



Luiz Alberto Oliveira Lima Roque
Danilo Barcelos da Silva
Gabriel Francesco Trotta Barbosa
João Cesar Campello Maia
Lucas Furlan Araújo

MONOGRAFIAS PARA ENGENHARIA DE AUTOMAÇÃO

Luiz Alberto Oliveira Lima Roque
Danilo Barcelos da Silva
Gabriel Francesco Trotta Barbosa
João Cesar Campello Maia
Lucas Furlan Araújo

MONOGRAFIAS PARA ENGENHARIA DE AUTOMAÇÃO

1ª ed.

Piracanjuba-GO
Editora Conhecimento Livre
Piracanjuba-GO

1ª ed.

Dados Internacionais de Catalogação na Publicação (CIP)

Roque, Luiz Alberto Oliveira Lima
R638M MONOGRAFIAS PARA ENGENHARIA DE AUTOMAÇÃO

/ Luiz Alberto Oliveira Lima Roque. Danilo Barcelos da Silva. Gabriel Francesco Trotta Barbosa.
João Cesar Campello Maia. Lucas Furlan Araújo. – Piracanjuba-GO

Editora Conhecimento Livre, 2025

201 f.: il

DOI: 10.37423/2025.edcl1197

ISBN: 978-65-5367-709-8

Modo de acesso: World Wide Web

Incluir Bibliografia

1. automação 2. controladores-lógicos-programáveis 3. sistemas-supervisórios 4. iec-61131-3 5.
linguagem-ladder I. Roque, Luiz Alberto Oliveira Lima II. Silva, Danilo Barcelos da III. Barbosa,
Gabriel Francesco Trotta IV. Maia, João Cesar Campello V. Araújo, Lucas Furlan VI. Título

CDU: 620

<https://doi.org/10.37423/2025.edcl1197>

O conteúdo dos artigos e sua correção ortográfica são de responsabilidade exclusiva dos seus respectivos autores.

EDITORIA CONHECIMENTO LIVRE

Corpo Editorial

MSc Edson Ribeiro de Britto de Almeida Junior

MSc Humberto Costa

MSc Thays Merçon

MSc Adalberto Zorzo

MSc Taiane Aparecida Ribeiro Nepomoceno

PHD Willian Douglas Guilherme

MSc Andrea Carla Agnes e Silva Pinto

Dr. Walmir Fernandes Pereira

MSc Edisio Alves de Aguiar Junior

MSc Rodrigo Sanchotene Silva

MSc Adriano Pereira da Silva

MSc Frederico Celestino Barbosa

MSc Guilherme Fernando Ribeiro

MSc. Plínio Ferreira Pires

MSc Fabricio Vieira Cavalcante

PHD Marcus Fernando da Silva Praxedes

MSc Simone Buchignani Maigret

Dr. Adilson Tadeu Basquerote

Dra. Thays Zigante Furlan

MSc Camila Concato

PHD Miguel Adriano Inácio

Dra. Anelisa Mota Gregoleti

PHD Jesus Rodrigues Lemos

MSc Karine Moreira Gomes Sales

Dr. Saulo Cerqueira de Aguiar Soares

MSc Pedro Panhoca da Silva

MSc Helton Rangel Coutinho Junior

MSc Carlos Augusto Zilli

MSc Euvaldo de Sousa Costa Junior

Dra. Suely Lopes de Azevedo

Dr. Francisco Odecio Sales

MSc Ezequiel Martins Ferreira

MSc Eliane Avelina de Azevedo Sampaio

MSc Carlos Eduardo De Oliveira Gontijo

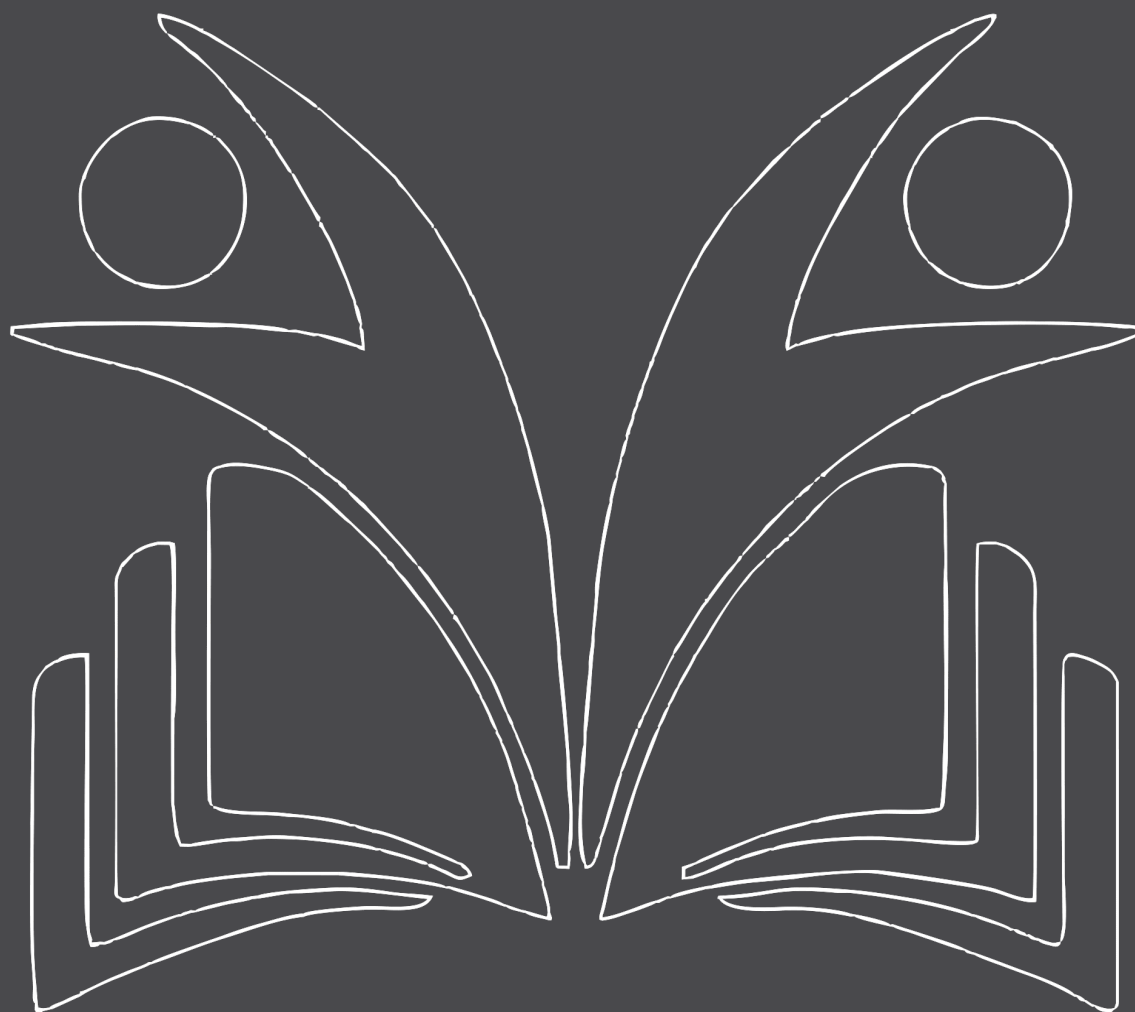
Dr. Rodrigo Couto Santos

Dra. Milena Gaion Malosso

PHD Marcos Pereira Dos Santos



10.37423/2025.edcl1197



Dedicamos aqui, à todos aqueles que nos apoiaram durante nossa jornada

AGRADECIMENTOS

João Cesar Campello Maia

A Deus, por ter permitido que eu tivesse saúde determinação para não desanimar durante a realização deste trabalho.

Aos meus familiares e amigos, por sempre me apoiar e motivar para enfrentar qualquer desafio que fosse necessário me possibilitando a me tornar uma pessoa melhor e mais capacitada.

Ao meu parceiro Lucas Furlan Araújo, por me acompanhar durante os longos anos do curso de engenharia, sempre me ajudando nos estudos e me mantendo motivado, assim como no desenvolvimento deste trabalho.

Aos docentes e servidores do Instituto Federal Fluminense, por me proporcionarem um ensino completo e desafiador que fez eu me tornar uma pessoa mais madura e preparada para o mercado de trabalho.

Ao professor Luiz Alberto Oliveira Lima Roque, pelos vários conhecimentos compartilhados e o acompanhamento do desenvolvimento deste trabalho, provento sua perspectiva e nos guiando nessa jornada.

Lucas Furlan Araújo

Primeiramente, a Deus por me guiar e proteger meu caminho até aqui. Sem ele, nada disso seria possível.

Aos meus pais, minha eterna fonte de amor, força e inspiração. Vocês foram e sempre serão meu maior motivo para continuar, mesmo nos momentos mais difíceis. Cada conquista que alcancei até hoje carrega um pedaço do esforço, do sacrifício e do carinho que sempre dedicaram a mim. Este trabalho é, antes de tudo, uma homenagem a vocês, que me ensinaram o valor da perseverança e a importância de acreditar nos meus sonhos.

Agradeço também à minha família, que esteve presente em cada etapa desta jornada, seja com palavras de incentivo, abraços silenciosos ou simplesmente com a certeza de que eu nunca estive sozinho. Foram essas presenças, de tantas formas, que me deram forças para seguir em frente.

Entre essas pessoas tão especiais, estão também meu querido avô Osmar e minha amada avó Teresinha, que já não estão fisicamente entre nós, mas permanecem vivos em cada lembrança, em cada ensinamento e em cada gesto de amor que guardo em meu coração. Hoje, ao alcançar esta

conquista, sinto que eles caminham ao meu lado, não apenas na memória, mas na essência do que sou e de tudo o que aprendi. Este trabalho, portanto, é dedicado também a eles, como uma forma de reconhecer que cada passo dado carrega o amor, a inspiração e o exemplo que deixaram para mim e para toda a nossa família.

Aos meus amigos, principalmente ao meu irmão, João Cesar Campello Maia, que, ao longo dessa longa caminhada, esteve ao meu lado em todos os momentos. Entre risadas e nervosismos durante todo o processo, sempre ajudando um ao outro, sem egoísmo.

Aos servidores do Instituto Federal Fluminense, sem o suporte deles, nada teria sido alcançado. Em especial, a um professor, um mestre, Luiz Alberto Oliveira Lima Roque. Cujas dedicação, sabedoria e paixão pelo ensino foram essenciais na minha formação.

Agradeço, do fundo do meu coração, a todos que estiveram comigo nesta árdua jornada.

"A ciência pode entreter-se com teorias, mas a engenharia precisa de soluções"

(Henry Petroski)

Resumo: Neste trabalho foi feito um estudo sobre a automatização de testes hidrostáticos realizados nos equipamentos e ferramentas offshore utilizados no sistema de completação de uma poço de petróleo. Os testes hidrostáticos são essenciais para garantir o correto funcionamento, capacidade de vedação e integridade estrutural dos equipamentos, prevenindo falhas que poderiam resultar em vazamentos e danos ambientais. A pesquisa aborda a implementação de um sistema automatizado, que substituem os métodos manuais, promovendo maior segurança, eficiência e precisão no processo. Além disso, são discutidos os benefícios da automação, como a redução do tempo de teste, a minimização de erros humanos e o aumento da confiabilidade dos resultados. O estudo também analisa os principais recursos utilizados, incluindo transdutores de pressão, válvulas solenoides e sistemas supervisórios. Por fim, conclui-se que a automação dos testes hidrostáticos representa um avanço significativo na indústria petrolífera, contribuindo para a otimização operacional e a conformidade com as normas de segurança e qualidade.

Palavras-chaves: Automação. Testes Hidrostáticos. Segurança Industrial. Equipamentos Offshore. Extração de Petróleo.

LISTA DE ILUSTRAÇÕES

Figura 1 – Conjunto ANM, BAP e MCV.....	19
Figura 2 – Manifold.....	20
Figura 3 – Hierarquia de Controle de Riscos.....	22
Figura 4 – Exemplo de Controlador Lógico Programável.....	26
Figura 5 – Exemplo de uma bomba tríplex.....	28
Figura 6 – Exemplo de uma bomba hidropneumática.....	29
Figura 7 – Exemplo de válvula solenoide.....	30
Figura 8 – Exemplo de válvula de alívio.....	30
Figura 9 – Exemplo de válvula de retenção.....	31
Figura 10 – Exemplo de manômetro.....	32
Figura 11 – Exemplo de Transdutor de Pressão.....	32
Figura 12 – Exemplo de filtro hidráulico.....	33
Figura 13 – Exemplo de Relé	34
Figura 14 – CODESYS.....	34
Figura 15 – Ignition SCADA.....	35
Figura 16 – Circuito hidráulico UTF.....	39
Figura 17 – Programação em Ladder no CodeSys.....	41
Figura 18 – Declaração das variáveis utilizadas.....	42
Figura 19 – Linha de programação - Comando do motor tríplex.....	42
Figura 20 – Linha de programação - Comando do motor hidropneumático.....	43
Figura 21 – Linha de programação - Comando da tranca da célula e válvulas de pressão	44
Figura 22 – Linha de programação - Comando da válvula de saída de flushing.....	44
Figura 23 – Linha de programação - Comando da válvula de retorno do fluido.....	45
Figura 24 – Linha de programação - Comando dos indicadores visuais de teste.....	45

Figura 25 – Linha de programação - Comando do alarme de segurança	44
Figura 26 – Etapas de configuração da comunicação no CodeSys	46
Figura 27 – Etapas de configuração da comunicação no Ignition	47
Figura 28 – Acesso das variáveis no Ignition através do servidor OPC UA	48
Figura 29 – Ignition - Aba perspective components	48
Figura 30 – Ignition - Representação do sistema	49
Figura 31 – Ignition - Configuração de cor das bombas e válvulas	50
Figura 32 – Ignition - Configuração do Interruptor	50
Figura 33 – Ignition - Configuração da porta da célula de teste	51
Figura 34 – Ignition - Configuração dos indicadores visuais	52
Figura 35 – Ignition - Configuração da simulação do valor de pressão	52
Figura 36 – Ignition - Configuração do valor da variável Pressurizado	53
Figura 37 – Ignition - Configuração do gráfico de pressão	53
Figura 38 – Ignition - Configuração do relatório de teste	54
Figura 39 – Sistema supervisorio do projeto	54
Figura 40 – Supervisorio em funcionamento - Flushing	55
Figura 41 – Supervisorio em funcionamento - Aumento de pressão	56
Figura 42 – Supervisorio em funcionamento - Teste hidrostático	56
Figura 43 – Supervisorio em funcionamento - Teste do alarme	57
Figura 44 – Supervisorio em funcionamento - Gráfico em tempo real	57
Figura 45 – Supervisorio em funcionamento - Relatório de teste	58

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
IFF	Instituto Federal Fluminense
API	American Petroleum Institute
ANM	Árvore de Natal Molhada
FPSO	Floating Production Storage and Offloading
BAP	Base Adaptadora de Produção
MCV	Modulo de Conexão Vertical
FIBAP	Ferramenta de Instalação da Base Adaptadora de Produção
THRT	Tubing Hanger Running Tool
TRT	Tree Running Tool
UTF	Unidade de Teste e Flushing
CLP	Controlador Lógico Programável
IHM	Interface Homem Máquina

Sumário

1 INTRODUÇÃO.....	19
2 OBJETIVOS	21
2.1 OBJETIVO GERAL	21
2.2 OBJETIVOS ESPECÍFICOS	22
2.3 JUSTIFICATIVA.....	22
3 FUNDAMENTAÇÃO TEÓRICA.....	23
3.1 SISTEMA MARÍTIMO DE PRODUÇÃO DE PETRÓLEO	23
3.2 CONCEITOS DE FUNCIONAMENTO DOS EQUIPAMENTOS E FERRAMENTAS SUBMARINAS	24
3.3 COMO FUNCIONA A MANUTENÇÃO E TESTE DE EQUIPAMENTOS OFFSHORE	24
3.4 CONTROLADOR LÓGICO PROGRAMÁVEL	25
3.4.1 CONCEITO E FUNCIONAMENTO	25
3.4.2 ESTRUTURA.....	26
3.4.3 LINGUAGENS DE PROGRAMAÇÃO	26
3.4.4 APLICAÇÕES	26
3.4.5 VANTAGENS.....	27
3.5 CIRCUITO HIDRÁULICO.....	27
3.5.1 BOMBAS	28
3.5.1.1 BOMBA TRIPLEX.....	28
3.5.1.2 BOMBA HIDROPNEUMÁTICA	29
3.5.2 VÁLVULAS	30
3.5.2.1 VÁLVULA SOLENOIDE	30
3.5.2.2 VÁLVULA DE ALÍVIO	31
3.5.2.3 VÁLVULA DE RETENÇÃO	32
3.5.3 MANÔMETRO	32
3.5.4 TRANSDUTOR DE PRESSÃO	33
3.5.5 FILTRO HIDRÁULICO	33
3.6 COMPUTADOR.....	34
3.7 RELÉ	34

3.8 CODESYS	35
3.9 IGNITION SCADA	35
3.9.1 BENEFÍCIOS DO SOFTWARE	36
3.10 OPEN PLATFORM COMMUNICATIONS (OPC)	36
3.10.1 ARQUITETURA UNIFICADA (OPC UA)	37
4 METODOLOGIA	38
4.1 CIRCUITO HIDRÁULICO	39
4.2 DESENVOLVIMENTO DA PROGRAMAÇÃO LADDER	41
4.3 DESENVOLVIMENTO DO SISTEMA SUPERVISÓRIO	47
4.3.1 COMUNICAÇÃO ENTRE IGNITION E CODESYS	47
4.3.1.1 CRIAÇÃO SERVIDOR OPC UA	48
4.3.1.2 CONEXÃO AO SERVIDOR PELO IGNITION	49
4.3.2 CRIAÇÃO DA INTERFACE HOMEM MÁQUINA	50
5 ANÁLISE E DISCUSSÃO DOS RESULTADOS	56
6 CONCLUSÃO	59
REFERÊNCIAS	61
DESENVOLVIMENTO E CONSTRUÇÃO DE UM PROTÓTIPO DE CONTROLADOR LÓGICO PROGRAMÁVEL E SOFTWARE COM PROGRAMAÇÃO EM LADDER UTILIZANDO TECNOLOGIAS DE CÓDIGO ABERTO	63
1 INTRODUÇÃO	72
1.1 JUSTIFICATIVA	73
2 OBJETIVOS	73
2.1 OBJETIVO GERAL	73
2.2 OBJETIVOS ESPECÍFICOS	73
2.3 ESTRUTURA DO TRABALHO	74
3 FUNDAMENTAÇÃO TEÓRICA	74
3.1 MICROCONTROLADORES	75
3.1.1 ORIGEM E FILOSOFIA	75
3.1.2 COMPONENTES PRINCIPAIS	75
3.1.3 VANTAGENS DOS MICROCONTROLADORES	76

3.2 CONTROLADOR LÓGICO PROGRAMÁVEL (CLP)	76
3.2.1 ORIGEM E FILOSOFIA	77
3.2.2 ELEMENTOS DE UM SISTEMA DE CONTROLE	77
3.2.3 COMPONENTES PRINCIPAIS DE UM CLP	77
3.2.4 VANTAGENS DOS CLPS	78
3.2.5 APLICAÇÕES	78
3.2.6 DESAFIOS E LIMITAÇÕES	78
3.2.7 CONCLUSÃO	79
3.3 ARDUINO	79
3.3.1 ORIGEM E FILOSOFIA	79
3.3.2 COMPONENTES PRINCIPAIS	79
3.3.3 CONCLUSÃO	80
3.4 LINGUAGEM LADDER	80
3.4.1 ORIGEM E FILOSOFIA	81
3.4.2 ESTRUTURA E FUNCIONAMENTO	81
3.4.3 VANTAGENS	82
3.4.4 APLICAÇÕES	82
3.4.5 DESAFIOS E LIMITAÇÕES	82
3.4.6 CONCLUSÃO	83
3.5 LINGUAGEM C++	83
3.5.1 ORIGEM E FILOSOFIA	83
3.5.2 CARACTERÍSTICAS PRINCIPAIS	84
3.5.3 APLICAÇÕES	84
3.5.4 VANTAGENS	85
3.5.5 DESAFIOS E LIMITAÇÕES	85
3.5.6 CONCLUSÃO	86
4 DESENVOLVIMENTO	86
4.1 METODOLOGIA	86
4.2 DESENVOLVIMENTO DO HARDWARE	87

4.2.1 MODELO DE REFERÊNCIA	87
4.2.1.1 SÍNTESE DAS CARACTERÍSTICAS RELEVANTES.....	87
4.2.2 ESPECIFICAÇÕES DO PROTÓTIPO.....	88
4.2.2.1 CARACTERÍSTICAS GERAIS	88
4.2.2.2 ETAPAS DO DESENVOLVIMENTO DO PROTÓTIPO	90
4.2.2.2.1 ELABORAÇÃO DO ESQUEMA ELETRÔNICO	90
4.2.2.2.2 PROJETO DA PLACA DE CIRCUITO IMPRESSO (PCB)	95
4.2.2.2.3 SELEÇÃO DOS COMPONENTES.....	97
4.2.2.2.4 PRODUÇÃO DA PCB.....	98
4.2.2.2.5 DESENVOLVIMENTO DA CARCAÇA 3D	98
4.2.3 MONTAGEM FINAL E VERIFICAÇÃO.....	100
4.2.4 GRAVAÇÃO DO BOOTLOADER NO ATMEGA328P	101
4.3 IMPLEMENTAÇÃO DO SOFTWARE.....	109
4.3.1 PROGRAMAÇÃO EM C COM ARDUINO IDE.....	109
LISTING 4.1 – ENTRADAS ANALÓGICAS E SAÍDAS PWM COM ARDUINO	111
LISTING 4.2 – ENTRADAS DIGITAIS E ACIONAMENTO DE RELÉS	113
4.3.2 PROGRAMAÇÃO EM LADDER COM LDMICRO	114
5 TESTES E AVALIAÇÃO DO PROTÓTIPO	118
5.1 TESTES FUNCIONAIS.....	118
5.1.1 ENTRADAS DIGITAIS E SAÍDAS A RELÉ	118
5.1.2 ENTRADAS ANALÓGICAS (0–10V).....	119
5.1.3 SAÍDAS PWM	119
5.1.4 PROGRAMAÇÃO EM LADDER COM LDMICRO	119
5.2 AVALIAÇÃO DOS RESULTADOS.....	119
5.2.1 COMPARATIVO DE CUSTOS.....	120
6 RESULTADOS E DISCUSSÃO	121
6.1 DESEMPENHO FUNCIONAL	121
6.2 COMPARAÇÃO COM CLPS COMERCIAIS	121
6.3 LIMITAÇÕES E DESAFIOS	121

6.4 ANÁLISE GERAL	122
7 CONCLUSÃO E TRABALHOS FUTUROS	122
7.1 CONCLUSÃO	122
CONSIDERAÇÕES SOBRE O APRENDIZADO ADQUIRIDO	123
7.2 TRABALHOS FUTUROS.....	124
REFERÊNCIAS	125
APLICAÇÃO DE VISÃO COMPUTACIONAL PARA DETECÇÃO E MONITORAMENTO DE VEÍCULOS	127
1 INTRODUÇÃO.....	135
1.1 CONSIDERAÇÕES INICIAIS	135
1.2 PROBLEMA.....	135
1.3 OBJETIVOS GERAIS	136
1.4 OBJETIVOS ESPECÍFICOS.....	136
1.5 JUSTIFICATIVA.....	136
2 FUNDAMENTAÇÃO TEÓRICA.....	137
2.1 PROCESSAMENTO DIGITAL DE IMAGENS	137
2.1.1 ETAPAS DO PROCESSAMENTO DIGITAL DE IMAGENS.....	137
2.1.1.1 AQUISIÇÃO DE IMAGEM	138
2.1.1.2 PRÉ-PROCESSAMENTO DE IMAGENS	138
2.1.1.3 SEGMENTAÇÃO	139
2.1.1.4 EXTRAÇÃO DE CARACTERÍSTICAS	139
2.1.1.5 RECONHECIMENTO E INTERPRETAÇÃO	139
2.1.2 APLICABILIDADE EM VISÃO COMPUTACIONAL.....	140
2.2 DEEP LEARNING	141
2.2.1 TRANSFERÊNCIA DE APRENDIZADO	142
2.3 VISÃO COMPUTACIONAL	142
2.3.1 AVANÇOS HISTÓRICOS	142
2.4 REDES NEURAIS CONVOLUCIONAIS.....	143
2.4.1 ETAPAS DE UMA CNN.....	144
2.5 ALGORITMO YOU ONLY LOOK ONCE	146

2.5.1 ETAPAS DE FUNCIONAMENTO	147
2.6 BIBLIOTECA OPEN SOURCE COMPUTER VISION.....	148
2.7 RASTREAMENTO DE OBJETOS	148
2.7.1 INTEGRAÇÃO DE DETECÇÃO E RASTREAMENTO	148
2.7.2 TÉCNICAS DE RASTREAMENTO DE OBJETOS.....	149
2.7.3 O ALGORITMO DEEPSORT	149
2.7.4 O ALGORITMO STRONGSORT	150
2.7.5 APLICAÇÕES NO MONITORAMENTO DE VEÍCULOS.....	151
2.7.6 CONSIDERAÇÕES FINAIS.....	151
2.8 SISTEMA DE DETECÇÃO DE VEÍCULOS	152
2.8.1 ARQUITETURA DE SISTEMAS DE DETECÇÃO DE VEÍCULOS.....	152
2.8.2 SISTEMAS EXISTENTES.....	153
2.8.2.1 SYDNEY COORDINATED ADAPTIVE TRAFFIC SYSTEM	153
2.8.3 DESAFIOS E GARGALOS	153
2.8.4 CONSIDERAÇÕES FINAIS.....	155
2.9 MÉTRICAS DE AVALIAÇÃO PARA DETECÇÃO E RASTREAMENTO	156
2.9.1 PRECISÃO	156
2.9.2 RECALL	156
2.9.3 F1-SCORE	157
2.9.4 MEAN AVERAGE PRECISION (MAP)	157
2.9.5 TAXA DE FALSOS POSITIVOS (FALSE POSITIVE RATE - FPR)	157
2.9.6 IOU (INTERSECTION OVER UNION)	158
2.10 DATASETS PARA TREINAMENTO E VALIDAÇÃO	158
2.10.1 KITTI VISION BENCHMARK SUITE	158
2.10.2 COCO (COMMON OBJECTS IN CONTEXT).....	158
2.10.3 MOTCHALLENGE	159
2.10.4 MIO-TCD (TRAFFIC CAMERA DATASET)	159
2.11 CONSIDERAÇÕES FINAIS.....	159
3 METODOLOGIA	160

3.1 DEFINIÇÃO DOS REQUISITOS DE FUNCIONAMENTO	160
3.2 AQUISIÇÃO DE IMAGENS	161
3.3 PREPARAÇÃO DOS DADOS	161
3.4 PRÉ-PROCESSAMENTO DE IMAGENS.....	161
3.4.1 TREINAMENTO DA REDE YOLO.....	163
3.5 DETECÇÃO DE VEÍCULOS	163
3.6 RASTREAMENTO DE VEÍCULOS.....	164
3.7 SIMULAÇÃO DE CONTROLE DE TRÁFEGO COM SEMÁFORO INTELIGENTE (SUMO)	166
4 RESULTADOS E DISCUSSÃO	169
4.1 RESULTADOS.....	169
4.1.1 RESULTADOS DO TREINAMENTO	169
4.1.2 RESULTADOS DA VALIDAÇÃO DO MODELO	171
4.1.3 RESULTADOS DO RASTREAMENTO	173
4.1.4 RESULTADOS DA SIMULAÇÃO EM SUMO	175
4.2 DISCUSSÃO	176
4.2.1 DESEMPENHO DO TREINAMENTO INICIAL	176
4.2.2 REFINAMENTO POR TRANSFERÊNCIA DE APRENDIZADO	177
4.2.3 ANÁLISE DO RASTREAMENTO	178
4.2.4DISCUSSÃO DA SIMULAÇÃO NO SUMO	179
4.2.4.1 LEITURA DETALHADA DOS GRÁFICOS.....	179
4.2.4.2 INTERPRETAÇÃO DOS GANHOS.....	179
4.2.4.3 ASPECTOS AINDA NÃO EXPLORADOS.....	179
4.2.4.4 IMPLICAÇÕES PARA AUTOMAÇÃO DE TRÁFEGO	180
5 CONCLUSÃO E TRABALHOS FUTUROS	180
5.1 CONCLUSÃO	180
5.2 TRABALHOS FUTUROS.....	181
REFERÊNCIAS	183
APÊNDICES.....	186

1 INTRODUÇÃO

A necessidade crescente de sistemas automatizados em ambientes industriais exige do mercado soluções tecnológicas que garantam, além de operação eficaz, segurança para os trabalhadores e os equipamentos. Em relação a isso, os sistemas de supervisão se mostram como uma ferramenta essencial para o gerenciamento e acompanhamento de processos críticos, através da visualização em tempo real, coleta de informações e intervenção imediata em operações. Uma atenção especial deve ser dada para esse tipo de observação em ambientes que controlam pressão, visto que, segundo dados da análise de impacto regulatório da NR13 (GOV, 2021) no período entre janeiro de 2016 e julho de 2020, foram registrados 9.347 acidentes relacionados a trabalho com pressão, dos quais 49 foram fatais, assim indicando o quão intrigante é o sistema e os perigos que variações inesperadas na pressão podem representar.

Dessa forma, manter a integridade do processo requer um monitoramento constante, capaz de antecipar falhas e garantir que as operações continuem sem interrupções. Neste cenário específico mencionado anteriormente, pode-se sugerir o desenvolvimento de um sistema de monitoramento para um ambiente de teste sob pressão que priorize tanto aspectos relacionados à segurança quanto à eficiência do processo (SPAGNOLETTI, 2024). Esse sistema terá como objetivo acompanhar indicadores críticos como pressão, temperatura e fluxos, enquanto disponibiliza uma interface acessível ao usuário para garantir a rápida identificação de falhas ou a tomada imediata de decisões em situações emergenciais. A segurança é assegurada através de sistemas de alarmes personalizados pelo usuário e desativação automática em emergências, já a eficiência decorre da otimização do uso dos recursos com menos paralisações não planejadas.

A indústria do petróleo possui papel estratégico no desenvolvimento econômico e social do Brasil, sendo uma das principais responsáveis pela geração de energia e pela sustentação da matriz econômica nacional. O país ocupa posição de destaque no cenário mundial, especialmente em função das reservas localizadas no pré-sal, que impulsionaram significativamente a produção de petróleo e gás natural nas últimas décadas. Além de atender grande parte da demanda interna por combustíveis e derivados, o setor é um dos maiores geradores de receitas para o Estado, por meio de royalties e tributos, além de contribuir expressivamente para a balança comercial através da exportação de petróleo bruto. (IBP, 2023)

Figura 1 – Conjunto ANM, BAP e MCV



Fonte: (SIQUEIRA, 2021)

No âmbito social, a cadeia produtiva do petróleo e gás é responsável pela geração de milhares de empregos diretos e indiretos, fomentando a indústria nacional de bens e serviços e estimulando o desenvolvimento de tecnologia e inovação. Em especial, o segmento offshore brasileiro consolidou-se como referência mundial em engenharia e exploração em águas profundas e ultraprofundas, o que ressalta a importância do investimento em pesquisa, capacitação técnica e adoção de soluções automatizadas para aumentar a eficiência e a segurança das operações. (IBP, 2023)

No artigo de Froats (2023), é afirmado que a realização de testes hidrostáticos desempenha um papel essencial na indústria do petróleo, uma vez que equipamentos submarinos e de superfície estão constantemente expostos a condições extremas de pressão, temperatura e ambiente corrosivo. Esses testes têm como objetivo verificar a integridade estrutural e a vedação de componentes críticos, como árvores de natal, manifolds, válvulas e linhas de produção, garantindo sua segurança e confiabilidade antes da instalação ou após manutenções.

Entretanto, a execução desses ensaios apresenta diversas dificuldades práticas. Entre elas, destacam-se a necessidade de infraestrutura robusta para geração de pressões elevadas, o controle rigoroso de variáveis como temperatura e contaminação do fluido, além da garantia da segurança dos operadores envolvidos, uma vez que falhas podem resultar em acidentes graves. Outro desafio está relacionado ao alto custo operacional e ao tempo necessário para a realização dos testes, que muitas vezes demandam a paralisação parcial de atividades produtivas.

Segundo Spagnoletti (2024), a indústria do petróleo sofre com os riscos relacionados aos trabalhos com pressão em diversas partes de seu processo de produção e manutenção de equipamentos. Durante as manutenções dos equipamentos de produção de petróleo, como por exemplo a árvore de natal molhada (ANM), base adaptadora de produção (BAP) e o módulo de conexão vertical (MCV) indicados na Figura 1 ou o manifold indicado na Figura 2, é necessário que sejam feitos diversos testes hidrostáticos para garantir o funcionamento após manutenção, conforme imposto nas normas da American Petroleum Institute (API).

Figura 2 – Manifold



Fonte: (GABRIEL, 2014)

Dessa forma, esse estudo tem como objetivo desenvolver e analisar a viabilidade de um ambiente de teste com pressão utilizado durante as manutenções de equipamentos submarinos de produção de petróleo, para que seja possível minimizar os riscos relacionados ao trabalho com pressão. Para tal, foi pensado um sistema para realizar o controle da atividade remotamente, para garantir a segurança durante a execução.

2 OBJETIVOS

O texto a seguir descreve os objetivos do trabalho.

2.1 OBJETIVO GERAL

Como objetivo geral deste trabalho, destaca-se a elaboração de um sistema supervisório para ambientes de teste com pressão que aumente a segurança na execução da atividade por meio do isolamento da área e controle remoto das válvulas e bombas necessárias para o teste.

2.2 OBJETIVOS ESPECÍFICOS

- a) Gerenciamento da UTF, controlada através de uma aplicação, e gerenciada localmente por um CLP;
- b) Gerenciamento do acesso à área de teste controlado por meio de uma aplicação, e gerenciado via CLP;
- c) Uso de CLP para gerenciamento de equipamento;
- d) Elaboração de uma interface de controle supervisão dos equipamentos inseridos no sistema, que possa acionar, desativar e exibir o status atualizado do equipamento.

2.3 JUSTIFICATIVA

Para a execução de um sistema de controle automático em testes industriais, consideram-se, para tal, três fatores principais: proteção, supervisão de ações e eficiência.

Isolar a área de teste representa um fator imprescindível para proteção da saúde ocupacional desde o início, conforme a hierarquia de controle de risco na figura 2. Quando o acesso é limitado, o acesso direto ao teste sendo realizado por pessoas é restrito, assim a probabilidade de acidentes decorrentes dos testes é limitada. Na área industrial, este tema é de suma importância devido à grande manipulação com grandes máquinas, grandes pressões ou substâncias perigosas, que podem prejudicar a saúde ocupacional.

Para supervisionar remotamente os testes, o sistema supervisão é a melhor opção. Com o sistema, é possível acompanhar e controlar o teste remotamente, de modo centralizado e em tempo real, aumentando a qualidade da operação e reduzindo os erros humanos. Os controladores remotos asseguram segurança e eficiência para testes, sem necessidade de presença física e ininterrupta, possibilitando reações mais rápidas a imprevistos.

Finalmente, a automação dos processos garante aumento de produtividade através da limitação no deslocamento constante e na manipulação manual que tomam tempo. Além do aumento do tempo de testes, esta eliminação também remove dispêndios de operação, propiciando melhor utilização dos recursos. Com menos intervenção humana e um controle mais assentado dos processos, a ação da empresa tem aumento do desempenho e melhor efetividade com menos esforço.

Destarte, a combinação entre maior segurança, possibilidade de controle remoto e aumento da eficiência torna a aplicação de sistemas automatizados e supervisórios um recurso indispensável para a evolução da realização de testes em condições industriais.

Figura 3 – Hierarquia de Controle de Riscos



Fonte: (TEODORO, 2022)

3 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo serão apresentados os fundamentos teóricos relevantes para o desenvolvimento do projeto. Primeiramente será explicado os equipamentos offshore e seus processos de manutenção e teste e seus riscos. Em seguida, será mostrado em detalhes o funcionamento de um sistema de teste com pressão tradicional, onde não é utilizada a automação para redução de riscos e aumento de confiabilidade e eficiência e, por fim, serão detalhados os componentes necessários para o desenvolvimento deste projeto.

3.1 SISTEMA MARÍTIMO DE PRODUÇÃO DE PETRÓLEO

Para auxiliar na produção e extração de petróleo, são utilizados equipamentos submarinos. A seguir, será abordado a respeito dos sistemas submarinos, como também suas funções e sua importância.

Bai e Bai (2010) definem um sistema submarino de produção da seguinte forma: Um sistema submarino de produção é composto por poço submarino completado, sistema de cabeça de poço submarino, árvore de natal molhada (ANM), interligação submarina com sistema de escoamento, equipamentos submarinos e facilidades de controle para operação de poço. Ele pode variar, em complexidade, de um único poço satélite ligado por uma linha de fluxo a uma plataforma fixa, uma unidade flutuante de produção, armazenamento e transferência (FPSO) ou a facilidades em terra; a um sistema composto por vários poços conectados a um manifold, em configuração template ou clustered, com a produção sendo transferida para uma unidade fixa ou flutuante, ou ainda diretamente para facilidades em terra. (BAI, 2010, p. 5).

Os equipamentos submarinos do conjunto ANM são utilizados no fundo do mar e possuem válvulas conectadas ao poço e à unidade flutuante de produção (FPSO). É através dessas válvulas que irá ocorrer o fluxo de produção de óleo e gás, do poço para a superfície, permitindo também a injeção de fluido e gás da superfície para o poço quando necessário.

O conjunto ANM é considerado o principal equipamento de segurança de um poço submarino; ele é composto pelas seguintes partes: BAP (Base Adaptadora de Produção), Tubing Hanger (Suspensor de Coluna), ANM (Árvore de Natal Molhada), Tree Cap (Capa da árvore de Natal) e MCV (Módulo de Conexão Vertical).

Para instalar ou retirar os equipamentos descritos acima, são utilizadas as ferramentas FIBAP (Ferramenta de Instalação da BAP), THRT (Ferramenta de Instalação do Tubing Hanger) e TRT (Ferramenta de Instalação da ANM e Tree Cap). (BAI, 2010)

3.2 CONCEITOS DE FUNCIONAMENTO DOS EQUIPAMENTOS E FERRAMENTAS SUBMARINAS

Cada equipamento e ferramenta mencionados anteriormente possuem uma função distinta, como por exemplo, realizar a integração entre diferentes equipamentos na estrutura do poço, controlar o fluxo do poço visando à produção ou injeção, realizar a instalação de equipamentos através do controle do sistema de acoplamento por meio de suas funções hidráulicas.

Todos os equipamentos e ferramentas utilizam os mesmos princípios de funcionamento para atingir suas determinadas tarefas. São utilizadas válvulas e pistões acionados por linhas hidráulicas alimentadas por unidades de pressão a bordo da plataforma de produção ou, em caso de falha da alimentação externa, utilizando os acumuladores presentes nos próprios equipamentos e ferramentas. Assim, é possível realizar o acionamento das válvulas responsáveis pelo controle do fluxo de produção ou injeção e o acionamento dos pistões responsáveis por realizar as conexões entre os equipamentos. Logo, deve ser considerado que o funcionamento de todos os componentes do sistema de completação de um poço offshore depende da capacidade de vedação das suas linhas hidrostáticas; portanto, os testes hidrostáticos realizados durante as manutenções são de extrema importância para garantir o correto funcionamento dos equipamentos e ferramentas em campo, assim drasticamente reduzindo o risco a danos ambientais e às vidas humanas devido a falhas. (BAI, 2010)

3.3 COMO FUNCIONA A MANUTENÇÃO E TESTE DE EQUIPAMENTOS OFFSHORE

A exploração do petróleo e gás natural em alto-mar apresenta vários desafios, como a necessidade de resistência à corrosão em ambiente submarino por décadas e a necessidade de alta confiabilidade na capacidade de vedação dos equipamentos trabalhando com grandes pressões internas. Tendo em vista as condições extremas de funcionamento dos equipamentos e ferramentas do sistema de produção de petróleo e a necessidade de um grande grau de confiabilidade, faz-se necessária na indústria de óleo e gás uma cultura de manutenções preventivas e corretivas extremamente robusta.

Durante as manutenções gerais dos equipamentos, é feita a desmontagem completa do equipamento e suas peças são submetidas a inspeções relevantes, variando de inspeções visuais a inspeções por radiografia e partícula magnética. Caso sejam encontrados desvios, é feito o reparo, quando possível, ou a substituição por nova peça. Após todos os componentes estarem prontos, será realizada a remontagem do equipamento utilizando novas unidades para todos os fixadores e elementos de vedação. A última etapa do processo é a realização dos testes funcionais e de vedação, onde ele será posicionado em local isolado por barreiras de engenharia e terá suas linhas de funções conectadas a uma bomba externa, denominada de unidade de teste e flushing (UTF), para controlar seus acionamentos. Assim se faz possível, através de confirmação visual e monitoramento por transdutores de pressão, garantir o correto funcionamento e a capacidade de vedação de todas as linhas hidráulicas do equipamento. (ALMEIDA, 2020)

Existem outros níveis de manutenção mais simples além daquele descrito acima, onde alguns dos passos descritos anteriormente não são feitos ou são feitos de maneira reduzida. Aquilo que se mantém constante em todos os níveis de manutenção é a necessidade da realização dos testes funcionais e de vedação ao final de cada manutenção.

3.4 CONTROLADOR LÓGICO PROGRAMÁVEL

Os controladores lógicos programáveis desempenham um papel essencial na automação industrial, pois tornam possível não apenas comandar, mas também acompanhar o funcionamento de máquinas e processos. Fundamentalmente, eles operam a partir de um programa previamente armazenado em memória, responsável por interpretar sinais vindos de sensores, chaves ou botoeiras. Com base nesse processamento, o equipamento gera sinais de saída capazes de acionar diferentes atuadores, como motores elétricos e válvulas, garantindo que todo o sistema funcione de maneira integrada e eficiente. (FREITAS, 2023)

3.4.1 CONCEITO E FUNCIONAMENTO

O CLP é um computador industrial projetado para operar em ambientes adversos (com poeira, vibrações, temperaturas elevadas, etc.) e executar tarefas de controle lógico de maneira contínua e determinística. Seu funcionamento baseia-se em um ciclo de varredura (scan), que consiste em quatro etapas principais:

- **Leitura das Entradas:** O CLP lê os sinais provenientes dos sensores e dispositivos de entrada conectados, como: sensores de presença, chaves de fim de curso, botoeiras, entre outros).

- **Processamento do Programa:** Com base nas instruções do programa armazenado em sua memória, o CLP processa as informações das entradas e define os estados das saídas.
- **Atualização das Saídas:** O CLP envia sinais para os atuadores (motores, válvulas, relés, entre outros.), ativando ou desativando-os conforme a lógica definida.
- **Autodiagnóstico e Comunicação:** Realiza verificações internas e comunica-se com outros dispositivos, se necessário.

