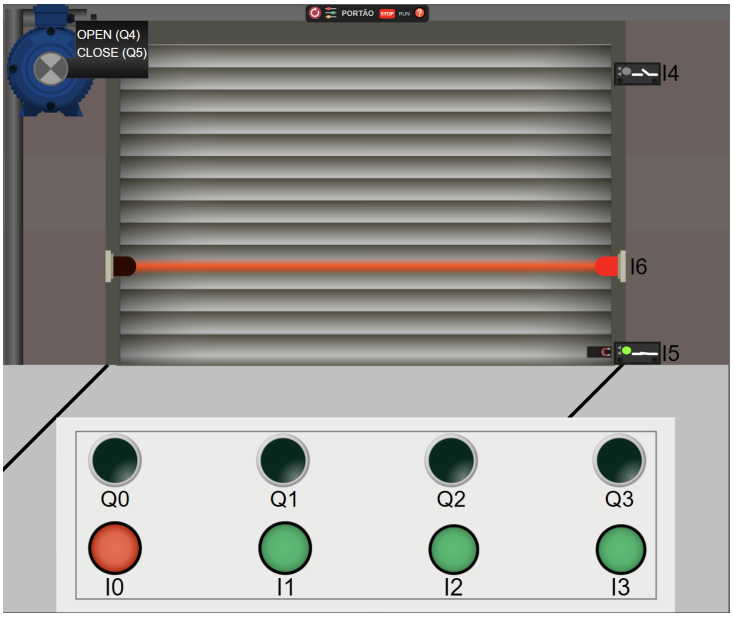

Figura 86 – Simulação do Portão. Fonte: o autor.

Figura 87 – Motor pegando fogo devido a erro na programação. O portão terminou de abrir mas o motor não foi desligado. Fonte: o autor.

Na programação vista na Figura 88, o botão presente em I1 liga e mantém o motor do portão ligado (Q4), abrindo-o. Quando o fim de curso I4 é acionado, indicando que o portão já está aberto, o motor é desligado. I2 ou I0 também param a abertura do portão.

Ao energizar botão presente em I2, é acionado o fechamento do portão, ligando o motor no sentido reverso através da saída Q5. A portão continua fechando até que aciona o fim de curso inferior, que aciona a entrada I5, desligando o motor.

5.3.4 Simulação dos Semáforos

Esta simulação (exibida na Figura 89) contém um cruzamento de ruas onde carros podem passar de um lado para o outro, sendo controlados por dois semáforos. Os carros

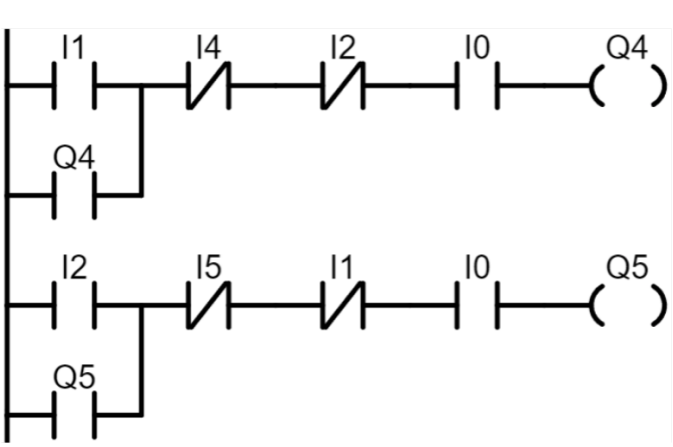

Figura 88 – Exemplo programação para o portão. Disponível em <https://gvensino.com.br/exemploportao>. Fonte: o autor.

andam sozinhos dependendo do semáforo que estiver aceso em sua rua: se verde ou amarelo, andam; se vermelho, param. O aluno deve programar as lâmpadas dos semáforos no CLP para alternar a liberação do tráfego nos dois cruzamentos em intervalos de tempos regulares, impedindo que haja colisão dos veículos, que pode ser vista na Figura 90.

Esta é uma ótima simulação para o ensino de temporizadores e de lógica sequencial.

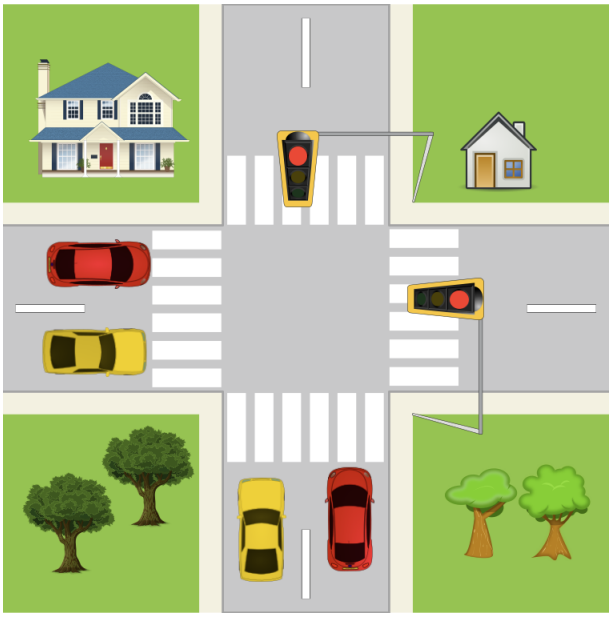

Figura 89 – Simulação dos Semáforos - carros parados. Fonte: O autor.

Um exemplo de programação completa da simulação do semáforo pode ser visto e acessado através do Apêndice A.2.

5.3.5 Simulação do Tanque

A simulação do tanque (Figura 91) permite a programação do enchimento e esvaziamento de um tanque, fazendo o controle do nível do mesmo; também conta com um

Figura 90 – Simulação dos Semáforos - carros colidindo devido a programação errada. Fonte: O autor.

aquecedor capaz de esquentar a temperatura da substância do tanque e um sensor de temperatura, para que o controle de temperatura também possa ser feito.

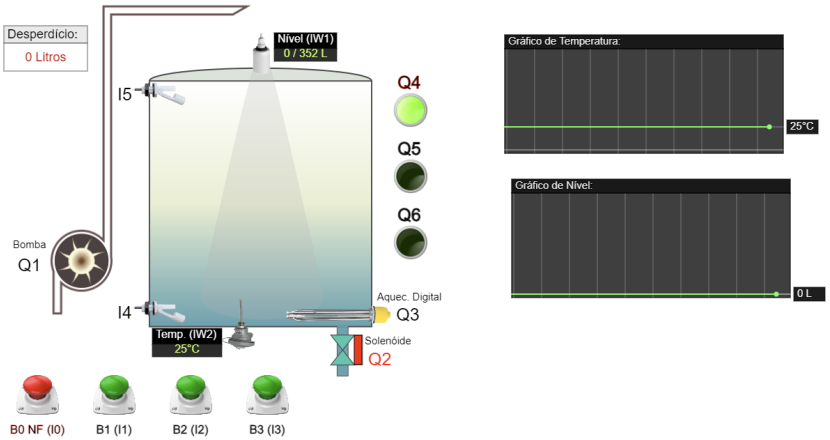

Figura 91 – Simulação do Tanque. Fonte: O autor.

A simulação também conta com botões, lâmpadas e gráficos animados mostrando a alteração da temperatura e do nível ao longo do tempo.

5.3.5.1 Controle de nível do tanque

O controle do nível do tanque é composto por uma Bomba (saída Q1) capaz de encher o tanque; uma válvula solenóide (Q2), capaz de escoar a substância para fora do tanque; e três sensores para detecção de nível, sendo um sensor do tipo chave boia de nível inferior (I4), outro sensor do tipo chave boia de nível superior (I5) e um transmissor de nível analógico (IW1). A Figura 92 mostra o tanque enchendo.

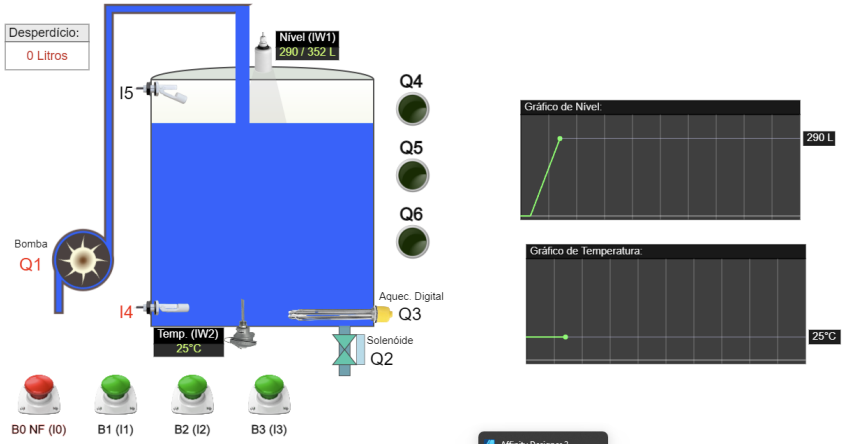

Figura 92 – Tanque sendo cheio pela bomba. Fonte: O autor.

Caso a bomba não desligue após o tanque já estar cheio, o líquido começa a transbordar (Figura 93), o cenário começa a se inundar e então é apresentado um erro no simulador (Figura 94).

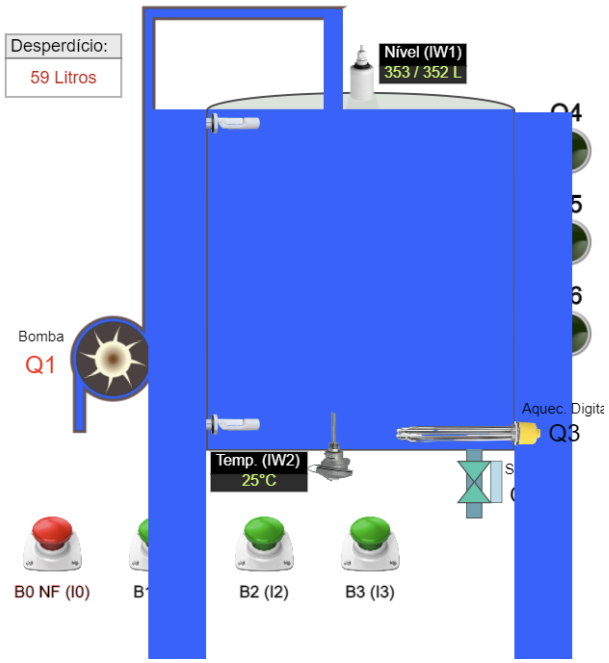

Figura 93 – Tanque transbordando devido a erro na programação. Fonte: O autor.