3.4.2 ESTRUTURA

Um CLP é composto pelos seguintes elementos:

- **Unidade Central de Processamento (CPU):** É o "cérebro" do sistema, responsável por interpretar e executar o programa.
- **Memória:** Armazena o programa do usuário e os dados necessários à operação.
- **Módulos de Entrada/Saída (E/S):** Fazem a interface entre o CLP e os dispositivos do processo. Entradas recebem sinais (digitais ou analógicos), e saídas enviam comandos.
- **Fonte de Alimentação:** Fornece energia ao sistema.
- **Interface de Programação:** Permite que o usuário desenvolva, edite e transfira o programa ao CLP, geralmente por meio de um software específico.

3.4.3 LINGUAGENS DE PROGRAMAÇÃO

Os CLPs podem ser programados em diferentes linguagens padronizadas pela norma IEC 61131-3, sendo as mais utilizadas:

- **Ladder:** É um modelo de programação gráfica aplicada ao desenvolvimento de softwares dos Controladores Lógicos Programáveis. Resumidamente, é responsável por informar quais ações devem ser realizadas a partir do comando solicitado. Essa linguagem gráfica é baseada em diagramas de relés, amplamente utilizada por sua semelhança com esquemas elétricos. (ABB, 2025)
- **Texto Estruturado:** Semelhante a linguagens de programação convencionais (C ou C++), ideal para cálculos e estruturas complexas.
- **Bloco de Funções e Diagrama de Sequência de Funções:** O diagrama de blocos é uma representação visual de um sistema, composta por retângulos (os blocos) conectados por setas que indicam o fluxo de informação, sinal ou processo. Cada bloco representa uma função ou operação específica. A proposta é sintetizar estruturas complexas de forma clara, reduzindo a ambiguidade na comunicação técnica. Em vez de mostrar todos os detalhes internos de um sistema, ele apresenta apenas os elementos funcionais principais e como se conectam. Sendo assim, utilizadas para representar processos contínuos ou sequenciais de forma gráfica e intuitiva. (FM2S, 2025)

3.4.4 APLICAÇÕES

Os CLPs estão presentes em diversos segmentos industriais, como:

- Linhas de produção automatizadas;
- Sistemas de transporte e embalagem;
- Controle de motores e máquinas;
- Processos de envase e dosagem;
- Sistemas de tratamento de água e efluentes;

3.4.5 VANTAGENS

- Flexibilidade na modificação de processos, sem a necessidade de alterações físicas no painel elétrico;
- Confiabilidade em ambientes industriais hostis;
- Facilidade de diagnóstico e manutenção, por meio de softwares de monitoração;
- Redução de espaço físico e cabeamento, comparado com sistemas baseados em relés;

Figura 4 – Exemplo de Controlador Lógico Programável



Fonte: (ROCKWELL, 2024)

3.5 CIRCUITO HIDRÁULICO

Um circuito hidráulico é formado por um conjunto de componentes que trabalham com um fluido sob pressão, normalmente óleo, para gerar força e movimento. Nesse sistema, a bomba é responsável por movimentar o fluido, que é conduzido por tubulações até válvulas e atuadores, como cilindros ou motores hidráulicos. Esses atuadores realizam o trabalho desejado, como empurrar, levantar ou girar objetos, dependendo da aplicação.

Portanto, para a realização do projeto, utilizaremos os seguintes componentes:

3.5.1 BOMBAS

3.5.1.1 BOMBA TRIPLEX

As Bombas Triplex são amplamente conhecidas por seu design robusto e eficiência na geração de alta pressão. Essas bombas operam utilizando três pistões ou êmbolos, que se movem alternadamente para gerar um fluxo contínuo de líquido sob alta pressão. Esse movimento alternado é o responsável pela capacidade da bomba de manter uma pressão constante e elevada.

Cada um dos três pistões de uma bomba triplex é acionado por um eixo de manivelas. Quando um pistão é empurrado para frente, o líquido é forçado para fora do cilindro e para dentro da linha de descarga. Enquanto isso, outros dois pistões estão em seus ciclos de sucção e compressão, garantindo que o fluxo do líquido não seja interrompido. Esta configuração garante um funcionamento suave e contínuo.

Um componente essencial no funcionamento das Bombas Triplex é o engate rápido duas polegadas, que permite uma conexão rápida e segura dos sistemas de mangueiras, contribuindo para a eficiência operacional. Além disso, outras válvulas e componentes trabalham em conjunto para garantir a segurança e a eficiência das bombas. São comumente empregados feitos materiais resistentes à corrosão, como aço inoxidável ou ligas especiais, garantindo sua durabilidade em diversas condições operacionais.

Componentes principais de uma bomba triplex:

- Cabeçote e pistões: Geralmente fabricados em latão ou aço inox, dependendo das exigências de resistência química e sanitária;
- Eixo de manivelas: Converte o movimento rotativo em alternado para ativar os pistões;
- Válvulas: Acionadas por diferença de pressão, controlam o fluxo em cada pistão;
- Vedações especiais: Evitam vazamentos entre partes móveis e suportam altas pressões;

Figura 5 – Exemplo de uma bomba triplex



Fonte: (LEWA, 2025)

A bomba triplex se destaca por sua eficiência em sistemas que exigem alta pressão e funcionamento constante. Devido ao uso de três pistões, anteriormente citados, operando de maneira alternada, ela consegue manter o fluxo de fluido mais uniforme, o que favorece o desempenho do sistema como um todo. Sua estrutura foi projetada para suportar longos períodos de trabalho sem perda de rendimento, mesmo em condições intensas. Além disso, o acesso facilitado aos componentes internos torna as manutenções mais rápidas e menos complexas. Por essas características, é amplamente utilizada em aplicações industriais que exigem força, resistência e confiabilidade. (JYVALVULAS, 2025)

3.5.1.2 BOMBA HIDROPNEUMÁTICA

As bombas pneumáticas para líquidos, também conhecidas como bombas hidropneumáticas, operam utilizando ar comprimido ou gás inerte (como o nitrogênio) para gerar pressões hidráulicas elevadas. Elas funcionam com base em um sistema de pistões: um pistão de maior diâmetro é movido pelo ar comprimido e transfere essa força a um pistão menor, que comprime o fluido hidráulico. Essa diferença de área entre os pistões cria um efeito multiplicador de pressão, permitindo que uma pressão de entrada relativamente baixa gere uma pressão de saída muito mais alta no fluido.

Esse tipo de bomba se destaca por manter a pressão constante no sistema sem consumir energia continuamente, já que só opera quando há necessidade de reposição de pressão. Além disso, o controle de velocidade e de resposta é altamente preciso, o que permite aumentar ou manter a pressão de forma gradual e controlada. Outro aspecto relevante é que essas bombas não requerem lubrificação contínua e operam com baixo ruído, o que contribui para uma manutenção mais simples e maior confiabilidade.

Figura 6 – Exemplo de uma bomba hidropneumática



Fonte: (FLUTROL, 2025)

As bombas hidropneumáticas são utilizadas principalmente em sistemas onde se exige alta pressão com segurança, controle fino de pressão e confiabilidade operacional, como em testes de pressão, injeções controladas de fluido e acionamentos hidráulicos de precisão. (FLUTROL, 2025)

Componentes principais de uma bomba hidropneumática:

- Pistão pneumático: É acionado pelo ar comprimido e gera a força que movimenta o pistão hidráulico. Possui maior diâmetro para garantir força suficiente;
- Pistão hidráulico: Recebe a força do pistão pneumático e pressuriza o fluido. Sua área menor permite multiplicar a pressão;
- Câmara de ar: Espaço onde o ar comprimido atua. Controla os ciclos da bomba por meio de válvulas de entrada e escape de ar;
- Câmara hidráulica: Onde o fluido é comprimido e enviado sob pressão ao sistema externo;
- Válvulas de retenção: Controlam a direção do fluido, impedindo o retorno e garantindo fluxo correto;

3.5.2 VÁLVULAS

3.5.2.1 VÁLVULA SOLENOIDE

É um dispositivo eletromecânico usado para controlar o fluxo de fluidos (líquidos ou gases) em um sistema. Ela funciona por meio de uma bobina (solenóide), que, quando energizada, cria um campo magnético e aciona o núcleo móvel, abrindo ou fechando o fluxo de fluido na válvula.

Figura 7 – Exemplo de válvula solenóide



Fonte: (JEFFERSON.IND, 2024)

3.5.2.2 VÁLVULA DE ALÍVIO

As válvulas de alívio hidráulicas atuam como dispositivos de segurança nos sistemas hidráulicos, impedindo que a pressão ultrapasse os limites estabelecidos e cause danos aos componentes. Quando a pressão do sistema fica acima do valor ideal, a válvula se abre automaticamente, permitindo que parte do fluido seja desviada para o reservatório ou para outra linha de menor pressão. Isso ajuda a equilibrar o sistema e evitar sobrecargas.

Seu funcionamento é simples, mas eficiente: uma mola mantém a válvula fechada até que a pressão do fluido se torne maior que a força da mola. Nesse momento, a válvula se abre e libera o excesso de fluido. Assim que a pressão volta ao valor seguro, a mola empurra o mecanismo de volta para a posição fechada, restabelecendo o funcionamento normal do circuito. (HYDRACO, 2025)

Figura 8 – Exemplo de válvula de alívio



Fonte: (HYDRACO, 2025)

3.5.2.3 VÁLVULA DE RETENÇÃO

As válvulas de retenção hidráulicas têm a função de garantir que o fluido circule em apenas uma direção dentro do sistema. Elas operam de forma automática e independente, ou seja, não precisam de comandos externos para funcionar. Quando o fluido se move no sentido certo, a válvula se abre naturalmente. Mas, se houver alguma tentativa de retorno do fluido, ela se fecha de imediato, impedindo o refluxo e protegendo o sistema.

O funcionamento é simples. Dentro da válvula há uma peça móvel, como uma esfera ou um pistão, que se desloca com a força do fluido quando ele flui corretamente. Caso a direção do fluxo se inverta, essa peça é pressionada contra a vedação da válvula, bloqueando completamente a passagem. Em muitos modelos, há também uma mola que ajuda a manter a válvula fechada quando não há pressão suficiente, o que oferece ainda mais segurança, especialmente em momentos de parada ou baixa atividade do sistema. (HYDRACOR, 2025)

Figura 9 – Exemplo de válvula de retenção



Fonte: (HYDRACOR, 2025)

3.5.3 MANÔMETRO

Os manômetros industriais são equipamentos essenciais que permitem acompanhar a pressão de fluidos, gases ou vapores em sistemas industriais, garantindo a segurança do funcionamento e otimizando a eficiência dos processos.

Esses dispositivos exibem visualmente a pressão interna de tubulações ou equipamentos por meio de mostrador analógico ou visor digital. Eles ajudam a identificar rapidamente pressões fora dos limites, antecipando problemas como falhas operacionais, vazamentos ou acidentes.

Figura 10 – Exemplo de manômetro



Fonte: (NAUCO, 2025)

Em modelos tradicionais, como os tubo Bourdon, a pressão atua sobre um elemento metálico flexível, normalmente em formato de "C" ou espiral, que se deforma com o aumento da pressão. Esse movimento mecânico é transferido a um ponteiro que indica o valor da pressão em escala calibrada.

Já os modelos digitais usam sensores capacitivos para converter pressão em sinal elétrico, exibindo o valor em visor digital com precisão e permitindo integração com sistemas automáticos. (NAUCO, 2025)

3.5.4 TRANSDUTOR DE PRESSÃO

É um dispositivo que converte a pressão de um fluido em um sinal elétrico. Utilizado em diversas áreas, como automação industrial e dispositivos médicos, ele mede a deformação causada pela pressão e transforma essa variação em um sinal que pode ser monitorado e controlado por sistemas eletrônicos.

Figura 11 – Exemplo de Transdutor de Pressão



Fonte: (DIGITROL, 2024)

3.5.5 FILTRO HIDRÁULICO

Nos sistemas hidráulicos industriais, o filtro hidráulico desempenha um papel essencial ao garantir a limpeza do fluido que circula no circuito. Ele é responsável por reter impurezas e partículas sólidas

que, se não removidas, podem causar danos aos componentes do sistema, como bombas, válvulas e cilindros, reduzindo a eficiência e a vida útil dos equipamentos.

Figura 12 – Exemplo de filtro hidráulico



Fonte: (NEWTEC, 2025)

O filtro funciona fazendo com que o fluido passe por um elemento filtrante que retém as partículas contaminantes. Essa filtragem pode ocorrer em diferentes pontos do sistema: na sucção da bomba, na linha de pressão ou no retorno do fluido, dependendo das necessidades específicas do equipamento e do processo.

A escolha do filtro adequado deve considerar o tipo de fluido, a quantidade e o tipo de contaminantes presentes, além das condições operacionais, como pressão e temperatura. Ademais, é fundamental realizar a manutenção periódica do filtro, com limpeza ou substituição do elemento filtrante, para garantir que o sistema continue funcionando de maneira eficiente e evitar falhas inesperadas.

3.6 COMPUTADOR

É o dispositivo principal que processa informações e controla outros equipamentos no sistema. Ele pode rodar softwares de supervisão, controle, e coleta de dados, além de ser usado para programação e interface com CLPs (Controladores Lógicos Programáveis) e sistemas SCADA (Supervisory Control and Data Acquisition).

3.7 RELÉ

O relé é um dispositivo eletromecânico que funciona como um interruptor controlado por um circuito elétrico. É muito usado para controle de grandes cargas elétricas com baixa potência de controle.

Figura 13 – Exemplo de Relé

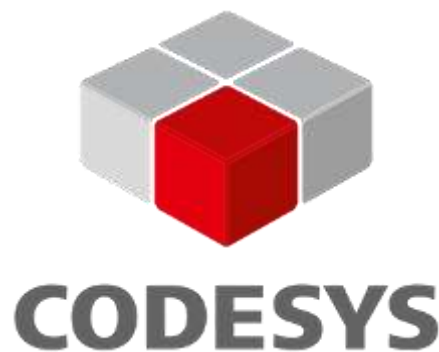


Fonte: (WEG, 2024)

3.8 CODESYS

O CODESYS, criado pela 3S-Smart Software Solutions em 1994, é uma ferramenta de desenvolvimento essencial para a automação industrial. Sua missão é padronizar a programação de controladores lógicos programáveis (CLPs), oferecendo uma plataforma universal que facilita a programação independente do fabricante. Essa uniformidade é importante, pois permite que engenheiros e técnicos se adaptem rapidamente a diferentes sistemas, sem a necessidade de aprender interfaces específicas para cada marca de CLP. (INETEC, 2025)

Figura 14 – CODESYS



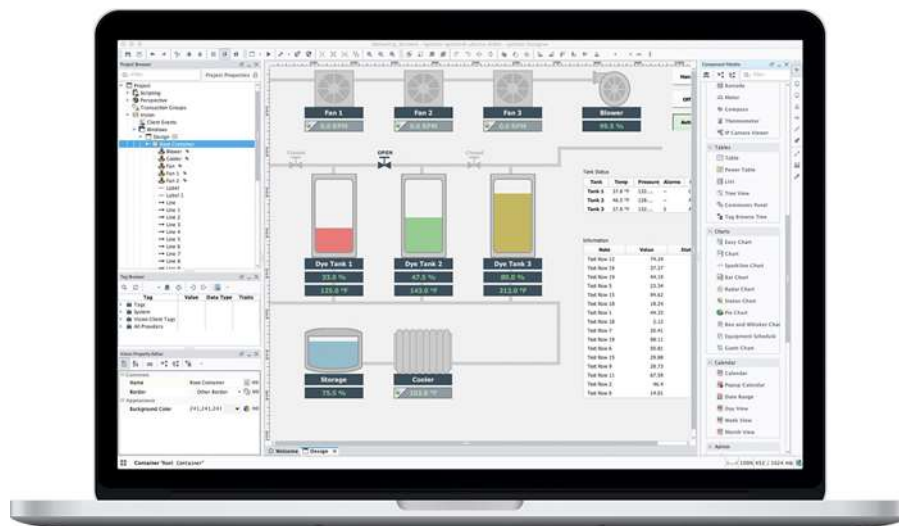
Fonte: (INETEC, 2025)

3.9 IGNITION SCADA

Dentro do conjunto de ferramentas de automação industrial estão os sistemas de Controle de Supervisão e Aquisição de Dados, ou SCADA. Os sistemas SCADA são soluções de software projetadas para conectar ativos físicos a bancos de dados digitais, processar e gerenciar grandes volumes de dados operacionais e permitir que os usuários monitorem o desempenho dos ativos e os processos de controle. A Engineering Industries eXcellence tem parceria com a Inductive Automation para fornecer soluções SCADA inovadoras para as necessidades de fabricação modernas de hoje. A plataforma

Ignition SCADA da Inductive Automation combina um modelo de licenciamento ilimitado com implantação instantânea baseada na Web, bem como um conjunto de ferramentas líder do setor para controle de supervisão e aquisição de dados, tudo em uma plataforma universal aberta e dimensionável. (GROUP, 2025)

Figura 15 – Ignition SCADA



Fonte: (GROUP, 2025)

3.9.1 BENEFÍCIOS DO SOFTWARE

- Aumentar a visibilidade em tempo real das operações;
- Melhorar a transparência e o controle do processo;
- Otimizar a eficiência de processos, pessoas e máquinas;
- Facilitar o fluxo de trabalho contínuo da manufatura;
- Impulsionar a tomada de decisões orientada por dados;
- Aumentar a interoperabilidade do sistema;

3.10 OPEN PLATFORM COMMUNICATIONS (OPC)

É um protocolo de comunicação industrial que visa facilitar a troca de informação entre os componentes físicos entre um sistema industrial. O padrão OPC representa um marco significativo no desenvolvimento de sistemas de automação industrial, sendo reconhecido como um dos principais protocolos de interoperabilidade para a troca segura e confiável de dados entre dispositivos, sistemas e plataformas distintas. Sua principal função é garantir a comunicação contínua e padronizada entre equipamentos de diferentes fabricantes, promovendo a integração e o desempenho eficiente de

processos industriais. A entidade responsável pela criação, manutenção e evolução deste padrão é a OPC Foundation.

Originalmente lançado em 1996, o OPC foi concebido para resolver a complexidade da comunicação entre sistemas supervisórios (HMI/SCADA) e Controladores Lógicos Programáveis (CLPs), ao abstrair os protocolos proprietários como Modbus, Profibus, entre outros, por meio de uma interface comum e padronizada. Com isso, surgiu uma indústria especializada no desenvolvimento de servidores OPC, permitindo que usuários finais pudessem construir arquiteturas abertas e interoperáveis com equipamentos de múltiplos fornecedores, utilizando uma camada intermediária responsável pela conversão das requisições de leitura e escrita.

As especificações do OPC foram inicialmente desenvolvidas para a plataforma Microsoft Windows, fazendo uso da tecnologia OLE (Object Linking and Embedding), o que deu origem à denominação OPC Classic. Essa versão foi amplamente adotada em setores diversos como manufatura, automação predial, petróleo e gás, energia e utilities em geral.

Com o avanço das arquiteturas orientadas a serviços (SOA – Service-Oriented Architecture) e a crescente necessidade de segurança, escalabilidade e modelagem de dados mais robusta, tornou-se evidente a limitação do OPC Classic. Em resposta a essas demandas, a OPC Foundation desenvolveu o padrão OPC UA (Unified Architecture), uma evolução que proporciona uma arquitetura de plataforma aberta, multiplataforma, rica em funcionalidades e preparada para o futuro da indústria digital.

O OPC UA não apenas mantém os princípios fundamentais do OPC Classic, como também oferece suporte nativo à comunicação segura, modelagem semântica de informações, interoperabilidade em nuvem e integração com novas tecnologias. Por essas razões, o OPC UA tem sido cada vez mais adotado por empresas e organizações globais como uma base confiável para a construção de soluções interoperáveis e modernas.

(FOUNDATION, 2025a)

3.10.1 ARQUITETURA UNIFICADA (OPC UA)

A Arquitetura Unificada OPC, lançada oficialmente em 2008, representa uma evolução significativa em relação ao padrão OPC Classic. Desenvolvida pela OPC Foundation, essa nova arquitetura foi concebida com base em princípios de orientação a serviços, adotando uma abordagem independente de plataforma e visando suprir as limitações observadas nas versões anteriores do padrão.

O OPC UA integra, em uma estrutura unificada e extensível, todas as funcionalidades das especificações individuais do OPC Classic, como o acesso a dados em tempo real, alarmes e eventos, e acesso a dados históricos, entre outros. Embora seja funcionalmente compatível com o OPC Classic, o OPC UA foi projetado para ir além, oferecendo maior segurança, escalabilidade, interoperabilidade e suporte nativo à modelagem de informações complexas.

Dessa forma, o OPC UA consolida-se como uma plataforma de comunicação robusta, segura e preparada para o futuro da automação industrial, sendo cada vez mais adotada como base tecnológica por fabricantes de equipamentos, desenvolvedores de software e integradores de sistemas ao redor do mundo.

Ademais, ela é funcionalmente melhor adaptada para o mercado atual, tendo os seguintes recursos:

- Descoberta: Encontrar a disponibilidade de servidores OPC em PCs e/ou redes locais;
- Espaço de endereço: Todos os dados são representados hierarquicamente (por exemplo, arquivos e pastas), permitindo que estruturas simples e complexas sejam descobertas e utilizadas por clientes OPC;
- Sob demanda: Ler e gravar dados/informações com base nas permissões de acesso
- Assinaturas: Monitoram dados/informações e relatam por exceção quando os valores mudam com base nos critérios do cliente;
- Eventos: Notificar informações importantes com base nos critérios do cliente;
- Métodos: Os clientes podem executar programas com base em métodos definidos no servidor;

(FOUNDATION, 2025b)

4 METODOLOGIA

O trabalho se tratou de um estudo sobre a viabilidade técnica da implementação de um sistema supervisório de controle para ambientes de teste com pressão. Tendo como foco os testes feitos nas manutenções de equipamentos submarinos de produção de petróleo, o estudo se dividirá em diferentes etapas, desde o dimensionamento dos componentes necessários para a implementação física até o desenvolvimento de todas as simulações necessárias para validar o funcionamento do sistema supervisório.

Este projeto visou realizar um estudo de viabilidade técnica, portanto, optou-se por utilizar simulações para replicar uma implementação física do projeto, assim permitindo prever o comportamento real do sistema. As etapas de desenvolvimento foram delineadas de acordo com pesquisa e experimentação, dividindo-se em três áreas: dimensionamento e criação do circuito hidráulico,

programação em linguagem ladder do CLP e desenvolvimento do sistema supervisor e sua comunicação.

Para realizar o dimensionamento e criação do circuito hidráulico, foi necessário primeiramente mapear todos os componentes relacionados à geração de energia hidráulica necessários para implementação do projeto, como válvulas, bombas e sensores. Os componentes foram selecionados conforme os requisitos de segurança relevantes e a capacidade de atender ao projeto.

No desenvolvimento da programação do CLP, foi utilizada a linguagem ladder para automatizar o controle do circuito hidráulico, assim como implementar medidas de segurança diretamente na programação, proporcionando maior segurança e confiabilidade no sistema e diminuindo o risco de acidentes devido ao erro humano durante a atividade.

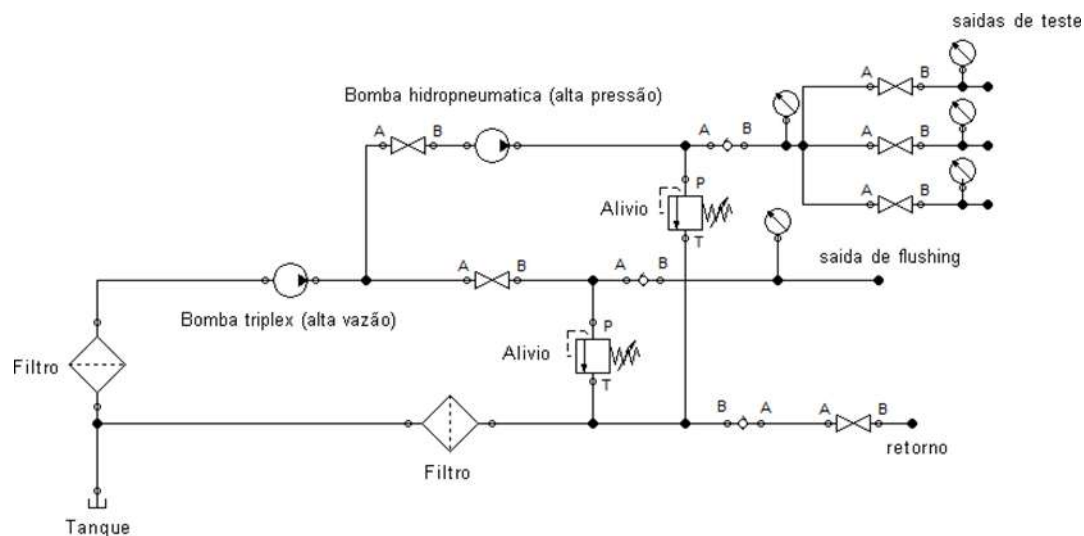
No desenvolvimento do sistema supervisor, foi utilizado o software Ignition para criação de uma interface homem-máquina que permitisse ao usuário ter o controle do funcionamento do circuito hidráulico em tempo real. Esse sistema supervisor possibilita ao operador uma compreensão clara do sistema, permitindo a realização de testes de forma eficiente e segura.

Esta seção irá detalhar cada um dos passos citados acima, provendo as explicações necessárias sobre as decisões tomadas e a metodologia adotada para o desenvolvimento do sistema supervisor.

4.1 CIRCUITO HIDRÁULICO

O desenvolvimento do projeto se iniciou com o mapeamento dos componentes necessários para a montagem do circuito responsável por gerar a energia hidráulica, denominado unidade de teste, e a elaboração do circuito mostrado na figura 16 a seguir.

Figura 16 – Circuito hidráulico UTF



Fonte: Dos Autores (2025)

O sistema hidráulico desenvolvido neste projeto tem como objetivo simular as condições reais de operação de um circuito de testes hidrostáticos, com foco tanto em testes funcionais quanto em testes de vedação de equipamentos. Para isso, foi projetado um circuito contendo diversos componentes com funções específicas, conforme representado na Figura 16.

O sistema é composto por duas bombas principais. A primeira é a bomba triplex de alta vazão, responsável por fornecer grande volume de fluido a uma pressão moderada. Esta bomba é utilizada nas operações de flushing, cujo objetivo é a limpeza do sistema hidráulico, bem como para alimentar a bomba hidropneumática. A segunda bomba, hidropneumática de alta pressão, é responsável por gerar pressão elevada no fluido, sendo essencial para a realização de testes de vedação e funcionais, que exigem condições de pressão mais rigorosas.

O fluido é armazenado inicialmente em um tanque reservatório, de onde é puxado e conduzido ao sistema através de filtros posicionados estrategicamente. Esses filtros são responsáveis por reter impurezas e garantir que apenas fluido limpo circule pelas bombas e demais componentes, protegendo assim o sistema contra danos e contaminações.

As válvulas de retenção são instaladas para garantir que o fluxo de fluido ocorra em um único sentido, evitando o retorno que poderia danificar ou comprometer o desempenho das bombas e do sistema como um todo. Além disso, o circuito conta com válvulas de alívio, localizadas próximo às bombas, que têm como função liberar o fluido em caso de sobrepressão, assegurando a integridade dos componentes.

A distribuição do fluido para as diferentes finalidades é controlada por válvulas solenóides, que são acionadas eletricamente e permitem direcionar o fluxo para as saídas de teste, saída de flushing ou linha de retorno ao tanque. Essas válvulas garantem flexibilidade operacional e permitem alternar entre os modos de teste conforme a necessidade.

Para monitoramento, o sistema conta com manômetros instalados nas saídas principais, possibilitando ao operador acompanhar a pressão durante os testes em tempo real. Essa medição é fundamental tanto para fins de controle quanto para garantir a segurança do procedimento.

O funcionamento geral do sistema ocorre em dois modos principais:

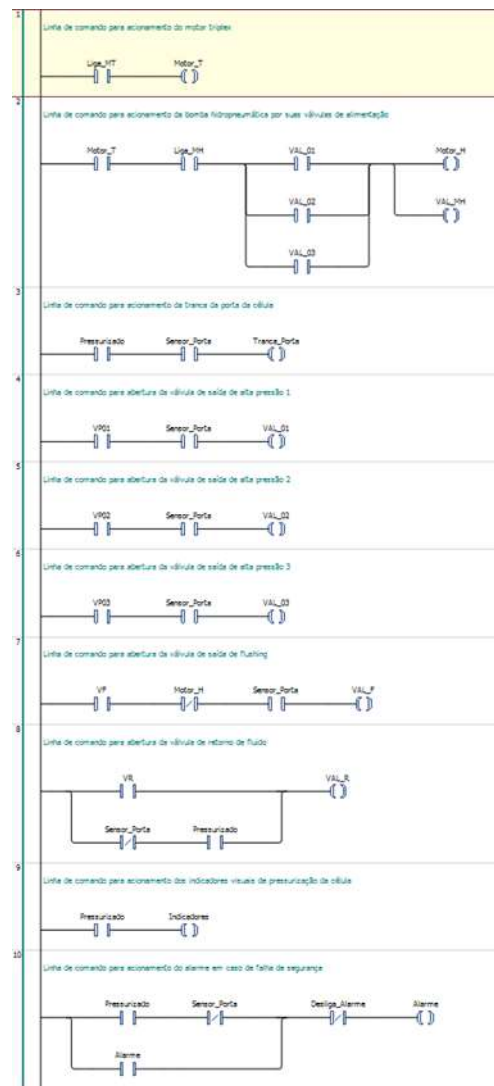
- Modo Flushing: a bomba triplex é acionada, gerando fluxo contínuo de fluido a alta vazão, que é direcionado para a saída de flushing. Este modo permite a limpeza das tubulações antes da realização dos testes.
- Modo Teste de Pressão: a bomba triplex alimenta a bomba hidropneumática, que amplifica a pressão do fluido e o envia às saídas de teste. As válvulas são acionadas para direcionar o fluido, e os manômetros informam a pressão atingida no sistema.

A combinação de componentes permite que o sistema atue de forma flexível, eficiente e segura, simulando fielmente as condições de um ambiente real de teste hidrostático.

4.2 DESENVOLVIMENTO DA PROGRAMAÇÃO LADDER

Com a finalidade de realizar o controle do sistema de testes hidrostáticos de uma célula de teste através de um controlador lógico programável (CLP), foi utilizado o software CodeSys para elaborar um programa em linguagem Ladder, apresentado na figura 17. A seguir, será feita a explicação do programa, apresentando o funcionamento de todas as linhas.

Figura 17 – Programação em Ladder no CodeSys



Fonte: Dos Autores (2025)

O programa é composto por uma linha de acionamento para o motor da bomba triplex, uma linha de acionamento para o motor pneumático da bomba hidropneumática e sua válvula de entrada, uma linha para o acionamento do mecanismo de tranca da porta de acesso à célula de teste, cinco linhas para o acionamento das válvulas de saída de alta pressão, flushing e retorno, uma linha para o acionamento dos indicadores visuais de andamento de teste e uma linha para acionamento de alarme caso ocorra uma falha de segurança.

Além do código em linguagem ladder, o ambiente CodeSys exige que todas as variáveis utilizadas nos contatos e bobinas sejam previamente declaradas, especificando o tipo de dado que será armazenado. Neste projeto, todas as variáveis foram definidas como do tipo booleano, uma vez que representam estados binários, ou seja, ligado ou desligado.

A Figura 18, apresentada abaixo, ilustra essa configuração.

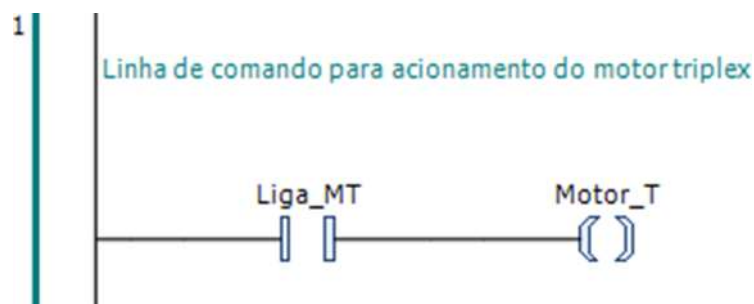
Figura 18 – Declaração das variáveis utilizadas

```
1  PROGRAM PLC_PRG
2  VAR
3      Liga_MT: BOOL;
4      Motor_T: BOOL;
5      Liga_MH: BOOL;
6      VAL_01: BOOL;
7      VAL_02: BOOL;
8      VAL_03: BOOL;
9      Motor_H: BOOL;
10     Pressurizado: BOOL;
11     Sensor_Porta: BOOL;
12     Tranca_Porta: BOOL;
13     VP01: BOOL;
14     VAL_F: BOOL;
15     VAL_R: BOOL;
16     VR: BOOL;
17     VF: BOOL;
18     VP03: BOOL;
19     VP02: BOOL;
20     Indicadores: BOOL;
21     Alarme: BOOL;
22     Desliga_Alarme: BOOL;
23     VAL_MH: BOOL;
24 END_VAR
25
```

Fonte: Dos Autores (2025)

Na figura 19 a seguir, é possível analisar a seção do código responsável pelo acionamento do motor da bomba triplex, formada por um contato normalmente aberto (Liga MT) ligado diretamente a uma bobina (Motor T), assim permitindo o controle da bomba triplex por um interruptor relacionado à variável "Liga MT".

Figura 19 – Linha de programação - Comando do motor triplex

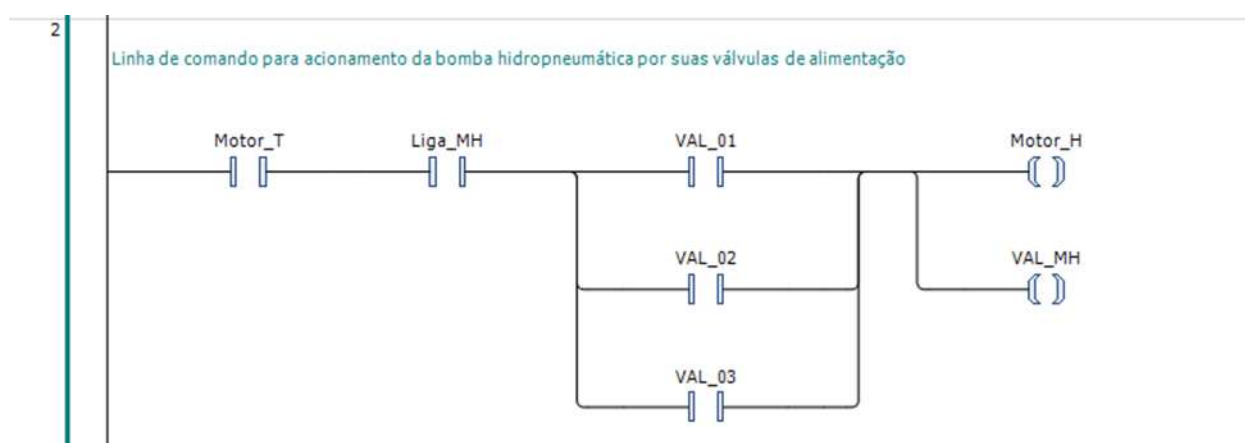


Fonte: Dos Autores (2025)

Em relação ao trecho do programa responsável pelo acionamento da bomba hidropneumática, ilustrado na Figura 20, a lógica é composta por dois contatos normalmente abertos em série, associados às variáveis "Motor T" e "Liga MH", que indicam, respectivamente, o estado da bobina de

acionamento da bomba triplex e o estado do interruptor que comanda a bomba hidropneumática. Esses contatos estão em série com três outros contatos normalmente abertos dispostos em paralelo, vinculados ao estado das válvulas de saída de pressão, garantindo que, para acionar a bomba, ao menos uma dessas saídas esteja aberta. Por fim, o circuito aciona duas bobinas associadas às variáveis "Motor H" e "VAL MH", que representam, respectivamente, o acionamento do motor pneumático da bomba e a válvula de entrada de fluido para a bomba.

Figura 20 – Linha de programação - Comando do motor hidropneumático



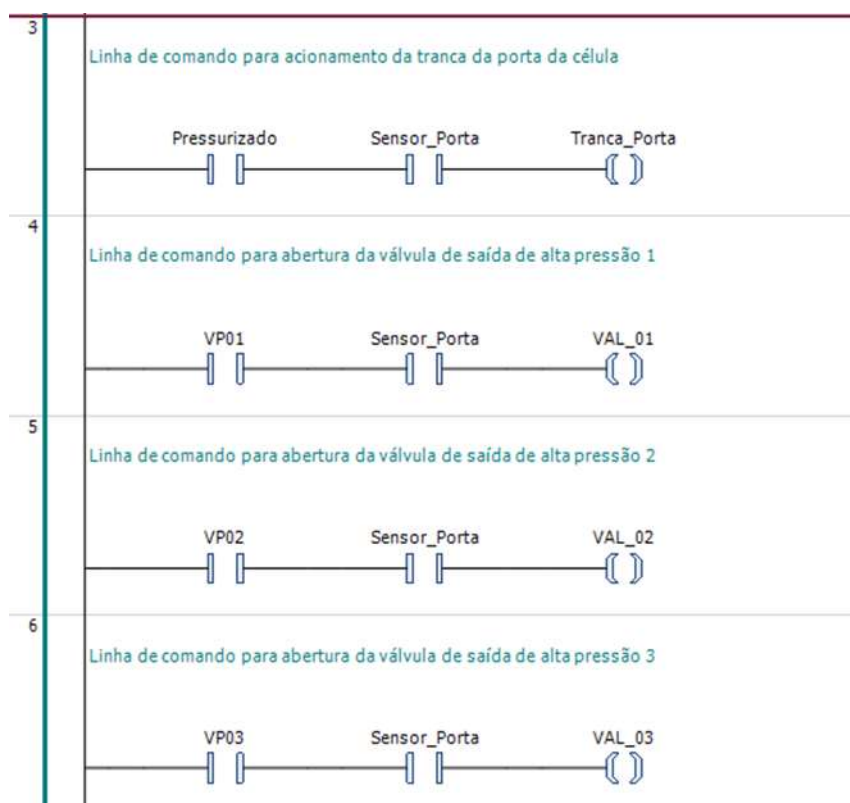
Fonte: Dos Autores (2025)

Em relação às linhas de comando da tranca da porta de acesso à célula de teste e das válvulas de saída de alta pressão, ilustradas na figura 21, ambas são compostas por dois contatos normalmente abertos em série, que acionam uma bobina responsável pelas variáveis de saída correspondentes.

No sistema de trancamento da célula, os contatos estão associados às variáveis que indicam o estado de pressurização da célula e à leitura do sensor que detecta se a porta está devidamente fechada. A bobina, por sua vez, está vinculada à variável responsável por enviar o sinal de trancamento da porta.

Já nas linhas de acionamento das válvulas de saída de alta pressão, os contatos estão conectados, respectivamente, à variável que representa o estado do interruptor de acionamento das válvulas e ao sensor da porta. A bobina correspondente está associada à variável que envia o sinal de abertura das válvulas.

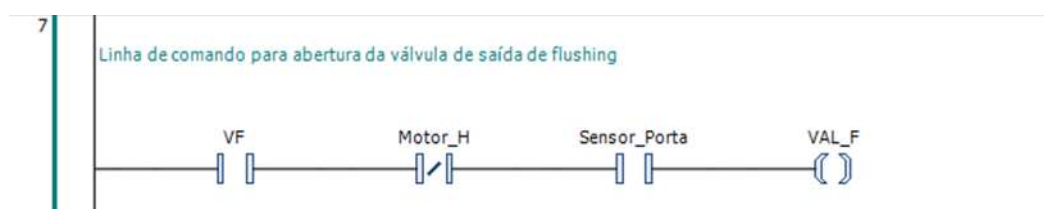
Figura 21 – Linha de programação - Comando da tranca da célula e válvulas de pressão



Fonte: Dos Autores (2025)

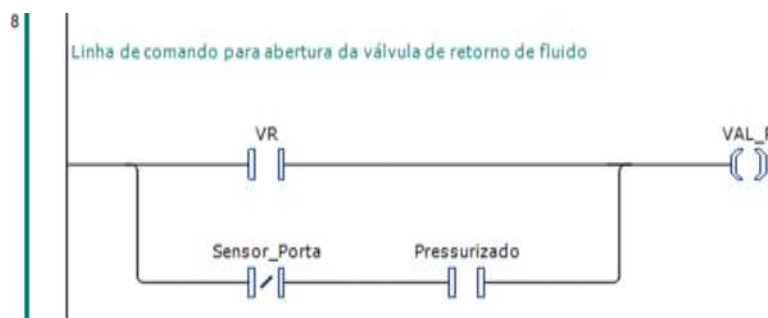
A linha de controle da válvula de saída de flushing, ilustrada na figura 22, é composta por uma lógica que inclui três contatos em série. O primeiro é um contato normalmente aberto vinculado à variável "VF", que representa o interruptor de acionamento da válvula. Em seguida, há um contato normalmente fechado associado ao estado da variável de acionamento do motor hidropneumático, impedindo o acionamento da válvula durante o funcionamento da bomba. Por fim, há um contato normalmente aberto vinculado ao sensor que indica o fechamento da porta da célula de teste. Essa lógica em série conduz à bobina responsável pelo acionamento da válvula de saída de flushing.

Figura 22 – Linha de programação - Comando da válvula de saída de flushing



Fonte: Dos Autores (2025)

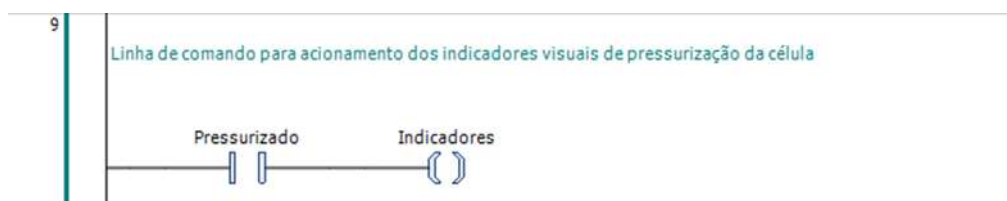
Figura 23 – Linha de programação - Comando da válvula de retorno do fluido



Fonte: Dos Autores (2025)

Na Figura 24 a seguir, é possível observar a seção do código responsável pelo acionamento dos indicadores visuais da célula pressurizada. Essa lógica é composta por um contato normalmente aberto, associado à variável "Pressurizado", conectado diretamente a uma bobina representada pela variável "Indicadores". Dessa forma, o acionamento dos indicadores é controlado por meio do estado de pressurização da célula vinculado à variável "Pressurizado".

Figura 24 – Linha de programação - Comando dos indicadores visuais de teste



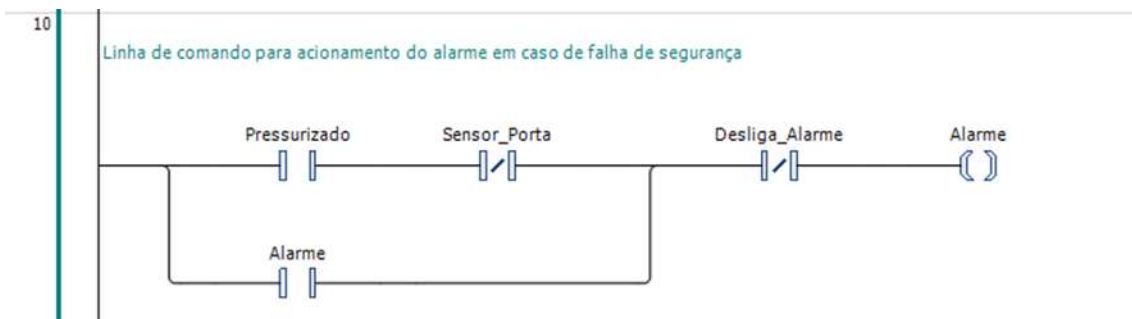
Fonte: Dos Autores (2025)

Por fim, a Figura 25 apresenta a linha de código responsável pelo acionamento do alarme sempre que a célula estiver pressurizada com a porta aberta, caracterizando uma condição de risco. Essa linha é composta por quatro contatos: três conectados em série e um quarto em paralelo com os dois primeiros, formando uma lógica de auto-manutenção (ou selo).