O medidor de Desperdício mostra quantos litros da substância foram perdidos no transbordamento.

Esta simulação do tanque também pode ser utilizada manualmente para se demonstrar o funcionamento dos elementos separadamente: ao clicar na bomba, ela é acionada; ao clicar na válvula solenoide, ela também é acionada. Ao clicar em cada uma das chaves boia, idem.

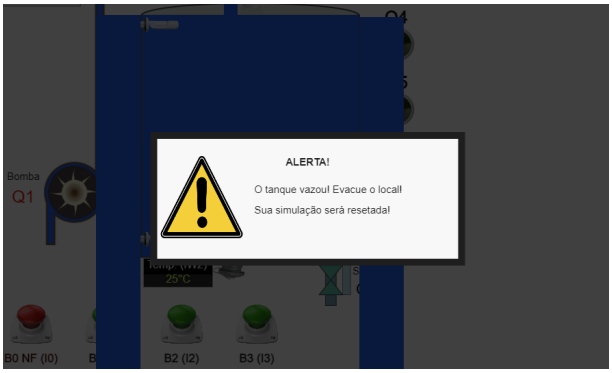

Figura 94 – Mensagem de erro após tanque transbordar. Fonte: O autor.

5.3.5.2 Controle da temperatura do tanque

O controle da temperatura do tanque é composto por um Aquecedor Digital (Q3) que pode também ser alterado para analógico através do menu de Configurações da simulação. Além do aquecedor, existe também um transmissor de temperatura analógico (IW2), que faz a leitura da temperatura do fluído dentro do tanque.

A ideia do controle de temperatura é elevar a temperatura do fluído dentro do tanque para um valor pré-determinado pelo usuário e lá se manter, fazendo o aluno ter de elaborar essa lógica, como pode ser visto, por exemplo, na Figura 95.

Aqui existem também algumas situações de falha possíveis: se o aluno mandar ligar a resistência sem que esta esteja mergulhada no fluído, ela vai ficando vermelha e queima após um certo tempo, representando um comportamento realista da situação, conforme é mostrado na Figura 96.

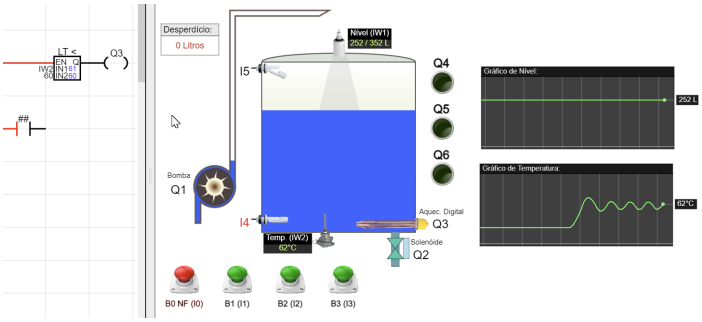

Figura 95 – Exemplo de controle On/Off da temperatura do tanque em 60 graus. Fonte: O autor.

5.3.6 Simulação dos Cilindros

Esta simulação, que pode ser vista na Figura 97 conta com 6 cilindros pneumáticos que podem ser configurados individualmente em três tipos distintos:

- Simples Ação com válvula 3/2 vias;

Figura 96 – Demonstrando a resistência queimada. Fonte: O autor.

- Dupla Ação com válvula 5/2 vias;
- Dupla Ação com válvula 5/3 vias.

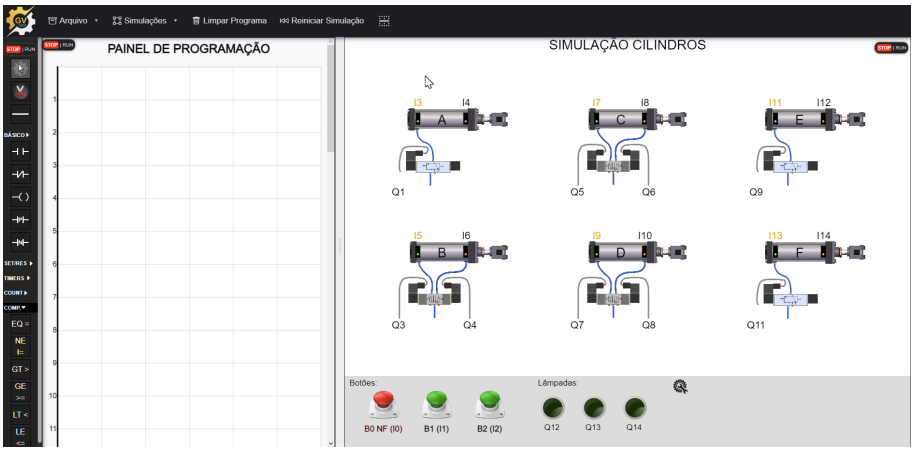

Figura 97 – Simulação dos Cilindros. Fonte: O autor.

A janela de configuração dos cilindros pode ser vista na Figura 98.

Figura 98 – Configuração dos Cilindros. Fonte: O autor.

Os cilindros possuem sensores do tipo magnético acoplados em seus êmbolos, enviando sinais de quando estão avançados ou recuados.

Essa simulação também conta com alguns botões e lâmpadas, permitindo que diversas lógicas possam ser criadas. A simulação dos cilindros é ótima para ser utilizada

em exercícios de lógica sequenciais, onde os cilindros devem ser acionados e reacionados em sequências variadas, o que geralmente é um desafio para alunos em estágio intermediário de programação em linguagem ladder.

Através da roda do mouse é possível aumentar ou diminuir o zoom da tela, permitindo posicionar o painel de simulação conforme for mais confortável e conveniente ao aluno.

5.3.7 Simulação da Casa Inteligente

A simulação da casa inteligente (Figura 99) simula uma casa com diversos equipamentos elétricos e eletrônicos podendo ser ligados ou desligados. É possível monitorar o consumo individual de cada aparelho ao colocar o mouse sobre ele e também o consumo total da casa e o valor financeiro que está sendo gasto em tempo real, de acordo com o consumo dos equipamentos ligados.

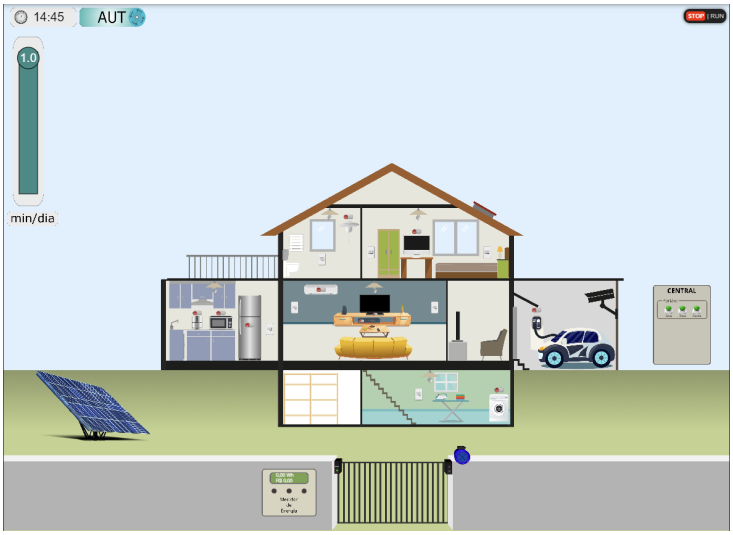

Figura 99 – Simulação Casa Inteligente. Fonte: O autor.

É possível navegar até os cômodos da casa, acender ou apagar as luzes, ligar ou desligar TV, Ar-condicionado, máquina de lavar, entre outros eletrodomésticos. Existe também um sistema de captação de energia solar, capaz de, dependendo da hora do dia, gerar mais ou menos energia, compensando um pouco os gastos com energia de outros equipamentos. O consumo em tempo real de cada equipamento pode ser visto ao passar o mouse sobre o aparelho, conforme mostra a Figura 101.

A simulação permite a passagem controlada do tempo, onde pode-se fazer o tempo passar mais rápido ou devagar. O cenário escurece quando atinge o período noturno, como se pode ver na Figura 100.

Essa simulação possui os modos automático e manual. No modo automático, todo o funcionamento da casa é feito desconsiderando qualquer programação feita no painel

Figura 100 – Simulação Casa Inteligente - noite. Fonte: O autor.

Figura 101 – Simulação Casa Inteligente - consumo em tempo real. Fonte: O autor.

de programação. Esse modo serve mais para demonstrar aos alunos como o consumo de energia funciona de acordo com os equipamentos ligados ou desligados na casa.

5.3.8 Simulação das ligações elétricas

Essa simulação (Figura 102) contém um CLP e alguns dispositivos de entrada (como botões, sensores capacitivos e indutivos, potenciômetro) e de saída (lâmpadas, relés, motores). O aluno deverá montar todo o circuito exatamente como faria na prática, tendo de alimentar o CLP corretamente, alimentar e fazer as ligações das entradas (botões/sensores ligados corretamente nas entradas do CLP) e saídas (lâmpadas e outros atuadores ligados corretamente nas saídas do CLP).

Através da roda do mouse é possível aumentar ou tirar o zoom e aproximar de áreas específicas da tela. Para se criar uma ligação basta clicar num dos bornes/terminais que o fio começa a surgir. Conforme se posiciona o mouse e clica na tela, o fio vai sendo conduzido, até chegar ao seu terminal de destino.

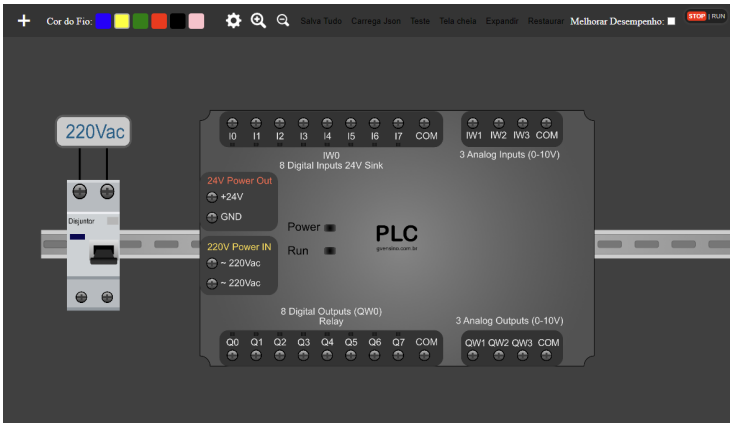

Figura 102 – Simulação das Ligações Elétricas. Fonte: O autor.