O primeiro contato é normalmente aberto e está associado à variável "Pressurizado"; o segundo é normalmente fechado, vinculado à variável "Sensor Porta", que indica se a porta está aberta; o terceiro é também normalmente fechado, associado à variável "Desliga Alarme", correspondente ao botão de desativação do alarme. O quarto contato, por sua vez, é normalmente aberto e está vinculado à própria variável do alarme, permitindo que, uma vez acionado, o alarme se mantenha ativo até que o botão de desligamento seja pressionado.

Essa lógica garante que o alarme seja ativado automaticamente em situações perigosas e permaneça acionado até intervenção manual, contribuindo para a segurança do sistema.

Figura 25 – Linha de programação - Comando do alarme de segurança



Fonte: Dos Autores (2025)

O código em ladder, desenvolvido no software CodeSys, tem como objetivo automatizar o processo de controle e acionamento das bombas geradoras de pressão e das válvulas de controle de fluxo, operações que, em muitos ambientes de teste hidrostático, ainda são realizadas manualmente. Além disso, o programa incorpora medidas de segurança que asseguram que os testes só possam ser iniciados quando o ambiente estiver devidamente isolado por barreiras de engenharia.

Por meio da programação no CodeSys, o sistema garante que a execução dos testes hidrostáticos ocorra de forma segura e eficiente, otimizando a produtividade e reforçando a proteção dos profissionais envolvidos na atividade.

4.3 DESENVOLVIMENTO DO SISTEMA SUPERVISÓRIO

Com o objetivo de controlar o circuito hidráulico e o ambiente de teste, foi desenvolvido um sistema supervisório utilizando o software Ignition. Esse sistema proporciona ao operador uma visualização clara e intuitiva de todas as variáveis envolvidas na operação, além de permitir o controle direto sobre seus estados, viabilizando a execução segura e eficiente dos testes.

4.3.1 COMUNICAÇÃO ENTRE IGNITION E CODESYS

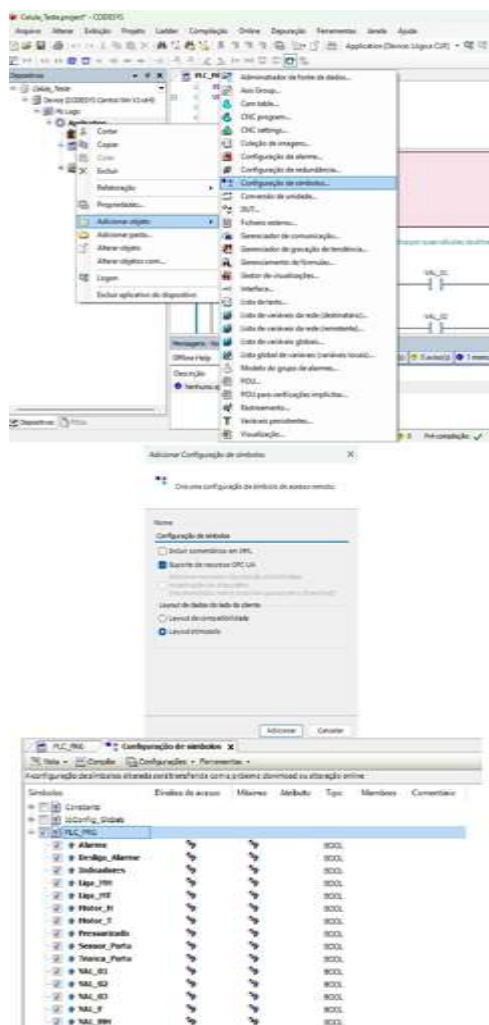
Considerando que o objetivo do projeto é avaliar a viabilidade técnica do sistema e de seus componentes por meio de simulações, em vez da utilização de equipamentos físicos reais, torna-se necessária a comunicação entre os diferentes softwares responsáveis por simular cada parte do sistema. Para isso, será adotado o protocolo de comunicação industrial OPC UA (Open Platform Communications Unified Architecture), que permitirá a troca de informações entre os sistemas

envolvidos, viabilizando a recriação virtual do ambiente real de operação. Para tal, foram seguidas as etapas indicadas a seguir.

4.3.1.1 CRIAÇÃO SERVIDOR OPC UA

Primeiramente, se faz necessário realizar a criação do servidor OPC UA no software CodeSys e disponibilizar as informações referentes aos valores das variáveis do programa em ladder para acesso aos clientes que futuramente se conectarão ao servidor. Portanto, conforme indicado na figura 26, deve-se adicionar ao projeto o objeto "configuração de símbolos", habilitando o suporte ao recurso OPC UA. Em seguida, dentro desse objeto, devem ser selecionadas todas as variáveis que estarão disponíveis para acesso por meio do servidor OPC UA.

Figura 26 – Etapas de configuração da comunicação no CodeSys

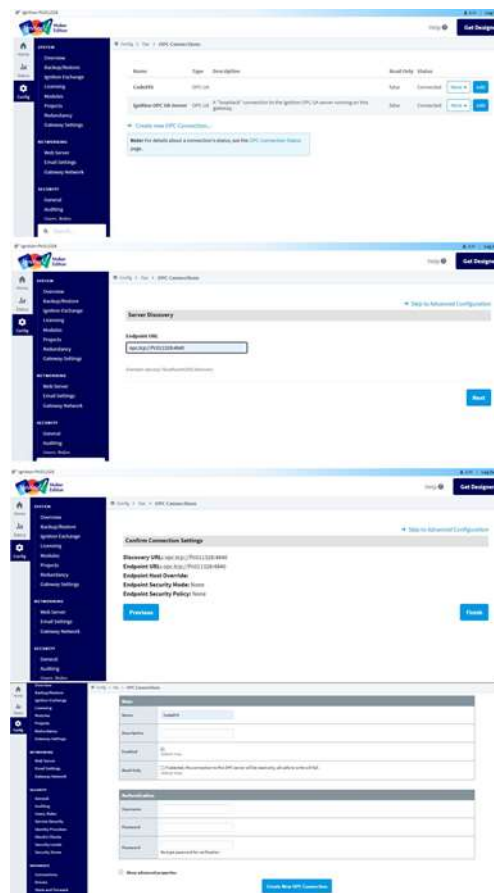


Fonte: Dos Autores (2025)

4.3.1.2 CONEXÃO AO SERVIDOR PELO IGNITION

Após a configuração do servidor, é necessário estabelecer a conexão como cliente por meio do software Ignition, indicado na figura 27. Para isso, deve-se acessar o portal local do Ignition inserindo o endereço "http://localhost:8088/web/home" em qualquer navegador. Em seguida, no menu de configurações, deve-se localizar a aba OPC Connections e criar uma nova conexão, inserindo o endpoint do servidor OPC UA e definindo um nome para a conexão.

Figura 27 – Etapas de configuração da comunicação no Ignition



Fonte: Dos Autores (2025)

Com a conexão estabelecida, torna-se possível acessar, no ambiente de desenvolvimento do Ignition, as variáveis do programa desenvolvido no CodeSys. Conforme ilustrado na Figura 28, ao navegar pelo Tag Browser e selecionar a opção Browse Devices, é possível acessar o servidor OPC e importar as variáveis desejadas para utilizá-las no sistema supervisório.

Figura 28 – Acesso das variáveis no Ignition através do servidor OPC UA

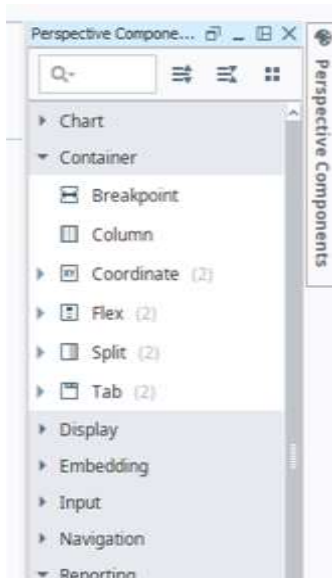


Fonte: Dos Autores (2025)

4.3.2 CRIAÇÃO DA INTERFACE HOMEM MÁQUINA

No software Ignition, para inserir os símbolos que representam os componentes reais do sistema — como válvulas, bombas e a estrutura do ambiente de teste — deve-se acessar a aba Perspective Components, conforme indicado na Figura 29. A partir dessa seção, é possível adicionar ao supervisório os elementos gráficos que refletem o estado real dos componentes durante o funcionamento do sistema.

Figura 29 – Ignition - Aba perspective components

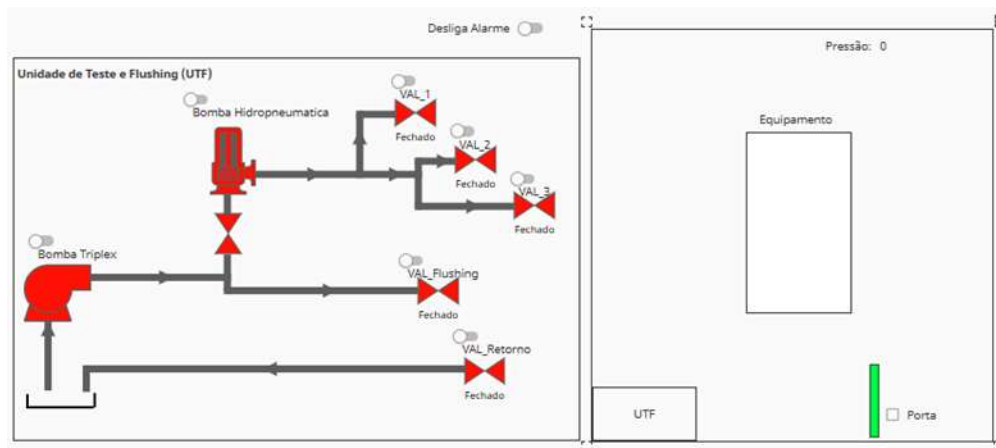


Fonte: Dos Autores (2025)

Utilizando os símbolos disponíveis, foi criada no supervisório uma representação visual do circuito hidráulico e da estrutura da célula de teste. Essa interface inclui o equipamento a ser testado, os

indicadores visuais de pressão e estado do teste, além da porta de acesso à área de ensaio, conforme ilustrado na Figura 30 abaixo.

Figura 30 – Ignition - Representação do sistema

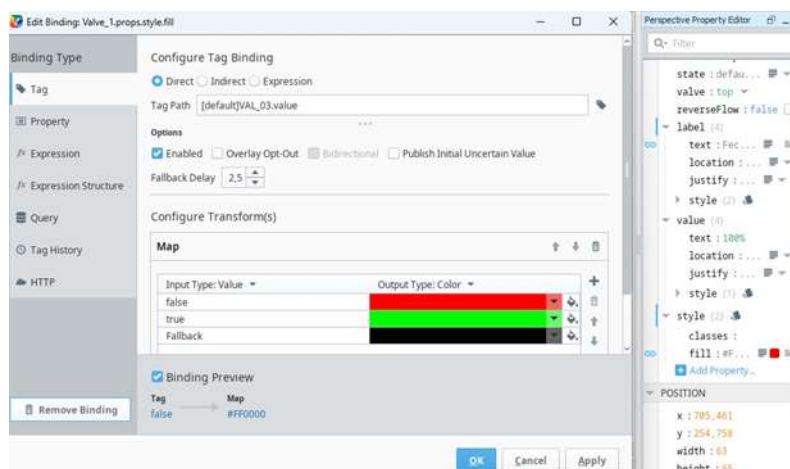


Fonte: Dos Autores (2025)

Para facilitar a identificação do estado em que os componentes se encontram, foi implementada uma alteração de cor dos símbolos.

Os símbolos que representam as bombas e válvulas no supervisório são exibidos na cor vermelha quando estão desativados — ou seja, com as bombas desligadas e as válvulas fechadas — e mudam para a cor verde durante o seu acionamento. A configuração dessa propriedade visual é realizada na aba Perspective Property Editor, onde é possível vincular dinamicamente a cor do componente ao valor de uma variável específica, conforme ilustrado na Figura 31.

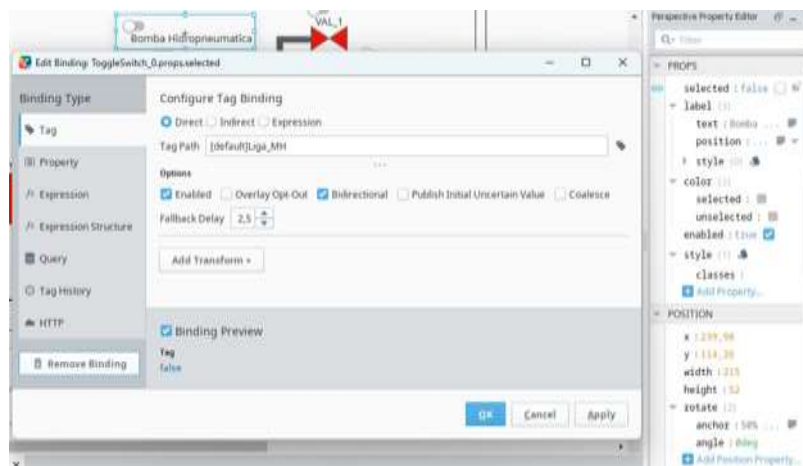
Figura 31 – Ignition - Configuração de cor das bombas e válvulas



Fonte: Dos Autores (2025)

Para permitir o controle de acionamento e desacionamento dos componentes, foram adicionados interruptores no supervisório, vinculados às variáveis responsáveis pelo acionamento dos contatos na programação do CLP. Cada componente que o operador precisa manipular durante a execução da atividade conta com seu respectivo interruptor, conforme indicado na Figura 32 abaixo.

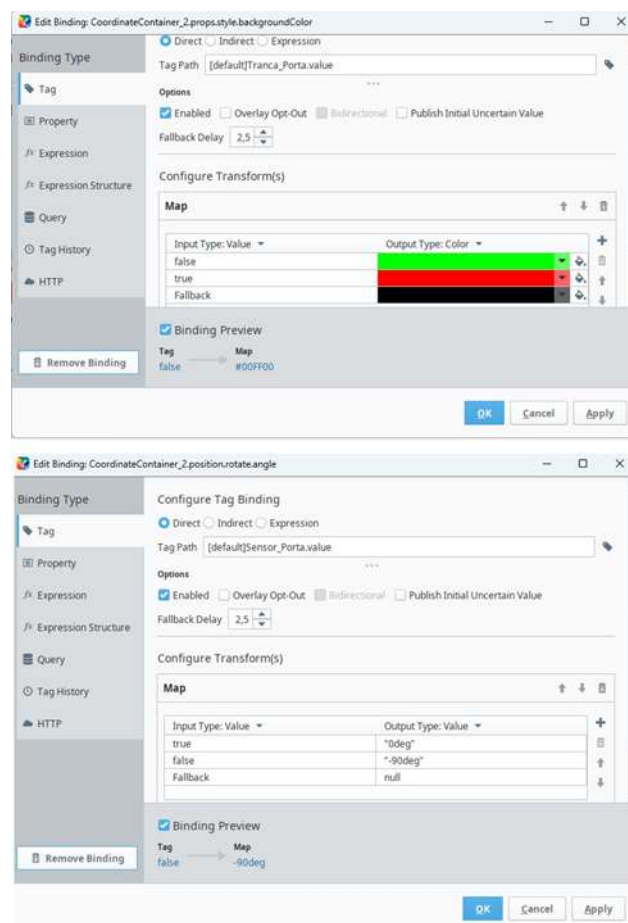
Figura 32 – Ignition - Configuração do Interruptor



Fonte: Dos Autores (2025)

A porta de acesso da célula de teste foi configurada na simulação de forma que sua cor seja exibida em verde quando estiver destrancada e em vermelho quando estiver trancada. Além disso, para simular o sensor que indica se a porta está fechada, foi utilizado um checkbox, cujo estado altera a posição gráfica da porta na interface, representando se ela está aberta ou fechada. As etapas de configuração desses recursos podem ser visualizadas na Figura 33.

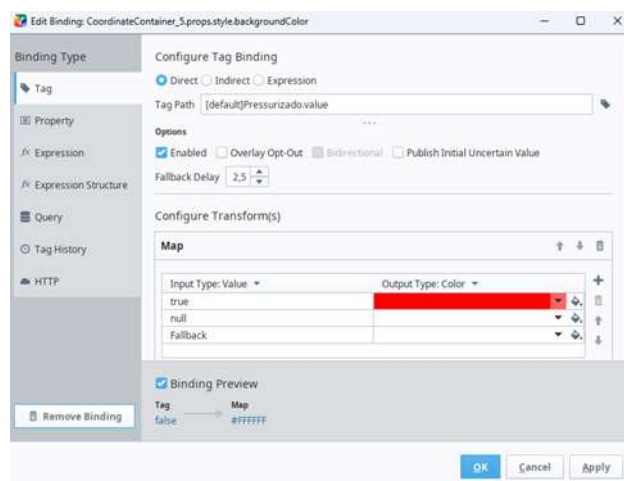
Figura 33 – Ignition - Configuração da porta da célula de teste



Fonte: Dos Autores (2025)

Os indicadores visuais de teste — normalmente implementados fisicamente por meio de lâmpadas vermelhas instaladas nos quatro cantos da célula, acionadas sempre que há pressão em seu interior — foram simulados utilizando o componente Container como representação gráfica das lâmpadas. Na simulação, ao ser detectado que a célula está pressurizada, os Containers mudam automaticamente para a cor vermelha, indicando o estado de risco. Esse comportamento está ilustrado na Figura 34 abaixo.

Figura 34 – Ignition - Configuração dos indicadores visuais

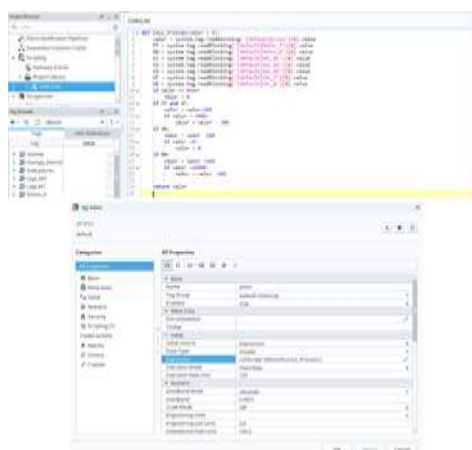


Fonte: Dos Autores (2025)

Para simular a variação do valor de pressão em função do estado das válvulas e bombas, foi utilizado o módulo de scripting do software Ignition, que permite a criação de códigos em linguagem Python para representar fenômenos físicos reais.

Inicialmente, é necessário criar um novo script contendo a lógica que simula o comportamento da pressão no sistema. Em seguida, o valor da variável de pressão é vinculado ao resultado gerado por essa função utilizando a função "runScript()" na definição da fonte do valor da variável, permitindo que a simulação responda dinamicamente às condições do processo, conforme indicado na figura 35.

Figura 35 – Ignition - Configuração da simulação do valor de pressão

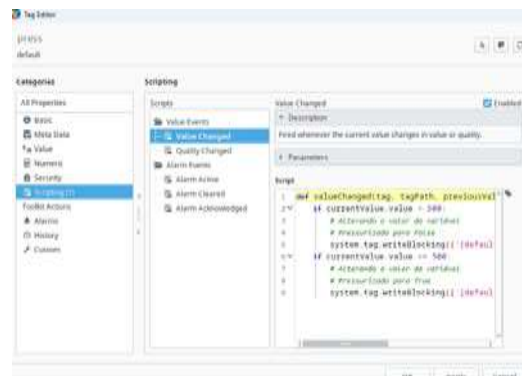


Fonte: Dos Autores (2025)

Além da configuração para a simulação da pressão, a variável de pressão também possui um script associado, executado sempre que seu valor é alterado. Esse script é responsável por atualizar o valor

da variável booleana Pressurizado, com base no valor atual de pressão. Conforme ilustrado na Figura 36.

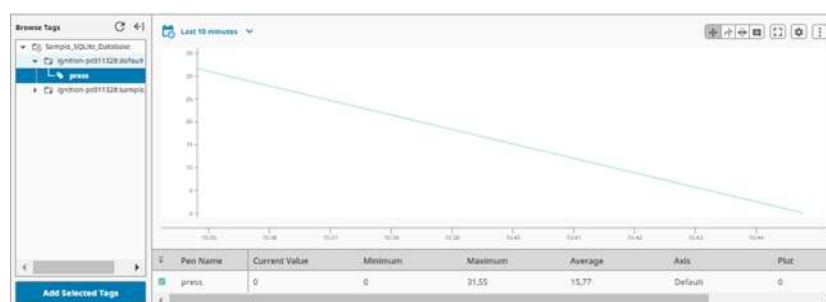
Figura 36 – Ignition - Configuração do valor da variável Pressurizado



Fonte: Dos Autores (2025)

Foi adicionado ao supervisório um gráfico em tempo real para exibir o valor da pressão durante a execução do teste, permitindo que o técnico responsável acompanhe seu progresso e identifique eventuais quedas de pressão que possam indicar a ocorrência de vazamentos. Para isso, foi utilizado o componente Power Chart, configurado para monitorar e exibir a variável de pressão continuamente, conforme ilustrado na Figura 37.

Figura 37 – Ignition - Configuração do gráfico de pressão

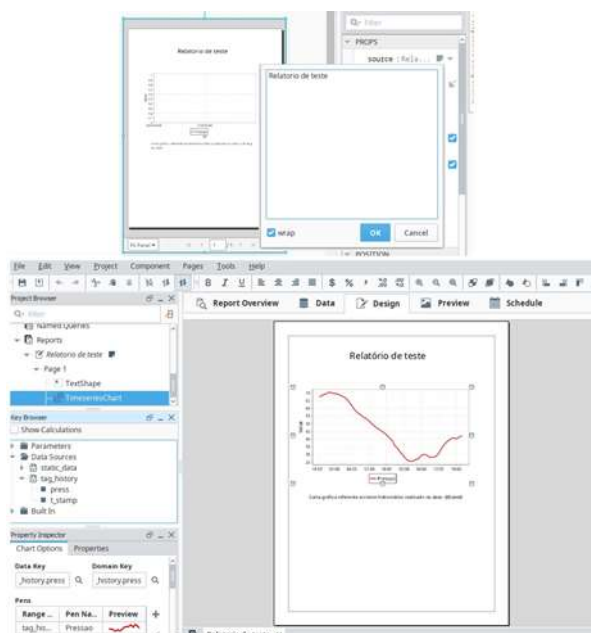


Fonte: Dos Autores (2025)

Além disso, foi implementado no supervisório um gerador de relatórios, com o objetivo de criar um documento que comprove a realização da atividade. Para isso, foi adicionado o componente Report Viewer à interface supervisória e utilizado o módulo Reports do Ignition para estruturar o relatório.

O relatório gerado inclui a representação gráfica do teste realizado e um texto informativo com a data de execução da atividade, servindo como registro oficial do procedimento. A configuração detalhada pode ser visualizada na Figura 38 abaixo.

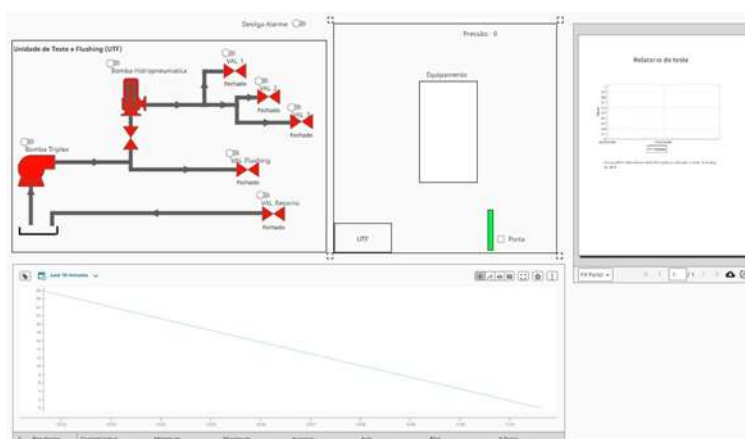
Figura 38 – Ignition - Configuração do relatório de teste



Fonte: Dos Autores (2025)

Por fim, foram adicionados textos para facilitar o entendimento do sistema. Dessa forma, na figura 39 é apresentado o sistema supervisorio completo do projeto, responsável pelo controle e supervisão de todo o processo de teste hidrostático em uma célula de teste.

Figura 39 – Sistema supervisorio do projeto



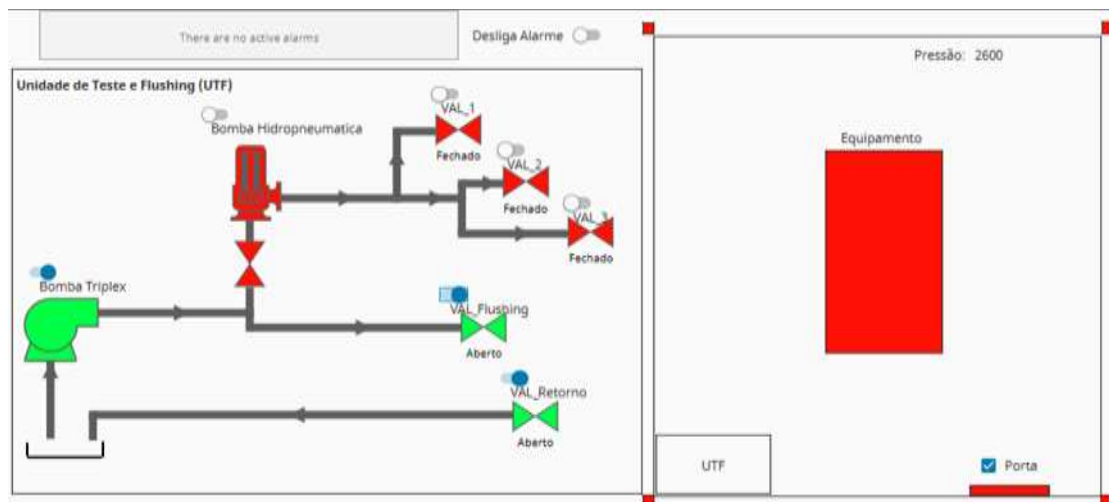
Fonte: Dos Autores (2025)

5 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Concluindo o sistema supervisorio do projeto, é possível verificar de forma mais concisa os resultados obtidos durante o processo. A fim de ilustrar todo o funcionamento do sistema, são apresentadas imagens do supervisorio durante as etapas de uma atividade de manutenção.

Na figura 40, é possível observar o funcionamento do sistema durante a realização de um flushing. Observa-se que ocorreu uma alteração da cor dos componentes ativados durante a atividade, indicando seu estado de funcionamento. Adicionalmente, é possível observar o aumento do valor de pressão, implementado via script, levando à ativação da tranca da porta e luzes indicadores de teste, sinalizados em cor vermelha na célula de teste.

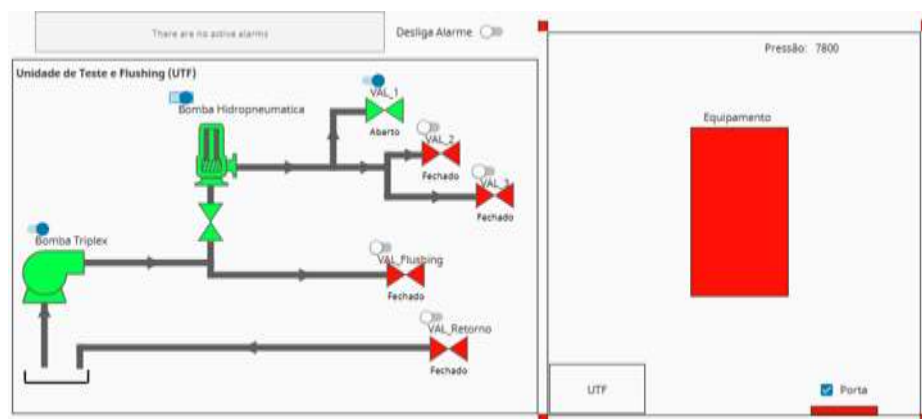
Figura 40 – Supervisório em funcionamento – Flushing



Fonte: Dos Autores (2025)

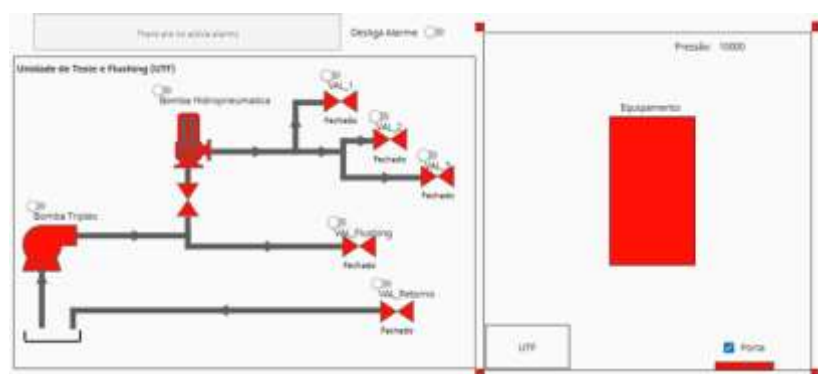
As figuras 41 e 42 demonstram o sistema supervisório durante a transição para um teste hidrostático. Verifica-se que, ao trocar o acionamento da linha hidráulica de flushing para linha hidráulica de teste, ocorre o aumento de pressão dentro da célula. Então é realizado o teste hidrostático ao manter a pressão pelo tempo determinado sem que sejam observadas quedas da mesma.

Figura 41 – Supervisório em funcionamento - Aumento de pressão



Fonte: Dos Autores (2025)

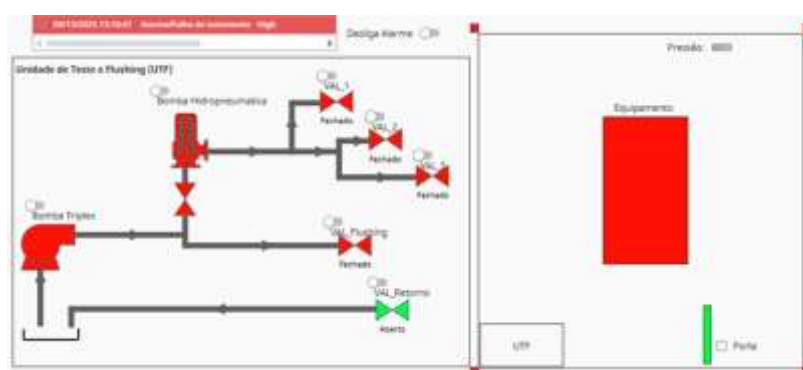
Figura 42 – Supervisório em funcionamento - Teste hidrostático



Fonte: Dos Autores (2025)

Com intuito de visualizar o funcionamento do alarme, foi forçada a abertura da porta da célula durante o teste, resultando na abertura da válvula de retorno e acionamento do alarme de falha de isolamento, indicado na figura 43.

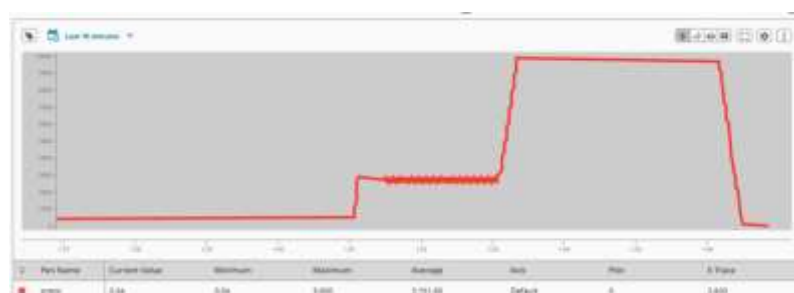
Figura 43 – Supervisório em funcionamento - Teste do alarme



Fonte: Dos Autores (2025)

Na figura 44, é possível verificar a visualização do gráfico em tempo real que o usuário do sistema irá acompanhar durante a efetuação das atividades exemplificadas anteriormente.

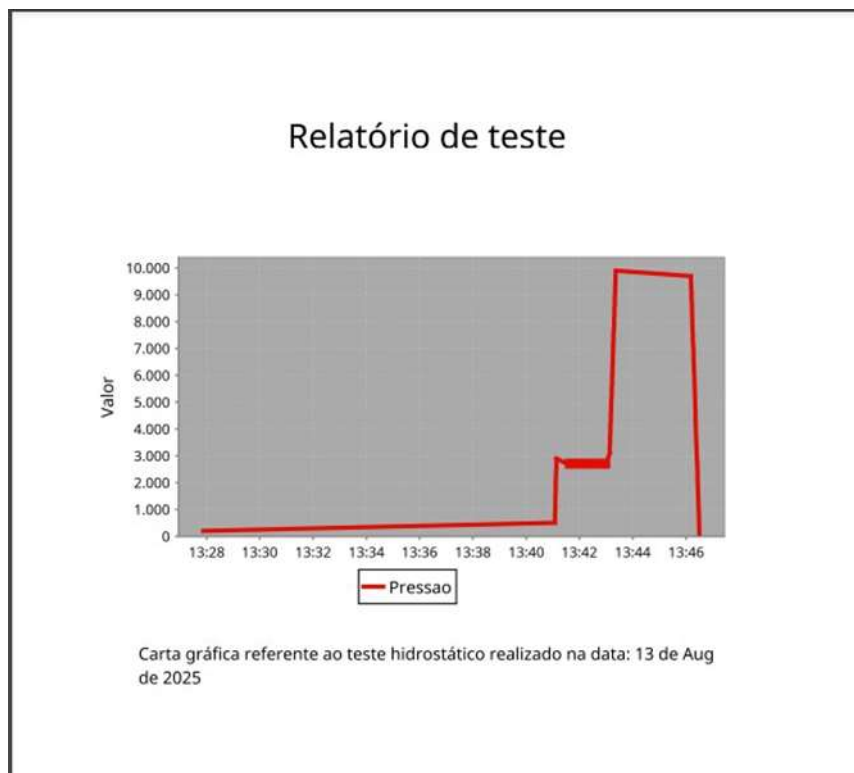
Figura 44 – Supervisório em funcionamento - Gráfico em tempo real



Fonte: Dos Autores (2025)

Por fim, na figura 45, observa-se o relatório de teste extraído do sistema supervisório após a realização das atividades exemplificadas durante esta seção.

Figura 45 – Supervisório em funcionamento - Relatório de teste



Fonte: Dos Autores (2025)

Após todo o processo, os resultados do projeto evidenciam a efetividade do desenvolvimento de um sistema supervisório voltado ao ambiente de teste hidrostático de equipamentos submarinos. Essa implementação permitiu observar a operação do sistema em condições de maior segurança e precisão, demonstrando o potencial da automação aplicada a processos industriais desse tipo.

6 CONCLUSÃO

Portanto, após toda a análise, pode-se afirmar que o objetivo principal do projeto foi plenamente alcançado. Com o desenvolvimento de um sistema supervisório eficaz para um ambiente de teste hidrostático para equipamentos de produção de petróleo. Através da integração e comunicação entre os softwares CodeSys e Ignition, foi possível monitorar e controlar com precisão todos os componentes do circuito hidráulico e as variáveis reais, garantindo o funcionamento correto e seguro do sistema.

O desenvolvimento de um sistema supervisório voltado para o controle e monitoramento de células de teste hidrostático de equipamentos offshore demonstrou-se essencial para garantir a segurança, eficiência e rastreabilidade das operações envolvidas nesses testes. A integração dos softwares, anteriormente citados, possibilitou a criação de uma solução robusta e flexível, capaz de simular com fidelidade as condições reais enfrentadas no ambiente industrial.

Durante o projeto, foi possível implementar um controle preciso dos componentes hidráulicos, monitoramento em tempo real de variáveis críticas como pressão e estado das válvulas, além da geração de relatórios automáticos para registro das atividades executadas. A simulação do sistema hidráulico e a criação de uma interface gráfica intuitiva permitiram que operadores interagissem com a célula de teste de maneira segura e eficaz, reforçando a importância da automação e digitalização em ambientes de testes industriais.

Em suma, o trabalho realizado evidencia como tecnologias modernas de automação podem ser aplicadas para otimizar processos complexos e críticos na indústria de petróleo e gás, contribuindo para a redução de falhas, aumento da produtividade e melhoria na qualidade dos serviços prestados. O sistema proposto pode servir de base para futuras melhorias e expansões, como a incorporação de inteligência artificial para análise preditiva ou a integração com sistemas de gestão corporativa.

REFERÊNCIAS

ABB. 2025. Disponível em: <<https://loja.br.abb.com/blog/post/linguagem-ladder-principais-fundamentos#:~:text=O%20que%20%C3%A9%20linguagem%20ladder,a%20partir%20do%20comando%20solicitado.>> Citado na página 23.

ALMEIDA, J. S. de. Uso da ferramenta pert/cpm no serviço de manutenção de equipamentos submarinos. Perspectivas Online, 2020. Disponível em: <https://scholar.archive.org/work/zak7oziimnbbebfkokokeicu7i/access/wayback/https://ojs3.perspectivasonline.com.br/exatas_e_engenharia/article/download/578/495>. Citado na página 22.

BAI, Q. B. Y. Sistemas Marítimos de Produção de Petróleo. [S.l.]: Elsevier, 2010. Citado 2 vezes nas páginas 20 e 21.

DIGITROL. 2024. Disponível em: <<https://digitrol.com.br/produto/transdutores-de-pressao-para-polimeros-dyn-x-dynisco/>>. Citado na página 30.

FLUTROL. 2025. Disponível em: <<https://www.flutrol.com.br/arquivos/flutrol-bombaspneumaticas.pdf>>. Citado na página 27.

FM2S. 2025. Disponível em: <<https://www.fm2s.com.br/blog/diagrama-de-blocos>>. Citado na página 23.

FOUNDATION, O. 2025. Disponível em: <<https://opcfoundation.org/about/what-is-opc/>>. Citado na página 34.

FOUNDATION, O. U. 2025. Disponível em: <<https://opcfoundation.org/about/opc-technologies/opc-ua/>>. Citado na página 35.

FREITAS, C. M. Clp – controlador lógico programável – parte 1. Embarcados, 2023. Disponível em: <<https://embarcados.com.br/clp-parte1/>>. Acesso em: 05 dez. 2024. Citado na página 22.

FROATS, H. Hydrostatic Testing for Oil Pipelines: Ensuring Safety and Compliance. 2023. Disponível em: <https://hydrotechtesting.com/recertifying-and-upgrading-pipeline-systems-with-hydrostatic-testing/?utm_source=chatgpt.com>. Citado na página 16.

GABRIEL, P. Equipamentos submarinos. TecPetro, 2014. Disponível em: <<https://tecpetro.wordpress.com/2014/05/11/equipamentos-submarinos/>>. Acesso em: 28 mar. 2025. Citado na página 17.

GOV. Relatório de AIR - NR 13. [S.l.], 2021. Disponível em: <<https://www.gov.br/trabalho-e-emprego/pt-br/assuntos/inspecao-do-trabalho/seguranca-e-saude-no-trabalho/relatorios-air/relatorio-air-nr-13.pdf>>. Citado na página 15.

GROUP, E. 2025. Disponível em: <https://www.indx.com/pt_br/product/ignition-scada>. Citado na página 33.

HYDRACO. 2025. Disponível em: <<https://www.hydraco.com.br/blog/valvula-de-alivio-de-pressao-qual-a-funcao/>>. Citado na página 28.

HYDRACOR. 2025. Disponível em: <<https://www.hydraco.com.br/blog/veja-como-funciona-valvula-de-retencao-e-onde-e-instalada/>>. Citado na página 29.

IBP. PANORAMA GERAL DO SETOR DE PETRÓLEO E GÁS. 2023. Disponível em:<<https://www.ibp.org.br/personalizado/uploads/2023/02/panorama-setor-og-ibp-2023.pdf>>. Citado 2 vezes nas páginas 15 e 16.

INETEC. 2025. Disponível em: <<https://inetec.com.br/o-que-e-codesys-e-sua-importancia-no-mundo-da-automacao/>>. Citado na página 32.

JEFFERSON.IND. 2024. Disponível em: <<https://www.jefferson.ind.br/conteudo/valvula-solenoides.html>>. Citado na página 28.

JYVALVULAS. 2025. Disponível em: <<https://www.jyvalvulas.com.br/blog/categorias/artigos/bombas-triplex-tecnologia-avancada-para-alta-pressao-e-eficiencia>>. Citado na página 26.

LEWA. 2025. Disponível em: <<https://www.lewa.com/pt-BR/bombas/bombas-de-processo/bombas-processo-triplex-para-processos-de-alta-pressao>>. Citado na página 26.

NAUCO. 2025. Disponível em: <<https://www.nauco.com.br/blog/manometro-industrial/manometro-industrial-o-que-e-sua-importancia-e-aplicacoes-no-setor-industrial/>>.Citado na página 30.

NEWTEC. 2025. Disponível em: <<https://filtrosnewtec.com.br/sistemas-hidraulicos-qual-a-funcao-de-cada-filtro/>>. Citado na página 31.

ROCKWELL. Sistemas de controlador lógico programável. Rockwell Automation, 2024. Disponível em: <<https://www.rockwellautomation.com/pt-br/products/hardware/allen-bradley/programmable-controllers/micro-controllers/micrologix-family/micrologix-1400-controllers.html>>. Acesso em: 14 nov. 2024. Citado na página 24.

SIQUEIRA, C. A. Technipfmc e onesubsea perto de megacontratos com a petrobras. Brasil Energia, 2021. Disponível em: <<https://brasilenergia.com.br/petroleoegas/technipfmc-e-onesubsea-perto-de-megacontratos-no-brasil/>>. Acesso em: 13 nov. 2024. Citado na página 16.

SPAGNOLETTI. The Risks of High-Pressure Valve Failure in the Oilfield: Causes, Dangers, and Legal Considerations. 2024. Disponível em: <https://www.spaglaw.com/blog/2024/12/the-risks-of-high-pressure-valve-failure-in-the-oilfield-causes-dangers-and-legal-considerations/?utm_source=chatgpt.com>. Citado 2 vezes nas páginas 15 e 17.

TEODORO. Como funciona a hierarquia de controle de riscos? OnSafety, 2022. Disponível em: <<https://onsafety.com.br/como-funciona-a-hierarquia-de-controle-de-riscos/>>.Acesso em: 14 nov. 2024. Citado na página 19.

WEG. 2024. Disponível em: <<https://www.weg.net/catalog/weg/RS/pt/Controls/Relés-Eletrônicos/Linha-22%2C5-mm/RTW---Relés-temporizadores-eletrônicos/RELE-TEMPORIZADOR-RTW-CI02-U030SE05/p/10118424>>. Citado na página 32.