Na Figura 103 pode-se ver a alimentação básica de um CLP através de um disjuntor, ligações essas que deve ser realizada pelo aluno.

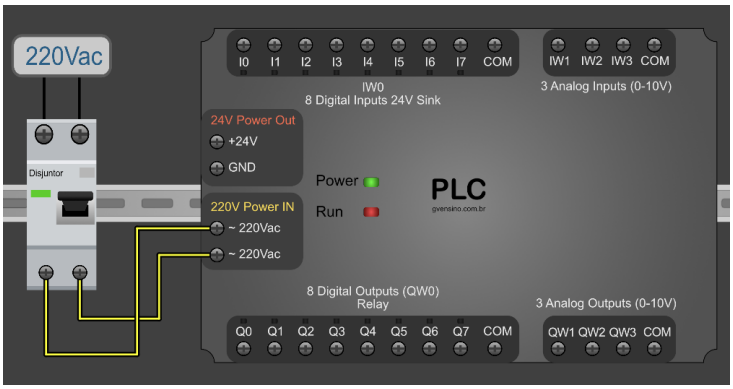

Figura 103 – Alimentando o CLP ao ligar o disjuntor. Fonte: O autor.

Existe um menu (Figura 104) onde o usuário pode inserir quaisquer componentes na tela, posicioná-los livremente e então fazer as ligações.

Os componentes disponíveis nessa simulação até a data de publicação deste documento são:

Componentes de Ligação:

- Barramento Horizontal
- Barramento Vertical

Chaves/Sensores:

- Botão NA/NF Verde;
- Botão NA/NF Vermelho;
- Sensor Capacitivo NPN

Figura 104 – Menu com alguns dos componentes disponíveis. Fonte: O autor.

- Sensor Capacitivo PNP
- Sensor Indutivo NPN
- Sensor Indutivo PNP

Atuadores

- Lâmpada 24V Verde;
- Lâmpada 24V Vermelha;
- Relé 24V;
- Motor (em estágio final de desenvolvimento).

Em Analógicos existe um Potenciômetro, que pode ser utilizado em entradas analógicas do CLP. E em Medidor existe um Voltímetro, que pode ser usado tanto para medir o nível de tensão de uma saída analógica, quanto também para se medir a tensão em outros pontos do circuito montado.

Caso o usuário faça alguma ligação que resulte em um curto-circuito, o disjuntor é automaticamente desarmado, permitindo que o aluno veja que há algum erro e faça a sua correção antes de uma nova tentativa.

A Figura 105 demonstra a simulação das ligações elétricas, onde um circuito com dois botões, um sensor indutivo e uma lâmpada são interligados ao CLP.

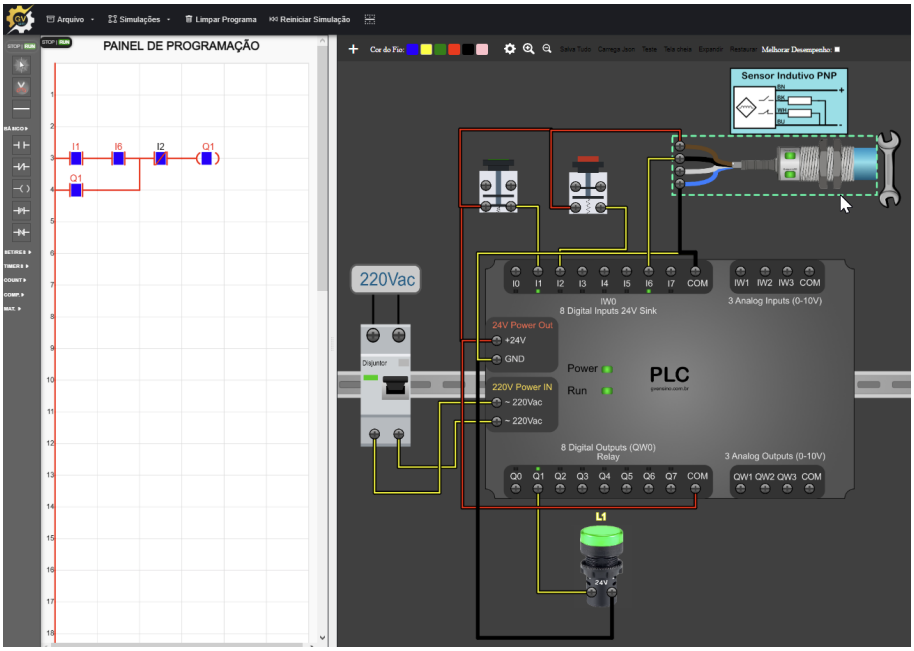

Figura 105 – Circuito simples demonstrando a simulação. Fonte: O autor.

5.4 Possibilidades de uso no ensino técnico

Dentro do ensino técnico de programação de CLP, o simulador pode ser empregado na sequência sugerida a seguir:

1. Simulação Botões e Lâmpadas 1 e 2: pode ser usada no início do ensino, quando se está aprendendo os fundamentos básicos da programação ladder, como Contato N.A., N.F, bobinas e selos;

2. Simulação do Portão: pode ser usada para aprimorar os conhecimentos sobre retenção de saídas (selo ou set/reset) e algumas lógicas mais complexas, já que envolve um portão que pode ser danificado;

3. Simulação da Casa Inteligente: pode ser usada para aprimorar os conhecimentos gerais sobre ligar e desligar saídas, programação de horários e outras lógicas similares;

4. Simulação do Tanque: pode ser usada para elaborar controle de nível e de temperatura, depois que os alunos já estão dominando os fundamentos básicos da programação ladder;

5. Simulação do Semáforo: pode ser usada para exercícios mais avançados envolvendo temporização e sequenciamento;

6. Simulação dos Cilindros: pode ser usada para exercícios envolvendo Contadores ou lógicas sequenciais mais complexas;

7. Simulação de Ligações Elétricas: pode ser usada para mostrar aos alunos como fazer as ligações das entradas e saídas dos CLP com sensores e atuadores, ensinando-os o aspecto prático da instalação de um CLP;

5.5 Possibilidades de uso no Ensino Médio e Divulgação Científica

O simulador de CLP pode ser empregado de maneira positiva em várias áreas do ensino médio, especialmente aquelas que envolvem ciência, tecnologia, engenharia e matemática (STEM). Aqui estão algumas sugestões:

Física: O simulador pode ser usado para demonstrar conceitos de física, como eletricidade e magnetismo. Os alunos podem usar o simulador para criar circuitos e observar como a corrente flui através deles. A simulação de cilindros pode ser usada para demonstrar aplicações envolvendo força e pressão. A simulação de nível pode ser usada para demonstrar conceitos envolvidos na temperatura e na propagação de ondas/sons (já que essa simulação utiliza um sensor ultrasônico para medir o nível do tanque).

Raciocínio Lógico: A programação de CLP envolve muitos conceitos lógicos e de análise. O simulador pode ser usado para reforçar esses conceitos e mostrar aos alunos como eles são aplicados no mundo real.

Tecnologia da Informação: O simulador pode ser usado para ensinar os alunos sobre programação e controle de sistemas. Pode-se ensinar a criar animações ou mesmo simulações simples usando linguagens de programações acessíveis (como p5.js).

Ciências em geral: Em aulas de ciências, o simulador pode ser usado para demonstrar como os sistemas de controle são usados em experimentos científicos e na coleta de dados. Por exemplo, a simulação de um cruzamento de trânsito pode mostrar como a automação está presente e tem grande impacto até mesmo nas nossas ruas sem nem mesmo percebermos. A simulação do tanque pode remeter a processos de distribuição de água e saneamento básico, enquanto o controle de temperatura ali presente pode ser uma forma de apontar aos alunos equipamentos tanto em casa (microondas, fornos, air-fryers) quanto na indústria sobre a importância do controle automático de sistemas de aquecimento. As simulações também podem ser usadas no aspecto ético, para se mostrar as consequências de erros em cálculos ou em programações, dando ênfase à responsabilidade que cada profissional deve ter ao realizar seu trabalho da melhor forma.

5.6 O futuro - melhorias e novos horizontes

O simulador desenvolvido tem diversas características que dão a ele bastante flexibilidade para expansões no futuro:

5.6.1 Novas Simulações, inclusive em 3D

O fato de ter sido desenvolvido em ambiente Web e de haver o desacoplamento entre o painel de programação e as simulações permite que a criação de novas simulações seja simples e não afete o núcleo principal do simulador.

Todas as simulações existentes até agora foram em 2D, mas existe a possibilidade também de inserir simulações em 3D (que ficam um pouco mais pesadas) ou a utilização de outros recursos para fazer simulações em 2D mesmo. Essas novas simulações podem ser feitas utilizando qualquer tecnologia Web existente, tais como:

5.6.1.1 HTML, CSS e JavaScript

Simulações simples, sem muitas animações, podem ser feitas utilizando apenas as tecnologia Web mais puras, HTML, CSS e JavaScript.

5.6.1.2 SVG + JavaScript

O SVG (Scalable Vector Graphics) é um formato de imagens vetoriais (ou seja, montadas matematicamente) que podem ser ampliadas ou reduzidas sem perda de qualidade. Utilizando imagens SVG com JavaScript é possível criar imagens interativas e animações com ótimo desempenho no navegador, sendo esta uma ótima opção para criação de simulações que tenham menos de 100 elementos renderizados na tela.