DESENVOLVIMENTO E CONSTRUÇÃO DE UM PROTÓTIPO DE CONTROLADOR LÓGICO PROGRAMÁVEL E SOFTWARE COM PROGRAMAÇÃO EM LADDER UTILIZANDO TECNOLOGIAS DE CÓDIGO ABERTO

Dedico este trabalho aos professores e amigos que fizeram parte dessa jornada.

"Tudo aquilo que o homem ignora, não existe para ele. Por isso o universo de cada um, se resume no tamanho de seu saber."

Resumo: Este projeto tem como objetivo o desenvolvimento de um protótipo funcional de Controlador Lógico Programável (CLP) didático, voltado para o ensino em laboratórios educacionais de controle e automação. O trabalho envolveu uma revisão bibliográfica sobre CLPs e suas aplicações, bem como a análise de diferentes abordagens de hardware e software. A partir dessa investigação, foram selecionados componentes que equilibram desempenho e custo-benefício, priorizando interfaces intuitivas, modularidade e compatibilidade com ferramentas livres.

O protótipo foi construído com base no microcontrolador ATmega328P, distribuído em três camadas de circuito impresso: entradas/saídas, controle lógico e sinalização. Foram implementadas 6 entradas digitais opto-isoladas (12–24V), 2 entradas analógicas (0–10V), 6 saídas a relé (5A/250VAC) e 3 saídas PWM, além de RTC e comunicação RS485.

A programação foi realizada por meio da plataforma Arduino IDE, utilizando linguagem C, e por meio do software LDmicro, com lógica Ladder. Ambas as abordagens comprovaram a compatibilidade do hardware com metodologias adotadas na indústria e no meio acadêmico. Em conclusão, o projeto apresenta uma solução acessível, eficiente e replicável para o ensino de automação, explorando tecnologias de código aberto.

Palavras-chave: Automação industrial. Controlador lógico programável. Programação Ladder. Tecnologias de código aberto. Arduino.

LISTA DE ILUSTRAÇÕES

Figura 1 – Controlador Lógico Programável estruturado em blocos. Adaptado de (ROQUE, 2014).....	77
Figura 2 – Placa Arduino Uno, uma das versões mais utilizadas em projetos didáticos e de prototipagem (ARDUINO, 2025).....	79
Figura 3 – Programa de automação em linguagem Ladder. Adaptado de (ROQUE, 2014)..	81
Figura 4 – Exemplo de código em C++ aplicado no ambiente Arduino, ilustrando a integração entre programação textual e sistemas embarcados. Fonte:	
Autoria própria (2025).....	84
Figura 5 – Modelos de CLP analisados como referência: (a) Siemens LOGO! 8 24CEo, (b) WEG CLIC-02, (c) Allen Bradley Micrologix 1000. Fonte: Siemens (2025); WEG (2025); Allen Bradley (2025).....	87
Figura 6 – Diagrama de blocos representando a organização modular em três camadas do protótipo. Fonte: Autoria própria (2025).....	90
Figura 7 – Esquema eletrônico da Camada 1: Entradas Digitais, Entradas Analógicas, Saídas a Relé e Fonte de Alimentação. Fonte: Autoria própria (2025).....	91
Figura 8 – Esquema eletrônico da Camada 2: Controle, Comunicação, RTC e Saídas PWM. Fonte: Autoria própria (2025).....	92
Figura 9 – Diagrama de pinagem (pinout) do microcontrolador ATmega328P, utilizado como núcleo de processamento do protótipo.....	92
Figura 10 – Esquema eletrônico da Camada 3: LEDs de Sinalização e conector USB. Fonte: Autoria própria (2025).....	93
Figura 11 – Camada de energia. Fonte: Autoria própria (2025).....	95
Figura 12 – Camada de controle Fonte: Autoria própria (2025).....	95
Figura 13 – Camada de indicação Fonte: Autoria própria (2025).....	96
Figura 14 – Interface de seleção de componentes no catálogo SMT da JLCPCB.....	97
Figura 15 – Renderização 3D da carcaça com as três camadas de PCB instaladas	98

Figura 16 – Partes da carcaça 3D projetada para o protótipo: (a) base, (b) tampa e (c) suporte interno.....	98
Figura 17 – Camadas do protótipo: energia, controle e indicação.....	99
Figura 18 – Etapas de montagem do protótipo na carcaça.....	100
Figura 19 – Protótipo final montado em carcaça, com fixação em trilho padrão industrial.....	100
Figura 20 – Montagem física da gravação ISP entre Arduino UNO e o protótipo PLC 51	Figura
21 – Etapa 1 – Abertura do exemplo ArduinoISP na IDE do Arduino.....	101
Figura 22 – Etapa 2 – Seleção da porta COM do Arduino UNO.....	102
Figura 23 – Etapa 3 – Seleção da placa “Arduino UNO”	103
Figura 24 – Etapa 4 – Upload do código “ArduinoISP” para o Arduino UNO.....	104
Figura 25 – Etapa 5 – Seleção da porta COM do protótipo (Nano).....	105
Figura 26 – Etapa 6 – Seleção da placa “Arduino Nano”	106
Figura 27 – Etapa 7 – Seleção do processador “ATmega328P (Old Bootloader)”	106
Figura 28 – Etapa 8 – Seleção do programador “Arduino as ISP”	107
Figura 29 – Etapa 9 – Execução do comando “Gravar Bootloader”	107
Figura 30 – Diagrama de blocos – Conversão de entrada analógica (0–10V) em saída PWM (0–10V).....	110
Figura 31 – Interface do LDmicro com lógica Ladder desenvolvida	114
Figura 32 – Seleção do microcontrolador no Ldmicro.....	114
Figura 33 – Exportação do código C para uso na IDE Arduino.....	115
Figura 34 – Exportação do arquivo .hex para gravação via AVRDUDESS.....	116
Figura 35 – Tela de configuração do AVRDUDESS para gravação do arquivo .hex	116

LISTA DE TABELAS

Tabela 1 – Comparativo entre CLPs comerciais. Fonte: Autoria própria (2025), com base em dados dos fabricantes.....	86
Tabela 2 – Especificações técnicas completas do protótipo. Fonte: Autoria própria (2025).....	88
Tabela 3 – Comparativo entre o protótipo desenvolvido e CLPs comerciais. Fonte: Autoria própria (2025); dados dos CLPs extraídos de seus respectivos manuais.....	89
Tabela 4 – Mapeamento de conexões entre Arduino Nano e o conector ISP (J5) do protótipo....	101
Tabela 5 – Mapeamento de pinos entre o PLC e o ATmega328P.....	108
Tabela 6 – Número de portas disponíveis no protótipo.....	109
Tabela 7 – Comparativo de custos entre CLPs comerciais e o protótipo desenvolvido.....	119

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog-to-Digital Converter (Conversor Analógico-Digital)	AI	Analog Input	(Entrada Analógica)
Arduino IDE	Arduino Integrated Development Environment (Ambiente de Desenvolvimento Integrado Arduino)			
ATmega328P	Microcontrolador AVR da família Atmel, modelo 328P, utilizado no protótipo			
AVR	Alf and Vegard's RISC processor (Arquitetura de Microcontroladores)			
AVRDUDESS	AVR Downloader/UploaDEr			
Graphical Interface	(Interface gráfica para o programa AVRDUde)			
BOM	Bill of Materials (Lista de Materiais)			
C++	Linguagem de Programação C++			
CAD	Computer-Aided Design (Projeto Assistido por Computador)	CLP	Controlador Programável	Lógico
CNI	Confederação Nacional da Indústria			
CNT	Counter (Contador)			
CPU	Central Processing Unit (Unidade Central de Processamento)			
DC	Direct Current (Corrente Contínua)			
EEPROM	Electrically Erasable Programmable Read-Only Memory (Memória Programável e Apagável Eletricamente)			
Ethernet	Rede local baseada em protocolo IEEE 802.3			
FBD	Function Block Diagram (Diagrama de Blocos de Função)			
FIEP	Federação das Indústrias do Estado do Paraná			
Gerber	Formato de arquivo padrão para fabricação de PCBs	I2C		
Inter-Integrated Circuit	(Circuito Inter-Integrado)			
IA	Artificial Intelligence (Inteligência Artificial)			
IEEE 802.3				

I/O Padrão IEEE para Ethernet (tecnologia de rede local com fio)

Input/Output (Entrada e Saída)

IEC International Electrotechnical Commission (Comissão Eletrotécnica Internacional)

IDE Integrated Development Environment (Ambiente de Desenvolvimento Integrado)

IHM Interface Homem-Máquina

IoT Internet of Things (Internet das Coisas)

IPC Institute of Printed Circuits (Instituto de Conexões e Embalagens Eletrônicas)

ISP In-System Programming (Programação no Sistema)

JLCPCBJiaLiChuang Printed Circuit Board (Fabricante chinesa de PCBs)

Ladder Linguagem de programação gráfica baseada em diagramas de contato (Diagrama Ladder)

LCD Liquid Crystal Display (Display de Cristal Líquido)

LDmicro Software livre para programação de microcontroladores em Ladder

LED Light Emitting Diode (Diodo Emissor de Luz)

Modbus Modular Digital Bus (protocolo de comunicação serial industrial)

MISO Master In, Slave Out (Entrada do Mestre, Saída do Escravo)

MOSI Master Out, Slave In (Saída do Mestre, Entrada do Escravo)

MOV Metal-Oxide Varistor (Varistor de Óxido Metálico)

NA Normally Open (Contato Normalmente Aberto)

NF Normally Closed (Contato Normalmente Fechado)

PCB Printed Circuit Board (Placa de Circuito Impresso)

PLC Programmable Logic Controller (Controlador Lógico Programável)

PNP Positive-Negative-Positive (Tipo de transistor/sensor de saída)

POO Programação Orientada a Objetos

Profibus Process Field Bus (protocolo de comunicação industrial)

PWM Pulse Width Modulation (Modulação por Largura de Pulso)

RAM	Random Access Memory (Memória de Acesso Aleatório)
RTC	Real-Time Clock (Relógio de Tempo Real)
RS485	Standard de comunicação serial balanceada
SCADA	Supervisory Control and Data Acquisition (Sistema de Supervisão e Aquisição de Dados)
SCK	Serial Clock (Sinal de Clock Serial)
SMT	Surface-Mount Technology (Tecnologia de Montagem em Superfície)
SPI	Serial Peripheral Interface (Interface Periférica Serial)
ST	Structured Text (Texto Estruturado)
STL	Standard Template Library (Biblioteca de Modelos Padrão)
TIM	Timer (Temporizador)
TVS	Transient Voltage Suppressor (Diodo Supressor de Transiente)
UART	Universal Asynchronous Receiver-Transmitter (Transmissor/Receptor Assíncrono Universal)
USB	Universal Serial Bus (Barramento Serial Universal)
VAC	Volts Alternating Current (Volts em Corrente Alternada)
Vca	Volts Corrente Alternada
Vcc	Voltage Common Collector (Tensão de Coletor Comum)
VDC	Volts Direct Current (Volts em Corrente Contínua)
Wi-Fi	Wireless Fidelity (Padrão de rede sem fio)

1 INTRODUÇÃO

Este capítulo apresenta as fundamentações e definições necessárias para o entendimento do projeto de forma coesa e sólida. Inicialmente, contextualiza-se o tema, abordando os principais conceitos que norteiam os estudos, além da motivação e da relevância do projeto. Em seguida, são apresentados os objetivos gerais e específicos, que auxiliam na compreensão dos resultados esperados.

O avanço tecnológico e a automação têm desempenhado um papel fundamental na indústria moderna, impulsionando a eficiência, a produtividade e a qualidade dos processos. Nesse contexto, o ensino de controle e automação torna-se uma área de extrema importância para a formação de profissionais capacitados a lidar com os desafios da indústria contemporânea.

A automação industrial está passando por uma transformação sem precedentes, impulsionada pela Indústria 4.0 e pela integração de tecnologias inovadoras, como Inteligência Artificial (IA), Internet das Coisas (IoT), robótica colaborativa e manufatura aditiva. Para se ter uma ideia, o mercado da Indústria 4.0 no Brasil movimentou US\$ 1,77 bilhão em 2022, com um crescimento anual médio de 18,8% entre 2017 e 2022. As projeções indicam que esse valor pode chegar a US\$ 5,62 bilhões até 2028, com um ritmo ainda mais acelerado de 21% ao ano entre 2023 e 2028. Os dados são do estudo Monitor da Indústria 4.0, da International Market Analysis Research and Consulting (IMARC), analisado pelo Observatório Nacional da Indústria, da Confederação Nacional da Indústria (CNI). Essa revolução representa uma oportunidade estratégica para a indústria nacional aumentar sua participação no PIB e fortalecer sua competitividade global. A interconexão de máquinas e sistemas digitais permite um gerenciamento mais eficiente da produção, reduzindo falhas e otimizando recursos. (CNI, 2025)

No âmbito educacional, os laboratórios desempenham um papel crucial no processo de aprendizagem prática, proporcionando aos estudantes a oportunidade de aplicar os conhecimentos teóricos adquiridos em sala de aula. No entanto, a disponibilidade de recursos didáticos que simulem de forma eficaz os sistemas de controle industriais é essencial para um ensino mais completo e abrangente.

Diante desse cenário, surge a necessidade de desenvolver um protótipo de um **Controlador Lógico Programável (CLP)** didático e funcional, que possa auxiliar o aprendizado em laboratórios educacionais de controle e automação. O objetivo principal deste trabalho é criar uma solução acessível e eficiente para o ensino desses conceitos, utilizando a plataforma **Arduino** (ARDUINO, 2025) e a linguagem **Ladder** (ROQUE, 2014) como alternativas baseadas em tecnologias de código aberto.

1.1 JUSTIFICATIVA

O desenvolvimento deste protótipo de Controlador Lógico Programável (CLP) didático e funcional se justifica pela carência de recursos adequados atualmente disponíveis nos laboratórios educacionais. Muitas vezes, os equipamentos utilizados são caros, complexos e voltados para aplicações industriais, dificultando o acesso e a compreensão por parte dos estudantes em fase de aprendizagem.

Ao criar um protótipo didático baseado em tecnologias amplamente disponíveis e de baixo custo, busca-se democratizar o acesso a recursos de controle e automação (ROQUE, 2014), permitindo que estudantes de diferentes instituições tenham a oportunidade de desenvolver habilidades práticas nessa área.

Além disso, o uso da plataforma Arduino e da linguagem Ladder como alternativas de código aberto permite que o protótipo seja personalizável e adaptável às necessidades específicas de cada laboratório educacional. Isso proporciona flexibilidade no desenvolvimento e amplia o potencial de uso do CLP didático em diferentes contextos.

Com a finalização deste protótipo, espera-se contribuir significativamente para o ensino de controle e automação, oferecendo uma ferramenta prática, intuitiva e de baixo custo que simula as operações de um CLP industrial. Acredita-se que essa solução acessível proporcionará aos estudantes uma experiência de aprendizagem enriquecedora, preparando-os de forma mais eficaz para enfrentar os desafios da indústria moderna.

2 OBJETIVOS

Nesta seção, são apresentados os objetivos que se pretendem alcançar com este trabalho.

2.1 OBJETIVO GERAL

Desenvolver um protótipo didático de Controlador Lógico Programável (CLP) e um software com programação em Ladder, utilizando tecnologias de código aberto, com o intuito de oferecer uma ferramenta acessível e eficiente para o ensino de controle e automação em laboratórios educacionais.

2.2 OBJETIVOS ESPECÍFICOS

- a) Projetar e construir o hardware do protótipo de CLP;
- b) Desenvolver o software de programação em Ladder;
- c) Testar e validar o protótipo em cenários de simulação industrial;

- d) Documentar o processo de desenvolvimento e os resultados obtidos.

2.3 ESTRUTURA DO TRABALHO

Neste capítulo, foram apresentados os objetivos geral e específicos deste trabalho, seguidos da descrição da estrutura adotada para o desenvolvimento do estudo.

O **Capítulo 3** apresenta a fundamentação teórica necessária para a compreensão do projeto. São discutidos os conceitos de microcontroladores e controladores lógicos programáveis (CLPs), incluindo suas arquiteturas, interfaces de entrada e saída, e a importância desses dispositivos no contexto educacional.

O **Capítulo 4** detalha a metodologia aplicada no desenvolvimento do protótipo, abordando desde a seleção e modelagem dos componentes até a montagem física e a implementação do software, utilizando a linguagem Ladder e a plataforma Arduino.

O **Capítulo 5** descreve os testes realizados e a avaliação prática do protótipo. São apresentados os procedimentos de verificação funcional e os resultados obtidos.

O **Capítulo 6** discute os resultados, analisando as potencialidades do protótipo desenvolvido em comparação com soluções comerciais, além de destacar suas vantagens e limitações.

O **Capítulo 7** apresenta as conclusões do trabalho, sintetizando os principais achados e propondo melhorias e possíveis desdobramentos futuros.

Ao final, são listadas as referências bibliográficas utilizadas ao longo do desenvolvimento do trabalho.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos e teorias que embasam o desenvolvimento do protótipo de Controlador Lógico Programável (CLP) didático. A fundamentação teórica é essencial para compreender as bases tecnológicas e metodológicas que sustentam este trabalho, além de fornecer uma visão clara sobre os principais componentes e tecnologias envolvidos.

Inicialmente, discute-se a importância dos microcontroladores no contexto da automação industrial, destacando suas funcionalidades e aplicações. Em seguida, são apresentados os Controladores Lógicos Programáveis (CLPs), sua arquitetura, interfaces e papel essencial na automação de processos.

Também são abordadas a plataforma Arduino e as linguagens de programação C++ e Ladder, ferramentas utilizadas no desenvolvimento do protótipo. Por fim, analisa-se o ecossistema dos CLPs e

sua relevância educacional, visando compreender as potencialidades e limitações de uma solução didática para o ensino de controle e automação.

3.1 MICROCONTROLADORES

Um microcontrolador é um circuito integrado que combina, em um único chip, uma unidade central de processamento (CPU), memória e periféricos de entrada e saída (I/O). Amplamente empregado em sistemas embarcados, permite controlar dispositivos eletrônicos em aplicações que vão desde eletrodomésticos até sistemas industriais complexos.

Criado na década de 1970, o microcontrolador revolucionou o design de sistemas eletrônicos ao integrar funcionalidades que anteriormente exigiam múltiplos componentes, possibilitando a automação inteligente de dispositivos com baixo consumo de energia e custo reduzido.

Entre as arquiteturas mais utilizadas destacam-se as famílias PIC (Microchip), MCS-51 (Intel) e AVR (Atmel). Esta última foi adotada na plataforma Arduino, devido à sua simplicidade de uso e à ampla comunidade de suporte (IEEE, 2025b).

3.1.1 ORIGEM E FILOSOFIA

O microcontrolador surgiu como uma evolução natural dos microprocessadores, com a proposta de integrar, em um único encapsulamento, os recursos necessários para o controle autônomo de sistemas. A Intel e a Texas Instruments foram pioneiras nessa tecnologia, com destaque para o modelo Intel 8048 lançado em 1976 (IEEE, 2025a).

Sua filosofia é baseada na união de hardware e software, permitindo a criação de sistemas embarcados autossuficientes e personalizáveis para tarefas específicas.

3.1.2 COMPONENTES PRINCIPAIS

Os microcontroladores integram diversos blocos funcionais em um único chip. Entre os principais componentes destacam-se:

- **Unidade Central de Processamento (CPU):** executa as instruções armazenadas na memória e controla os periféricos conectados ao microcontrolador.
- **Memória:**
 - **Flash:** armazena o código do programa (não volátil).
 - **RAM:** utilizada durante a execução para armazenamento temporário.
 - **EEPROM:** permite armazenar dados de forma permanente, mesmo com o desligamento.

- Periféricos de Entrada/Saída (I/O): pinos digitais ou analógicos usados para interagir com sensores, atuadores e outros dispositivos externos.
- Conversores Analógico-Digitais (ADC): convertem sinais analógicos (ex.: tensão de 0–5 V) em valores digitais.
- Timers e contadores: recursos internos utilizados em temporizações, medições e geração de pulsos (PWM).
- Interfaces de comunicação: protocolos como UART, SPI, I2C e USB viabilizam a integração com dispositivos externos e outros microcontroladores.
- Oscilador: circuito interno ou externo que gera o sinal de clock para sincronização das operações.

Segundo Tocci (TOCCI; WIDMER; MOSS, 2011) e Bolton (BOLTON, 2015),

a integração desses componentes em um único chip possibilitou a difusão de sistemas embarcados acessíveis, com baixo consumo energético e grande aplicabilidade. No caso específico do ATmega328P, utilizado na plataforma Arduino, seu datasheet (INC., 2016) destaca a presença de CPU RISC de 8 bits, memória Flash de 32 kB, EEPROM de 1 kB, SRAM de 2 kB, além de periféricos de comunicação e temporização.

3.1.3 VANTAGENS DOS MICROCONTROLADORES

- Integração de componentes: reduz a necessidade de circuitos externos e simplifica o projeto.
- Baixo custo e consumo: ideal para aplicações embarcadas de pequeno porte ou com restrições energéticas.
- Facilidade de desenvolvimento: plataformas como o Arduino abstraem a complexidade do hardware e facilitam a programação.
- Versatilidade: aplicável em contextos residenciais, automotivos, industriais e de consumo.

3.2 CONTROLADOR LÓGICO PROGRAMÁVEL (CLP)

O Controlador Lógico Programável (CLP), ou em inglês Programmable Logic Controller (PLC), é um dispositivo eletrônico digital utilizado na automação de processos industriais. Desenvolvido na década de 1960 para substituir sistemas baseados em relés, tornou-se uma ferramenta essencial para o controle preciso de máquinas, linhas de produção e processos complexos. Sua flexibilidade, confiabilidade e robustez permitem sua aplicação em diversos setores, como manufatura, energia, petróleo e gás (BOLTON, 2015; ROQUE, 2014; FRANCHI, 2008).

3.2.1 ORIGEM E FILOSOFIA

O CLP surgiu como solução para os desafios da indústria automobilística, que necessitava de sistemas de controle mais flexíveis do que os painéis de relés convencionais. A primeira geração foi desenvolvida pela empresa Bedford Associates, em 1968, sob o nome Modicon (Modular Digital Controller). Sua filosofia é pautada na simplicidade da programação, confiabilidade operacional e adaptabilidade a diferentes cenários industriais (FRANCHI, 2008).

Esses dispositivos são projetados para executar tarefas lógicas, sequenciais e tempo- rizadas com base em programas armazenados em sua memória — geralmente escritos em linguagens como o diagrama Ladder (ROQUE, 2014). Os CLPs recebem sinais de sensores (como botões e sensores de proximidade), processam-nos conforme a lógica implementada e atuam sobre saídas conectadas a motores, válvulas e lâmpadas.

3.2.2 ELEMENTOS DE UM SISTEMA DE CONTROLE

O sistema de controle industrial é comumente dividido em três elementos principais: sensores, atuadores e controladores (ROQUE, 2014):

- **Sensores:** captam grandezas físicas (temperatura, pressão, posição) e as convertem em sinais elétricos.
- **Atuadores:** executam ações físicas com base nos comandos dos controladores (motores, válvulas, relés).
- **Controladores:** processam dados dos sensores, aplicam algoritmos de controle e emitem sinais para os atuadores. O CLP é o exemplo clássico desse elemento.

O funcionamento do sistema é cíclico: sensores informam o estado do processo, controladores tomam decisões e atuadores realizam ações — repetindo-se continuamente.

3.2.3 COMPONENTES PRINCIPAIS DE UM CLP

- **Unidade Central de Processamento (UCP):** executa o programa de controle, processa entradas e gera saídas. É composta por microcontrolador, memória e interfaces de comunicação.
- **Módulos de Entrada e Saída (I/O):** captam sinais externos (entradas) e acionam dispositivos (saídas), podendo ser digitais ou analógicos.
- **Fonte de Alimentação:** fornece energia estável para os módulos do CLP.
- **Interface de Programação:** software utilizado para programar o CLP, geralmente conforme a norma IEC 61131-3 (Ladder Diagram, Structured Text, Function Block Diagram).
- **Módulos de Comunicação:** permitem integração com redes industriais (Ethernet, Modbus, Profibus) e sistemas supervisórios.

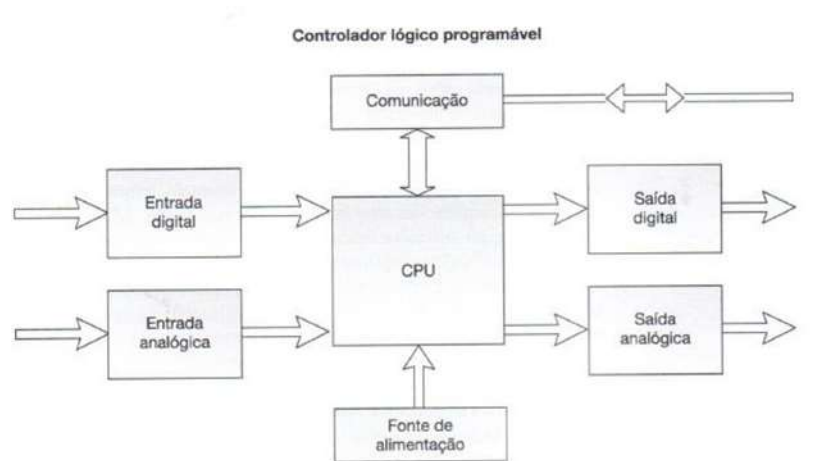


Figura 1 – Controlador Lógico Programável estruturado em blocos. Adaptado de (ROQUE, 2014).

3.2.4 VANTAGENS DOS CLPS

- **Confiabilidade:** projetados para ambientes industriais adversos, resistem a vibração, calor e interferência eletromagnética.
- **Flexibilidade:** podem ser reprogramados para novas aplicações sem alterações no hardware.
- **Facilidade de manutenção:** a estrutura modular facilita diagnósticos e substituições rápidas.
- **Escalabilidade:** permitem expansão com novos módulos de I/O e comunicação.
- **Integração com sistemas supervisórios:** compatíveis com SCADA e IHMs, viabilizam monitoramento remoto.

3.2.5 APLICAÇÕES

Os CLPs são amplamente utilizados em (FRANCHI, 2008; ROQUE, 2014):

- Controle de máquinas: máquinas-ferramenta, impressoras, braços robóticos.
- Processos industriais: controle de temperatura, vazão e pressão.
- Linhas de produção: sincronização de esteiras, robôs e sistemas de montagem.
- Sistemas de energia: controle de geradores e subestações.
- Automação predial: iluminação, climatização e segurança.

3.2.6 DESAFIOS E LIMITAÇÕES

- Custo inicial elevado: especialmente em projetos de grande porte.
- Complexidade de programação: aplicações avançadas exigem conhecimento técnico.
- Limitações de hardware: processamento e memória restritos.
- Integração com sistemas externos: depende da compatibilidade com sensores, atuadores e redes.

3.2.7 CONCLUSÃO

O CLP é um dos pilares da automação industrial, unindo confiabilidade, adaptabilidade e robustez. Sua evolução, incorporando recursos como IoT e inteligência artificial, tem expandido suas capacidades. No escopo deste

3.3 ARDUINO

O Arduino é uma plataforma de prototipagem eletrônica de código aberto amplamente utilizada em projetos de automação, robótica, Internet das Coisas (IoT), controle embarcado e educação técnica. Sua popularidade deve-se à acessibilidade, simplicidade de uso, baixo custo e vasta comunidade de suporte (ARDUINO, 2025).

3.3.1 ORIGEM E FILOSOFIA

Criado em 2005 por Massimo Banzi e sua equipe na Itália, o Arduino nasceu como uma solução acessível para estudantes e educadores desenvolverem projetos eletrônicos e interativos (BANZI, 2011). A plataforma adota a filosofia de open-source, oferecendo gratuitamente o código-fonte do software (Arduino IDE) e os esquemáticos do hardware das placas.

Essa abordagem colaborativa permitiu a criação de uma grande comunidade global, que contribui continuamente com bibliotecas, tutoriais e exemplos de aplicação em diversas áreas, ampliando o alcance da plataforma para além do meio educacional, chegando até o mercado profissional e de prototipagem rápida.

3.3.2 COMPONENTES PRINCIPAIS

- **Placa de Circuito Impresso (PCI):**
- Núcleo do sistema, composta por um microcontrolador (como o ATmega328P) e circuitos auxiliares de alimentação, comunicação e interface com o ambiente externo (INC., 2016).
- Existem diversas versões: Arduino Uno, Mega, Nano, Due, entre outras, que variam em capacidade, tamanho e recursos disponíveis.
- Ambiente de Desenvolvimento Integrado (IDE):
- O Arduino IDE permite escrever, compilar e transferir o código para a placa por meio de uma interface simples.
- A linguagem de programação é baseada em C/C++, com diversas bibliotecas prontas para uso.
- **Shields e Módulos:**

- Shields são placas de expansão que se encaixam sobre o Arduino para adicionar funções, como controle de motores, comunicação sem fio, ethernet, entre outras.
- Módulos são dispositivos conectados externamente via fios aos pinos da placa, como sensores de temperatura, displays LCD, módulos Bluetooth, etc.
- **Bootloader:**
- Pequeno programa pré-instalado no microcontrolador que permite carregar o código via porta USB, sem a necessidade de gravadores externos.
- **Conectividade e Comunicação:**
- A maioria das placas oferece conexão USB para programação e comunicação serial. Algumas versões possuem conectividade adicional, como Wi-Fi, Bluetooth ou Ethernet.



Figura 2 – Placa Arduino Uno, uma das versões mais utilizadas em projetos didáticos e de prototipagem (ARDUINO, 2025).

3.3.3 CONCLUSÃO

O Arduino é uma plataforma consolidada e amplamente adotada no ensino e desenvolvimento de sistemas embarcados, oferecendo uma excelente relação custo-benefício. Sua abordagem de código aberto, somada à grande disponibilidade de recursos, torna-o ideal para projetos didáticos, prototipagem rápida e aplicações de baixo a médio nível de complexidade. Neste trabalho, sua utilização permitiu a construção de um CLP didático funcional, acessível e alinhado às práticas de ensino e aprendizado da automação industrial.

3.4 LINGUAGEM LADDER

A Linguagem Ladder, também conhecida como Diagrama Ladder ou Ladder Logic, é uma das linguagens de programação gráficas mais difundidas no contexto da automação industrial. Criada a partir da representação esquemática dos circuitos de relés, sua principal característica é a semelhança

visual com uma escada (ladder, em inglês), formada por dois trilhos verticais que simbolizam a alimentação elétrica e diversos degraus horizontais que representam os circuitos lógicos de controle (ROQUE, 2014).

3.4.1 ORIGEM E FILOSOFIA

A Linguagem Ladder foi desenvolvida na década de 1970, durante a transição dos sistemas de comando baseados em relés para os primeiros CLPs. Seu propósito era facilitar a migração dos profissionais da época, mantendo uma simbologia próxima dos diagramas elétricos convencionais. Assim, técnicos e engenheiros, já habituados a esquemas de contadores e relés, puderam programar CLPs de forma intuitiva, sem a necessidade de aprender linguagens textuais complexas (ROQUE, 2014).

Com o passar dos anos, a Ladder evoluiu significativamente, incorporando novos elementos lógicos e funções que permitem implementar desde controles discretos simples até processos industriais complexos. A padronização promovida pela norma internacional IEC 61131-3 consolidou seu uso, garantindo compatibilidade entre diferentes fabricantes de CLPs e tornando a Ladder uma linguagem de referência mundial.

3.4.2 ESTRUTURA E FUNCIONAMENTO

Um programa em Ladder é estruturado como uma sequência de "degraus"(rungs) dispostos entre dois trilhos verticais. Cada degrau representa uma instrução lógica composta por elementos básicos como contatos e bobinas. A varredura do programa é feita ciclicamente pelo CLP, da esquerda para a direita e de cima para baixo, garantindo a atualização contínua das entradas, do processamento lógico e das saídas (ROQUE, 2014).

- **Contatos:** simulam dispositivos de entrada, podendo ser normalmente abertos (NA) ou normalmente fechados (NF).
- **Bobinas:** representam dispositivos de saída que são energizados conforme as condições lógicas dos contatos.
- **Elementos Especiais:** incluem temporizadores (TIM), contadores (CNT), comparadores, registradores e blocos de função, que permitem a criação de rotinas avançadas de controle.

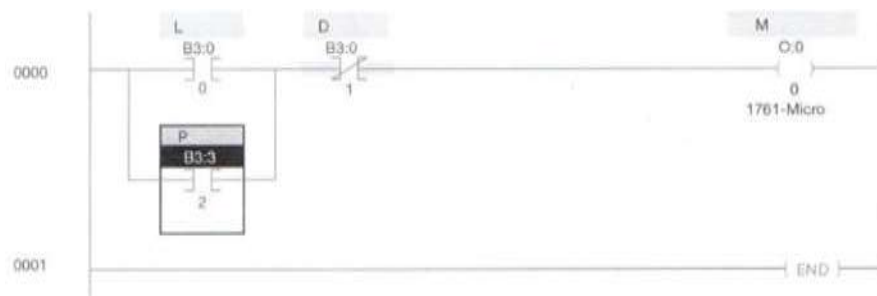


Figura 3 – Programa de automação em linguagem Ladder. Adaptado de (ROQUE, 2014).

A Figura 3 apresenta um exemplo prático de programa Ladder, no qual a lógica de contatos e bobinas é aplicada para o acionamento de dispositivos, evidenciando a clareza visual da linguagem.

3.4.3 VANTAGENS

- **Facilidade de Aprendizado:** sua base em diagramas elétricos facilita a assimilação por profissionais de automação e eletrotécnica.
- **Visualização Intuitiva:** a representação gráfica permite rápida interpretação, diagnóstico de falhas e manutenção em tempo real.
- **Compatibilidade Industrial:** consolidada como padrão de mercado, é suportada pela maioria dos CLPs comerciais.
- **Integração com Outras Linguagens:** pode ser combinada com Structured Text (ST) e Function Block Diagram (FBD), conforme a norma IEC 61131-3.

3.4.4 APLICAÇÕES

A Ladder é utilizada em praticamente todos os setores da automação industrial, destacando-se em (ROQUE, 2014):

- **Controle de Máquinas:** esteiras, prensas, empacotadoras e injetoras.
- **Linhas de Produção:** sincronização de motores, sensores e atuadores em sistemas automatizados.
- **Sistemas de Segurança:** implementação de intertravamentos, bloqueios e barreiras de proteção.
- Automação Predial: controle de iluminação, bombas, climatização e elevadores.

3.4.5 DESAFIOS E LIMITAÇÕES

Apesar de suas vantagens, a Ladder apresenta limitações quando aplicada a sistemas de maior complexidade:

- Escalabilidade: programas muito extensos podem se tornar de difícil organização visual.

- Manipulação de Dados: operações complexas com estruturas ou cadeias de caracteres são menos eficientes em Ladder.
- Curva de Complexidade: para implementar algoritmos avançados ou cálculos matemáticos, pode demandar esforço significativo.

3.4.6 CONCLUSÃO

A Linguagem Ladder permanece como um dos pilares da programação de CLPs, unindo tradição, clareza e confiabilidade. Sua adoção global e padronização pela IEC 61131-3 reforçam sua importância como ferramenta de ensino e prática industrial. No contexto deste trabalho, a Ladder foi escolhida por sua compatibilidade com a plataforma Arduino e por sua relevância didática, oferecendo aos estudantes um ambiente de programação intuitivo e alinhado às práticas industriais reais (ROQUE, 2014).

3.5 LINGUAGEM C++

A linguagem C++ é uma das linguagens de programação mais poderosas e versáteis disponíveis atualmente. Desenvolvida por Bjarne Stroustrup no início da década de 1980, o C++ é uma extensão da linguagem C, incorporando recursos de Programação Orientada a Objetos (POO). Essa combinação de eficiência, flexibilidade e abstração consolidou o C++ como escolha popular em áreas que vão desde sistemas embarcados até softwares de alto desempenho (STROUSTRUP, 2013; International Organization for Standardization, 2020).

3.5.1 ORIGEM E FILOSOFIA

O C++ foi criado com o objetivo de adicionar suporte a classes, herança, polimorfismo e outros conceitos de POO à linguagem C, preservando sua eficiência e compatibilidade com hardware de baixo nível. A primeira versão, chamada C with Classes, foi lançada em 1983. Em 1985, a linguagem foi renomeada para C++ — em alusão ao operador de incremento da linguagem C, simbolizando uma evolução (STROUSTRUP, 2013).

Desde então, o C++ passou por diversas atualizações, como os padrões C++98, C++11, C++14, C++17 e C++20. Cada versão trouxe melhorias de desempenho, novos recursos e maior facilidade de uso, consolidando o C++ como uma linguagem moderna e relevante (International Organization for Standardization, 2020).

3.5.2 CARACTERÍSTICAS PRINCIPAIS

- Programação Orientada a Objetos (POO):
- Classes e Objetos: permitem modelar entidades do mundo real, encapsulando dados e comportamentos.
- Herança: facilita a reutilização de código e a criação de hierarquias de classes.
- Polimorfismo: possibilita que objetos de diferentes classes sejam tratados de maneira uniforme.
- Encapsulamento: protege os dados internos de uma classe, expondo apenas métodos públicos.
- **Alto Desempenho:**
- Linguagem compilada, com código altamente otimizado.
- Controle direto de recursos de hardware, como memória e registradores.
- Flexibilidade:
- Suporte a múltiplos paradigmas: procedural, orientado a objetos e genérico (templates).
- Aplicável tanto em sistemas de baixo nível (como drivers) quanto em softwares de alto nível (interfaces gráficas).
- Biblioteca Padrão (STL - Standard Template Library):
- Estruturas de dados (vetores, listas, mapas) e algoritmos (ordenação, busca) prontos para uso.
- Compatibilidade com C:
- Total compatibilidade com bibliotecas escritas em C, facilitando a integração de códigos existentes.

3.5.3 APLICAÇÕES

O C++ é amplamente utilizado em diferentes domínios (TOCCI; WIDMER; MOSS, 2011):

- Sistemas Operacionais: componentes do Windows, Linux e macOS.
- Jogos e Gráficos: motores como Unreal Engine são escritos em C++.
- Sistemas Embarcados: empregado em microcontroladores e automação industrial.
- Aplicações de Alto Desempenho: servidores, bancos de dados e simulações científicas.
- Ferramentas de Desenvolvimento: compiladores, interpretadores e IDEs.

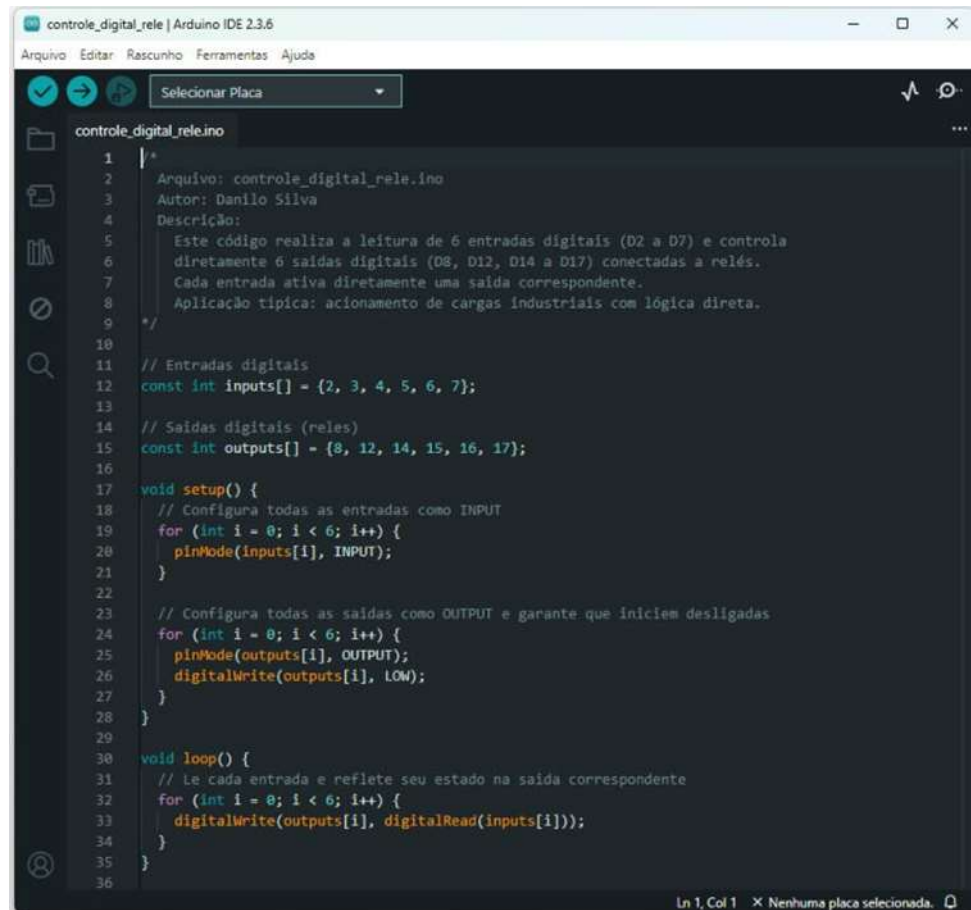


Figura 4 – Exemplo de código em C++ aplicado no ambiente Arduino, ilustrando a integração entre programação textual e sistemas embarcados. Fonte: Autoria própria (2025).

3.5.4 VANTAGENS

- Desempenho: uma das linguagens mais rápidas e eficientes disponíveis.
- Controle de Recursos: permite gerenciamento manual de memória e controle de baixo nível.
- Portabilidade: pode ser compilada em diversas plataformas.
- Amplo Suporte: extensa comunidade, bibliotecas e frameworks disponíveis.

3.5.5 DESAFIOS E LIMITAÇÕES

- Complexidade: sintaxe robusta e conceitos avançados que podem dificultar o aprendizado.
- Gerenciamento de Memória: controle manual pode gerar falhas como vazamentos ou ponteiros inválidos.
- Curva de Aprendizado: exige familiaridade com ponteiros, templates e programação genérica.

3.5.6 CONCLUSÃO

O C++ é uma linguagem poderosa no universo da programação, combinando performance, versatilidade e suporte a múltiplos paradigmas. Neste trabalho, foi utilizada para integrar a linguagem Ladder ao protótipo de CLP, demonstrando sua eficácia em aplicações de automação baseadas em tecnologias de código aberto. Sua ampla adoção e evolução contínua consolidam o C++ como uma linguagem essencial no ensino e na indústria (STROUSTRUP, 2013; ROQUE, 2014).

4 DESENVOLVIMENTO

Neste capítulo, detalha-se o processo de desenvolvimento do protótipo de Controlador Lógico Programável (CLP) didático e funcional. O trabalho abrangeu diversas etapas, iniciando pela seleção de componentes e definição de requisitos, passando pela construção e implementação do hardware e do software, e culminando na validação do protótipo.

4.1 METODOLOGIA

A metodologia adotada neste trabalho foi organizada em duas frentes complementares: (i) desenvolvimento do hardware e (ii) desenvolvimento do software. Cada frente foi planejada e executada de modo a garantir a eficácia e a funcionalidade do protótipo de CLP didático.