5.6.1.3 Bibliotecas JavaScript 2D

Existem bibliotecas JavaScript que utilizam o elemento Canvas do HTML para facilitar o desenvolvimento de gráficos e animações interativas 2D. Algumas dessas bibliotecas são:

- P5.js: trás recursos que facilitam o desenho e interação no elemento Canvas e foi utilizado no simulador desenvolvido neste trabalho;
- Pixi.js: mais avançado que o P5.js, também utiliza o Canvas e utiliza aceleração de hardware para criar animações e interações em 2D em páginas web;
- Fabric.js e Konva.js: permitem criar imagens interativas na tela, sendo ótimos para criação de simulações onde o usuário deve posicionar os elementos na tela.

5.6.1.4 Bibliotecas JavaScript 3D

Existem bibliotecas JavaScript que utilizam o elemento Canvas do HTML para facilitar o desenvolvimento de gráficos e animações interativas em três dimensões (3D), contando com aceleração por hardware. Algumas dessas bibliotecas são:

- Babylon.js: biblioteca gratuita e de código aberto que permite a criação direto em JavaScript de elementos 3D, podendo adicionar a eles interação, física, diversos ângulos de visão, dentre vários outros recursos;
- PlayCanvas.js: similar ao Babylon.js, possui também um editor visual 3D que facilita o desenvolvimento; apesar da biblioteca em si do PlayCanvas ser código aberto, o editor 3D não é;
- Three.js: mais simples do que as opções anteriores, o Three.js não conta com motores de física embutidos (embora possam ser acrescentados) e sua grande vantagem é a enorme disponibilidade de materiais e tutoriais disponíveis, que facilitam o desenvolvimento de simulações em 3D;

5.6.2 Interação com outros softwares de Automação

Por usar JavaScript, é possível através de tecnologias como Node.Js, Deno.js ou Bun.js a criação de um servidor local para que o simulador possa se comunicar com outras aplicações, usando protocolos abertos como Modbus ou OPC. Isso inclui outros softwares de CLP, softwares de supervisório, planilhas, etc.

Dentro do Node.js, por exemplo, existem bibliotecas gratuitas e de código aberto que permitem a comunicação do Node com o simulador, através de uma porta HTTP ou HTTPS, e do Node com protocolos de redes abertas, como os já referidos Modbus e OPC, por exemplo. Dessa forma, criam-se várias outras possibilidades para o uso do simulador:

- O uso de um software de programação mais completo, como por exemplo o CodeSys, permitindo que se possa programar as simulações em qualquer uma das cinco linguagens de programação da norma IEC 61131-3;
- O ensino de softwares supervisórios SCADA ou IHM, recebendo e enviando informações diretamente para o simulador;
- A coleta de dados do simulador através de planilhas ou banco de dados.

6 Pesquisa - Avaliação de impacto

Para ter uma métrica sobre o impacto do simulador em ambiente educacional o simulador foi utilizado por algumas turmas de diversas cidades de cursos técnicos do SENAI MG. Ao todo, o simulador foi utilizado por mais de 200 alunos e por pelo menos 8 docentes diferentes.

Para se ter uma ideia da opinião dos usuários, foi feita duas pesquisas distintas: uma com os alunos e outra com os docentes.

6.1 Pesquisa com os Alunos

A pesquisa com os alunos foi feita através de um formulário do Google e encaminhado para os mais de 200 alunos que fizeram a disciplina de Sistemas Lógicos Programáveis do curso Técnico em Automação Industrial pelo SENAI MG. Esses alunos fazem parte do curso técnico EaD (Educação à Distância) e pertencem a várias cidades diferentes.

Foram coletadas, ao todo, 20 respostas voluntárias dos alunos. O título da pesquisa foi: Avaliação da Experiência de utilização do Simulador de CLP.

Objetivo da Pesquisa: Avaliar o impacto do uso do simulador de CLP na aprendizagem de programação de CLP por alunos e docentes.

Participantes: Alunos e docentes envolvidos no ensino e aprendizagem de programação de CLP.

Instrumento de Coleta de Dados: Questionário on-line contendo perguntas relacionadas à experiência dos participantes com o simulador de CLP, a facilidade de uso, a percepção de aprendizagem e a satisfação geral. O questionário pode incluir perguntas de escala Likert, perguntas abertas e perguntas de múltipla escolha.

Procedimento: Os participantes serão convidados a usar o simulador de CLP durante um período de tempo determinado (por exemplo, um semestre). Após esse período, eles serão convidados a preencher o questionário on-line.

Análise dos Dados: As respostas ao questionário serão analisadas usando técnicas estatísticas adequadas para determinar o impacto do uso do simulador de CLP na aprendizagem de programação de CLP.

6.1.1 Perguntas do Questionário para alunos:

Seção 1: Perguntas introdutórias

Pergunta: Qual é o seu nome completo (pergunta opcional)

Pergunta: Como você classifica seu grau de conhecimento em informática?

Respostas possíveis: (nunca havia mexido no computador antes, estou aprendendo agora / Já mexia antes, mas tinha dificuldade / Já mexia antes e tenho facilidade, mas as vezes preciso da ajuda de outras pessoas / Já mexia antes e consigo me virar bem sozinho(a)).

Seção 2: CLP e o simulador

Pergunta: Qual era o seu nível de experiência com a programação de CLP antes de usar o simulador?

Respostas possíveis: (Não conhecia nada / Conhecia pouco / Tinha conhecimentos intermediários / Tinha conhecimentos avançados)

Pergunta: Quantas vezes você usou o simulador de CLP?

Respostas possíveis: (1-2 vezes / 3-5 vezes / Mais de 5 vezes)

Pergunta: Quão fácil ou difícil você achou o uso do simulador de CLP?

Respostas possíveis: (Fácil / Médio / Difícil)

Seção 3: Impacto na Aprendizagem

Pergunta: O uso do simulador de CLP ajudou a melhorar meu entendimento da programação de CLP?

Respostas possíveis: (SIM/NÃO/TALVEZ)

Pergunta: O uso do simulador reforçou os conhecimentos adquiridos nas aulas?

Respostas possíveis: (SIM/NÃO/Não tenho certeza)

Pergunta: O uso do simulador de CLP tornou meu processo de aprendizado de programação mais divertido e dinâmico?

Respostas possíveis: (SIM/NÃO/Não tenho certeza)

Pergunta: As simulações contribuíram para melhorar minha visão sobre as complexidades de um processo de automação real.

Respostas possíveis: (SIM/NÃO/Não tenho certeza)

Seção 4: Feedback e Sugestões

Pergunta: Pra finalizar, descreva aqui suas impressões sobre o simulador, os pontos positivos, negativos e sugestões de melhorias (opcional)

6.1.2 Resultados coletados - pesquisa com alunos

Pergunta 1: Nome completo -> 16 alunos quiseram se identificar na pesquisa, inserindo seu nome completo;

As demais perguntas podem ser vistas nas imagens abaixo, da Figura 106 até a Figura 112:

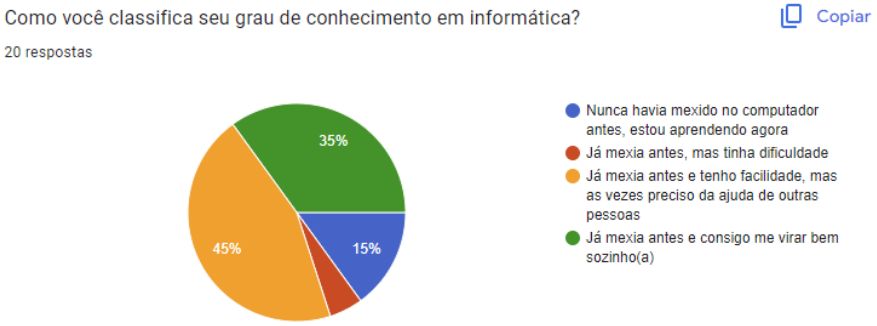

Figura 106 – Questionário alunos - Pergunta 2. Fonte: O autor.

Figura 107 – Questionário alunos - Pergunta 3. Fonte: O autor.

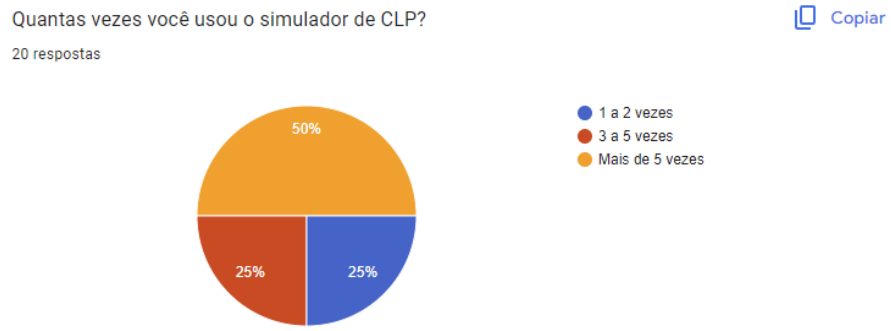

Figura 108 – Questionário alunos - Pergunta 4. Fonte: O autor.

Figura 109 – Questionário alunos - Pergunta 5. Fonte: O autor.

Figura 110 – Questionário alunos - Pergunta 6. Fonte: O autor.

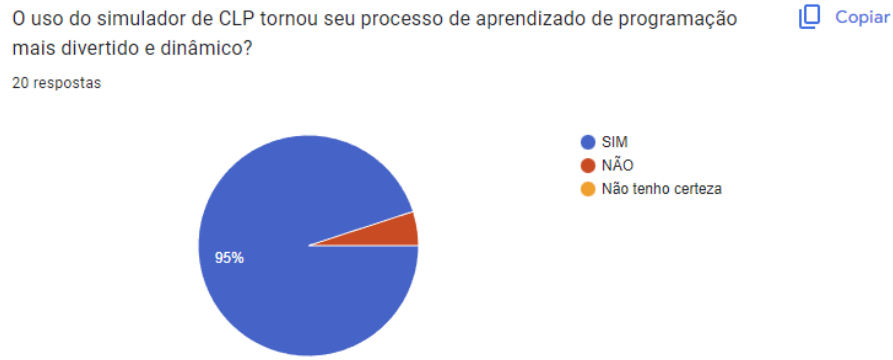

Figura 111 – Questionário alunos - Pergunta 7. Fonte: O autor.