- Desenvolvimento do hardware: Tomaram-se como referência modelos comerciais de CLPs industriais para orientar as especificações do protótipo didático, considerando arquitetura, interfaces de entrada/saída (I/O) e requisitos de comunicação. Definiram-se as especificações técnicas (microcontrolador, quantidade de entradas e saídas e interfaces de comunicação) e selecionaram-se componentes aderentes a tais requisitos, adotando-se a plataforma Arduino pela versatilidade, acessibilidade e ampla comunidade de suporte (ARDUINO, 2025). A montagem seguiu o diagrama de hardware, com as devidas conexões elétricas, instalação de sensores/atuadores e módulos de comunicação. Testes preliminares foram realizados para verificar o funcionamento do conjunto.
- Desenvolvimento do software: Utilizou-se a IDE Arduino para a programação do microcontrolador em C/C++, pela compatibilidade com a plataforma e pela disponibilidade de bibliotecas. Foram elaboradas rotinas de controle em linguagem Ladder, amplamente empregada em CLPs industriais, com foco em cenários típicos de automação (ROQUE, 2014). Além disso, empregaram-se ferramentas complementares como o LDmicro, para geração de código Ladder em C, e o AVRDUDESS, para gravação de programas no microcontrolador via interface USB-Serial. Procedeu-se à integração hardware–software, configurando-se as entradas e saídas conforme as especificações do protótipo, seguida de testes funcionais e correções. Todo o processo foi documentado (diagramas, códigos-fonte e instruções de montagem) visando reprodutibilidade e uso didático.

Após o desenvolvimento do hardware e do software, realizaram-se testes para validar a funcionalidade do protótipo, incluindo verificação do correto acionamento de todas as entradas e saídas, execução das rotinas em Ladder e interação com sensores e atuadores. Também foram conduzidas avaliações de robustez frente a diferentes condições operacionais (variações de temperatura e interferências eletromagnéticas), além de aplicações em cenários de ensino para aferir a eficácia do protótipo como ferramenta didática em laboratórios de controle e automação.

4.2 DESENVOLVIMENTO DO HARDWARE

4.2.1 MODELO DE REFERÊNCIA

Durante a análise dos principais modelos de Controladores Lógicos Programáveis (CLPs) disponíveis no mercado, foram considerados critérios como custo-benefício, interfaces intuitivas, documentação abrangente e características padrão da maioria dos CLPs industriais. Essa avaliação resultou na seleção de três modelos amplamente utilizados em automação de pequeno e médio porte: o CLIC-02 da WEG (ELÉTRICOS, 2025), o LOGO! da Siemens (AG, 2025) e o Micrologix 1000 da Allen Bradley (AUTOMATION, 2025b).

Tabela 1 – Comparativo entre CLPs comerciais. Fonte: Autoria própria (2025), com base em dados dos fabricantes.

Característica	Siemens LOGO! 8 24CEo	WEG CLIC-02	Micrologix 1000
Alimentação	24Vcc	24Vcc ou 110-220Vca	24Vcc ou 100-240Vca
Entradas Digitais	8 (4 config. como 0-10V)	8	10
Entradas Analógicas	4 (0-10V)	Não	Não
Saídas Digitais	4 (transistor, 0.3A)	6 (relê, até 10A)	6 (relê/transistor)
Saídas PWM	3 (0-10V)	Não	Não
Expansibilidade	Sim	Limitada	Sim (custo elevado)
Programação	Ladder, FBD	Ladder	Ladder
Custo Relativo	Médio	Baixo	Alto

4.2.1.1 SÍNTESE DAS CARACTERÍSTICAS RELEVANTES

Como evidenciado na Tabela 3 e Figura 5, os modelos apresentam diferenças significativas:

- LOGO! 24CEo: Destacou-se pela flexibilidade (entradas analógicas nativas e expansibilidade visível na Fig. 5a), sendo tecnicamente alinhado com automação moderna
- CLIC-02: Ofereceu melhor custo-benefício (Fig. 5b mostra sua construção simplificada), com saídas a relê robustas.



(a) Siemens LOGO! 8 24CEo



(b) WEG CLIC-02



(c) Allen Bradley Micrologix 1000

Figura 5 – Modelos de CLP analisados como referência: (a) Siemens LOGO! 8 24CEo, (b) WEG CLIC-02, (c) Allen Bradley Micrologix 1000. Fonte: Siemens (2025); WEG (2025); Allen Bradley (2025).

- Micrologix 1000: Representou o padrão industrial (Fig. 5c evidencia sua estrutura modular), porém com custo elevado

Esta análise permitiu identificar os requisitos mínimos desejáveis para o protótipo:

1. Entradas analógicas integradas (inspiradas no LOGO!)
2. Saídas a relê para cargas industriais (como no CLIC-02)
3. Alimentação 24Vcc (padrão nos três modelos)

A definição das especificações técnicas do protótipo será detalhada na Seção 4.2.2.

4.2.2 ESPECIFICAÇÕES DO PROTÓTIPO

Com base na análise dos modelos comerciais (Seção 4.2.1), o protótipo foi desenvolvido com arquitetura modular e recursos ampliados, conforme detalhado a seguir.

4.2.2.1 CARACTERÍSTICAS GERAIS

1. Módulo de Entradas:

- Digitais: Circuito opto-isolado com proteção contra transientes (TVS)
- Analógicas: Condicionamento com amplificador operacional (INA826) e ADC de 12-bit

2. Módulo de Saídas:

- Relés: Contato NA-NF 5A com supressão de arco (varistor MOV)
- PWM: Conversor DAC de 10-bit com filtro ativo LC

3. Núcleo de Processamento:

- Microcontrolador ATmega328P (8-bit AVR, 16 MHz) (INC., 2016).
- Memória: 32 kB Flash, 2 kB SRAM, 1 kB EEPROM (INC., 2016).
- RTC DS1307 (bateria CR2032) (INTEGRATED, 2008).

4. Condições Operacionais

- **Temperatura:** Operação garantida entre -40°C e $+85^{\circ}\text{C}$
- **Umidade:** 10% a 90% não condensante
- **Dimensões:** 74mm $\times$ 63mm $\times$ 25mm (L $\times$ P $\times$ A)
- **Proteção:** Ensaio segundo IEC 61131-2 (vibração/choque) (COMMISSION, 2007)

As características descritas anteriormente foram organizadas de forma consolidada na Tabela 2, a qual apresenta um resumo das especificações técnicas completas do protótipo, incluindo dados elétricos, condições operacionais, quantidade de entradas e saídas, interfaces de comunicação e recursos adicionais embarcados. Essa tabela serve como referência para a construção, validação e documentação técnica do sistema.

Tabela 2 – Especificações técnicas completas do protótipo. Fonte: Autoria própria (2025).

Dados Elétricos		
Tensão de alimentação	24 VDC	$\pm 10\%$
Corrente de alimentação	50 mA	@24VDC
Consumo típico	1.2 W	
Condições Ambientais e Dimensões		
Temperatura mínima	-40°C	
Temperatura máxima	85°C	
Dimensões físicas	74 $\times$ 63 mm	
Entradas		
Digitais (24VDC)	6	Opto-isoladas
Analógicas (0–10V)	2	Resolução 12-bit
Saídas		
Relé (RLY)	6	5A/250VAC ou 30VDC
PWM (0–10V)	3	1kHz max
Comunicação		
USB Tipo-C	1	Programação/dados
RS485	1	Modbus RTU
ISP	1	Atualização de firmware
Recursos Especiais		

RTC (Relógio Tempo Real)	1	Bateria CR2032
LEDs de status	4	Sistema/I/O

Com o objetivo de contextualizar as características do protótipo desenvolvido em relação a soluções comerciais amplamente utilizadas na indústria, a Tabela 3 apresenta um comparativo entre os principais parâmetros técnicos dos controladores Siemens LOGO! 8 24CEo (AG, 2025), WEG CLIC-02 (ELÉTRICOS, 2025) e Allen Bradley Micrologix 1000 (AUTOMATION, 2025b). A comparação inclui aspectos como alimentação, número de entradas e saídas, capacidade analógica, forma de programação, expansibilidade e custo relativo. Essa análise permite evidenciar os diferenciais, limitações e o posicionamento didático do protótipo frente às alternativas disponíveis no mercado.

Tabela 3 – Comparativo entre o protótipo desenvolvido e CLPs comerciais. Fonte: Autoria própria (2025); dados dos CLPs extraídos de seus respectivos manuais.

Característica	Protótipo	Siemens LOGO! 8 24CEo	WEG CLIC-02	Micrologix 1000
Alimentação	24 Vcc	24 Vcc	24 Vcc ou 110–220 Vca	24 Vcc ou 100–240 Vca
Entradas Digitais	6 opto-isoladas	8 (4 como 0–10 V)	8	10
Entradas Analógicas	2 (0–10 V)	4 (0–10 V)	Não	Não
Saídas Digitais	6 (relé)	4 (transistor, 0.3 A)	6 (relé, até 10 A)	6 (relé/transistor)
Expansibilidade	Não	Sim	Limitada	Sim (custo elevado)
Programação	Ladder (LDMicro / C++)	Ladder, FBD	Ladder	Ladder
Custo Relativo	Muito Baixo	Médio	Baixo	Alto

4.2.2.2 ETAPAS DO DESENVOLVIMENTO DO PROTÓTIPO

Com base nas especificações técnicas definidas anteriormente, o desenvolvimento do protótipo foi conduzido por meio de uma sequência estruturada de etapas, desde a modelagem inicial até a produção física do hardware. A seguir, são descritas as principais fases:

4.2.2.2.1 ELABORAÇÃO DO ESQUEMA ELETRÔNICO

A etapa de elaboração do esquema eletrônico consistiu na representação gráfica de todos os circuitos necessários para o funcionamento do protótipo, com base nos requisitos previamente definidos. Essa modelagem foi desenvolvida utilizando um software de CAD eletrônico, como o EasyEDA, o qual oferece integração direta com serviços de manufatura e bibliotecas de componentes amplamente disponíveis no mercado.

O circuito eletrônico completo foi modularizado em três placas independentes (camadas), interligadas por soquetes, com o objetivo de facilitar a montagem, manutenção e compreensão didática do sistema. As camadas foram organizadas da seguinte forma, conforme ilustra a Figura 6.

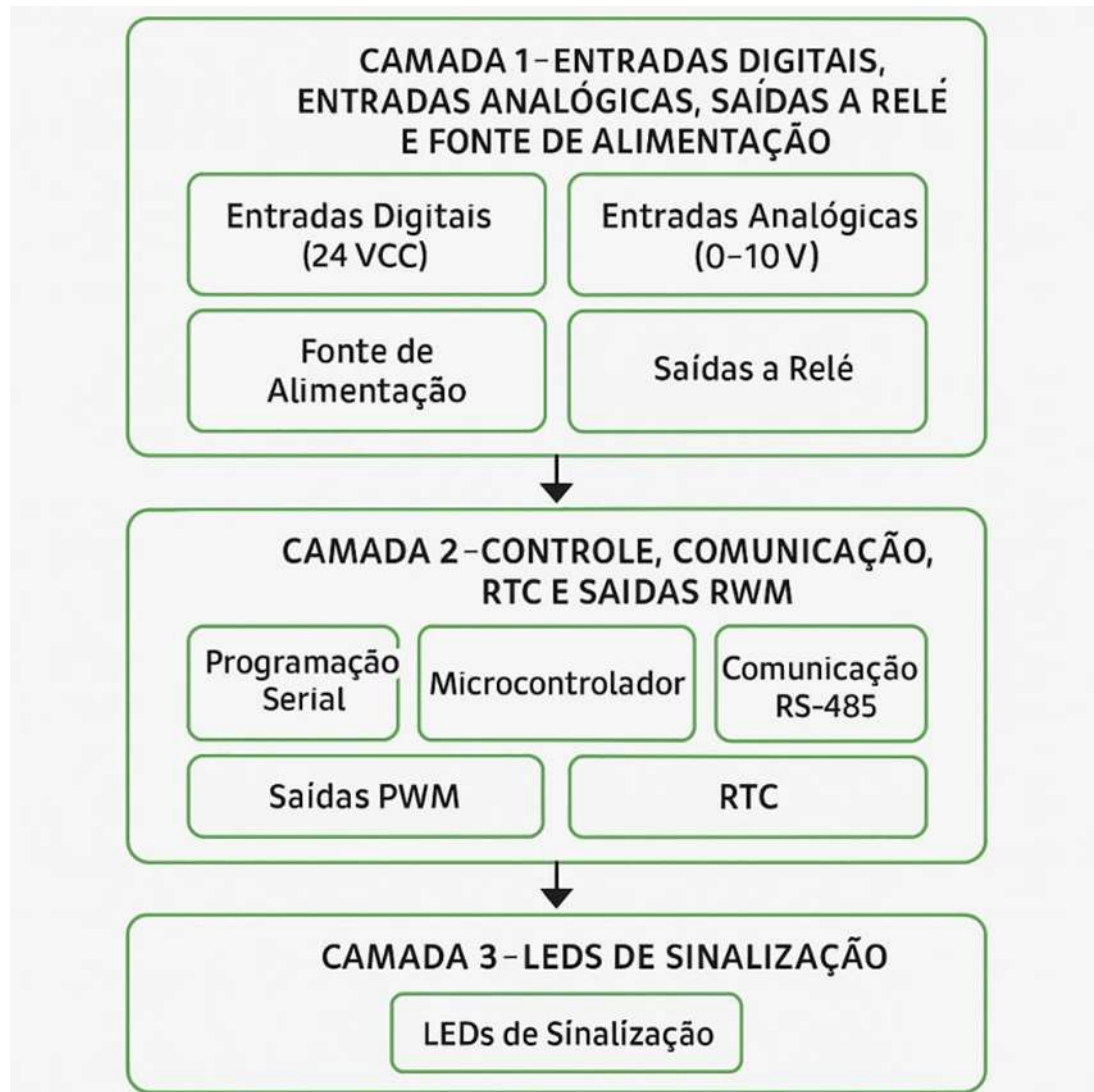


Figura 6 – Diagrama de blocos representando a organização modular em três camadas do protótipo.

Fonte: Autoria própria (2025).

Camada 1 – Entradas Digitais, Entradas Analógicas, Saídas a Relé e Fonte de Alimentação Esta camada concentra os blocos responsáveis pela aquisição e acionamento de sinais, além da conversão e regulação da alimentação elétrica. O esquema eletrônico correspondente é apresentado na Figura 7.

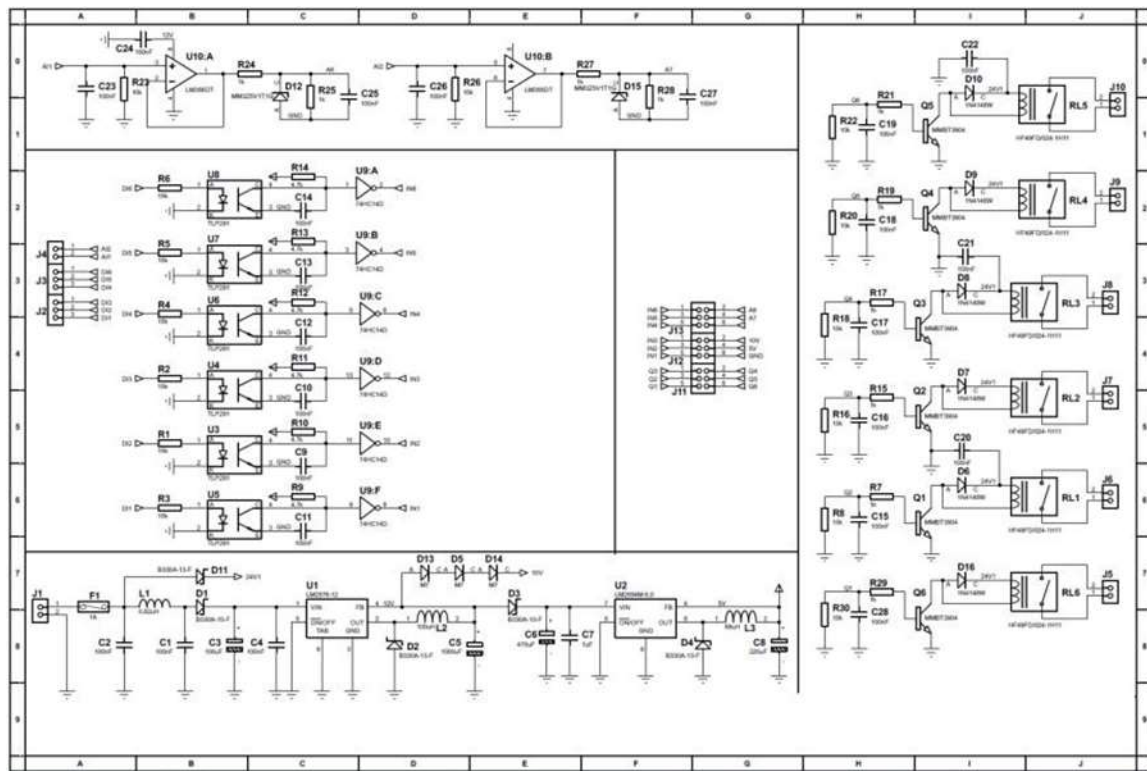


Figura 7 – Esquema eletrônico da Camada 1: Entradas Digitais, Entradas Analógicas, Saídas a Relé e Fonte de Alimentação. Fonte: Autoria própria (2025).

- **Entradas Digitais (24 VCC):** seis canais com proteção por optoacopladores (PC817), seguidos por inversores Schmitt Trigger (74HC14), garantindo isolamento galvânico e imunidade a ruídos.
- **Entradas Analógicas (0–10 V):** duas entradas com condicionamento por amplificadores operacionais LM358, divisores resistivos e proteção por diodo Zener de 5,1 V, permitindo adaptação ao microcontrolador.
- **Saídas a Relé:** seis relés compactos (HF46F), acionados por transistores NPN (BC817) e protegidos por diodos de flyback (1N4148).
- **Fonte de Alimentação:** entrada de 24 V regulada para 12 V (usada nas saídas PWM e entradas analógicas) e 5 V (para a lógica de controle), com reguladores step-down.

O layout da placa foi otimizado para ocupar o menor espaço possível, com dimensões finais de **74 mm × 63 mm**, respeitando normas como a IPC-2221 para separação de sinais de diferentes potências e tipos (COMMISSION, 1998).

Camada 2 – Controle, Comunicação, RTC e Saídas PWM Essa camada abriga os circuitos de controle lógico, temporização e comunicação. O microcontrolador é responsável por executar a lógica do

programa, gerenciar as entradas e saídas, e estabelecer comunicação com dispositivos externos. O circuito completo pode ser visualizado na Figura 8.

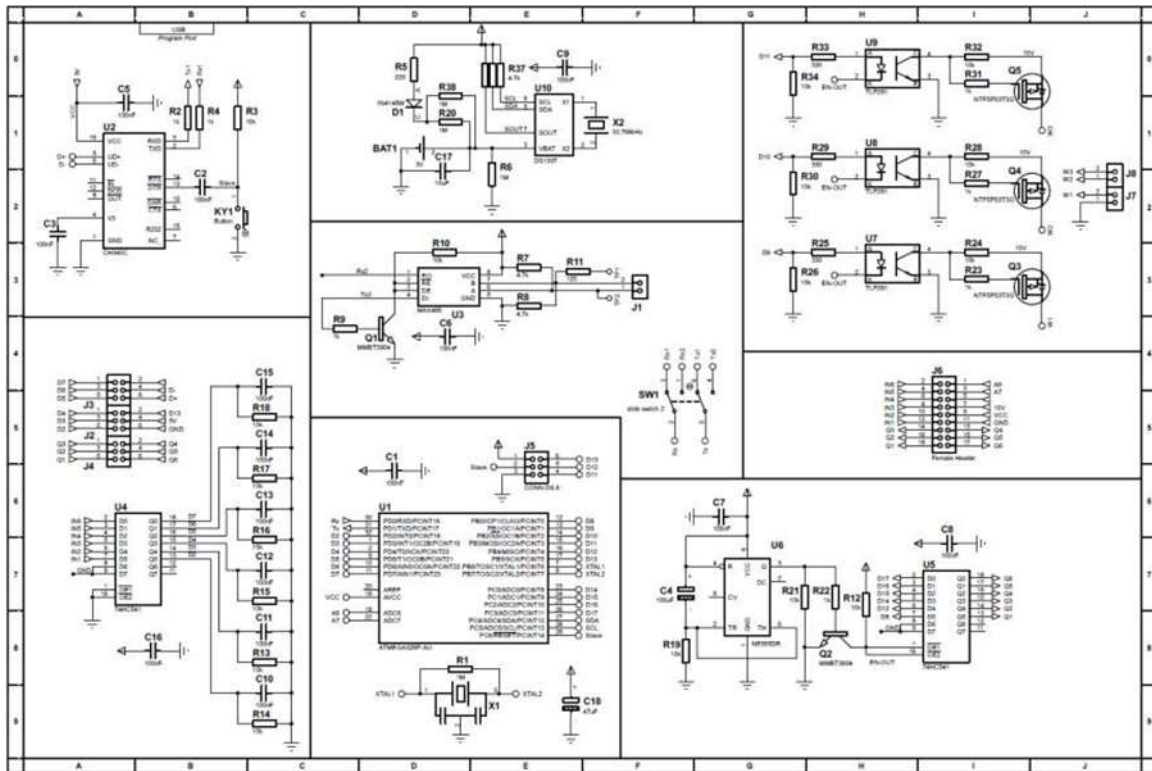


Figura 8 – Esquema eletrônico da Camada 2: Controle, Comunicação, RTC e Saídas PWM. Fonte: Autoria própria (2025).

- **Microcontrolador:** ATmega328P, compatível com a plataforma Arduino (INC., 2016).

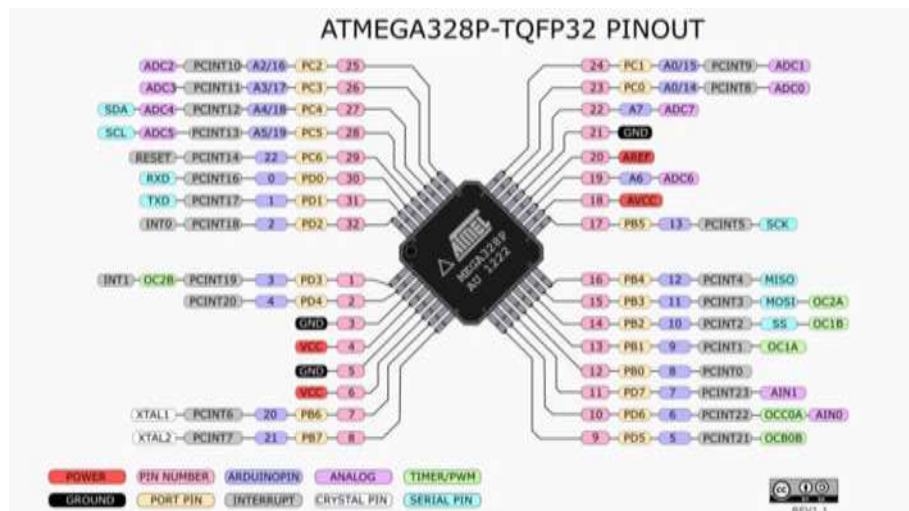


Figura 9 – Diagrama de pinagem (pinout) do microcontrolador ATmega328P, utilizado como núcleo de processamento do protótipo.

Fonte: Microchip Technology Inc. (2025).

- Interface de Programação: USB via conversor CH340C.
- Relógio de Tempo Real (RTC): DS1307 com bateria CR2032 (INTEGRATED, 2008).
- Comunicação Serial RS-485: implementada com driver MAX485, compatível com protocolo Modbus RTU.
- Saídas PWM: três canais isolados com optoacopladores e acionados por MOSFETs de canal P (IRF9540N).

Camada 3 – LEDs de Sinalização A terceira e última camada foi destinada a facilitar o diagnóstico e a visualização do funcionamento do sistema. Ela contém apenas indicadores visuais simples que espelham os estados lógicos das entradas e saídas digitais. Para isso, foram utilizados LEDs com resistores limitadores conectados em paralelo aos sinais de I/O.

Além disso, essa camada também abriga o conector USB Tipo-C, responsável pela comunicação serial com o computador. O esquema eletrônico correspondente é apresentado na Figura 10.

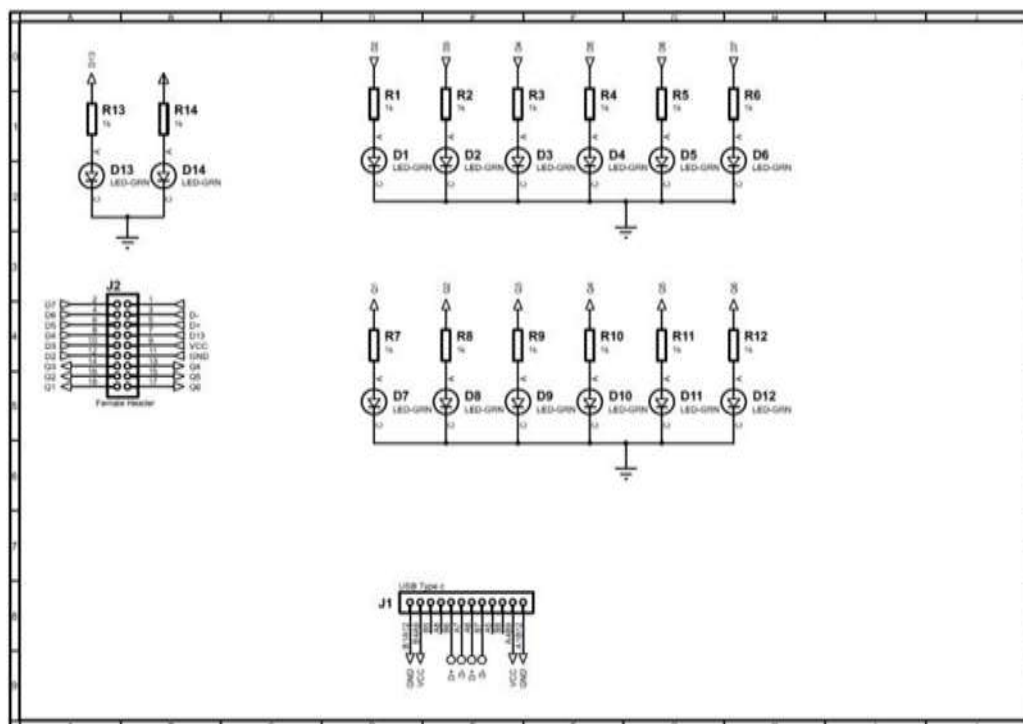


Figura 10 – Esquema eletrônico da Camada 3: LEDs de Sinalização e conector USB. Fonte: Autoria própria (2025).

- LEDs Indicadores: LEDs verdes para as entradas e vermelhos para as saídas, conectados em paralelo aos respectivos sinais.
- Conectores: nesta camada também está presente o conector USB Tipo-C, utilizado tanto para alimentação quanto para comunicação serial.

Essa organização em três camadas independentes otimizou o aproveitamento do espaço interno da carcaça e favoreceu a modularidade do sistema, contribuindo também para o caráter didático do projeto. A separação por soquetes facilita substituições futuras, manutenções e ampliações, tornando o protótipo robusto e flexível.

Além disso, a escolha por um layout modular em três camadas foi inspirada na arquitetura dos CLPs da linha LOGO! da Siemens, que adota a filosofia de separação funcional por blocos, promovendo maior organização, escalabilidade e clareza no projeto de sistemas embarcados voltados para automação (AG, 2025). Essa abordagem também reforça a compatibilidade visual e funcional com o ambiente industrial real, aproximando a experiência educacional dos padrões adotados em campo.

4.2.2.2 PROJETO DA PLACA DE CIRCUITO IMPRESSO (PCB)

Utilizando o software EasyEDA, foi realizada a conversão do esquema eletrônico para o layout físico da PCB (Printed Circuit Board). Esta etapa envolveu:

- Definição do empilhamento de camadas (2 camadas: sinal, terra, alimentação e controle);
- Posicionamento dos componentes conforme blocos funcionais;
- Roteamento das trilhas com largura adequada e espaçamento segundo a norma IPC-2221 (COMMISSION, 1998);
- Aplicação de barreiras de isolamento entre sinais de alta e baixa tensão.

A PCB foi projetada com dimensões finais de 74 mm × 63 mm. As Figuras 11, 12 e 13 ilustram, respectivamente, a camada de controle de energia, a camada de controle e a camada de indicação.

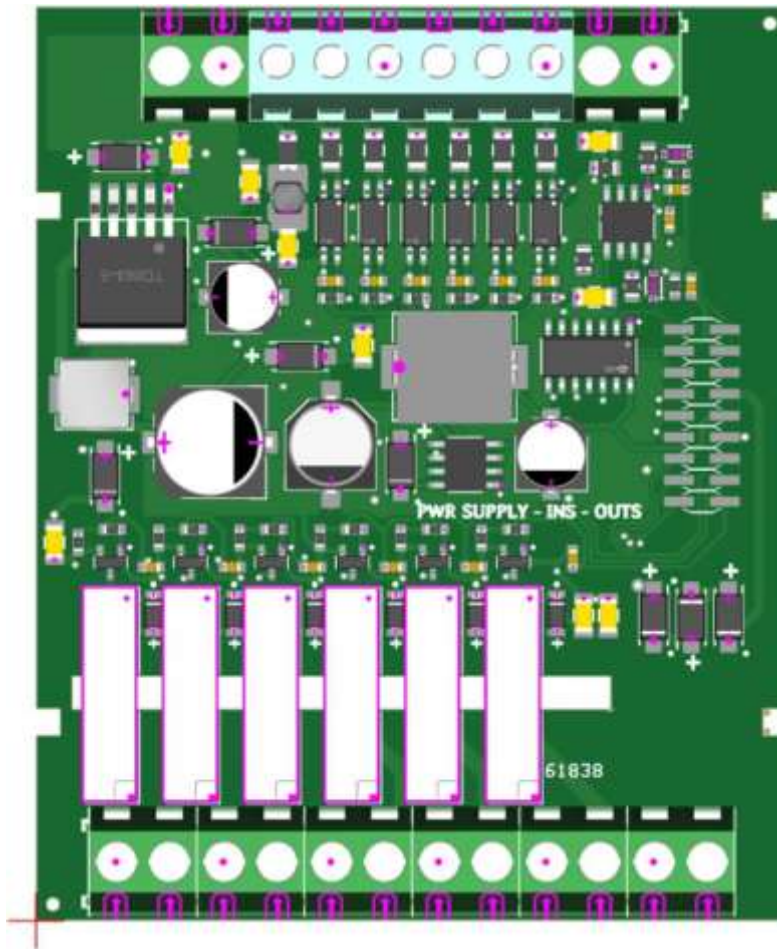


Figura 11 – Camada de energia. Fonte: Autoria própria (2025).

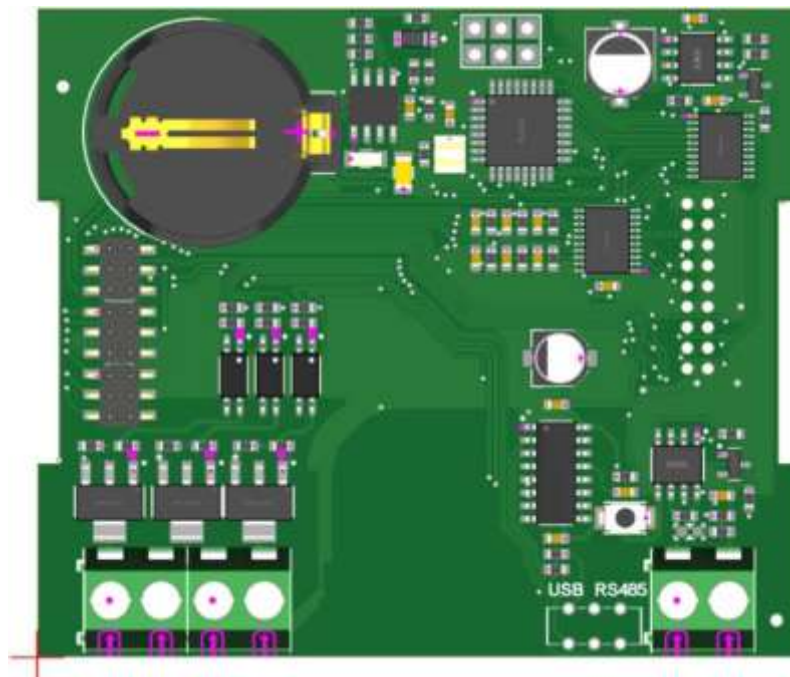


Figura 12 – Camada de controle Fonte: Autoria própria (2025).

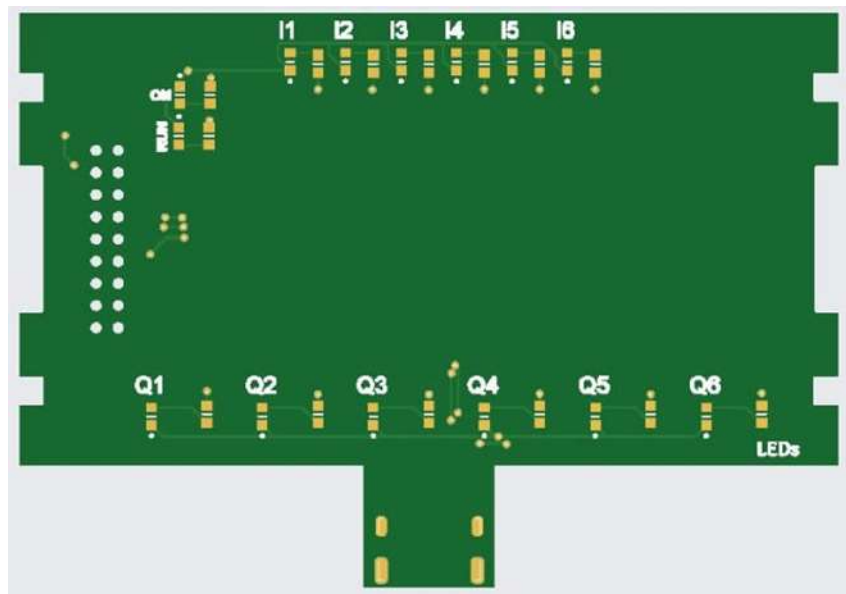


Figura 13 – Camada de indicação Fonte: Autoria própria (2025).

4.2.2.2.3 SELEÇÃO DOS COMPONENTES

Com o circuito definido, procedeu-se à seleção dos componentes eletrônicos com base em critérios de:

- Disponibilidade no mercado internacional, especialmente no catálogo da JLCPCB;
- Custo-benefício para produção em escala didática;
- Confiabilidade técnica, considerando tolerância, tensão de operação, potência dissipada e encapsulamento.

Foram priorizados componentes com datasheets acessíveis, montagem SMT sempre que possível e compatibilidade com soldagem automatizada.

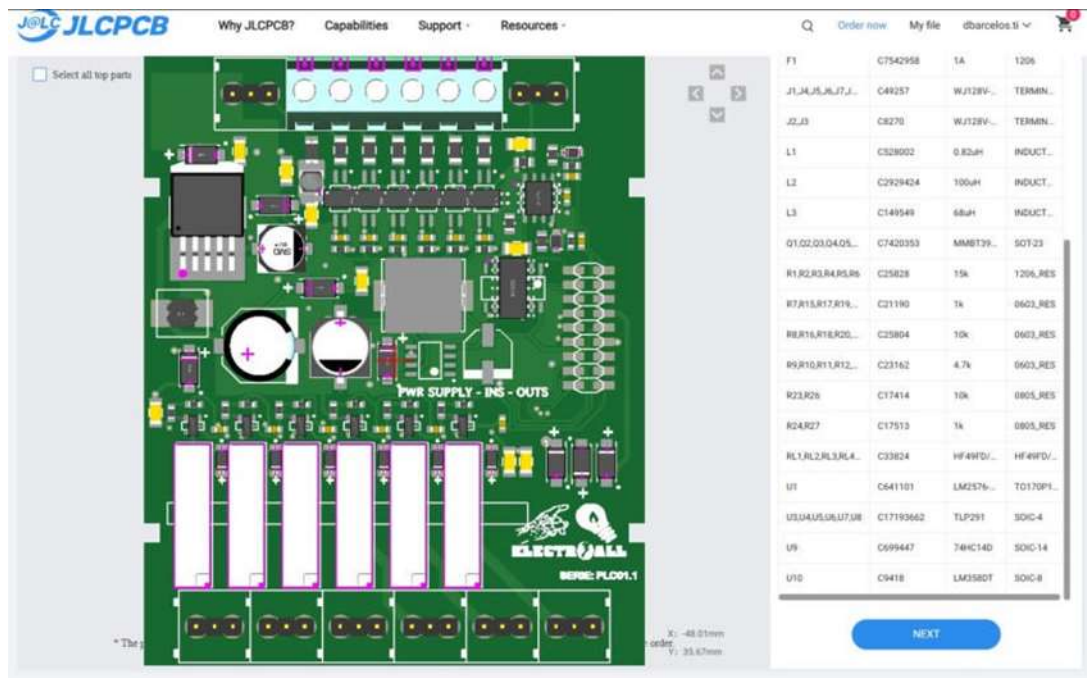


Figura 14 – Interface de seleção de componentes no catálogo SMT da JLCPCB.

Fonte: Autoria própria (2025).

4.2.2.2.4 PRODUÇÃO DA PCB

O arquivo Gerber gerado no software de projeto foi enviado para fabricação na empresa JLCPCB, especializada na produção de placas com montagem SMT e PTH. Foram solicitadas:

- PCB de 2 camadas, com espessura de 1,6 mm;
- Espessura de cobre de 2 oz;
- Máscara de solda preta e silk screen branco;
- Montagem automática de componentes SMT (SMT Assembly Service).

4.2.2.2.5 DESENVOLVIMENTO DA CARCAÇA 3D

Para proteger e organizar o conjunto eletrônico, foi projetado um gabinete em 3D por meio de software de modelagem (como Fusion 360 ou FreeCAD). O projeto considerou:

- Compatibilidade com as dimensões e furos de fixação da PCB;
- Aberturas para bornes, conectores USB, RS485 e LEDs indicadores;
- Fixação com encaixes ou parafusos M3.

A Figura 15 apresenta a visualização tridimensional da carcaça montada com as três placas eletrônicas internas conectadas. Essa renderização oferece uma visão geral da integração do conjunto, destacando a organização interna e a adaptação estrutural do gabinete 3D ao projeto eletrônico.

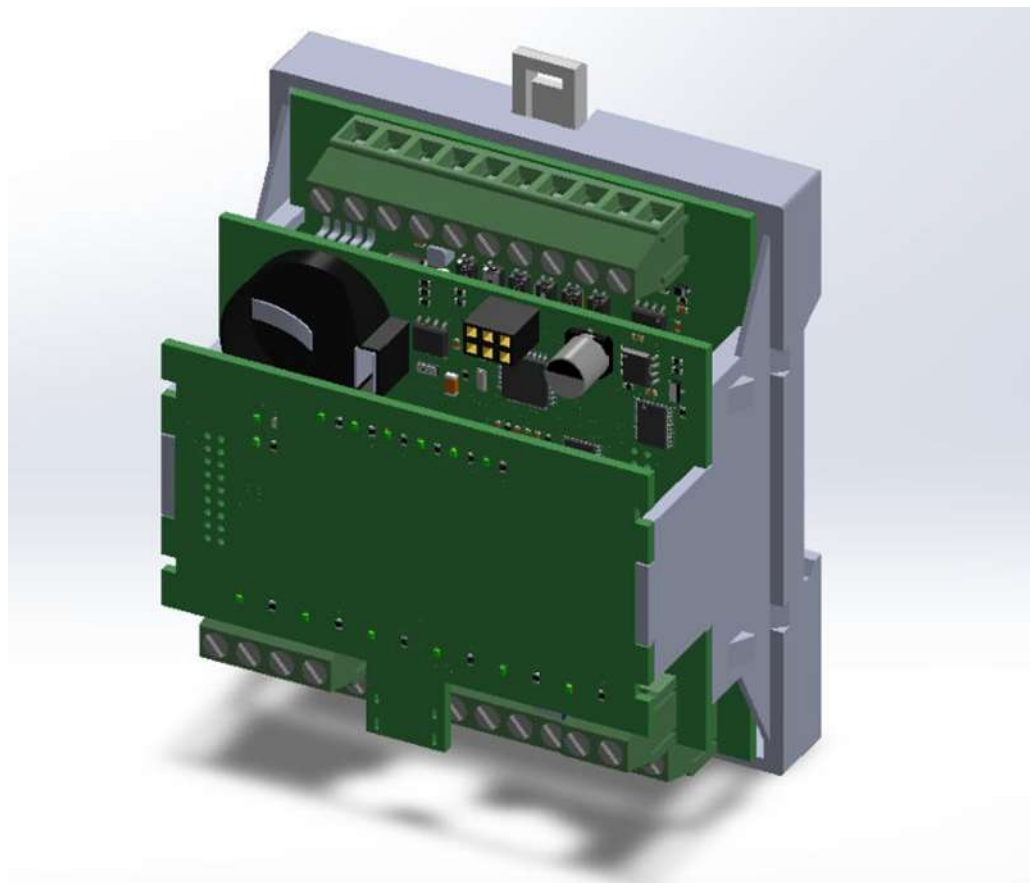


Figura 15 – Renderização 3D da carcaça com as três camadas de PCB instaladas

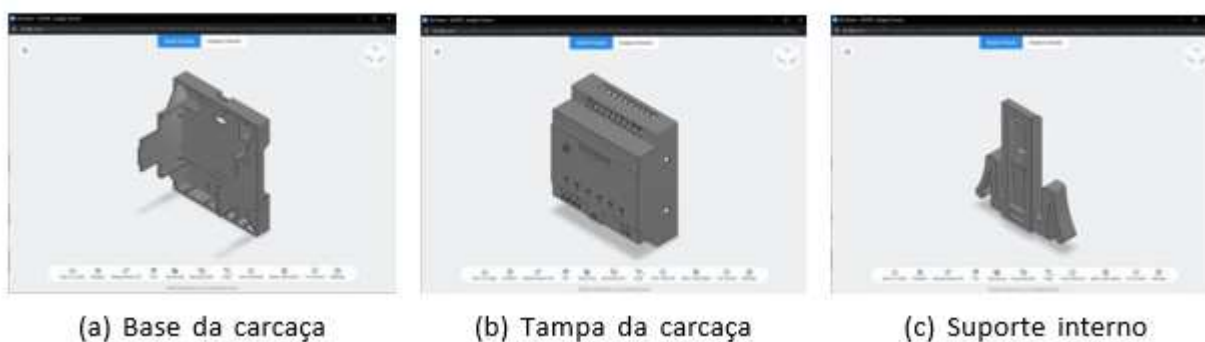


Figura 16 – Partes da carcaça 3D projetada para o protótipo: (a) base, (b) tampa e (c) suporte interno.

A carcaça foi encomendada via serviço de manufatura 3D da JLCPCB, utilizando material ABS industrial de alta resistência.

4.2.3 MONTAGEM FINAL E VERIFICAÇÃO

Após a chegada da PCB montada e da carcaça, foi realizada a montagem final do protótipo:

- Instalação da placa na carcaça com espaçadores de nylon;
- Teste individual de entradas e saídas;
- Verificação da comunicação USB e RS485;
- Instalação do RTC e bateria;
- Conferência de LEDs e alimentação geral.

Essa etapa assegurou que o protótipo estivesse operacional e de acordo com o projeto teórico.