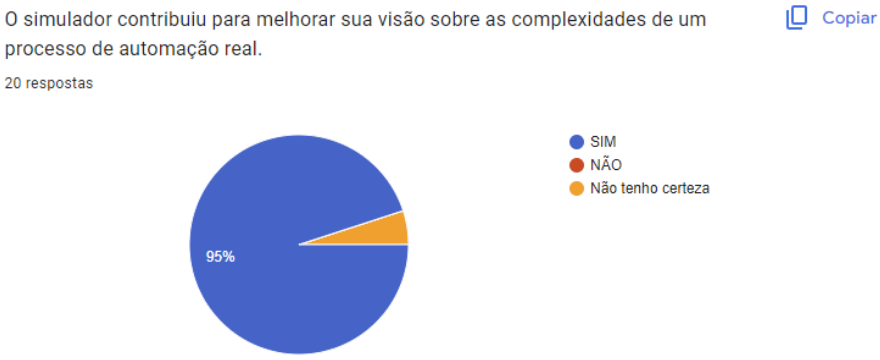

Figura 112 – Questionário alunos - Pergunta 8. Fonte: O autor.

6.2 Pesquisa com os docentes

A pesquisa com os docentes foi feita através de um formulário do Google e encaminhado para os docentes que lecionaram disciplinas de CLP em seus cursos técnicos pelo SENAI MG e utilizaram o simulador com seus alunos. Foram coletadas 7 respostas voluntárias dos docentes. O título da pesquisa foi: Avaliação de Impacto de utilização do Simulador de CLP (docentes). Todos os instrutores se identificaram com nome completo.

6.2.1 Perguntas do Questionário para docentes:

Pergunta: Qual é o seu nome completo?

Pergunta: Você utilizou o simulador de CLP com quantos alunos, ao todo?

Respostas possíveis: (Menos de 10 / Entre 10 e 20 / Entre 20 e 30 / Entre 30 e 40 / Entre 40 e 50 / Entre 50 e 70 / Entre 70 e 100 / Mais de 100 alunos)

Pergunta: Como você avalia a eficiência geral do simulador de CLP em relação ao desenvolvimento das habilidades de programação de CLPs dos alunos?

Respostas possíveis: (Pouco eficiente / Moderadamente eficiente / Muito eficiente)

Pergunta: O simulador de CLP contribuiu para uma compreensão mais aprofundada dos conceitos de programação dos alunos?

Respostas possíveis: (SIM / NÃO / Incerto)

Pergunta: O simulador de CLP contribuiu para uma compreensão mais aprofun-

dada dos conceitos de programação dos alunos?

Respostas possíveis: (SIM / NÃO / Incerto)

Pergunta: Na sua opinião, o simulador de CLP facilitou o engajamento e a participação ativa dos alunos durante as atividades de aprendizagem?

Respostas possíveis: (SIM / NÃO / Incerto)

Pergunta:Você considera que o simulador de CLP é uma ferramenta adequada para o ensino da programação de CLPs?

Respostas possíveis: (SIM / NÃO / Incerto)

6.2.2 Resultados coletados - pesquisa docentes

As Perguntas e resultados do questionário para os docentes que usaram o simulador com seus alunos podem ser vistos da Figura 113 até a Figura 117.

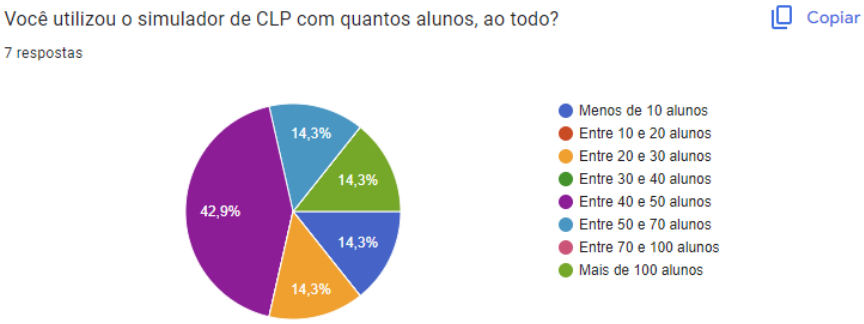

Figura 113 – Questionário docentes - Pergunta 2. Fonte: O autor.

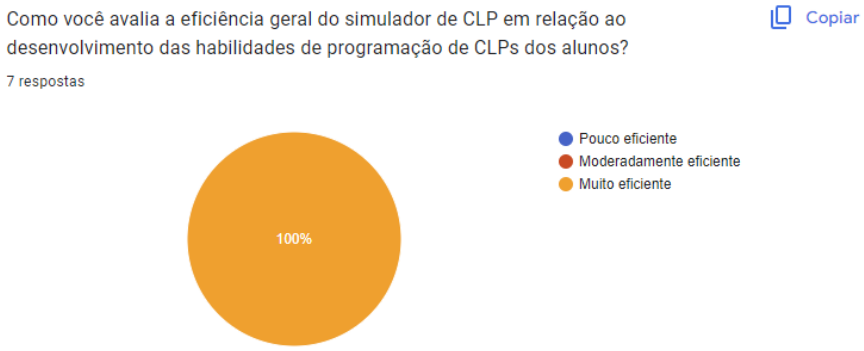

Figura 114 – Questionário docentes - Pergunta 3. Fonte: O autor.

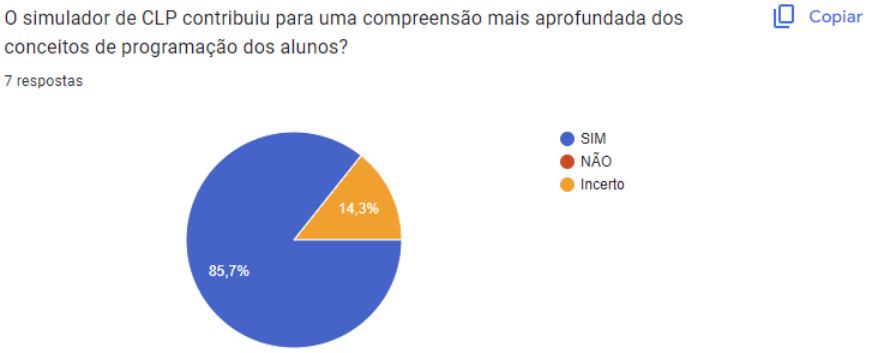

Figura 115 – Questionário docentes - Pergunta 4. Fonte: O autor.

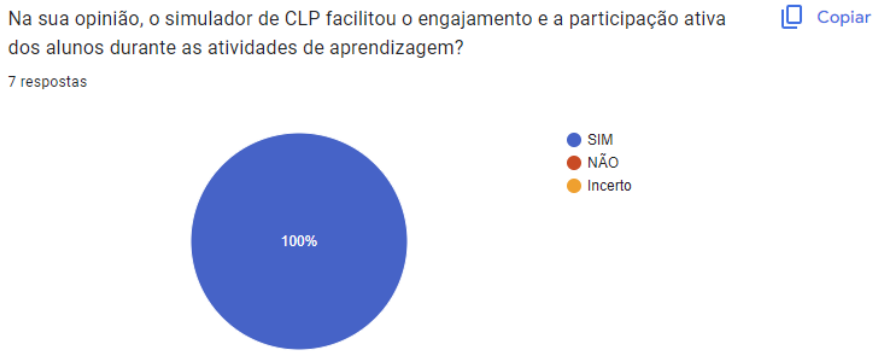

Figura 116 – Questionário docentes - Pergunta 5. Fonte: O autor.

Figura 117 – Questionário docentes - Pergunta 6. Fonte: O autor.

6.3 Considerações sobre a pesquisa

Os resultados coletados mostram que o uso do simulador foi muito bem aceito pelos docentes, que relataram que seu uso não apenas facilitou o ensino da matéria em suas aulas, como também contribuiu para uma compreensão mais aprofundada dos conceitos de programação e aumentou o engajamento dos alunos durante as aulas. Além disso, todos os docentes questionados disseram que o simulador foi muito eficiente no desenvolvimento das habilidades de programação dos alunos.

Apesar disso, deve-se ressaltar que a pesquisa teve uma amplitude pequena de entrevistados, já que poucas dezenas de alunos optaram por responder o questionário e o simulador foi utilizado até o momento por menos de 10 docentes. Novas pesquisas no futuro, conforme o simulador é divulgada e ganha mais adeptos, deverão ser realizadas.

7 Conclusão

Esta dissertação apresentou o desenvolvimento e a implementação de um simulador de CLP on-line, acessível via navegador web em diversos dispositivos, incluindo computadores e celulares. A ferramenta desenvolvida busca suprir uma lacuna existente no ensino de automação e controle, oferecendo uma plataforma interativa e prática para o aprendizado de conceitos relacionados a Controladores Lógicos Programáveis.

O simulador proporciona uma variedade de simulações, desde as mais básicas, envolvendo botões e lâmpadas, até simulações mais complexas e avançadas, como controle de semáforos, controle de nível, de temperatura e montagem de circuitos. Essa diversidade de simulações visa atender a um amplo espectro de necessidades educacionais, possibilitando a alunos e profissionais aprimorarem suas habilidades práticas e teóricas em CLP.

A utilização do simulador demonstrou ser uma ferramenta eficaz no processo de ensino-aprendizagem, proporcionando um ambiente rico e interativo para a experimentação e prática de conceitos de automação. Além disso, a acessibilidade via navegador web contribui para a democratização do acesso a ferramentas de aprendizagem em automação, permitindo que indivíduos de diferentes regiões e contextos tenham a oportunidade de desenvolver suas competências na área.

Apesar dos resultados positivos obtidos com a implementação do simulador, reconhece-se que há espaço para melhorias e expansões futuras. A inclusão de novas funcionalidades, simulações mais complexas e aprimoramento da interface de usuário são aspectos que podem ser explorados em trabalhos futuros, visando enriquecer ainda mais a experiência de aprendizagem dos usuários.

Dessa forma, o simulador de CLP on-line desenvolvido nesta dissertação se mostrou uma ferramenta valiosa no contexto educacional, proporcionando um meio prático e acessível para o estudo e prática de automação industrial. Acredita-se que sua implementação contribuirá significativamente para o avanço do ensino de automação, auxiliando na formação de profissionais mais qualificados e preparados para os desafios do mercado de trabalho.

Referências

1 CARRINO, G. Prévia: Novo flight simulator é uma verdadeira escola de aviação. *Start UOL*, 07 2020. Disponível em: <https://www.uol.com.br/start/ultimas-noticias/2020/07/30/previa-microsoft-flight-simulator-e-uma-verdadeira-escola-de-aviacao.htm>. 10, 24

2 Universidade do Colorado. *Kit de Construção de Circuitos: Laboratório Virtual DC*. 2023. <https://phet.colorado.edu/sims/html/circuit-construction-kit-dc-virtual-lab/latest/circuit-construction-kit-dc-virtual-lab_all.html?locale=pt_BR>. Acesso em: 22 jun. 2023. 10, 25

3 FERRAZ, E. *Modicon: 50 anos de Pioneirismo e Inovações*. 2018. Blog Schneider Electric. Disponível em: <https://blog.se.com/br/automacao-industrial/2018/11/30/modicon-50-anos-de-pioneirismo-e-inovacoes/>. 10, 32

4 SIMATIC S7-300 - Proven multiple times! [S.l.]: Siemens, 2023. <https://www.siemens.com/in/en/products/automation/systems/industrial/plc/simatic-s7-300.html>. Acessado em 2023-04-13. 10, 32

5 SISTEMAS de controlador lógico programável MicroLogix 1400. [S.l.]: Rockwell Automation, 2023. <https://www.rockwellautomation.com/pt-br/products/hardware/allen-bradley/programmable-controllers/micro-controllers/micrologix-family/micrologix-1400-controllers.html>. Acessado em 2023-04-08. 10, 35

6 CONTROLADORES SLC 500. [S.l.]: Rockwell Automation, 2023. <https://www.rockwellautomation.com/pt-br/products/hardware/allen-bradley/programmable-controllers/small-controllers/slc-500-controllers.html>. Acessado em 2023-04-08. 10, 36

7 PLC Chassis - Maverick Industrial Sales. [S.l.]: Maverick Industrial Sales, 2023. <https://maverickindustrialsales.com/collections/plc-chassis>. Acessado em 2023-04-12. 10, 36

8 GREENFIELD, D. Peer-to-peer faq: Controllers. *Automation World*, 02 2023. Acessado em 2023-04-14. Disponível em: <https://www.automationworld.com/control/article/22724233/industrial-controller-technologies>. 10, 44

9 CAPTURA de tela do site PLC Fiddle. [S.l.]: PLC Fiddle, 2023. <https://plcfiddle.com>. Acesso em 19 de abril de 2023. 12, 66

10 CAPTURA de tela do site PLC Simulator Online. [S.l.]: PLC Simulator Online, 2023. <https://plcsimulator.on-line>. Acesso em 19 de abril de 2023. 12, 66

11 CAPTURA de tela do site PLC-Simulator Virtual Labs. [S.l.]: PLC-Simulator Virtual Labs, 2023. <https://plc-coep.vlabs.ac.in/exp/up-down-counter/simulation/>. Acesso em 19 de abril de 2023. 12, 67

12 CAPTURA de tela do site PLC Simulator.NET. [S.l.]: PLC Simulator.NET, 2023. <http://www.plcsimulator.net>. Acesso em 14 de abril de 2023. 12, 67

13 IMAGEM obtida do site CODESYS. [S.l.]: CODESYS, 2023. <https://www.codesys.com>. Acesso em 20 de abril de 2023. 12, 68

14 IMAGEM obtida do site WTE PLC Simulator. [S.l.]: WTE PLC Simulator, 2023. <https://www.wte.co.nz/plc-simulator.html>. Acesso em 20 de abril de 2023. 12, 69

15 IMAGEM obtida do site OpenPLC Editor. [S.l.]: Autonomy Logic, 2023. <https://autonomylogic.com/download/>. Acesso em 20 de abril de 2023. 12, 70

16 IMAGEM obtida do software WinProLadder. [S.l.]: FATEK Automation Corp., 2023. <https://www.fatek.com>. Acesso em 21 de abril de 2023. 12, 70

17 IMAGEM obtida do aplicativo PLC Ladder Simulator 2. [S.l.]: Google Play Store, 2023. <https://play.google.com/store/apps/details?id=com.casdata.plcladdersimulator2>. Acesso em 21 de abril de 2023. 12, 71

18 IMAGEM obtida do site Factory I/O. [S.l.]: Factory I/O, 2023. <https://factoryio.com>. Acesso em 14 de abril de 2023. 12, 72

19 BR, C. *Pesquisa sobre o uso das tecnologias de informação e comunicação: pesquisa TIC Domicílios, ano 2023*. [S.l.]: Núcleo da Informação e Coordenação do Ponto BR-NIC. br São Paulo, 2023. 15, 23

20 SHARIF, K. H.; AMEEN, S. Y. Game engines evaluation for serious game development in education. In: IEEE. *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. [S.l.], 2021. p. 1–6. 15, 28

21 HANSON, L. G. Interactive mr simulation in a browser: Flexible reimplementation of the educational bloch simulator. In: *MRI together 2021: A global workshop on Open Science and Reproducible MR Research*. [S.l.: s.n.], 2021. 15, 28

22 GRECA, I. M.; SEOANE, E.; ARRIASSECQ, I. Epistemological issues concerning computer simulations in science and their implications for science education. *Science & Education*, Springer, v. 23, p. 897–921, 2014. 20

23 JONG, T. D.; LINN, M. C.; ZACHARIA, Z. C. Physical and virtual laboratories in science and engineering education. *Science*, American Association for the Advancement of Science, v. 340, n. 6130, p. 305–308, 2013. 20

24 ZACKIEWICZ, M. A economia do software e a digitalização da economia. *Revista Brasileira de Inovação*, v. 14, n. 2, p. 313–336, 2015. 22

25 WARSCHAUER, M. *Technology and social inclusion: Rethinking the digital divide*. [S.l.]: MIT press, 2004. 22

26 BR, C. *TIC Educação: 2022*. [S.l.]: Cetic. br São Paulo, 2022. 23

27 HARGITTAI, E. Second-level digital divide: Mapping differences in people's online skills. *arXiv preprint cs/0109068*, 2001. 23

28 ASSIS, P. Educação e tecnologias: o novo ritmo da informação. *Bolema Boletim de Educação Matemática*, v. 29, p. 428–434, 04 2015. 23

29 PAZIN-FILHO, A.; SCARPELINI, S. SimulaÇÃo: DefiniÇÃo. *Medicina (Ribeirao Preto. Online)*, v. 40, p. 162, 06 2007. 23

30 CHERNIKOVA, O. et al. Simulation-based learning in higher education: a meta-analysis. *Review of Educational Research*, Sage Publications Sage CA: Los Angeles, CA, v. 90, n. 4, p. 499–541, 2020. 24, 25

31 Di Serio Ángela; IBáñEZ, M. B.; KLOOS, C. D. Impact of an augmented reality system on students' motivation for a visual art course. *Computers Education*, v. 68, p. 586–596, 2013. ISSN 0360-1315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131512000590>. 24

32 SO, H. Y. et al. Simulation in medical education. *Journal of the Royal College of Physicians of Edinburgh*, SAGE Publications Sage UK: London, England, v. 49, n. 1, p. 52–57, 2019. 24

33 TEGTMEYER, K.; IBSEN, L.; GOLDSTEIN, B. Computer-assisted learning in critical care: from eniac to hal. *Critical care medicine*, v. 29, p. N177–82, 09 2001. 24

34 GAVIRA, M. d. O. *Simulação computacional como uma ferramenta de aquisição de conhecimento.* Tese (Doutorado) — Universidade de São Paulo, 2003. 24

35 ZAMPIER, L.; BRAGUINI, W. L. Simulador educacional como ferramenta de apoio em aulas de ciências. *Os desafios da escola pública paranaense na perspectiva do professor PDE, editora Secretaria de Educação do Governo do Estado do Paraná*, v. 2, 2016. 25

36 HUANG, H.-M.; RAUCH, U.; LIAW, S.-S. Investigating learners' attitudes toward virtual reality learning environments: Based on a constructivist approach. *Computers & Education*, Elsevier, v. 55, n. 3, p. 1171–1182, 2010. 25

37 NAVARRO, E. Simse: A software engineering simulation environment for software process education dissertation. *University of California, Irvine*, Citeseer, 2006. 26

38 SUNG, K.; GREGORY, S. *Basic Math for Game Development with Unity 3D.* [S.l.]: Springer, 2019. 28

39 EL-WAJEH, Y. A.; HATTON, P. V.; LEE, N. J. Unreal engine 5 and immersive surgical training: translating advances in gaming technology into extended-reality surgical simulation training programmes. *British Journal of Surgery*, Oxford University Press, v. 109, n. 5, p. 470–471, 2022. 28

40 VAKALIUK, T. A.; POCHTOVIUK, S. I. Analysis of tools for the development of augmented reality technologies. In: CEUR WORKSHOP PROCEEDINGS. [S.l.], 2021. 29

41 JOHANSSON, J. *Performance and Ease of Use in 3D on the Web: Comparing Babylon. js with Three. js.* 2021. 29

42 SPUY, R. Van der. *Learn Pixi. js.* [S.l.]: Apress, 2015. 29

43 LAU, W. W. et al. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In: *Proceedings of the 40th ACM technical symposium on Computer science education.* [S.l.: s.n.], 2009. p. 504–508. 29

44 ANTUNES, F. C. de C. Introdução a robótica: uma proposta de sequência didática para o ensino de programação. *Com a Palavra, o Professor*, v. 7, n. 19, p. 175–183, 2022. 30

45 NEVES, Y.; FREITAS, D. Construção de controlador lógico programável de baixo custo para fins didáticos. *Revista de Engenharia e Pesquisa Aplicada*, v. 4, n. 4, p. 28–39, 2019. 31

46 COSTA, G. H. S. Utilização de um controlador lógico programável em um sistema de bombeamento de água. Universidade Federal de Uberlândia, 2022. 31, 33

47 NASCIMENTO, A. B. Um breve tópico na história da automação: do painel elétrico ao controlador lógico programável. 2022. 31

48 FRANCHI, C. M.; CAMARGO, V. L. A. de. *Controladores Lógicos Programáveis Sistemas Discretos*. [S.l.]: Saraiva Educação SA, 2008. 32, 33, 34