(a) Camada de energia

Fonte: Autoria própria (2025)



(b) Camada de controle

Fonte: Autoria própria (2025)



(c) Camada de indicação

Fonte: Autoria própria (2025)

Figura 17 – Camadas do protótipo: energia, controle e indicação.

Após a verificação individual das placas, iniciou-se a etapa de montagem das três camadas do protótipo na carcaça. Esse processo envolveu a fixação da base, a inserção das placas internas (energia, controle e indicação) e, por fim, o ajuste final com a tampa do gabinete. As Figuras a seguir ilustram as principais etapas dessa montagem.

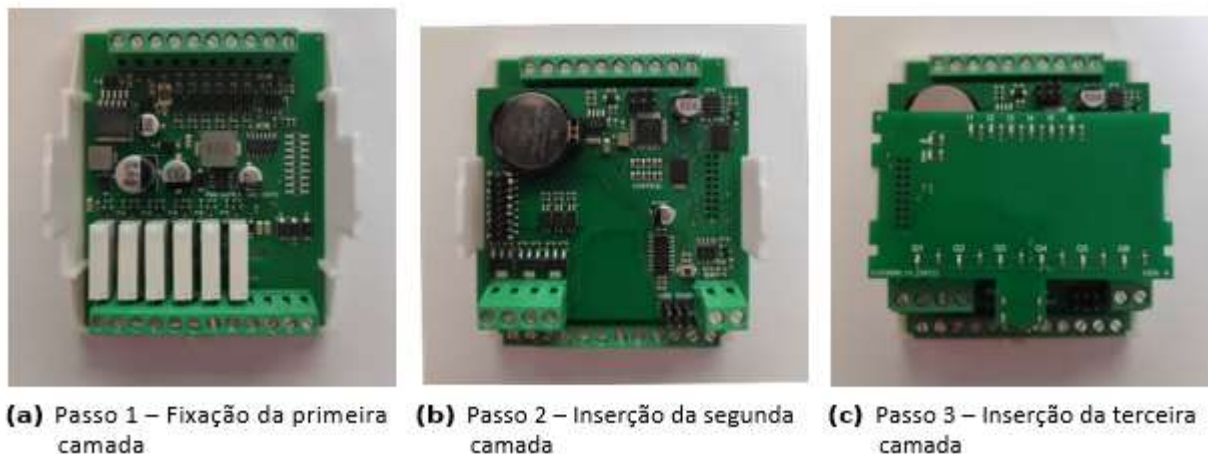


Figura 18 – Etapas de montagem do protótipo na carcaça.

Concluída a montagem, o protótipo foi fixado em sua carcaça final. As imagens a seguir mostram a visão lateral, frontal e traseira do dispositivo já montado, destacando o trilho e a trava utilizados para fixação em padrão industrial.



Figura 19 – Protótipo final montado em carcaça, com fixação em trilho padrão industrial.

4.2.4 GRAVAÇÃO DO BOOTLOADER NO ATMEGA328P

Para que o microcontrolador ATmega328P do protótipo possa ser programado via interface USB, é necessário realizar previamente a gravação do bootloader. Esse pequeno firmware, presente na maioria das placas Arduino comerciais, é responsável por permitir que o microcontrolador aceite códigos carregados via porta serial (USB), sem a necessidade de utilizar um gravador externo a cada reprogramação.

Na versão utilizada neste projeto, o ATmega328P é fornecido sem o bootloader gravado. Portanto, essa etapa é fundamental para que o protótipo possa ser programado posteriormente utilizando apenas um cabo USB Tipo-C, através do conversor CH340C integrado à placa.

Montagem física e ligações ISP

Antes da gravação propriamente dita, é necessário montar fisicamente o circuito de gravação ISP (In-System Programming) entre um Arduino UNO (utilizado como gravador) e o microcontrolador ATmega328P.

As conexões são feitas utilizando os seguintes pinos, conforme demonstrado na Tabela 4.

Tabela 4 – Mapeamento de conexões entre Arduino Nano e o conector ISP (J5) do protótipo

Função ISP	Pino Arduino Nano	Pino ISP no Protótipo (J5)
VCC (Alimentação)	5V	Pin 1
RESET (Slave Reset)	D10	Pin 2
GND (Terra)	GND	Pin 3
MOSI (Master Out, Slave In)	D11	Pin 4
MISO (Master In, Slave Out)	D12	Pin 5
SCK (Serial Clock)	D13	Pin 6

A Figura 20 mostra a montagem realizada com fios jumper entre o Arduino UNO e o conector ISP da placa do protótipo.

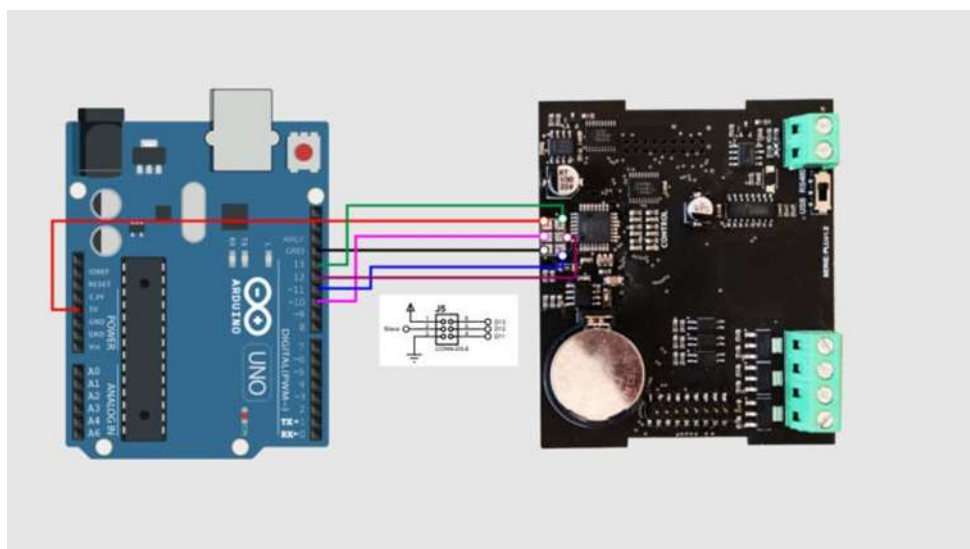


Figura 20 – Montagem física da gravação ISP entre Arduino UNO e o protótipo PLC

Importante: essa etapa é necessária apenas uma vez por microcontrolador — normalmente logo após a montagem ou substituição do chip. Uma vez gravado, o bootloader permanece armazenado no microcontrolador.

Procedimento de gravação do bootloader com Arduino UNO como gravador ISP

Após a montagem física do circuito, a gravação do bootloader pode ser feita diretamente pela IDE do Arduino, seguindo os passos abaixo:

1. Abrir o exemplo ArduinoISP, localizado em Arquivo > Exemplos > 11.ArduinoISP > ArduinoISP.

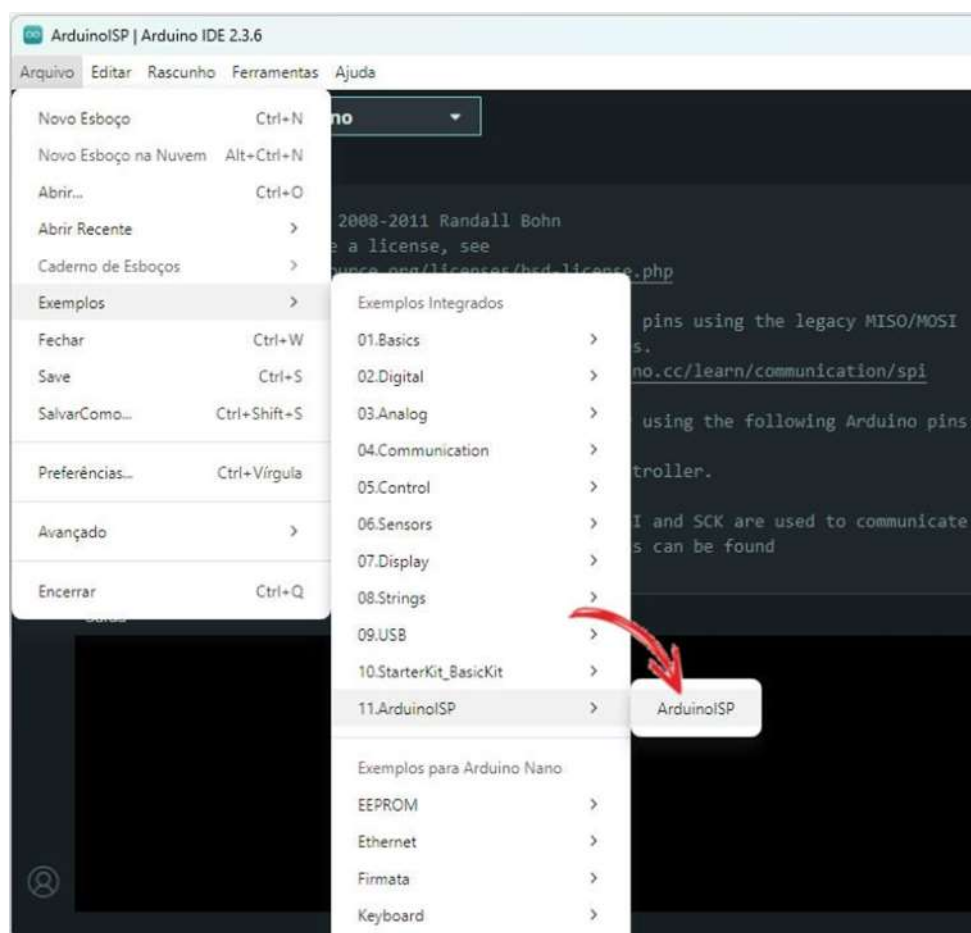


Figura 21 – Etapa 1 – Abertura do exemplo ArduinoISP na IDE do Arduino

2. Selecionar a porta COM do Arduino programador (UNO ou MEGA).

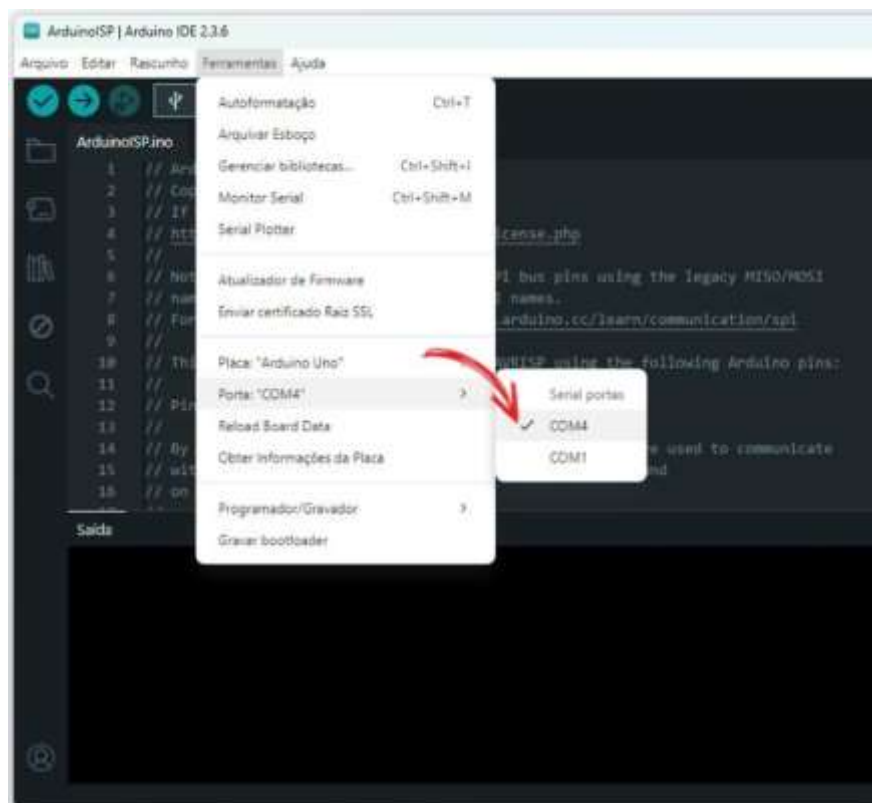


Figura 22 – Etapa 2 – Seleção da porta COM do Arduino UNO

3. Selecionar a placa “Arduino UNO” no menu Ferramentas > Placa.

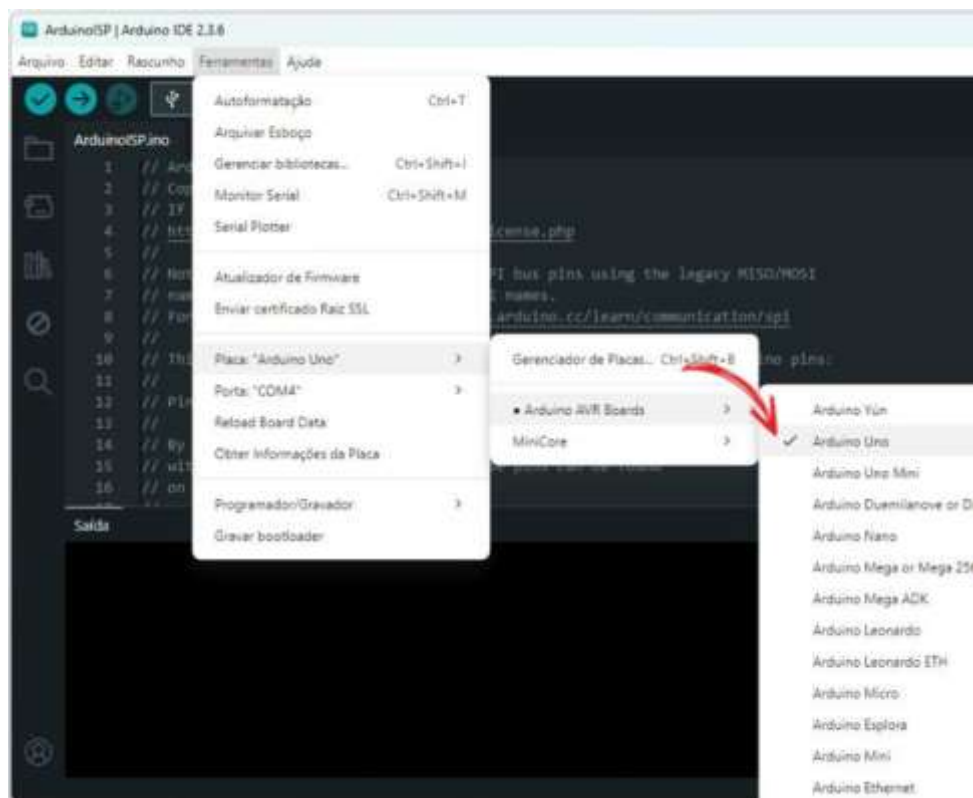


Figura 23 – Etapa 3 – Seleção da placa “Arduino UNO”

4. Carregar o código ArduinoISP no Arduino UNO.

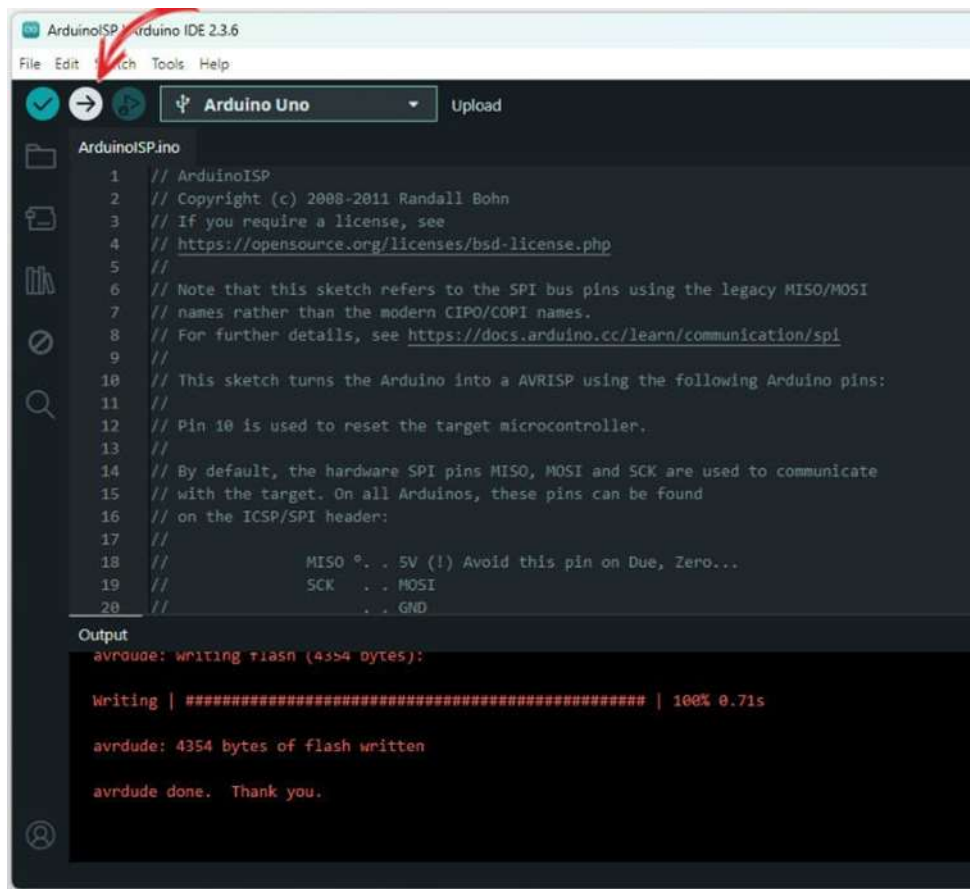


Figura 24 – Etapa 4 – Upload do código “ArduinoISP” para o Arduino UNO

5. Selecionar a porta COM correspondente ao protótipo (Nano/ATmega328P).

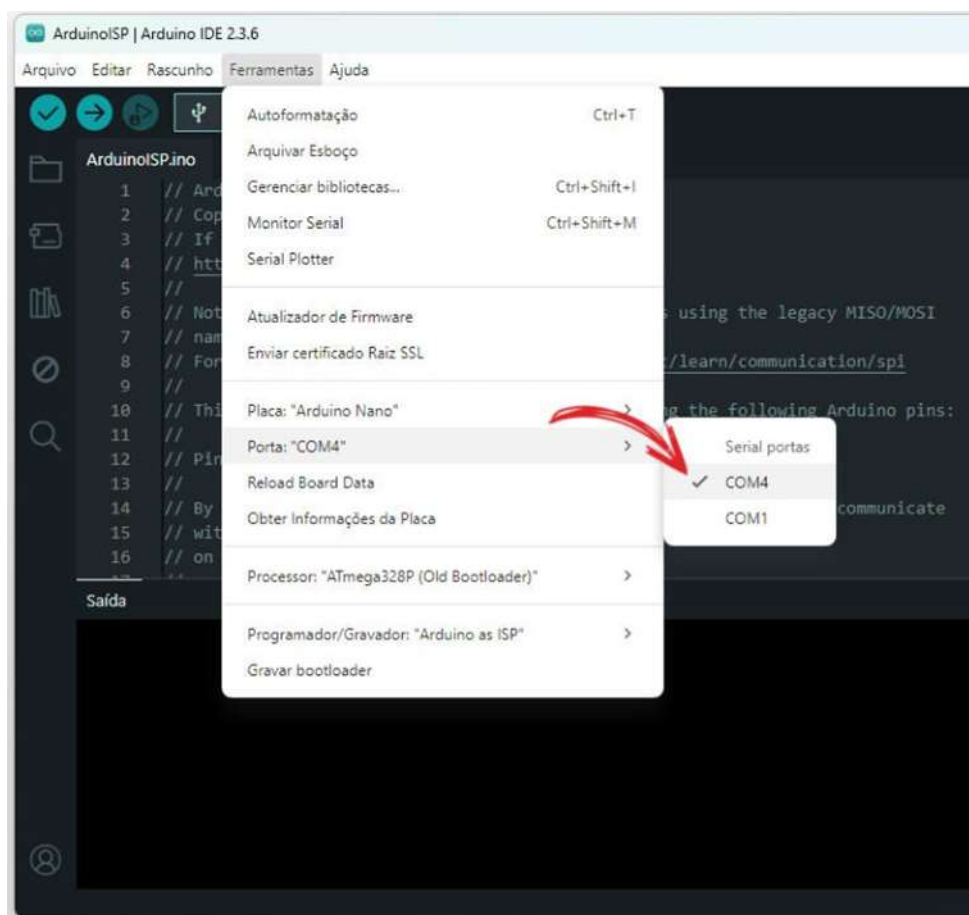


Figura 25 – Etapa 5 – Seleção da porta COM do protótipo (Nano)

6. Selecionar a placa “Arduino Nano”, utilizada no protótipo.

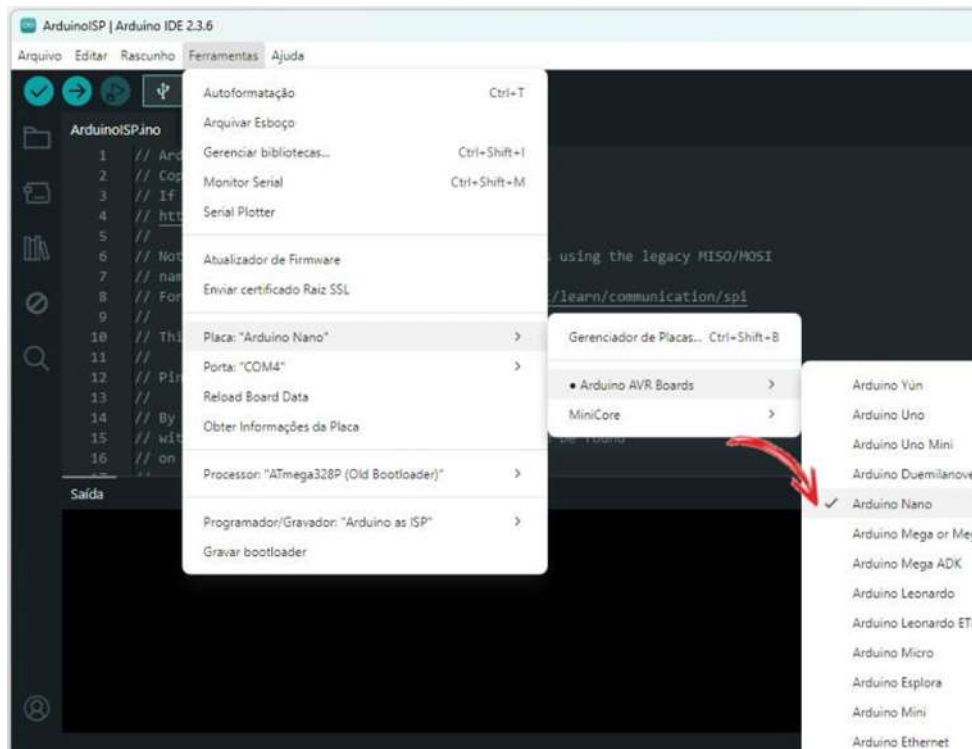


Figura 26 – Etapa 6 – Seleção da placa “Arduino Nano”

7. Selecionar o processador “ATmega328P (Old Bootloader)”.

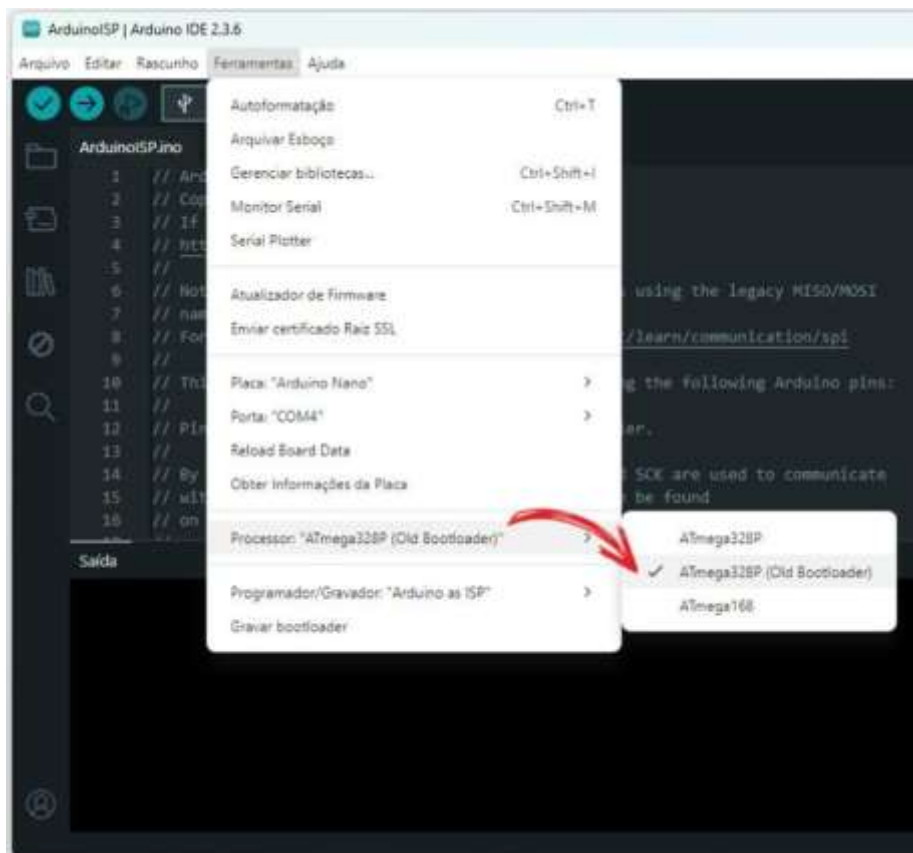


Figura 27 – Etapa 7 – Seleção do processador “ATmega328P (Old Bootloader)”

8. Selecionar o programador “Arduino as ISP”.

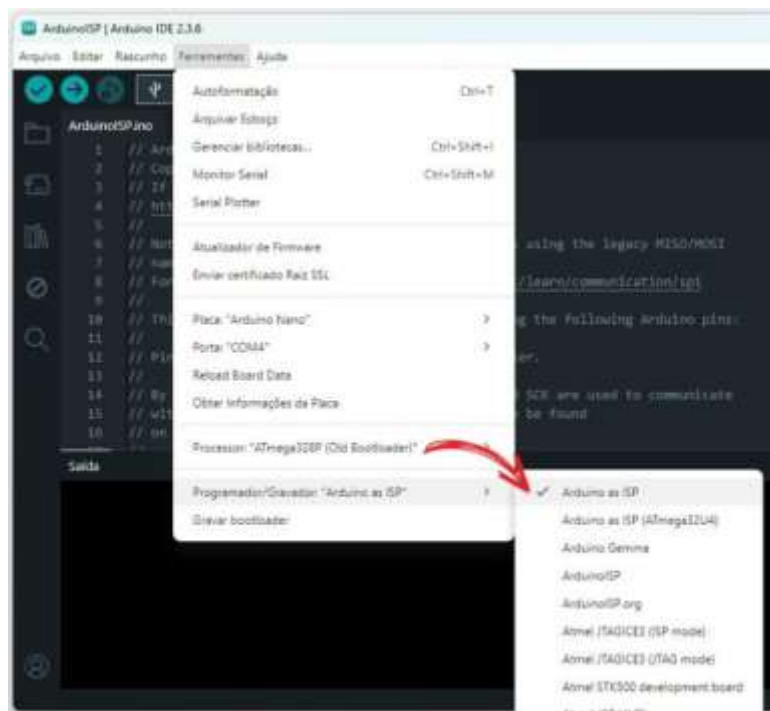
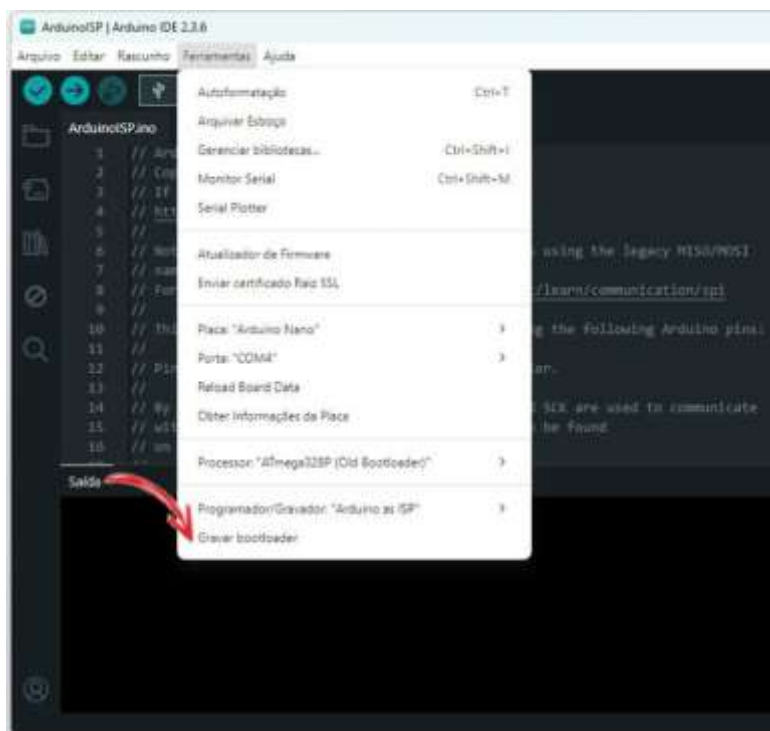


Figura 28 – Etapa 8 – Seleção do programador “Arduino as ISP”

9. Clicar em Ferramentas > Gravar Bootloader para iniciar a gravação.



Uma vez concluída essa etapa, o microcontrolador está pronto para ser programado normalmente pela IDE do Arduino ou AVRDUDESS, utilizando a interface USB Tipo-C do protótipo.

4.3 IMPLEMENTAÇÃO DO SOFTWARE

Durante a fase inicial de planejamento deste trabalho, previa-se o desenvolvimento de um software autoral com interface gráfica para construção de diagramas Ladder, com posterior conversão automática para código em linguagem C voltado ao microcontrolador ATmega328P. No entanto, ao longo das atividades de pesquisa e análise técnica, identificou-se a existência de soluções consolidadas e amplamente utilizadas na comunidade, como o LDmicro e o OpenPLC, que já oferecem suporte robusto à programação em Ladder com geração de código em C.

Diante disso, e considerando as limitações de tempo e escopo, optou-se por direcionar o foco do projeto para a construção, teste e validação do protótipo físico de CLP, integrando-o com essas ferramentas existentes. Essa decisão estratégica possibilitou maior aprofundamento na etapa de hardware e garantiu a viabilidade do projeto dentro do cronograma acadêmico, sem comprometer os objetivos propostos. Ressalta-se que o protótipo foi projetado com plena compatibilidade para uso com essas ferramentas, assegurando a coerência com o título do trabalho.

A seguir, são descritas as duas abordagens adotadas na implementação de software: a primeira utilizando a plataforma Arduino IDE com linguagem C e a segunda com o software LDmicro baseado em programação Ladder.

4.3.1 PROGRAMAÇÃO EM C COM ARDUINO IDE

Para validar a integração entre hardware e software, foi realizada uma abordagem prática utilizando a plataforma Arduino IDE. Inicialmente, mapeou-se os pinos do microcontrolador ATmega328P com as funções de entrada e saída do protótipo, conforme apresentado na Tabela 5.

Tabela 5 – Mapeamento de pinos entre o PLC e o ATmega328P

Entrada Digital	Pino	Entrada Analógica	Pino	Saída Relé	Pino
I1	D2	AI0	A6	Q1	D8
I2	D3	AI1	A7	Q2	D12
I3	D4	Saída PWM		Q3	D14
I4	D5	PWM W1	D9	Q4	D15
I5	D6	PWM W2	D10	Q5	D16
I6	D7	PWM W3	D11	Q6	D17

Além do mapeamento, a Tabela 6 apresenta um resumo da quantidade de portas disponíveis no protótipo desenvolvido:

Tabela 6 – Número de portas disponíveis no protótipo

Tipo de porta	Quantidade
Entradas digitais (12–24V)	6
Entradas analógicas (0–10V)	2
Saídas digitais via relé	6
Saídas PWM (0–10V)	3

Leitura Analógica e Controle PWM

A primeira abordagem consistiu na criação de um sketch em linguagem C/C++ na Arduino IDE, com o objetivo de testar a leitura de sinais analógicos e o acionamento de saídas PWM utilizando o microcontrolador ATmega328P. O código implementado está apresentado no Código 4.1.

```

/*
Arquivo: controle_pwm_analogico.ino
Autor: Danilo Silva

4   Descrição:
5   Este código realiza a leitura de duas entradas analógicas (A6 e A7)
6   e controla três saídas PWM (pinos 9, 10 e 11).
7   A entrada A6 controla o PWM no pino 9.
8   A entrada A7 controla os PWMs nos pinos 10 e 11 simultaneamente.
9   Ideal para controle proporcional de motores, LEDs, válvulas, etc.
10  */
11
12  // Mapeamento das saídas PWM
13  const int pwm[] = {9, 10, 11};
14
15  // Mapeamento das entradas analógicas (convertidas de 0-10V para 0-5V via
16  hardware)
17  const int analogIn[] = {A6, A7};
18
19  void setup() {
20    // Configura os pinos PWM como saídas
21    for (int i = 0; i < 3; i++) {
22      pinMode(pwm[i], OUTPUT);
23    }
24  }
25
26  void loop() {
27    // Leitura da entrada analógica A6 e mapeamento para faixa PWM (0-255)
28    int pwmValue1 = map(analogRead(analogIn[0]), 0, 1023, 0, 255);
29    analogWrite(pwm[0], pwmValue1); // Saída PWM1 (D9)
30
31    // Leitura da entrada analógica A7 e mapeamento
32    int pwmValue2 = map(analogRead(analogIn[1]), 0, 1023, 0, 255);
33    analogWrite(pwm[1], pwmValue2); // Saída PWM2 (D10)
34    analogWrite(pwm[2], pwmValue2); // Saída PWM3 (D11) recebe o mesmo
35    valor de A7
36  }

```

LISTING 4.1 – ENTRADAS ANALÓGICAS E SAÍDAS PWM COM ARDUINO

O protótipo desenvolvido possui duas entradas analógicas (AI0 e AI1), que recebem sinais variando de 0 a 10V. No entanto, o conversor analógico-digital (ADC) interno do microcontrolador ATmega328P opera com uma faixa de entrada de 0 a 5V, sendo necessário realizar uma conversão de nível de tensão via hardware. Para isso, utilizou-se um arranjo de resistores em divisor de tensão, ajustando o sinal analógico para uma faixa segura de 0 a 5V. Adicionalmente, um diodo zener de 5,1V foi utilizado para proteger a entrada contra sobretensões.

O ADC do ATmega328P possui resolução de 10 bits, o que significa que ele converte a tensão de entrada em um valor digital entre 0 e 1023. Dessa forma, a tensão convertida de 0–5V corresponde numericamente a essa faixa de valores. No programa, essa leitura é convertida para uma escala de 0 a 255 utilizando a função `map()`, que é compatível com o comando `analogWrite()`, responsável pela geração do sinal PWM nos pinos de saída digitais (D9, D10 e D11).

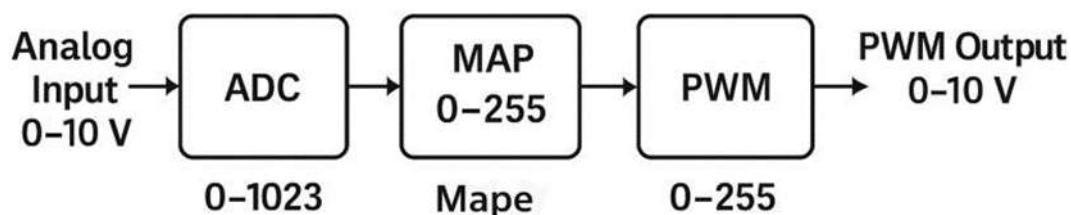


Figura 30 – Diagrama de blocos – Conversão de entrada analógica (0–10V) em saída PWM (0–10V)

O sinal PWM gerado apresenta uma largura de pulso proporcional à entrada analógica lida. Por exemplo, uma tensão de 10V aplicada à entrada analógica AI0 (convertida para 5V após o divisor resistivo) será convertida para o valor digital máximo (1023), o qual é mapeado para 255, resultando em um sinal PWM com 100% de ciclo de trabalho. Esse sinal é enviado à saída PWM (0–10V), que também possui condicionamento eletrônico para garantir compatibilidade com cargas externas, como atuadores e conversores de frequência.

Essa técnica garante a replicação analógica proporcional entre entrada e saída, permitindo que o CLP funcione como um intermediador em processos que exigem controle proporcional de sinais.

Leitura Digital e Controle de Relés

Neste código, a leitura dos sinais digitais ocorre nas portas D2 a D7, correspondentes às entradas I1 a I6. Os sinais de entrada são refletidos diretamente nas saídas D8, D12, D14 a D17, correspondentes às saídas Q1 a Q6 que controlam os relés responsáveis pelo acionamento de cargas.

O protótipo foi projetado para operar com até seis entradas digitais de 12 a 24VDC, compatíveis com os níveis de tensão normalmente utilizados em sensores industriais, botoeiras e outros dispositivos de campo. Cada entrada é opto-isolada por meio de um circuito com optoacopladores do tipo TLP291, garantindo isolamento elétrico entre os sinais externos e a lógica interna do microcontrolador, protegendo o sistema contra surtos de tensão e ruídos provenientes do ambiente industrial.

Internamente, os sinais de entrada passam por um estágio inversor com transistor, que também atua como filtro de sinal, assegurando um acionamento limpo e confiável no domínio lógico. Cada uma das seis entradas digitais está mapeada para um pino de entrada digital do ATmega328P (D2 a D7), conforme a Tabela 5.

As saídas digitais do protótipo são implementadas com seis relés miniatura de estado sólido, com contatos normalmente abertos (NA). Esses relés são acionados por transistores NPN, controlados diretamente pelo microcontrolador. Cada relé pode comutar correntes de até 5A em 250VAC ou 5A em 30VDC, o que torna o sistema apto para controlar cargas como lâmpadas, válvulas solenóides, contadores auxiliares e pequenos motores monofásicos.

Para garantir proteção e durabilidade dos contatos, foi adicionado um diodo de roda livre nos circuitos de acionamento, absorvendo a corrente induzida durante o desligamento de cargas indutivas.

O funcionamento do sistema segue uma lógica direta de espelhamento: ao detectar nível alto (lógico 1) em uma entrada digital, a saída correspondente é ativada. O mapeamento entre entradas (I1–I6) e saídas (Q1–Q6) é feito de forma fixa e direta, como evidenciado no Código 4.2.

```

1  /*
2   Arquivo: controle_digital_rele.ino
3   Autor: Danilo Silva
4   Descrição:
5   Este código realiza a leitura de 6 entradas digitais (D2 a D7) e controla
6   diretamente 6 saídas digitais (D8, D12, D14 a D17) conectadas a relés.
7   Cada entrada ativa diretamente uma saída correspondente.
8   Aplicação típica: acionamento de cargas industriais com ligação direta.
9  */
10
11 // Entradas digitais
12 const int inputs [] = {2, 3, 4, 5, 6, 7};
13
14 // Saídas digitais (relés)
15 const int outputs [] = {8, 12, 14, 15, 16, 17};
16
17 void setup() {
18     // Configura todas as entradas como INPUT
19     for (int i = 0; i < 6; i++) {
20         pinMode(inputs[i], INPUT);
21     }
22
23     // Configura todas as saídas como OUTPUT e garante que iniciem desligadas
24     for (int i = 0; i < 6; i++) {
25         pinMode(outputs[i], OUTPUT);
26         digitalWrite(outputs[i], LOW);
27     }
28 }
29
30 void loop() {
31     // Lê cada entrada e reflete seu estado na saída correspondente
32
33     for (int i = 0; i < 6; i++) {
34         digitalWrite(outputs[i], digitalRead(inputs[i]));
35     }
36 }

```

LISTING 4.2 – ENTRADAS DIGITAIS E ACIONAMENTO DE RELÉS

Programação do Microcontrolador

A gravação do firmware pode ser realizada de duas formas:

- Via porta USB do protótipo, utilizando o conversor USB-Serial CH340C e a própria interface da Arduino IDE;
- Via interface ISP (In-System Programming), utilizando outro Arduino como gravador e realizando a gravação direta no chip ATmega328P.

Essa abordagem permite testar a funcionalidade das interfaces de entrada e saída analógicas e digitais, bem como validar a compatibilidade entre o hardware do protótipo e a programação por meio de linguagens padrão no contexto de automação industrial didática.

4.3.2 PROGRAMAÇÃO EM LADDER COM LDMICRO

Como segunda abordagem de programação do protótipo, optou-se pela utilização do LDmicro, um software de código aberto amplamente utilizado em contextos acadêmicos e industriais para o desenvolvimento de lógicas Ladder compatíveis com microcontroladores. O LDmicro oferece uma interface gráfica intuitiva e permite gerar código em linguagem C diretamente compatível com a plataforma Arduino, facilitando a implementação em arquiteturas baseadas no ATmega328P.

Essa etapa teve como objetivo validar a compatibilidade do protótipo com ferramentas típicas da automação industrial, demonstrando que o hardware desenvolvido pode ser utilizado com linguagens gráficas padrão IEC 61131-3, como o diagrama Ladder.

No LDmicro, foi elaborada uma lógica Ladder simples, representando o controle direto dos relés a partir das entradas digitais do protótipo — funcionalidade equivalente ao comportamento já implementado anteriormente em linguagem C (ver Código 4.2). Essa lógica simula uma automação discreta clássica, baseada em contatos, amplamente utilizada na indústria para sistemas de intertravamento, comandos manuais e acionamentos simples.

A Figura 31 apresenta a tela principal do LDmicro com a lógica Ladder desenvolvida para o protótipo, utilizando elementos gráficos familiares aos profissionais da área.