49 LISSONE, M. I.; BONIFÁCIO, L. F. S. Elevador inteligente usando arduíno. 2018. 33

50 TÓFOLI, M. F.; HIGA, R. A. Estudo comparativo entre clp e microcontrolador em um elevador de baixa complexidade para carga. *Revista Eletrônica e-Fatec*, v. 4, n. 1, p. 10–10, 2014. 35

51 HANSSEN, D. H. *Programmable logic controllers: a practical approach to IEC 61131-3 using CODESYS*. [S.l.]: John Wiley & Sons, 2015. 37, 68

52 SEHGAL, P. P. S.; DWIVEDI, A.; CHOPRA, W. C. A. Special i/o modules in plc system. *International Refereed Journal of Engineering and Science (IRJES)*, v. 1, p. 20, 2012. 42, 43

53 RATA, M.; RATA, G. Speed control application with plc high speed outputs port. In: IEEE. *2023 13th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. [S.l.], 2023. p. 1–5. 42

54 KEN, B. How programmable logic controllers emerged from industry needs. *Control Engineering*, v. 9, 2008. 43

55 SEGOVIA, V. R.; THEORIN, A. History of control history of plc and dcs. *University of Lund*, 2012. 44, 45

56 BRUSSO, B. C. 50 years of industrial automation [history]. *IEEE Industry Applications Magazine*, IEEE, v. 24, n. 4, p. 8–11, 2018. 44

57 STANKOVSKI, S. et al. Using micro/mini plc/pac in the edge computing architecture. In: IEEE. *2020 19th International Symposium Infoteh-Jahorina (Infoteh)*. [S.l.], 2020. p. 1–4. 45

58 WANG, G. A new approach for plc ladder diagram design. In: IEEE. *2021 IEEE International Conference on Electro Information Technology (EIT)*. [S.l.], 2021. p. 021–026. 46

59 FRONCHETTI, F. et al. Language impact on productivity for industrial end users: A case study from programmable logic controllers. *Journal of Computer Languages*, Elsevier, v. 69, p. 101087, 2022. 46

60 BIBLIOTECA JavaScript Split.js. [S.l.]: Split.js, 2023. <https://split.js.org>. Acesso em 24 de abril de 2023. 76

Apêndices

APÊNDICE A – Programas Extras

A.1 Programa 1: Pisca-pisca de uma lâmpada em Ladder:

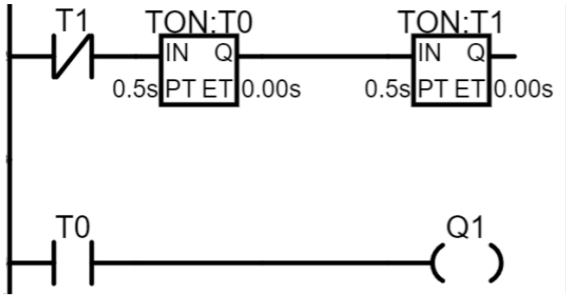

Figura 118 – Pisca-Pisca. Disponível em <https://gvensino.com.br/simpiscapisca>. Fonte: o autor.

A.2 Programa 2: Controle de Cruzamento de Semáforo

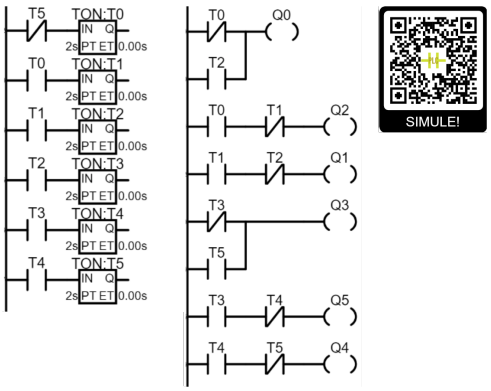

Figura 119 – Controle de Semáforo de um Cruzamento. Disponível em <https://gvensino.com.br/simsemaforo>. Fonte: o autor.

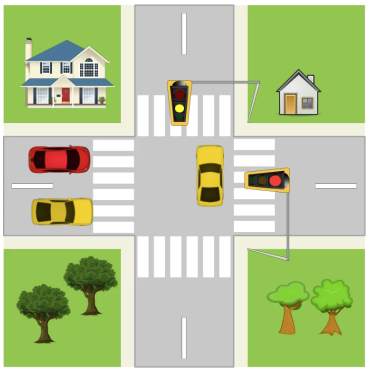

Figura 120 – Simulação do Semáforo. Fonte: o autor.

APÊNDICE B – Estrutura da Programação

B.1 Estrutura da programação do núcleo do simulador - Principais funções e Classes do painel de programação

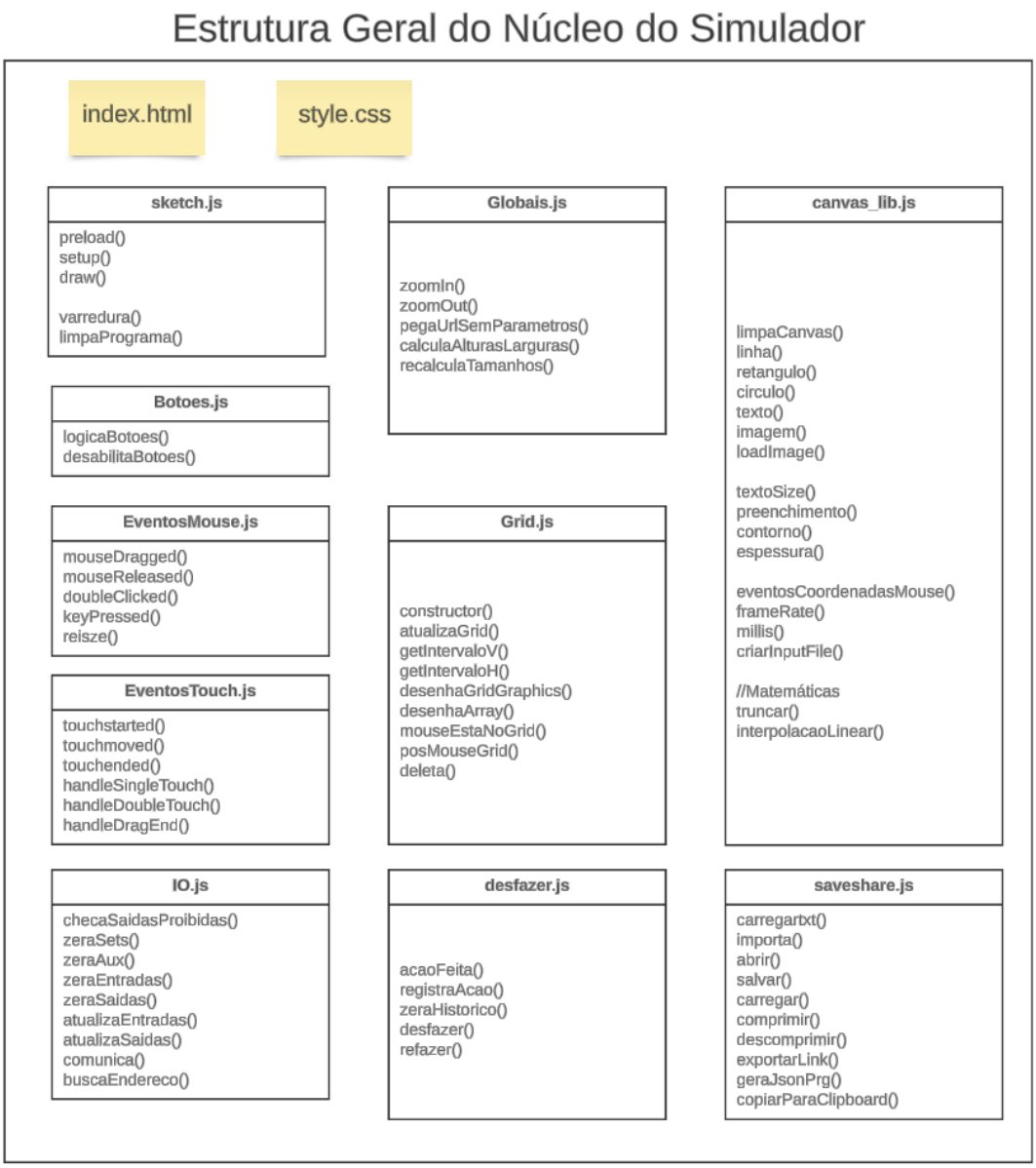

Figura 121 – Estrutura do Núcleo do simulador. Fonte: o autor.

B.1.1 sketch.js

É o arquivo principal do simulador. Dentro dele roda-se algumas funções importantes, que descreveremos a seguir:

preload(): esta função é a primeira a ser executada e sua função principal é pré-carregar as imagens que serão utilizadas pelo simulador.

setup(): logo após a execução do preload, é executado o setup(), onde é realizado os primeiros ajustes da simulação, tais como tamanho do grid da programação, detecção do tamanho da janela, inicialização da comunicação entre painel de programação e painel de simulação, inicialização dos eventos de detecção do mouse e do clique, dentre outras.

draw(): esta função é executada logo após a execução de setup() e é executada em loop, ou seja, uma execução atrás da outra, dentro do fluxo de desenho do navegador web. Nesta função é onde é feita toda a lógica de desenhar o programa na tela, desenhar mudanças no código, atualizar o grid e mandar executar a lógica de programação do ladder montada pelo usuário.

varredura(): esta função faz a varredura do programa ladder feito pelo usuário, simulando a energia passando por cada elemento.

limpaPrograma(): esta função limpa o grid de qualquer programação que nela esteja.

B.1.2 Botoes.js

Aqui temos as principais interações dos botões da barra lateral do simulador.

logicaBotoes(): faz a leitura de qual botão foi pressionado e realiza ações de acordo.

desabilitaBotoes(): desabilita ou habilita os botões, para que quando a programação estiver sendo executada (modo Run), nenhum botão da barra lateral possa ser clicado.

B.1.3 EventosMouse.js

Aqui é onde são processados os eventos do mouse do núcleo do simulador. As principais funções são:

mouseDragged(): detecta se o mouse foi arrastado pelo usuário, utilizada para detectar quando o usuário arrastou algum elemento no grid.

mouseReleased(): detecta quando o botão principal do mouse foi solto.

doubleClicked(): detecta quando há um duplo clique do botão do mouse.

keyPressed(): detecta quando alguma tecla do teclado é pressionada.

resize(): detecta quando o usuário muda o tamanho da janela, para que se possa alterar o tamanho dos elementos desenhados na tela.

B.1.4 EventosTouch.js

Aqui é onde são processados os eventos de touch do núcleo do simulador, para operações em celulares e tablets. As principais funções são:

touchstarted(): detecta quando há o início de um toque na tela.

touchmoved(): detecta quando o usuário arrasta o toque na tela.

touchended(): detecta quando o usuário tira o dedo da tela.

handleSingleTouch(): aciona quando o usuário dá apenas um toque breve na tela.

handleDoubleTouch(): aciona quando o usuário dá dois toques rápidos na tela.

handleDragEnd(): aciona quando o usuário termina de arrastar o dedo na tela.

B.1.5 IO.js

Aqui é onde são processados os eventos de comunicação entre a programação e a simulação. As principais funções são:

checaSaidasProibidas(): verifica se o usuário digitou algum nome que não deve ser usado no programa, tais como nomes de saídas repetidas.

zeraSets(): zera as saída setadas pelo operador SET.

zeraAux(): zera as variáveis auxiliares que controlam o fluxo do programa.

zeraEntradas(): zera as entradas do CLP.

zeraSaidas(): zera a saídas do CLP.

atualizaEntradas(): atualiza o valor das entradas de acordo com a simulação.

atualizaSaidas(): atualiza o valor das saídas de acordo com o processado no código.

comunica(): faz o envio e recebimento em tempo real das entradas e saídas para a simulação que estiver aberta.

buscaEndereco(): busca se um endereço digitado pelo usuário já existe no programa ou não.

B.1.6 Globais.js

Aqui ficam as variáveis e funções globais, que podem ser utilizadas em qualquer ponto da programação. As principais são:

zoomIn(): calcula o aumento do grid de programação quando o usuário quer aumentar o zoom.

zoomOut(): calcula a diminuição do grid de programação quando o usuário quer diminuir o zoom.

pegaUrlSemParametros(): separa o endereço do site dos parâmetros que foram inseridos no navegador.

calculaAlturasLarguras(): calcula a altura e largura da janela, para adequar o tamanho do grid na tela.

recalculaTamanhos(): quando usuário muda o tamanho do painel ou da tela, permite recalcular os tamanhos para melhor adquar o grid.

B.1.7 Grid.js

Aqui fica toda a lógica do Grid, que é onde é feita a programação pelo usuário. As principais funções são:

constructor(): cria o Grid inicialmente.

atualizaGrid(): redesenha o grid caso haja alterações em seu tamanho ou número de linhas exibidas.

gIntervaloV(): pega o tamanho vertical de cada célula do grid, em pixels.

getIntervaloH(): pega o tamanho horizontal de cada célula do grid, em pixels.

desenhaGridGraphics(): desenha as linhas do Grid ao fundo.

desenhaArray(): desenha a programação no Grid.

mouseEstaNoGrid(): detecta se o mouse está em cima do grid ou não.

posMouseGrid(): detecta em qual célula do grid o mouse está posicionado.

deleta(): deleta o elemento selecionado no grid.

B.1.8 desfazer.js

Aqui é onde ficam reunidas as lógicas de desfazer e refazer da programação. As principais funções ali são:

acaoFeita() e registraAcao(): são chamados sempre que o usuário faz algum acréscimo ou mudança na programação, para registrar no histórico e poder desfazer ou refazer mais tarde.

zeraHistorico(): apaga/inicializa o histórico de ações do usuário.

desfazer(): volta o programa para o estado anterior, antes da última modificação.

refazer(): refaz a última modificação desfeita.

B.1.9 canvas lib.js

Aqui é onde são criadas funções que facilitam o desenho de formas geométricas na tela (canvas). As principais funções são:

limpaCanvas(): limpa todo o conteúdo da tela. linha(): permite desenhar uma linha na tela. Recebe 4 parâmetros: ponto inicial X, ponto inicial Y, ponto final X, ponto final Y.

retangulo(): permite desenhar um retângulo na tela. Recebe 4 parâmetros: ponto inicial X, ponto inicial Y, largura e altura.

circulo(): permite desenhar um circulo na tela. Recebe 3 parâmetros: ponto inicial X, ponto inicial Y, raio.

texto(): permite escrever um texto na tela. Recebe 3 parâmetros: ponto inicial X, ponto inicial Y, texto.

imagem(): permite inserir uma imagem na tela. Recebe 3 parâmetros: ponto inicial X, ponto inicial Y, imagem.

loadImage(): permite o pré-carregamento das imagens, utilizadas no preload.

textoSize(): define o tamanho do texto a ser escrito no canvas.

preenchimento(): define a cor do preenchimento adotada pelo canvas. Pode receber como parâmetro o nome da cor em inglês, o código hexadecimal de cores ou um valor RGB.

contorno(): define a cor do contorno das formas que serão desenhadas, em pixels.

espessura(): define o tamanho da espessura das formas que serão desenhadas, em pixels.

eventosCoordenadasMouse(): inicializa os eventos que detectarão a posição do mouse ou do touch na tela.

frameRate(): calcula quantos quadros por segundos o simulador está desenhando na tela.

millis(): retorna o tempo, em milisegundos, a partir do momento em que o simulador é carregado. Útil para operações de tempo e temporizadores.

criarInputFile(): permite carregar um arquivo de simulação salvo.

truncar(): trunca um número. Recebe dois parâmetros: o número e a quantidade de casas que será truncada.

interpolacaoLinear(): faz interpolação linear de um valor, a partir de seu valor mínimo e máximo e qual o valor mínimo e máximo que será feita a interpolação. Útil para conversão de grandezas nas simulações, como por exemplo, conversão entre pixels e altura do nível do tanque, por exemplo.

B.1.10 saveshare.js

Aqui é onde fica a lógica de salvar e abrir uma programação feita pelo usuário. As principais funções são:

carregartxt(): carrega o código que está num arquivo txt.

importa(): permite importar o código que está num arquivo externo ou no endereço do site, através do parâmetro prg.

abrir(): permite abrir um arquivo externo.

salvar(): salva programação em um arquivo.

carregar(): carrega a programação que está em formato de texto.

comprimir(): comprime a programação em formato criptográfico, para ocupar menos espaço.

descomprimir(): descomprime a programação do formato criptográfico para formato legível pelo simulador.

exportarLink(): gera um link da programação do usuário que pode ser enviado para outros.

geraJsonPrg(): gera a programação da tela em formato texto (JSON), utilizada pelo próprio simulador para salvar o programa em formato texto.

copiarParaClipboard(): função que permite inserir a programação em formato texto na área de transferência do usuário.

B.2 Estrutura da programação das simulações - Principais funções e classes.

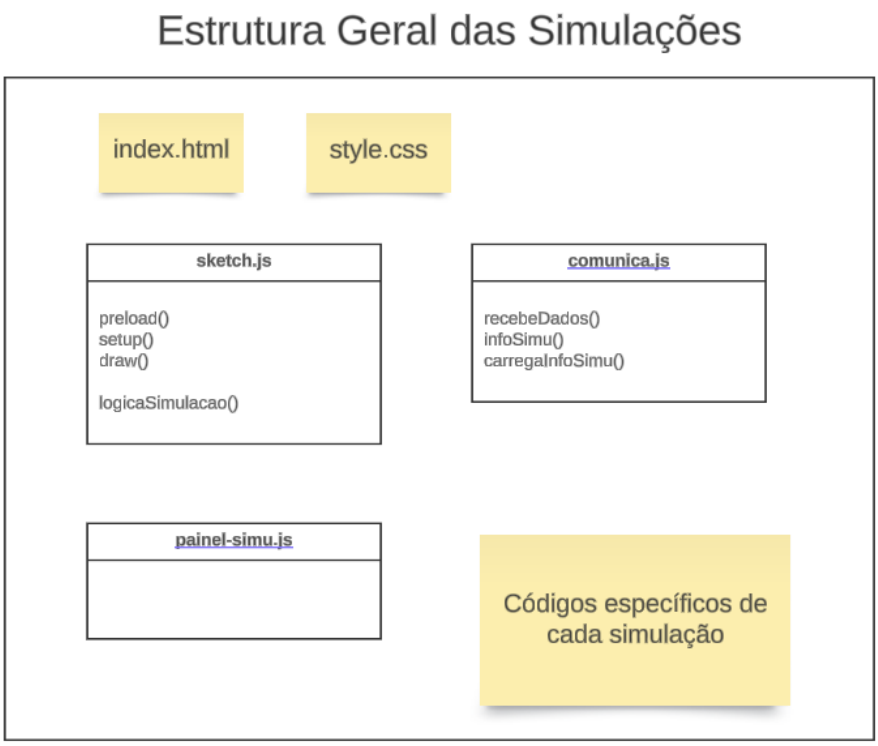

Figura 122 – Estrutura das Simulações. Fonte: o autor.

B.2.1 sketch.js

É o arquivo principal da simulação. Dentro dele roda-se toda a lógica da simulação, separada nas principais funções a seguir:

preload(): esta função é a primeira a ser executada e sua função principal é pré-carregar as imagens que serão utilizadas pela simulação.

setup(): logo após a execução do preload, é executado o setup(), onde é realizado os primeiros ajustes da simulação, tais como detecção do tamanho da janela, inicialização da comunicação entre painel de programação e painel de simulação, inicialização dos eventos de detecção do mouse e do clique, dentre outras.

draw(): esta função é executada logo após a execução de setup() e é executada em loop, ou seja, uma execução atrás da outra, dentro do fluxo de desenho do navegador web. Nesta função é onde é feita toda a lógica de desenhar a simulação na tela e fazer suas animações.

logicaSimulacao(): esta função varia de acordo com a simulação e é a responsável por fazer toda a lógica da simulação.

B.2.2 painel-simu.js

Controla o painel flutuante de simulação que fica no topo da simulação, no centro da tela.

B.2.3 comunica.js

Faz a comunicação entre a simulação e o painel de programação. Suas principais funções são:

recebeDados(): recebe os dados da programação, especialmente quais saídas estão acionadas ou não acionadas ou se o CLP está em Stop ou em Run.

infoSimu(): carrega informações da simulação que o usuário porventura tenha salvo.

carregaInfoSimu(): pega as configurações que o usuário fez na simulação para salvar.