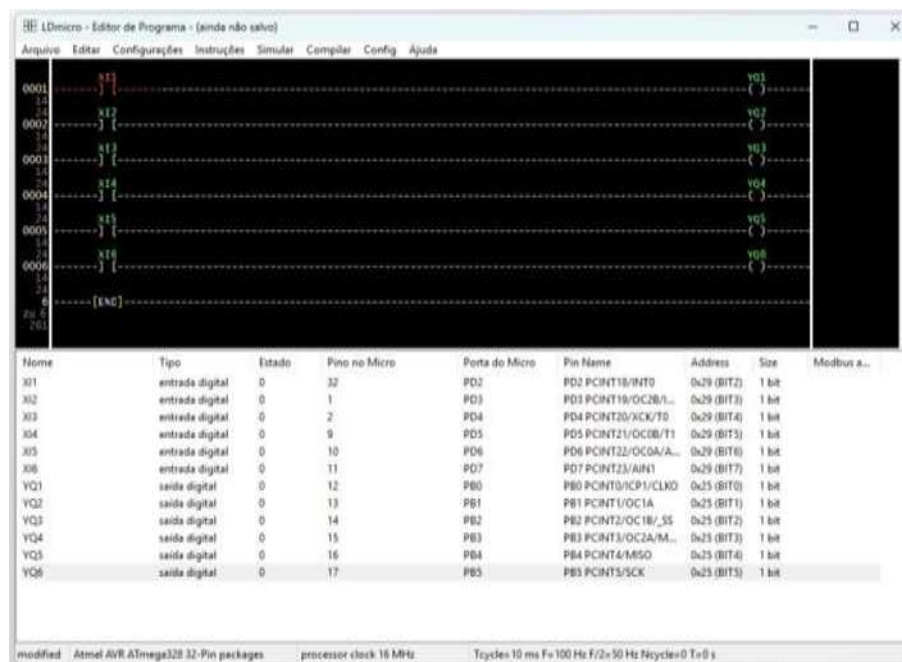


Figura 31 – Interface do LDmicro com lógica Ladder desenvolvida

Antes da exportação do programa, é necessário selecionar corretamente o micro- controlador alvo no LDmicro. Essa escolha define a pinagem, os registradores e o modelo de compilação. No caso deste projeto, foi selecionado o microcontrolador **ATmega328P (32-Pin packages)**, compatível com a arquitetura Arduino Nano utilizada no protótipo. A Figura 32 mostra a interface do software no momento da seleção do dispositivo:

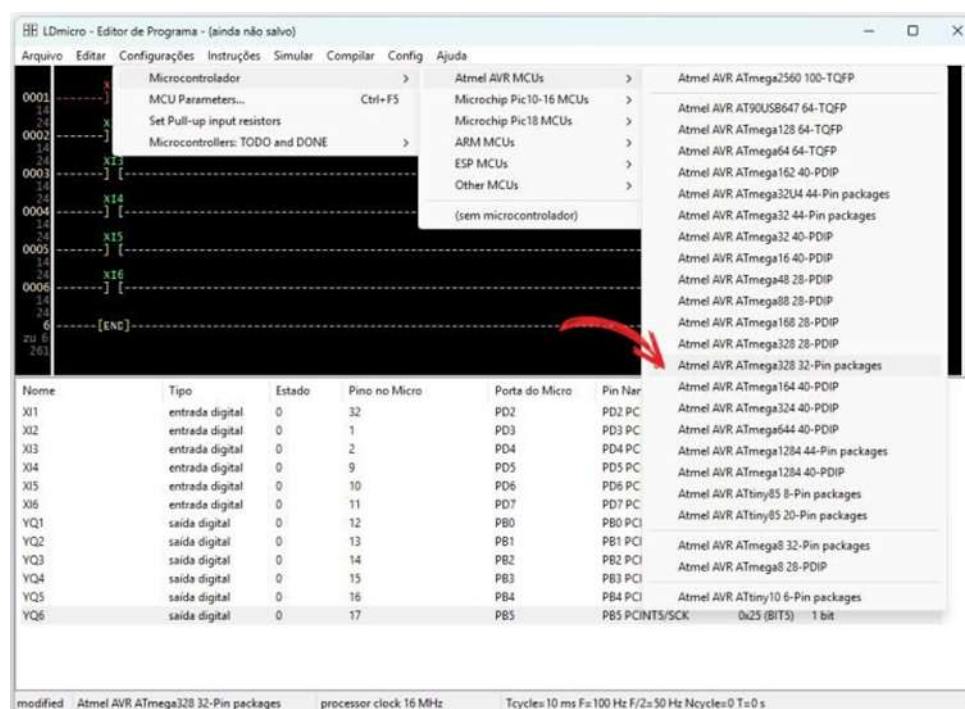


Figura 32 – Seleção do microcontrolador no Ldmicro

Após o desenvolvimento do fluxograma no ambiente gráfico do LDmicro, o software oferece diversas opções de exportação do programa para diferentes arquiteturas e plataformas. Neste projeto, serão destacadas duas formas principais de utilização prática:

- **Exportação como código C:** o LDmicro permite gerar automaticamente o código em linguagem C compatível com microcontroladores da família AVR, como o ATmega328P utilizado neste protótipo. Esse código pode ser aberto e compilado diretamente na Arduino IDE, sendo carregado na placa via porta USB-Serial CH340C (Figura 33).
- **Exportação como arquivo .hex:** também é possível gerar diretamente o arquivo binário compilado (.hex) a partir da lógica Ladder implementada. Esse arquivo pode ser gravado no microcontrolador utilizando ferramentas como o AVRDUDESS, por meio da interface USB-Serial CH340C, com controle detalhado do processo de gravação (Figura 34).

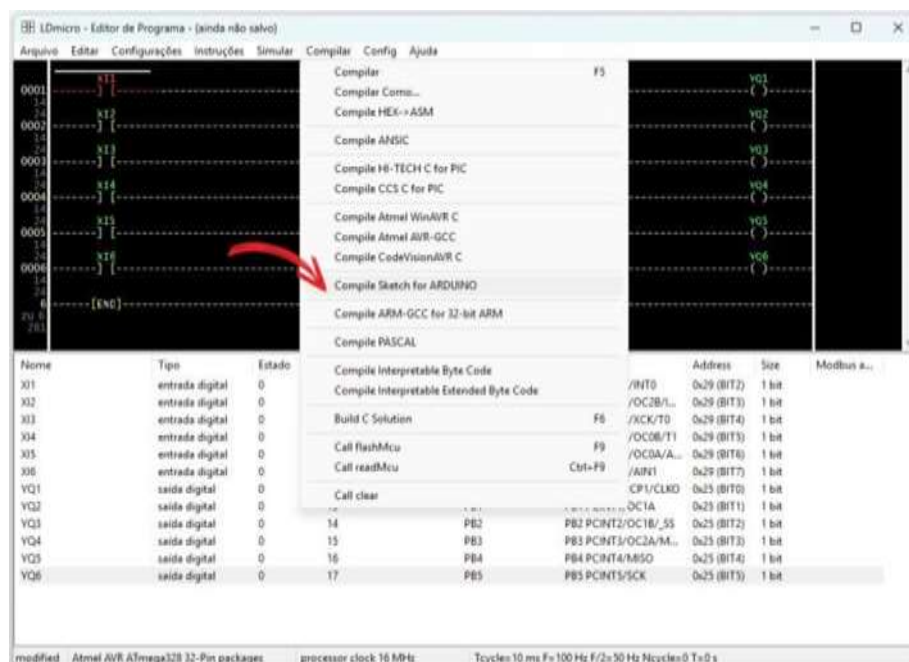


Figura 33 – Exportação do código C para uso na IDE Arduino

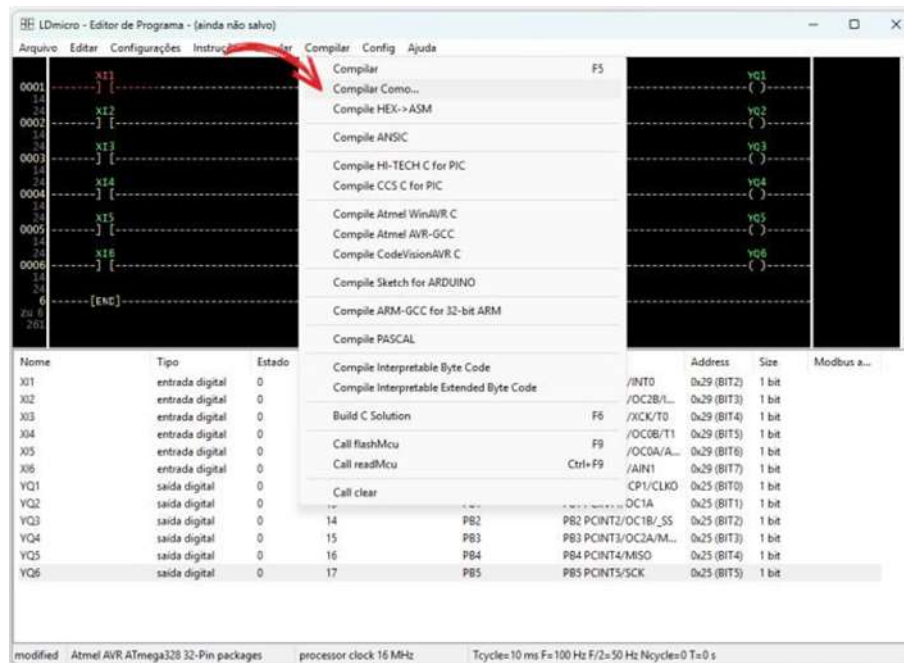


Figura 34 – Exportação do arquivo .hex para gravação via AVRDUDESS

Para a segunda opção de exportação, em que o programa Ladder é compilado diretamente como um arquivo .hex, é necessário utilizar um software gravador externo para transferir o código ao microcontrolador. Um dos programas mais utilizados para essa tarefa é o **AVRDUDESS**, que fornece uma interface gráfica para o utilitário AVRdude. Na Figura 35, observa-se a tela principal do software, onde o usuário deve configurar corretamente o microcontrolador (ATmega328P), a porta de comunicação (ex: COM4) e o arquivo .hex gerado pelo LDmicro.

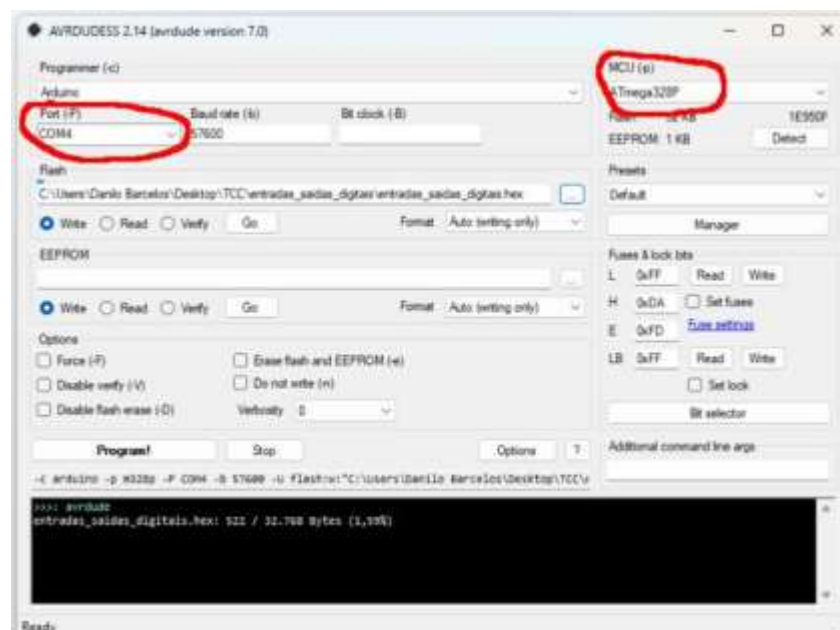


Figura 35 – Tela de configuração do AVRDUDESS para gravação do arquivo .hex

A adoção do LDmicro demonstrou que o protótipo desenvolvido é plenamente compatível com ferramentas gráficas baseadas na linguagem Ladder, amplamente utilizadas na indústria e no meio acadêmico. A possibilidade de exportação do programa tanto em linguagem C para compilação na IDE Arduino quanto como arquivo .hex para gravação via USB-Serial CH340C exemplifica a versatilidade do sistema em aceitar múltiplas abordagens de programação, sejam elas textuais ou gráficas. Essa flexibilidade amplia o espectro de aplicações do protótipo, viabilizando seu uso em contextos didáticos, projetos de ensino técnico e até mesmo em soluções de automação leve, onde microcontroladores podem substituir CLPs tradicionais em tarefas de controle discreto.

5 TESTES E AVALIAÇÃO DO PROTÓTIPO

Este capítulo apresenta os testes práticos realizados com o protótipo finalizado, visando validar o funcionamento das entradas e saídas, bem como a compatibilidade com diferentes métodos de programação. Os testes foram conduzidos após a montagem física e gravação do firmware via ISP ou USB, com o objetivo de simular aplicações reais de controle e automação.

5.1 TESTES FUNCIONAIS

Os testes foram divididos em quatro grupos principais, correspondendo aos sub-sistemas de entrada e saída implementados no protótipo: entradas digitais, saídas a relé, entradas analógicas e saídas PWM. Em cada etapa, foi aplicado um sketch de validação desenvolvido previamente na IDE Arduino ou exportado via LDmicro.

5.1.1 ENTRADAS DIGITAIS E SAÍDAS A RELÉ

O primeiro teste consistiu na simulação de sinais digitais de 24Vcc aplicados às seis entradas opto-isoladas do protótipo. Para isso, foram utilizados botões com lógica PNP conectados à alimentação. O acionamento de cada botão refletiu diretamente na saída digital correspondente, acionando os relés.

A validação foi confirmada com o acionamento de cargas reais, como lâmpadas de 127V ligadas aos contatos dos relés. O comportamento observado correspondeu ao esperado, com resposta imediata e sem falhas de comutação. O mapeamento dos pinos utilizado está descrito no Código 4.2 apresentado na Seção 4.3.1.

5.1.2 ENTRADAS ANALÓGICAS (0–10V)

Na segunda etapa, foram aplicados sinais analógicos de 0 a 10Vcc nas duas entradas analógicas do protótipo. Esses sinais foram previamente condicionados por divisores de tensão e proteção com diodos zener, sendo lidos nas portas A6 e A7 do ATmega328P.

A leitura foi validada por meio de conversão ADC (0–1023) e mapeamento para controle de saídas PWM. A variação dos valores foi observada no Monitor Serial da IDE Arduino, confirmando a funcionalidade dos circuitos de condicionamento e a integridade das leituras.

5.1.3 SAÍDAS PWM

As três saídas PWM, localizadas nos pinos D9, D10 e D11, foram testadas com base na leitura das entradas analógicas. Foi utilizado o sketch descrito no Código 4.1, onde a tensão analógica lida em A6 e A7 controla o ciclo de trabalho das saídas PWM, simulando um controle proporcional.

A saída PWM foi verificada com auxílio de um osciloscópio e também utilizando LEDs como carga indicadora. A resposta foi proporcional à variação dos potenciômetros aplicados na entrada, validando o controle.

5.1.4 PROGRAMAÇÃO EM LADDER COM LDMICRO

Com o intuito de validar a compatibilidade do protótipo com ferramentas gráficas de programação, foi realizado um teste utilizando uma lógica Ladder desenvolvida no software LDmicro. O diagrama elaborado, equivalente à lógica apresentada no Código 4.2, foi exportado para linguagem C e posteriormente compilado na IDE Arduino. Como alternativa, também foi gerado o arquivo .hex, possibilitando a gravação direta no microcontrolador por meio do AVRDUDESS. Ambas abordagens de exportação mostraram-se funcionais, demonstrando a compatibilidade do protótipo com fluxos gráficos de desenvolvimento amplamente utilizados em contextos industriais e didáticos.

5.2 AVALIAÇÃO DOS RESULTADOS

Os testes realizados permitiram validar os seguintes pontos do protótipo:

- As entradas digitais operam corretamente com sinais de 24Vcc e lógica PNP, com isolamento por optoacopladores.
- As saídas a relé suportam cargas de até 5A/250Vca, com comutação segura e resposta rápida.
- As entradas analógicas operam na faixa de 0–10Vcc com resolução de 10 bits (ADC), adequadas para controle proporcional.

- As saídas PWM respondem adequadamente aos sinais analógicos, com duty cycle proporcional.
- O firmware pode ser carregado tanto por USB (CH340C) quanto via ISP, inclusive com uso de bootloader externo.
- A programação com LDmicro foi eficaz, demonstrando compatibilidade total com o protótipo.

Todos os testes práticos corroboraram o funcionamento esperado das funcionalidades implementadas, confirmando a viabilidade técnica do protótipo como uma plataforma didática e funcional para aplicações de automação básica.

5.2.1 COMPARATIVO DE CUSTOS

Além dos testes funcionais, foi realizada uma avaliação econômica do protótipo em relação a Controladores Lógicos Programáveis (CLPs) comerciais de entrada. O objetivo é destacar a viabilidade financeira da solução proposta, reforçando sua aplicabilidade em contextos educacionais.

Enquanto os CLPs comerciais de entrada, como o Siemens LOGO! e o Allen-Bradley MicroLogix 1000, apresentam preços na faixa de R\$ 1.100 a R\$ 2.750, o protótipo desenvolvido teve custo aproximado de R\$ 550 por unidade. Esse valor considera a produção de duas unidades, o que significa que o custo unitário pode ser ainda mais reduzido em produções em escala.

O comparativo evidencia que o protótipo é uma alternativa significativamente mais acessível, mantendo todas as funcionalidades essenciais para fins didáticos, como entradas e saídas digitais, entradas analógicas, saídas PWM, comunicação e recursos adicionais.

Tabela 7 – Comparativo de custos entre CLPs comerciais e o protótipo desenvolvido

Modelo	Custo aproximado (R\$)
CLPs comerciais de entrada	1.100 – 2.750
Protótipo desenvolvido	550

Esse resultado demonstra que a proposta atende ao objetivo de oferecer uma solução acessível, econômica e prática, sem abrir mão das funcionalidades essenciais para o ensino e a experimentação em laboratórios de automação.

Nota: O custo do protótipo foi estimado a partir do processo de produção na JLCPCB (JLCPCB, 2025), incluindo montagem e frete. Já os valores de referência dos CLPs comerciais foram obtidos por meio de consultas a catálogos e sites oficiais da Siemens (SIEMENS, 2025), WEG (WEG, 2025) e Rockwell Automation (AUTOMATION, 2025a).

6 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os principais resultados obtidos com o desenvolvimento e testes do protótipo de Controlador Lógico Programável (CLP) didático, assim como a análise crítica do desempenho em comparação com dispositivos comerciais e os objetivos inicialmente propostos.

6.1 DESEMPENHO FUNCIONAL

Os testes práticos descritos no Capítulo 5 demonstraram o pleno funcionamento dos principais blocos do sistema: entradas digitais, entradas analógicas, saídas a relé e saídas PWM. As funcionalidades foram validadas individualmente por meio de códigos desenvolvidos em linguagem C e em linguagem Ladder via LDmicro, conforme descrito na Seção 4.3.

As saídas a relé suportaram o acionamento de cargas reais, como lâmpadas de 127V, com resposta imediata aos comandos das entradas. As entradas analógicas apresentaram comportamento linear e coerente, com leitura precisa de sinais de 0–10V convertidos via divisores resistivos. As saídas PWM geraram sinais modulados compatíveis com drivers de potência e controle de velocidade para pequenos motores ou variação de brilho em LEDs de potência.

6.2 COMPARAÇÃO COM CLPS COMERCIAIS

O protótipo desenvolvido foi comparado a três CLPs amplamente utilizados no setor educacional e industrial: Siemens LOGO! 8, WEG CLIC-02 e Allen Bradley MicroLogix 1000. A Tabela 3 evidencia que, apesar das limitações em robustez e certificações industriais, o protótipo oferece uma gama de funcionalidades comparável e, em alguns casos, superior, como a presença de RTC, saídas PWM e interface USB-C.

Essa comparação reforça o potencial da solução proposta como ferramenta de ensino, com custo reduzido, alto grau de customização e compatibilidade com linguagens de programação industriais.

6.3 LIMITAÇÕES E DESAFIOS

Durante o desenvolvimento do projeto, algumas limitações foram identificadas:

- A ausência de isolamento galvânico nas saídas PWM limita o uso direto com cargas de alta potência, exigindo drivers adicionais.
- A interface de programação via USB depende do correto reconhecimento do conversor CH340C, o que pode gerar dificuldades iniciais em alguns sistemas operacionais.

- A montagem modular em três camadas, embora favoreça o aspecto didático e a manutenção, aumenta o número de interconexões e pode gerar falhas de contato se não forem adequadamente encaixadas.

6.4 ANÁLISE GERAL

Apesar dos desafios, o protótipo atendeu aos objetivos propostos: demonstrar a viabilidade de construção de um CLP didático baseado em tecnologias de código aberto, com estrutura modular, baixo custo e compatibilidade com ambientes gráficos de programação como o LDmicro.

A adoção da linguagem Ladder e sua integração via LDmicro reforça a aplicabilidade do projeto em cursos técnicos, de engenharia e disciplinas introdutórias à automação industrial.

Além da validação funcional, o protótipo demonstrou também sua viabilidade econômica. O custo aproximado de R\$ 550 por unidade representa cerca de metade do valor dos CLPs comerciais de entrada (na faixa de R\$ 1.100 a R\$ 2.750), sem abrir mão das funcionalidades essenciais. Esse resultado reforça a proposta inicial do trabalho de oferecer uma solução prática, acessível e adequada para fins educacionais em automação.

7 CONCLUSÃO E TRABALHOS FUTUROS

7.1 CONCLUSÃO

Este trabalho resultou no desenvolvimento e construção de um protótipo de Controlador Lógico Programável (CLP) didático, fundamentado em tecnologias de código aberto, com estrutura modular e compatível com linguagens industriais, como a programação Ladder. O projeto demonstrou atender às demandas de ensino técnico e à experimentação em laboratórios de automação, destacando-se pela acessibilidade, flexibilidade e baixo custo. O protótipo foi projetado utilizando o microcontrolador ATmega328P, distribuído em três camadas físicas: entradas e saídas, controle lógico, e sinalização. Essa divisão favoreceu a organização do circuito, a manutenção e a compreensão didática dos blocos funcionais. Foram implementadas 6 entradas digitais opto-isoladas, 2 entradas analógicas (0–10V), 6 saídas a relé e 3 saídas PWM, além de recursos adicionais como RTC e comunicação RS485.

A programação foi realizada em duas abordagens complementares: linguagem C, por meio da plataforma Arduino IDE, e linguagem Ladder, utilizando o software LDmicro. Ambas as metodologias demonstraram plena compatibilidade com o hardware desenvolvido, comprovando a versatilidade do protótipo e sua aderência às práticas de automação industrial.

A validação prática dos blocos foi bem-sucedida, com resposta adequada das saídas aos comandos digitais e analógicos, conforme os testes apresentados. Comparado a CLPs comerciais de entrada, o protótipo demonstrou desempenho satisfatório para fins educacionais e experimentais, com destaque para funcionalidades avançadas como saídas PWM, porta USB-C e RTC.

CONSIDERAÇÕES SOBRE O APRENDIZADO ADQUIRIDO

A realização deste projeto representou uma jornada de aprendizado técnico, prático e acadêmico extremamente rica e multidisciplinar. Ao longo de sua execução, foi possível integrar diversas áreas do conhecimento em automação, eletrônica, programação embarcada e engenharia de produto.

Destaca-se, em primeiro lugar, o domínio de ferramentas de projeto eletrônico, como o EasyEDA, utilizado para a criação dos esquemas e roteamento das placas de circuito impresso (PCBs). Esse processo exigiu o estudo de boas práticas de layout, isolamento elétrico, dimensionamento de trilhas e uso de encapsulamentos padronizados, além da organização de bibliotecas e footprints.

A produção das placas em ambiente industrial foi outro marco importante. A utilização do serviço JLCPCB, com montagem automatizada SMT, envolveu a criação de arquivos Gerber, geração de listas de materiais (BOM) e posicionamento de componentes (Pick and Place). Esse processo proporcionou uma vivência próxima das etapas reais de fabricação de produtos eletrônicos, com interação direta com fornecedores e análise de critérios de produção em série.

No campo do desenvolvimento físico, o projeto envolveu também a modelagem 3D da estrutura do protótipo, utilizando software CAD para definir dimensões, encaixes e a disposição das camadas, aproximando a prática da prototipagem profissional.

Na parte lógica, o projeto foi responsável por consolidar conhecimentos fundamentais sobre programação embarcada, em especial com o microcontrolador ATmega328P, amplamente utilizado na plataforma Arduino. Foram exploradas funcionalidades como entradas e saídas digitais, leitura analógica via conversores ADC, geração de sinais PWM e controle por meio de protocolos como I2C (RTC) e comunicação serial.

Adicionalmente, o uso do LDmicro permitiu compreender na prática como a programação Ladder, tradicional em ambientes industriais, pode ser integrada com microcontroladores, reforçando o aprendizado das normas da IEC 61131-3 e das lógicas de controle baseadas em contatos.

Por fim, o processo envolveu a organização de documentos técnicos, como arquivos de firmware, esquemas eletrônicos, documentação acadêmica e registros de testes, o que consolidou boas práticas de engenharia, planejamento e comunicação técnica.

Essa experiência extrapolou os limites de um projeto de graduação e serviu como uma verdadeira introdução ao desenvolvimento de produtos eletrônicos reais, conectando teoria, prática e indústria.

7.2 TRABALHOS FUTUROS

Embora o projeto tenha alcançado seus principais objetivos, diversas melhorias e extensões podem ser exploradas em trabalhos futuros:

- Desenvolvimento de software próprio de programação Ladder: conforme previsto inicialmente, pode-se retomar a proposta de criação de uma interface gráfica própria para programação Ladder, integrada à geração de código C para o ATmega328P.
- Implementação de proteção galvânica nas saídas PWM: incluir optoacopladores ou drivers dedicados para ampliar a robustez do protótipo em aplicações com cargas de maior potência.
- Integração com supervisórios: desenvolver interfaces com protocolos industriais (como Modbus) para comunicação com sistemas SCADA e supervisórios.
- Ampliação da memória e processamento: substituir o ATmega328P por microcontroladores mais robustos (como ESP32, STM32 ou ARM Cortex-M) para permitir mais I/Os, conectividade Wi-Fi/Bluetooth e maior capacidade de processamento.
- Certificação e encapsulamento industrial: adaptar o projeto físico para um encapsulamento mais resistente e com melhor proteção elétrica, visando aplicações semi-industriais ou educacionais em campo.
- Documentação ampliada e guia didático: elaborar manuais de uso, exercícios práticos e conteúdos complementares para aplicação direta em disciplinas de CLPs, automação ou eletrotécnica.

A continuidade do projeto pode contribuir significativamente para o ensino prático em instituições públicas e privadas, popularizando o acesso ao aprendizado de sistemas de controle programáveis de forma acessível, aberta e personalizável.

REFERÊNCIAS

- AG, S. LOGO! System Manual (V8). [S.l.], 2025. Acesso em: 06 set. 2025. Disponível em: <<https://assets.new.siemens.com/siemens/assets/api/uuid:557e34c1-4111-4625-921a-0717d0053571/Manual-LOGO-2020.pdf>>. Citado 3 vezes nas páginas 36, 39 e 44.
- ARDUINO. Arduino IDE e documentação oficial. 2025. Acesso em: 06 set. 2025. Disponível em: <<https://www.arduino.cc/>>. Citado 5 vezes nas páginas 9, 18, 27, 28 e 35.
- AUTOMATION, R. Allen-Bradley MicroLogix Controllers. 2025. Acesso em: 21 set. 2025. Disponível em: <<https://www.rockwellautomation.com>>. Citado na página 71.
- AUTOMATION, R. MicroLogix 1000 Programmable Controllers User Manual. [S.l.], 2025. Acesso em: 06 set. 2025. Disponível em: <https://literature.rockwellautomation.com/idc/groups/literature/documents/um/1761-um003_-en-p.pdf>. Citado 2 vezes nas páginas 36 e 39.
- BANZI, M. Getting Started with Arduino. 2. ed. Sebastopol: Maker Media, O'Reilly, 2011. Citado na página 27.
- BOLTON, W. Programmable Logic Controllers. 6. ed. Oxford: Newnes, 2015. Citado 2 vezes nas páginas 23 e 24.
- CNI. Monitor da Indústria 4.0: dados do mercado brasileiro. 2025. Estudo IMARC analisado pelo Observatório Nacional da Indústria, Confederação Nacional da Indústria. Acesso em: 06 set. 2025. Disponível em: <<https://agenciafiep.com.br/2025/02/05/como-a-automacao-avancada-esta-revolucionando-a-industria/>>. Citado na página 18.
- COMMISSION, I. E. IPC-2221: Generic Standard on Printed Board Design. 1998. Norma internacional de projeto de PCBs. Citado 2 vezes nas páginas 41 e 45.
- COMMISSION, I. E. IEC 61131-2: Programmable Controllers – Part 2: Equipment requirements and tests. 2007. Norma internacional. Citado na página 38.
- ELÉTRICOS, W. E. Manual do Usuário – CLIC-02. [S.l.], 2025. Acesso em: 06 set. 2025. Disponível em: <<https://static.weg.net/medias/downloadcenter/h86/hcf/WEG-CLIC-02-user-manual-10009280784-pt-en-es.pdf>>. Citado 2 vezes nas páginas 36 e 39.
- FRANCHI, C. M. Controladores Lógicos Programáveis: Teoria e Aplicações. 2. ed. São Paulo: Érica, 2008. ISBN 9788536501994. Citado 2 vezes nas páginas 24 e 26.
- IEEE. A History of Early Microcontrollers, Part 4: The Intel 8048 and 8748. 2025. Acesso em: 13 fev. 2025. Disponível em: <<https://www.eejournal.com/article/a-history-of-early-microcontrollers-part-4-the-intel-8048-and-8748/>>. Citado na página 22.
- IEEE. O que é um microcontrolador. 2025. Acesso em: 13 fev. 2025. Disponível em: <<https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/>>. Citado na página 22.

INC., M. T. ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. [S.l.], 2016. Acesso em: 06 set. 2025. Disponível em: <<https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328P-Data-Sheet-DS40002061B.pdf>>. Citado 4 vezes nas páginas 23, 27, 38 e 42.

INTEGRATED, M. DS1307 - 64 x 8, Serial, I2C Real-Time Clock. [S.l.], 2008.

Acesso em: 06 set. 2025. Disponível em: <<https://www.analog.com/media/en/technical-documentation/data-sheets/ds1307.pdf>>. Citado 2 vezes nas páginas 38 e 43.

International Organization for Standardization. ISO/IEC 14882:2020 – Information technology — Programming languages — C++. 2020. Acesso em: 06 set. 2025. Disponível em: <<https://isocpp.org/std/the-standard>>. Citado na página 31.

JLPCB. PCB Prototype & PCB Fabrication Manufacturer. 2025. Acesso em: 21 set. 2025. Disponível em: <<https://jlcpcb.com>>. Citado na página 71.

ROQUE, L. A. O. L. Automação de Processo com Linguagem Ladder e Sistemas Supervisórios. 1. ed. Rio de Janeiro: LTC, 2014. Citado 12 vezes nas páginas 9, 18, 19, 24,25, 26, 28, 29, 30, 31, 34 e 35.

SIEMENS. LOGO! Logic Module. 2025. Acesso em: 21 set. 2025. Disponível em: <<https://new.siemens.com/global/en/products/automation/systems/industrial/plc/logo.html>>. Citado na página 71.

STROUSTRUP, B. The C++ Programming Language. 4. ed. Boston: Addison-Wesley, 2013. Citado 2 vezes nas páginas 31 e 34.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. Sistemas digitais: princípios e aplicações.11. ed. São Paulo: Pearson Education do Brasil, 2011. Citado 2 vezes nas páginas 23 e 32.

WEG. Controladores Programáveis WEG CLIC. 2025. Acesso em: 21 set. 2025. Disponível em: <<https://www.weg.net>>. Citado na página 71.

APLICAÇÃO DE VISÃO COMPUTACIONAL PARA DETECÇÃO E MONITORAMENTO DE VEÍCULOS

Dedico este trabalho aos meus pais, que me ensinaram o valor da educação; aos amigos, pelo apoio nos momentos difíceis; e ao professor Luiz Roque, pela orientação neste trabalho.

AGRADECIMENTOS

Agradeço primeiramente à minha família, pelo apoio constante, pelo incentivo e pelo exemplo de dedicação.

Ao professor Luiz Roque, pela orientação e pelo apoio durante o desenvolvimento deste trabalho.

Aos amigos e colegas, pelo companheirismo e pela motivação ao longo da minha jornada acadêmica.

Ao Instituto Federal Fluminense, pela oportunidade de formação, pela infraestrutura disponibilizada e pelo ambiente de aprendizado.

Por fim, a todos que, de alguma forma, contribuíram para minha formação, direta ou indiretamente, expresso minha sincera gratidão.

"A scientist discovers that which exists. An engineer creates that which never was."

(Theodore von Kármán)

Resumo: Este trabalho explora a concepção de um sistema voltado ao controle de tráfego, utilizando visão computacional para detectar veículos em tempo real. Propõe-se um modelo projetado para monitorar cruzamentos e rodovias, com a capacidade de coletar dados sobre o fluxo de tráfego e auxiliar na tomada de decisões, como o ajuste dinâmico de semáforos ou a identificação de congestionamentos. A abordagem sugere o uso de algoritmos de detecção de objetos baseados em redes neurais convolucionais (CNNs), com possíveis implementações em Python e bibliotecas como OpenCV e YOLO.

Palavras-chave: Controle de Tráfego. StrongSORT. SUMO. Visão Computacional. YOLOv5.

LISTA DE ILUSTRAÇÕES

Figura 1 – Etapas para processamento digital de Imagens.....	137
Figura 2 – Arquitetura Básica de uma CNN.....	143
Figura 3 – Efeito da Camada Max Pooling.....	144
Figura 4 – Detecção do YOLO com Bounding boxes.....	146
Figura 5 – DeepSORT aplicado no dataset de desafio MOT, demonstrando rastreamento em condições desafiadoras.....	149
Figura 6 – Condição de Ofuscamento Solar, perspectiva de um carro autônomo.....	153
Figura 7 – Geometria do cruzamento bidirecional modelado no SUMO (4 abordagens, 1 faixa por sentido).....	165
Figura 8 – Configuração controle fixo do Semáforo.....	166
Figura 9 – Convergência das Métricas de Avaliação Durante o Treinamento com o COCO.....	169
Figura 10 – Evolução das métricas durante o treinamento do YOLOv5 com o dataset MIO-TCD.....	170
Figura 11 – Detecção de veículos realizada pelo modelo treinado.....	172
Figura 12 – Exemplo de rastreamento com YOLOv5 + StrongSORT	174
Figura 13 – Progressão do tempo médio de espera (ciclo fixo × adaptativo).....	175
Figura 14 – Histograma dos tempos de viagem (ciclo fixo × adaptativo).....	176
Figura 15 – Imagem anotada do dataset.....	189

LISTA DE TABELAS

Tabela 1 – Comprimento das aproximações do cruzamento.....	166
Tabela 2 – Detalhamento das fases do semáforo fixo.....	167
Tabela 3 – Parâmetros do algoritmo Max-Pressure	168
Tabela 4 – Resultados das métricas de desempenho na validação com o dataset MIO-TCD.....	171
Tabela 5 – Métricas de rastreamento obtidas com o modelo YOLOv5 + StrongSORT	173
Tabela 6 – Indicadores médios por cenário de controle (1.200 veículos).....	175
Tabela 7 – Anotação YOLO correspondente à Figura 15. Cada linha representa um objeto detectado da classe "carro"(Classe 0).....	189

LISTA DE ABREVIATURAS E SIGLAS

CNN	Convolutional Neural Network
DeepSORT	Simple Online and Realtime Tracking with a Deep Association Metric
IA	Inteligência Artificial
IFF	Instituto Federal Fluminense
IPEA	Instituto de Pesquisa Econômica Aplicada
IoU	Intersection over Union
mAP	mean Average Precision
NMS	Non-Maximum Suppression
ONU	Organização das Nações Unidas
OpenCV	Open Source Computer Vision Library
PDI	Processamento Digital de Imagens
ReLU	Rectified Linear Unit
SCATS	Sydney Coordinated Adaptive Traffic System
SUMO	Simulation of Urban Mobility
TraCI	Traffic Control Interface
YOLO	You Only Look Once

LISTA DE SÍMBOLOS

AP_i	Average Precision da classe/limiar i
F_1	F1-Score (média harmônica entre Precisão e Revocação)
FN	Falsos Negativos
FP	Falsos Positivos
N	Número de classes/limiares considerados no cálculo do mAP
S	Dimensão da grade $S \times S$ usada pelo YOLO
TN	Verdadeiros Negativos
TP	Verdadeiros Positivos
Δq	Diferença de pressão entre eixos no controle Max-Pressure
q_{verde}	Comprimento de fila no eixo com sinal verde
$q_{vermelho}$	Comprimento de fila no eixo com sinal vermelho

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

O aumento das frotas de veículos é um fenômeno majoritariamente global, impulsionado pelo crescimento populacional, pela expansão das cidades e pela crescente acessibilidade a automóveis. Esse crescimento tem gerado impactos significativos, como trânsito, acidentes e dificuldades na gestão do tráfego urbano. Segundo dados da Organização das Nações Unidas (United Nations, 2018), mais de 55% da população mundial vive em áreas urbanas, e esse percentual deve alcançar 68% até 2050, aumentando a demanda por uma mobilidade eficiente. No Brasil, o Instituto de Pesquisa Econômica Aplicada (IPEA, 2021) informa que a frota de veículos cresceu mais de 120% entre 2001 e 2020, agravando os problemas de congestionamento nas principais capitais. Diante desse cenário, tecnologias como a visão computacional têm emergido como uma das soluções para lidar com os desafios da mobilidade urbana, permitindo a automação de sistemas de tráfego e oferecendo maior eficiência na circulação de veículos.

A visão computacional permite que sistemas autônomos sejam capazes de analisar e interpretar informações visuais de um ambiente físico ao invés de digital. No caso de um controle de tráfego, essa tecnologia pode ser utilizada para a detecção e contagem de veículos, fornecendo informações vitais para planejamentos urbanos, detecção de gargalos, gestão de semáforos e até mesmo no desenvolvimento de veículos autônomos.

Algoritmos modernos de detecção de objetos, como o YOLO (You Only Look Once), utilizado neste trabalho, têm se mostrado cada vez mais capazes nesta função de identificação de veículos, devido à sua capacidade de operar em tempo real com precisão e velocidade. YOLO utiliza redes neurais convolucionais para detectar e classificar objetos em um único estágio, tornando-o excelente em aplicações que exigem resposta rápida, como sistemas de controle de tráfego, com a capacidade do modelo de processar vídeos em tempo real com menos de 25 milissegundos de latência (REDMON; FARHADI, 2016).

1.2 PROBLEMA

A crescente urbanização e o aumento significativo da frota de veículos têm gerado uma grande pressão sobre os sistemas de tráfego nas cidades. Problemas como congestionamentos, atrasos, aumento no consumo de combustíveis e emissões de poluentes se intensificam a cada ano,

especialmente em regiões com alta densidade populacional. Esses desafios impactam diretamente a qualidade de vida dos cidadãos e a eficiência econômica, ocasionando atrasos, problemas de saúde e até riscos à segurança, devido à vulnerabilidade em grandes engarrafamentos. Apesar dos avanços na tecnologia de transporte, os métodos tradicionais de gestão de tráfego, como os sistemas fixos de temporização de semáforos, têm-se mostrado insuficientes para lidar com a complexidade e a dinâmica do fluxo de veículos. Diante disso, surge a necessidade de soluções mais inteligentes e adaptáveis, capazes de monitorar e reagir em tempo real às condições do tráfego, utilizando tecnologias avançadas, como a visão computacional, para detectar, classificar e contabilizar veículos de maneira eficiente, auxiliando no processo de tomada de decisão e no planejamento urbano.

1.3 OBJETIVOS GERAIS

A proposta deste trabalho é projetar um sistema capaz de detectar e contar veículos em tempo real a partir de vídeos. As informações obtidas pelo sistema podem então ser empregadas para otimizar a mobilidade urbana por meio da automação de semáforos, além de oferecer subsídios para aplicações em áreas como planejamento urbano.

1.4 OBJETIVOS ESPECÍFICOS

- Apresentar importantes tecnologias e características da visão computacional
- Projetar uma função de reconhecimento de carros por processamento de imagens com o modelo YOLO.
- Modelar conceitualmente a detecção em tempo real por meio de rastreamento de objetos
- Simular a atuação de um sistema de semáforo inteligente utilizando dados de detecção de veículos no simulador Simulation of Urban MObility (SUMO).

1.5 JUSTIFICATIVA

A utilização de tecnologias de visão computacional na automação do tráfego urbano surge como uma solução para os desafios enfrentados pelas grandes cidades em termos de mobilidade e segurança no trânsito. A implementação de sistemas de semáforos inteligentes, que ajustam dinamicamente os tempos de sinalização com base na detecção de veículos e velocidade de fluxo, pode resultar em benefícios significativos, como a redução de congestionamentos, o aumento da segurança em rodovias e a melhoria da qualidade de vida dos cidadãos por conta do menor tempo de deslocamento (REDIN et al., 2019). Além disso, contribui para a sustentabilidade urbana ao diminuir as emissões de gases poluentes e o consumo de combustíveis fósseis (Transport for NSW, s.d.). A automação proposta

também pode fornecer dados essenciais para o planejamento urbano e contribuir para a redução de custos operacionais na gestão do tráfego. Portanto, este trabalho justifica-se pela necessidade urgente de melhorar a mobilidade nas áreas urbanas e pela potencial contribuição das tecnologias de visão computacional para a construção de cidades mais inteligentes e sustentáveis.

2 FUNDAMENTAÇÃO TEÓRICA

A aplicação de técnicas de visão computacional para a detecção e monitoramento de veículos tem sido amplamente explorada nas últimas décadas. O crescimento da frota de veículos nas cidades e a necessidade de soluções inteligentes para o gerenciamento do tráfego vêm impulsionando o desenvolvimento de sistemas baseados em inteligência artificial (IA). Este capítulo apresenta uma revisão das principais abordagens, conceitos e métodos nessa área, com base em estudos relevantes.

2.1 PROCESSAMENTO DIGITAL DE IMAGENS

O processamento digital de imagens é um campo fundamental para o desenvolvimento de sistemas de visão computacional. Ele envolve a aplicação de algoritmos computacionais para o tratamento e análise de imagens digitais, possibilitando a extração de informações relevantes para diferentes aplicações.

Pode ser organizado em três níveis: baixo, médio e alto.

- **Baixo nível:** operações primitivas de pré-processamento, como redução de ruído e ajuste de contraste ou nitidez, input e output ambos são imagens.
- **Médio nível:** procedimentos mais sofisticados, como detecção e segmentação, que recebem uma imagem como entrada e produzem atributos extraídos como saída.
- **Alto nível:** análises e interpretações dos dados e das imagens obtidos nas etapas anteriores.

(GONZALEZ; WOODS, 2018)

2.1.1 ETAPAS DO PROCESSAMENTO DIGITAL DE IMAGENS

O processamento digital de imagens pode ser dividido em várias etapas principais, cada uma desempenhando um papel na transformação e análise de dados visuais. Essas etapas incluem desde a captura inicial até a extração de informações úteis para aplicações específicas. A seguir, é apresentado um fluxograma que ilustra o fluxo típico das etapas envolvidas no processamento de imagens digitais.

Figura 1 – Etapas para processamento digital de Imagens



Fonte: Elaboração do Autor

2.1.1.1 AQUISIÇÃO DE IMAGEM

A aquisição de imagem é a etapa inicial do processamento digital, sendo responsável por capturar a cena de interesse por meio de sensores, como câmeras digitais ou dispositivos ópticos. Esse processo envolve a conversão do sinal analógico (captado pelos sensores) em uma representação digital que possa ser tratada por algoritmos computacionais. A qualidade da imagem adquirida pode ser influenciada por fatores como a resolução espacial, a iluminação do ambiente e as características do sensor utilizado. Tecnologias como câmeras CCD (Charge-Coupled Device) e CMOS (Complementary Metal-Oxide-Semiconductor) são amplamente empregadas nessa etapa devido à sua alta eficiência e precisão (SZELISKI, 2022).

2.1.1.2 PRÉ-PROCESSAMENTO DE IMAGENS

O pré-processamento de imagens é uma etapa clássica em visão computacional e costuma melhorar o desempenho de detectores em tempo real, como o YOLO. Segundo Gonzalez e Woods (2018), operações de pré-processamento, como normalização de intensidades, correção de iluminação, redução de ruído e realce de contraste, ajudam a uniformizar as entradas e a aumentar a relação sinal-ruído. Na prática, para tarefas de detecção, costuma-se redimensionar as imagens para a resolução esperada pelo modelo e normalizar os valores de pixel (para o intervalo $[0, 1]$). A conversão para escala de cinza é uma possibilidade, mas não obrigatória: ela reduz a dimensionalidade ao custo de descartar informação cromática, o que pode ser indesejável quando a cor é discriminativa (por exemplo, luzes de freio, semáforos ou padrões de placa). Assim, em cenários dinâmicos de monitoramento de tráfego com foco na identificação de veículos, é comum preservar as três bandas (RGB) e limitar-se à reamostragem e à normalização, equilibrando precisão e velocidade.

2.1.1.3 SEGMENTAÇÃO

A segmentação divide a imagem em regiões ou objetos de interesse, viabilizando análises subsequentes, especialmente em cenários urbanos com sobreposições e oclusões (GONZALEZ; WOODS, 2018). Os métodos clássicos usam:

- Descontinuidade: detecção de bordas via mudanças abruptas de intensidade.
- Similaridade: agrupamento de pixels homogêneos por limiarização, crescimento ou fusão de regiões.
- Híbridos: combinação de bordas e superpixels para maior robustez.

Técnicas baseadas em grafos, morfologia matemática e redes convolucionais (Como o Mask R-CNN (HE et al., 2017)) elevam a precisão da segmentação. O YOLO, por sua vez, prevê caixas delimitadoras e classes em uma única etapa, dispensando a segmentação, garantindo alta velocidade em tempo real e um pipeline simples. Por outro lado, não gera máscaras detalhadas de objetos e pode apresentar menor precisão em contornos finos ou em cenários com objetos muito próximos.

2.1.1.4 EXTRAÇÃO DE CARACTERÍSTICAS

A extração de características é realizada após a segmentação, e desempenha um papel importante em várias aplicações de engenharia. Ela envolve duas atividades principais: detecção e descrição das características. A detecção refere-se à identificação de elementos significativos na imagem, como pontos de interesse, bordas ou regiões específicas, enquanto a descrição atribui atributos quantitativos a esses elementos, como localização, orientação ou intensidade, os quais permitem uma análise mais detalhada. Por exemplo, é detectada uma borda de um objeto e então descrita essa borda pela sua orientação e posição.

Conforme Gonzalez e Woods (2018), os métodos de extração de características podem ser classificados em bordas, regiões ou imagens completas. Para garantir robustez e eficiência, é fundamental que os descritores sejam invariantes a variações de escala, translação, rotação, iluminação e ponto de vista. Assim, mesmo em ambientes dinâmicos, como ruas de uma cidade, onde mudanças de iluminação, ângulo de câmera e velocidade dos veículos são constantes, as características extraídas permanecem estáveis e confiáveis.

2.1.1.5 RECONHECIMENTO E INTERPRETAÇÃO

A etapa de reconhecimento e interpretação é crucial no processamento de imagens, pois atribui significado às informações extraídas, permitindo a identificação e compreensão dos elementos

presentes em uma cena. Após a extração das características da imagem, essa etapa atribui significado aos dados processados, transformando-os em informações úteis para a tomada de decisão.

No reconhecimento, a tarefa principal é fazer a identificação dos objetos relevantes, como um carro, por exemplo, por meio de técnicas como caixas delimitadoras (Bounding boxes). Abordagens utilizando aprendizado profundo (deep learning) têm se mostrado líderes neste aspecto, redes neurais convolucionais (CNNs) se destacando por sua precisão e velocidade. Trabalhos como o de Krizhevsky, Sutskever e Hinton (2012), introduzindo o, hoje conhecido como, AlexNet, demonstraram a capacidade de redes neurais profundas em identificar padrões complexos em imagens, estabelecendo uma base sólida para aplicações modernas na área. Desde então, modelos mais avançados têm sido amplamente utilizados para tarefas de detecção, incluindo Faster R-CNN (REN et al., 2016) e YOLO (REDMON; FARHADI, 2016).

A interpretação, por sua vez, vai além do reconhecimento, tendo foco na análise das relações entre os objetos de interesse com o ambiente. No caso de um algoritmo visando a detecção de carros, a interpretação iria considerar a movimentação do carro, velocidades acima do esperado, comportamentos errôneos, análise de risco, entre outros.

Estudos, como o de Geiger, Lenz e Urtasun (2012), também destacam que um pré-requisito para o funcionamento adequado de um algoritmo de reconhecimento e interpretação é sua capacidade de manter a generalização. Como indicado no trabalho, diversos dos algoritmos de detecção mais avançados, quando sujeitos ao sobreajuste (overfitting) em conjuntos de dados como o Middlebury, tornam-se incapazes de manter uma performance satisfatória fora de um ambiente controlado. Por esse motivo, datasets como o KITTI Vision Benchmark Suite, voltado para navegação autônoma, têm sido desenvolvidos para proporcionar uma melhor aplicabilidade dos algoritmos em cenários do mundo real.

2.1.2 APLICABILIDADE EM VISÃO COMPUTACIONAL

A visão computacional, uma subdivisão da inteligência artificial (IA), é baseada no processamento digital de imagens para extrair informações visuais relevantes, permitindo a análise e interpretação do meio físico. O processamento digital de imagens (PDI) permite que sistemas computacionais compreendam o conteúdo visual, possibilitando desde a segmentação de objetos até a extração de características que alimentam algoritmos de reconhecimento e interpretação.

Na visão computacional, o PDI é amplamente aplicado em áreas como segurança pública, monitoramento de tráfego, saúde, agricultura de precisão e veículos autônomos. Por exemplo, em sistemas de detecção e monitoramento de veículos, etapas como a segmentação de imagens e a extração de características são utilizadas para identificar veículos em vídeos ou imagens estáticas, determinar suas posições e analisar comportamentos no tráfego urbano (GEIGER; LENZ; URTASUN, 2012).

2.2 DEEP LEARNING

O Deep Learning é uma subárea do aprendizado de máquina que utiliza redes neurais profundas para aprender representações hierárquicas de dados. Seu surgimento e avanço foram possíveis graças à combinação de aumento no poder computacional, crescimento das bases de dados disponíveis e desenvolvimento de algoritmos eficientes para treinamento de redes neurais. Essa abordagem tem revolucionado diversas áreas, incluindo a visão computacional, permitindo avanços significativos em tarefas como classificação de imagens e detecção de objetos.

A visão computacional, antes baseada em técnicas manuais de extração de características, foi transformada pelo Deep Learning. Redes neurais profundas aprendem automaticamente padrões diretamente das imagens, o que simplifica o processo e melhora a precisão em diversas aplicações. Esse salto tecnológico possibilitou o desenvolvimento de sistemas robustos e escaláveis para análise visual.

Redes Neurais Convolucionais (CNNs) são centrais nesse avanço. Com elas é possível aprender automaticamente padrões em imagens, como bordas, texturas e formas. Introduzidas por LeCun et al. (1998), e popularizadas por AlexNet ((KRIZHEVSKY; SUTSKEVER; HINTON, 2012)), as CNNs utilizam camadas convolucionais para analisar dados visuais de maneira eficiente.

O YOLO (You Only Look Once) exemplifica a aplicação do Deep Learning na detecção de objetos, processando imagens em tempo real. Baseado em CNNs, o YOLO realiza a detecção de forma unificada, dividindo a imagem em uma grade e prevendo objetos e suas localizações em uma única passagem pela rede. Essa abordagem simplifica e acelera tarefas de detecção, sendo amplamente utilizada em monitoramento de veículos e sistemas de vigilância.

Portanto, o Deep Learning, ao introduzir redes neurais profundas como as CNNs, possibilitou avanços significativos na visão computacional, transformando tarefas complexas, como detecção e monitoramento de veículos, em processos rápidos e precisos.

2.2.1 TRANSFERÊNCIA DE APRENDIZADO

A transferência de aprendizado aproveita modelos pré-treinados cujas camadas iniciais capturam características genéricas (bordas, texturas etc.), acelerando o treinamento em tarefas relacionadas e reduzindo o risco de sobreajuste em conjuntos de dados menores. Segundo Pan e Yang (2010), essa estratégia também pode melhorar a precisão e economizar recursos computacionais.

Neste trabalho, aplicou-se transferência de aprendizado ao YOLOv5: primeiro treinou-se na base COCO (classe "car"), depois refinou-se os pesos no MIO-TCD.

2.3 VISÃO COMPUTACIONAL

A visão computacional é uma área da inteligência artificial que se dedica ao desenvolvimento de algoritmos e sistemas capazes de interpretar e entender o conteúdo de imagens e vídeos, com o objetivo de automatizar e auxiliar tarefas normalmente realizadas pela visão humana. Essa disciplina envolve a extração de informações úteis a partir de dados visuais, permitindo que máquinas compreendam o ambiente ao seu redor e tomem decisões com base nas imagens capturadas. A aplicação de visão computacional é ampla, abrangendo áreas como automação industrial, monitoramento de segurança, medicina, robótica, veículos autônomos, agricultura, pecuária, militar, entre outros.

2.3.1 AVANÇOS HISTÓRICOS

A história da visão computacional começa nas décadas de 1950 a 1970, quando as primeiras tentativas de simular a visão humana em sistemas computacionais começaram a surgir, como "Perceptron", uma rede neural desenvolvida por Rosenblatt (1958), a qual serviu como base para avanços futuros. Durante esse período inicial, os estudos concentraram-se em entender como os computadores poderiam processar e interpretar imagens básicas. Pesquisas focaram na extração de características simples, como bordas e contornos, utilizando métodos baseados em álgebra linear e transformadas matemáticas, como a Transformada de Fourier, decompondo imagens em seus componentes de frequência. Nessa época, as limitações de hardware e software restringiam significativamente o alcance das aplicações práticas.

Nos anos 1980, a área de processamento digital de imagens registrou avanços significativos, com o desenvolvimento de técnicas mais robustas para análise de imagens, incluindo a segmentação de objetos e a detecção de padrões. Entre os marcos desse período, destaca-se o algoritmo de detecção

de bordas de Canny (1986), o "Canny Edge Detector", e a aplicação da transformada de Hough, uma técnica capaz de detectar formas geométricas como linhas e círculos (DUDA; HART, 1972). Surgindo assim como soluções para tarefas de processamento inicial de imagens, e com elas os primeiros experimentos com sistemas de reconhecimento de objetos começaram, embora fossem ainda limitados a cenários altamente controlados e experimentais.

Na década de 1990, o campo da visão computacional experimentou um crescimento significativo graças ao aumento do poder computacional e ao avanço de tecnologias de câmeras digitais, que permitiram a captura de imagens com maior qualidade. Durante esse período, o foco voltou-se para a integração de técnicas estatísticas e aprendizado de máquina, que começaram a ser utilizadas para melhorar a capacidade dos algoritmos em reconhecer padrões mais complexos. Métodos como os modelos de mistura gaussiana (GMM) para segmentação de fundo (STAUFFER; GRIMSON, 1999) e os primeiros sistemas baseados em reconhecimento facial ganharam destaque. Além disso, frameworks como o SIFT (Scale-Invariant Feature Transform), desenvolvido por Lowe (1999), trouxeram avanços significativos para a extração de características invariantes.

Contudo, foi a partir da década de 2010 que a visão computacional experimentou uma verdadeira revolução, impulsionada por três fatores principais: o crescimento exponencial da capacidade de processamento (especialmente com o uso de GPUs), a disponibilidade de grandes volumes de dados rotulados e o avanço das técnicas de aprendizado profundo. Redes neurais convolucionais (CNNs), como a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), demonstraram pela primeira vez o potencial do aprendizado profundo em tarefas de classificação de imagens, superando significativamente os métodos tradicionais.

Desde então, a visão computacional tem evoluído rapidamente, com a introdução de arquiteturas cada vez mais sofisticadas, com modelos como a Mask R-CNN, que combinam segmentação e detecção de objetos com alta precisão. Atualmente, a visão computacional está profundamente integrada em diversas áreas, refletindo um progresso que foi possibilitado por décadas de pesquisa e inovação tecnológica.

2.4 REDES NEURAIS CONVOLUCIONAIS

Dentre os algoritmos existentes para reconhecimento e classificação de imagens, as redes neurais convolucionais (Convolutional Neural Networks, CNNs) destacam-se pela sua eficiência no processamento de dados visuais. Inspiradas na organização do córtex visual, essas redes possuem uma

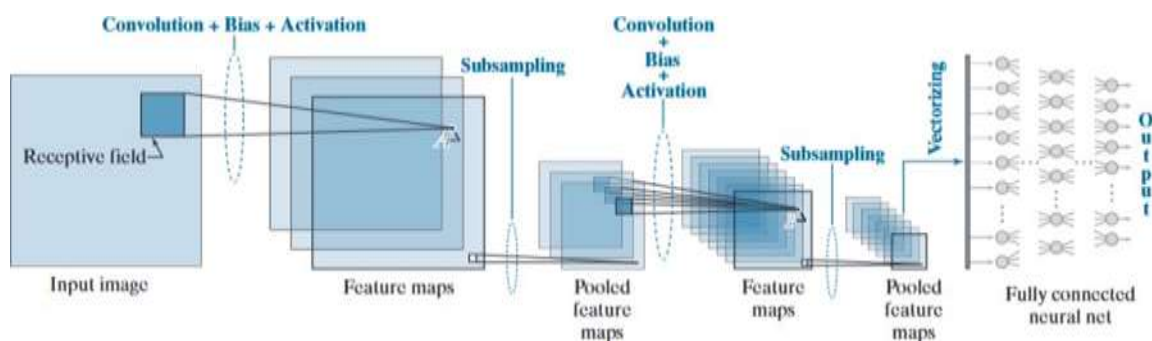
estrutura hierárquica em que camadas convolucionais iniciais aprendem a identificar características de baixo nível, como linhas e bordas, camadas intermediárias capturam padrões mais complexos, e camadas mais profundas aprendem representações abstratas e específicas.

De forma análoga aos neurônios do córtex visual em animais, cada filtro convolucional, pequenas matrizes de pesos utilizadas nas operações das CNNs, possui um campo receptivo correspondente a uma pequena área da imagem. Esses filtros extraem informações locais precisas, como texturas ou contornos. À medida que a informação avança pelas camadas da rede, os campos receptivos efetivos se expandem, permitindo que a CNN detecte padrões em contextos maiores, combinando características de diferentes níveis com alta precisão

2.4.1 ETAPAS DE UMA CNN

As Redes Neurais Convolucionais (CNNs) são compostas por diferentes camadas que trabalham juntas para processar e analisar imagens. Cada etapa desempenha um papel fundamental na extração e interpretação de características, permitindo que a rede reconheça padrões complexos. Abaixo, estão detalhadas essas camadas com suas respectivas funções:

Figura 2 – Arquitetura Básica de uma CNN



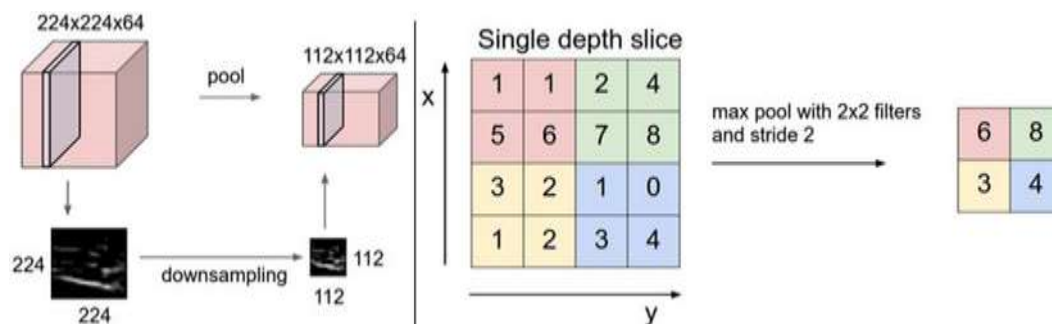
Fonte: Gonzalez e Woods (2018)

- **Entrada** - A entrada de uma CNN é uma imagem representada por uma matriz de pixels, onde cada pixel possui um valor de intensidade, ou em escala cinza (Matriz 2D) ou colorida (Matriz 3D com canais RGB). Essa matriz será analisada nas etapas subsequentes.
- **Camada Convolucional** - A camada convolucional é a principal responsável pela extração de características das imagens de entrada em redes neurais convolucionais (CNNs). Nessa camada, aplicam-se filtros, também chamados de kernels, que percorrem a imagem realizando operações de convolução para detectar padrões relevantes, como bordas, linhas e texturas.

Cada filtro gera uma parte de um mapa de características, que representa informações importantes da imagem, ajudando a rede a aprender os padrões presentes nela (GOODFELLOW; BENGIO; COURVILLE, 2016). Além disso, é nesta etapa que o bias é introduzido, um parâmetro adicional somado ao resultado das operações de convolução. O bias auxilia no ajuste da saída de cada neurônio, ou unidade de ativação, e é essencial para o processo de treinamento da rede, permitindo maior flexibilidade no aprendizado.

- **Função de Ativação** - Responsável por introduzir não linearidade nos modelos de redes neurais, permitindo que aprendam padrões complexos e aumentem sua capacidade de generalização. Existem diversas variações de funções de ativação, como ReLU (Rectified Linear Unit), Tanh, Swish, Mish, Sigmoid, entre outras. Sendo a ReLU amplamente reconhecida por sua simplicidade e eficiência computacional, além de apresentar desempenho superior em diversos cenários, como demonstrado por Hao et al. (2020).
- **Camada de Pooling** - Frequentemente inserida entre camadas convolucionais em redes neurais convolucionais, tendo como principal objetivo reduzir a dimensionalidade espacial das representações. Essa redução diminui a quantidade de parâmetros processados, reduzindo o tempo e o custo computacional, além de contribuir para mitigar o overfitting, ou seja, o sobreajuste do modelo aos dados de treinamento. Existem diferentes métodos de pooling, sendo o mais comum o Max Pooling, que divide o mapa de características em regiões específicas e seleciona o maior valor de cada região. Esse processo reduz o mapa para um conjunto mais compacto de valores, preservando informações relevantes, conforme ilustrado na imagem abaixo.

Figura 3 – Efeito da Camada Max Pooling



Fonte: Karpathy, Johnson e Li (s.d.)

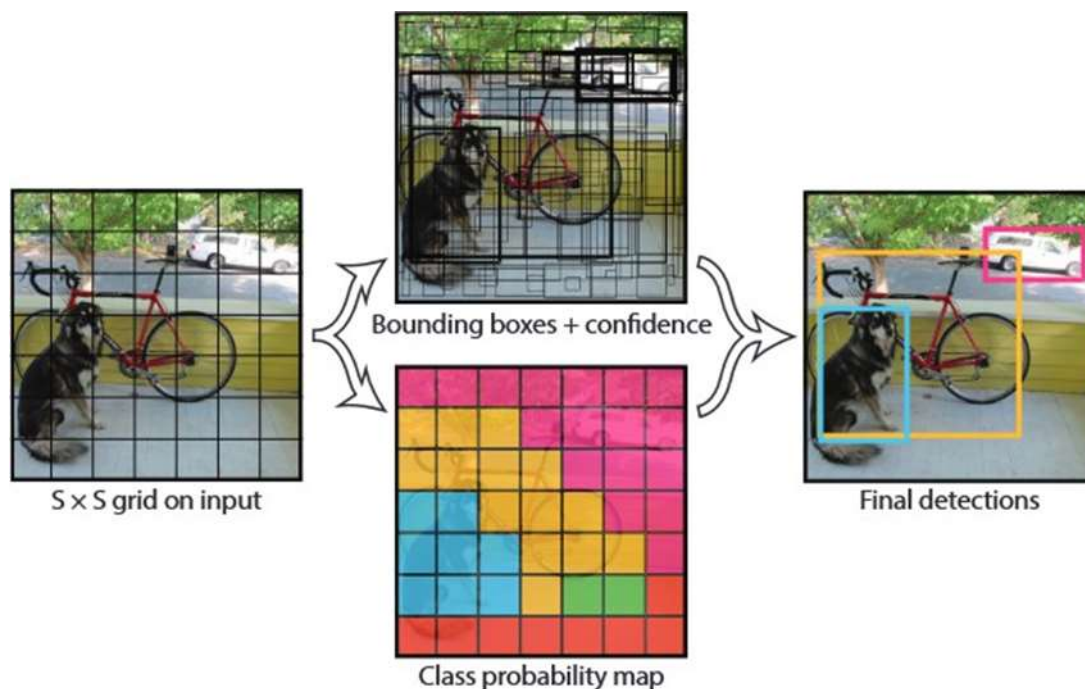
- **Camada totalmente conectada** - A camada totalmente conectada (Fully Connected - FC) é a etapa final em muitas arquiteturas de redes neurais convolucionais (CNNs), sendo responsável por traduzir as características extraídas nas camadas anteriores em uma predição final. Nessa camada, cada neurônio é conectado a todos os neurônios da camada anterior, permitindo a combinação das informações aprendidas ao longo da rede. Essa conexão total é essencial para tarefas como classificação, pois possibilita que o modelo relacione as características extraídas a classes específicas (KARPATHY; JOHNSON; LI, s.d.).
- **Saída - Última** etapa de uma rede neural convolucional (CNN), responsável por apresentar o resultado final do modelo. É nela que a rede retorna a predição, como a classificação encontrada para uma imagem.

2.5 ALGORITMO YOU ONLY LOOK ONCE

Para a detecção de objetos com alta velocidade e precisão, o algoritmo YOLO (You Only Look Once) desenvolvido e apresentado por Redmon e Farhadi (2016) , baseado em uma arquitetura de redes neurais convolucionais (CNN), destaca-se por sua capacidade de realizar classificação em tempo real de forma unificada.

Diferentemente de métodos tradicionais, que dividem o processo de detecção em múltiplas etapas, como a identificação de regiões de interesse e posterior classificação, etapas necessárias quando aplicando uma CNN para detecção de objetos, o YOLO processa a imagem inteira em uma única passagem pela rede. Sua metodologia consiste em dividir a imagem de entrada em uma grade e, para cada célula dessa grade, prever caixas delimitadoras (bounding boxes). Cada uma dessas caixas possui uma probabilidade associada de conter um objeto e uma classe correspondente ao objeto detectado. Permitindo não apenas a identificação da posição do objeto detectado, mas também a classificação de múltiplos objetos diferentes na mesma imagem, como pode ser visto na imagem abaixo.

Figura 4 – Detecção do YOLO com Bounding boxes



Fonte: Redmon e Farhadi (2016)

2.5.1 ETAPAS DE FUNCIONAMENTO

As principais etapas do funcionamento do YOLO são:

- **Divisão da Imagem** - A imagem de entrada é dividida em uma grade $S \times S$. Cada célula dessa grade se torna então responsável por detectar objetos cuja posição esteja centralizada nela.
- **Previsão de Caixas e Classes** - Para cada célula, o modelo prevê múltiplas caixas delimitadoras, associadas a um valor de confiança que indica a probabilidade de conter um objeto. Além disso, são previstas as probabilidades das classes para cada objeto detectado.
- **Refinamento** - Após a detecção inicial, técnicas como a supressão de não-máximos (Non-Maximum Suppression - NMS) são aplicadas para eliminar caixas redundantes e manter apenas as mais relevantes. A NMS funciona selecionando a caixa com a maior pontuação de confiança e a mantendo, enquanto remove todas as outras caixas que possuem uma grande sobreposição com a caixa selecionada, utilizando a métrica Intersection over Union (IoU). A IoU é calculada como a razão entre a área de interseção das caixas e a área de união delas. Quando o valor de IoU entre uma caixa prevista e a caixa selecionada é maior que um limite predefinido (por exemplo, 0,5), a caixa prevista é descartada como redundante, ajudando a reduzir múltiplas detecções para o mesmo objeto.

2.6 BIBLIOTECA OPEN SOURCE COMPUTER VISION

O OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto utilizada para fins de visão computacional, principalmente em tempo real, e processamento de imagens. Inicialmente desenvolvida pela Intel em 1999, e mantida por uma comunidade de desenvolvedores, ela é projetada para fornecer ferramentas rápidas e otimizadas para analisar, modificar e compreender imagens e vídeos, além de incluir funcionalidades de aprendizado de máquina.

Uma das principais características do OpenCV é sua extensa coleção de algoritmos otimizados, incluindo técnicas de detecção de bordas, segmentação de imagens e rastreamento de objetos (KAEHLER; BRADSKI, 2016). Esses recursos são fundamentais para implementar sistemas robustos de visão computacional em tempo real.

2.7 RASTREAMENTO DE OBJETOS

O rastreamento de objetos é uma tarefa fundamental em sistemas de visão computacional, particularmente em aplicações que envolvem o monitoramento de movimento, como no controle de tráfego e segurança pública. O objetivo do rastreamento é acompanhar a posição de um objeto ao longo do tempo em uma sequência de imagens ou vídeos, permitindo que o sistema registre sua trajetória e identifique o movimento do objeto de interesse.

2.7.1 INTEGRAÇÃO DE DETECÇÃO E RASTREAMENTO

Algoritmos de detecção de objetos, como o YOLO, são amplamente utilizados devido à sua capacidade de realizar detecções em tempo real com alta precisão. No entanto, esses algoritmos operam de forma independente em cada quadro de vídeo, ou seja, não conseguem associar uma detecção em um quadro com a mesma detecção em frames subsequentes. Essa limitação pode levar a contagens repetidas de objetos que passam por uma área de interesse, comprometendo a análise em tempo real.

O rastreamento de objetos entra como um complemento à detecção, permitindo identificar veículos individuais ao longo de sua trajetória em um fluxo de vídeo. Essa funcionalidade reduz erros em aplicações como contagem de veículos, análise de densidade de tráfego e detecção de anomalias, como veículos parados ou mudanças abruptas de direção. Em resumo, o rastreamento integra detecções isoladas em uma análise contínua, garantindo que cada veículo seja identificado e monitorado consistentemente.

2.7.2 TÉCNICAS DE RASTREAMENTO DE OBJETOS

Os métodos de rastreamento de objetos podem ser classificados em duas categorias principais: rastreamento baseado em características de movimento e localização, e rastreamento baseado em aprendizado profundo. Ambas as abordagens têm sido amplamente utilizadas em sistemas de monitoramento de tráfego.

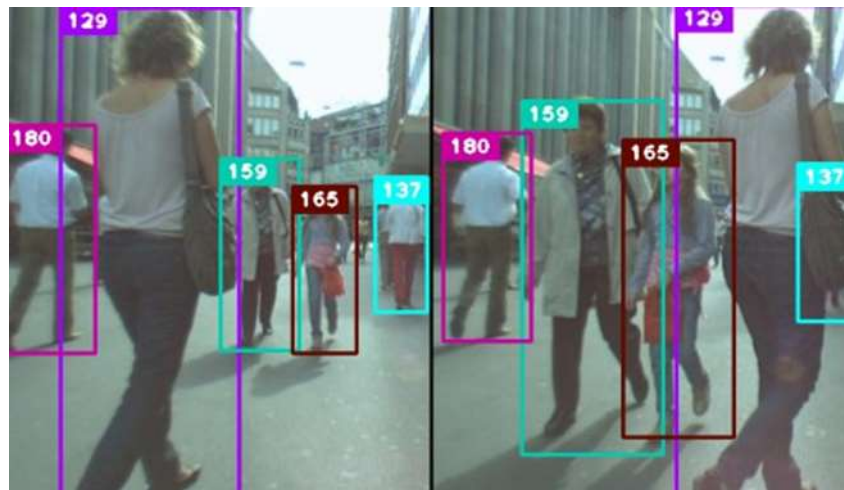
- **Modelos baseados em características de movimento e localização:**
- O filtro de Kalman (KALMAN, 1960) é uma técnica amplamente utilizada para prever a próxima posição de um objeto com base em seu estado anterior e no movimento observado. Apesar de ser eficiente e leve, o filtro de Kalman é limitado em cenários com múltiplos objetos ou movimentos não lineares.
- Métodos baseados em correlação espacial também são utilizados, onde a posição do objeto em um frame é associada a uma posição próxima no frame seguinte, considerando apenas dados de movimento.
- **Modelos baseados em aprendizado profundo:**
- O algoritmo DeepSORT (Simple Online and Realtime Tracking) (WOJKE; BEWLEY; PAULUS, 2017) combina características visuais e informações de posição para criar associações mais robustas. Ele utiliza embeddings de características visuais, extraídas por redes neurais, para diferenciar objetos semelhantes em movimento.
- Modelos mais avançados, como os baseados em Transformers, como o TrackFormer (MEINHARDT et al., 2021), utilizam mecanismos de atenção para capturar melhor os relacionamentos entre objetos ao longo do tempo, sendo promissores em cenários de alta complexidade.

2.7.3 O ALGORITMO DEEPSORT

O DeepSORT é amplamente utilizado em sistemas de rastreamento de múltiplos objetos devido à sua capacidade de reduzir identity switches, ou seja, erros de troca de identidade entre objetos rastreados. Ele funciona combinando as coordenadas espaciais fornecidas pelo YOLO com embeddings de características visuais do objeto, permitindo uma associação robusta, mesmo quando ocorrem mudanças temporárias de posição ou oclusões parciais. Além disso, o DeepSORT atribui IDs (Identificadores Únicos) a cada objeto identificado, garantindo que cada veículo detectado seja

monitorado de maneira consistente, prevenindo contagens duplicadas. Essa integração entre YOLO e DeepSORT garante um rastreamento eficiente e preciso.

Figura 5 – DeepSORT aplicado no dataset de desafio MOT, demonstrando rastreamento em condições desafiadoras.



Fonte: Wojke, Bewley e Paulus (2017)

2.7.4 O ALGORITMO STRONGSORT

Embora o DeepSORT tenha se tornado referência em rastreamento por detecção, suas premissas originais de 2017 já não exploravam totalmente os avanços recentes em detecção e re-identificação (ReID). Visando preencher essa lacuna, foi proposto o StrongSORT (DU et al., 2023), uma revisão do processo; ele mantém a filosofia “tracking-by-detection”, porém atualiza cada estágio do pipeline do detector aos módulos de associação para os padrões atuais de desempenho.

Principais melhorias

- **Detecção melhorada:** usa detectores atuais (por exemplo, YOLOv5) com aumentos de dados, gerando caixas de detecção mais precisas.
- **Re-identificação (ReID) mais forte:** troca o extrator de características por um modelo que cria descritores mais distintos para cada veículo.
- **Ajustes dinâmicos:** calibra o filtro de Kalman e os limites de associação para lidar melhor com cenas cheias, reduzindo trocas de ID.
- **Módulos extras:**
 1. AFLink: conecta trilhas curtas sem analisar aparência, adicionando apenas 1,7 ms por quadro.

2. GSI : suaviza trilhas e preenche detecções perdidas por interpolação gaussiana, com custo extra de cerca de 7,1 ms por quadro.

O StrongSORT foi avaliado na série pública de benchmarks **MOT** (Multiple Object Tracking), que reúne sequências de vídeo anotadas para comparar diferentes algoritmos de rastreamento. No conjunto MOT17, formado por cenas urbanas de densidade moderada, o algoritmo elevou o índice IDF1 (Critério o qual avalia a qualidade de detecções ao longo de um vídeo) do DeepSORT em mais de 4 pontos percentuais, chegando a **80.9%**. Já no MOT20, que traz vídeos ainda mais congestionados, a variante StrongSORT++ (com ALink e GSI) figura entre os melhores resultados registrados.

No rastreamento de veículos deste trabalho, o StrongSORT foi combinado com o YOLOv5 para:

- reduzir trocas de identidade (ID switches);
- recuperar trilhas perdidas em cruzamentos congestionados;
- manter a velocidade de processamento próxima ao tempo real.

2.7.5 APLICAÇÕES NO MONITORAMENTO DE VEÍCULOS

A integração de rastreamento de objetos com detectores como o YOLO tem aplicações diretas em sistemas de monitoramento e análise de tráfego. Algumas das principais aplicações incluem:

- Contagem precisa de veículos - O rastreamento evita que o mesmo veículo seja contado repetidamente ao longo de um vídeo ou em tempo real.
- Análise de fluxo - Permite monitorar a velocidade e direção dos veículos na área de interesse.
- Identificação de comportamentos anormais - Identificação de veículos em alta velocidade, realizando movimentos erráticos ou ilegais, paradas repentinas, entre outros comportamentos anômalos, que podem ser detectados e rastreados para análise.

2.7.6 CONSIDERAÇÕES FINAIS

O rastreamento de objetos desempenha um papel crucial em aplicações de monitoramento de tráfego, complementando algoritmos de detecção para garantir análises mais precisas e contínuas. Com o uso de algoritmos como o DeepSORT e avanços recentes baseados em aprendizado profundo, é possível melhorar significativamente a precisão e robustez de sistemas em cenários dinâmicos e de alta densidade.

2.8 SISTEMA DE DETECÇÃO DE VEÍCULOS

A detecção de veículos é o primeiro passo em sistemas de monitoramento de tráfego. Essa tarefa envolve localizar veículos em imagens ou vídeos e é frequentemente combinada com técnicas de rastreamento para monitorar os objetos ao longo do tempo. Sistemas modernos utilizam abordagens baseadas em redes neurais convolucionais (CNNs), que se destacam pela precisão e agilidade em cenários desafiadores, como mudanças de iluminação, oclusões, altas velocidades e tráfego intenso.

2.8.1 ARQUITETURA DE SISTEMAS DE DETECÇÃO DE VEÍCULOS

- Um sistema típico de detecção de veículos pode ser dividido em etapas ou módulos que operam de forma integrada, desde a captura de dados até a análise e geração de saídas úteis. Abaixo estão os componentes mais comuns em sistemas desse tipo.
- **Captura de Dados** - Imagens ou vídeos são capturadas por câmeras, geralmente estáticas, instaladas em pontos estratégicos sobre rodovias, cruzamentos, entre outros. Em casos específicos, câmeras móveis, seja sobre drones ou veículos também podem ser empregados.
- **Pré-Processamento de Imagens** - Antes de seguir para processamento dos detectores, a entrada passa por um pré-processamento digital, essa etapa envolve processos como normalização, redimensionamento e remoção de ruído.
- **Detecção de Veículos** - Nessa etapa os objetos de interesse, no caso, os veículos, são localizados e classificados. Ferramentas como o YOLO realizam a detecção em tempo real com base em redes neurais convolucionais.
- **Rastreamento de Veículos** - O rastreamento é uma ferramenta acompanhante da detecção e permite que o sistema monitore veículos em movimento ao longo do tempo. Algoritmos como StrongSORT usam informações de posição (bounding boxes) e embeddings visuais para associar detecções consecutivas, reduzindo erros como identity switches.
- **Análise e Saída dos Dados** - Após a detecção e rastreamento, os dados processados podem ser utilizados para várias finalidades, como contagem de veículos, estimativa de velocidade, geração de relatórios de densidade de tráfego, e até mesmo detecção de comportamentos ilegais.

2.8.2 SISTEMAS EXISTENTES

A gestão eficiente do tráfego urbano é um desafio enfrentado por diversas cidades ao redor do mundo, especialmente devido ao crescimento da frota de veículos e à complexidade das vias urbanas. Para mitigar esses problemas, diversos sistemas de monitoramento e controle de tráfego foram desenvolvidos ao longo dos anos, utilizando tecnologias como sensores, algoritmos adaptativos e processamento de dados em tempo real. Nesta seção, serão apresentados alguns dos principais sistemas existentes, destacando suas características, funcionalidades e aplicações práticas no contexto urbano.

2.8.2.1 SYDNEY COORDINATED ADAPTIVE TRAFFIC SYSTEM

Um dos sistemas mais amplamente utilizados para o gerenciamento adaptativo de tráfego é o SCATS (Sydney Coordinated Adaptive Traffic System). Desenvolvido na Austrália, o SCATS ajusta dinamicamente os tempos dos semáforos com base em dados coletados por sensores instalados nas vias, como fluxo, velocidade e ocupação. Esse sistema permite melhorar a fluidez do trânsito, reduzindo o tempo de viagem em até 28% e diminuindo as emissões de dióxido de carbono em cerca de 15% (Transport for NSW, s.d.). Além disso, o SCATS pode aumentar a capacidade de interseções em até 15%, otimizando o fluxo de veículos em tempo real (KUSTIJA; NUR, 2023).

Apesar dos benefícios, o SCATS enfrenta desafios relacionados ao custo inicial de implementação, à necessidade de infraestrutura tecnológica avançada e à manutenção contínua. Ainda assim, sua eficácia em melhorar a mobilidade urbana torna-o uma solução relevante para cidades que buscam inovações sustentáveis no gerenciamento de tráfego. A adoção desse sistema demonstra como a tecnologia pode ser aplicada de maneira estratégica para mitigar problemas urbanos complexos (KUSTIJA; NUR, 2023).

2.8.3 DESAFIOS E GARGALOS

Os sistemas de detecção de veículos enfrentam desafios técnicos e operacionais que impactam sua eficácia na gestão de tráfego urbano. Abaixo, alguns dos gargalos mais críticos, conforme apresentado por Dong e Cappuccio (2023):

- **Ofuscamento Solar e Clima Adverso** - O ofuscamento solar é um problema ambiental comum que prejudica imagens, reduzindo a eficácia de algoritmos de visão computacional. Ele pode ter impacto direto, quando o sol baixo causa brilho intenso nas câmeras, ou impacto indireto,

com a luz refletindo em superfícies molhadas ou brilhantes, dificultando a detecção. De forma semelhante, condições climáticas adversas, como chuva intensa ou neblina densa, afetam sensores como câmeras, prejudicando a percepção do ambiente e a detecção de objetos. Esses desafios reforçam a necessidade de avanços em tecnologias de sensores e algoritmos mais robustos para garantir a eficiência de sistemas de monitoramento em diversas condições climáticas.

Figura 6 – Condição de Ofuscamento Solar, perspectiva de um carro autônomo



Fonte: Dong e Cappuccio (2023)

- **Deteção de Falhas** - A detecção de falhas é um grande desafio para sistemas que utilizam visão computacional, especialmente em ambientes urbanos reais. Embora os testes de campo geralmente ocorram em regiões com clima favorável ou tráfego leve, as complexidades do ambiente real, como variações climáticas, tráfego intenso e comportamentos imprevisíveis, podem sobrecarregar os sistemas. A falha em detectar corretamente esses fatores pode comprometer a eficácia dos sistemas de monitoramento, tornando essencial o aprimoramento das tecnologias de percepção e detecção de falhas para garantir o bom funcionamento em condições adversas.
- **Volume de Dados, Capacidade de Armazenamento e Velocidade de Inferência em Tempo Real** - Em sistemas de semáforos inteligentes que utilizam visão computacional para identificar veículos e ajustar o tempo dos sinais, o volume de dados gerado pode ser significativo. Por

exemplo, um veículo autônomo pode gerar cerca de 4000 GB de dados por dia, o que equivale ao volume gerado por aproximadamente 3000 pessoas. Para sistemas de semáforos, isso implica na necessidade de plataformas de comunicação e armazenamento eficientes, além de um poder computacional elevado para processar os dados em tempo real, o que exige grande capacidade de processamento. Além disso, o consumo de energia dos equipamentos, como GPUs de alto desempenho, pode impactar a eficiência energética e a viabilidade do sistema em larga escala.

- **Custo Adicional** - Sistemas de semáforos inteligentes, assim como veículos autônomos, dependem de uma variedade de dispositivos a bordo, como câmeras e plataformas de computação. Esses sistemas exigem também dispositivos de comunicação e fontes de energia adicionais. O custo elevado desses sistemas é um desafio importante, já que, por exemplo, veículos autônomos podem custar até \$250.000, em comparação aos \$30.000 de um veículo convencional. Para os sistemas de semáforos inteligentes, é essencial buscar soluções que reduzam os custos operacionais, permitindo a adoção em larga escala e a viabilidade financeira dos projetos.

Apesar dos avanços significativos na tecnologia de detecção de veículos, os desafios e gargalos apresentados continuam a representar obstáculos para a implementação eficiente e em larga escala de sistemas de semáforos inteligentes. A superação desses problemas exige inovações contínuas em áreas como processamento de dados em tempo real, robustez contra condições ambientais adversas e otimização de custos. A busca por soluções que atendam a esses requisitos é fundamental para a evolução dos sistemas de monitoramento de tráfego e para a implementação segura e eficiente de infraestruturas inteligentes em cidades.

2.8.4 CONSIDERAÇÕES FINAIS

Em conclusão, os sistemas de detecção de veículos e semáforos inteligentes representam avanços significativos no gerenciamento do tráfego urbano, com o potencial de melhorar a fluidez do trânsito e reduzir o impacto ambiental. Sistemas como o SCATS exemplificam como a adaptação em tempo real das infraestruturas de tráfego pode otimizar a circulação urbana. No entanto, para alcançar uma implementação eficiente e em larga escala, ainda é necessário superar desafios tecnológicos, como a precisão na detecção sob condições adversas e a gestão de grandes volumes de dados.

2.9 MÉTRICAS DE AVALIAÇÃO PARA DETECÇÃO E RASTREAMENTO

Para avaliar a eficácia dos sistemas de detecção e rastreamento de objetos, como aqueles aplicados ao monitoramento de tráfego, é essencial utilizar métricas consolidadas e amplamente aceitas na área de visão computacional. Estas métricas ajudam a medir não apenas a acurácia das predições realizadas pelo modelo, mas também a consistência e a eficiência dos algoritmos empregados. Abaixo estão descritas as principais métricas utilizadas na detecção e rastreamento de objetos conforme demonstrado por Roy e Bhaduri (2021) :

2.9.1 PRECISÃO

A precisão mede a capacidade do modelo de evitar falsos positivos. Em outras palavras, é a proporção de detecções corretas em relação ao total de detecções realizadas. A precisão é calculada da seguinte forma:

$$\text{Precisão} = \frac{TP}{TP + FP}$$

Onde:

- **TP (True Positives):** Verdadeiros Positivos, ou seja, as detecções corretas feitas pelo modelo.
- **FP (False Positives):** Falsos Positivos, ou seja, detecções incorretas onde o modelo classificou um objeto onde não havia.

2.9.2 RECALL

A revocação é uma métrica que avalia a capacidade do modelo de detectar todos os objetos relevantes, ou seja, é a proporção de detecções corretas em relação ao total de objetos reais no conjunto de dados. A fórmula para o cálculo da revocação é:

$$\text{Revocação} = \frac{TP}{TP + FN}$$

Onde:

- **FN (False Negatives):** Falsos Negativos, ou seja, objetos que o modelo falhou em detectar.

2.9.3 F1-SCORE

O F1-Score é uma métrica combinada que leva em consideração tanto a precisão quanto a revocação, proporcionando um equilíbrio entre essas duas métricas. O F1-Score é útil em cenários onde é importante otimizar tanto a capacidade do modelo de identificar objetos corretamente (precisão) quanto a sua capacidade de detectar todos os objetos relevantes (revocação). A fórmula do F1-Score é:

$$\text{F1-Score} = \frac{2 \times \text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

2.9.4 MEAN AVERAGE PRECISION (MAP)

O mAP (Mean Average Precision) é uma das métricas mais utilizadas em tarefas de detecção de objetos. Ele calcula a precisão média, levando em consideração diferentes limiares de sobreposição (IoU - Intersection over Union) entre as caixas preditas e reais. A principal vantagem do mAP é que ele avalia o modelo com diferentes níveis de tolerância para a sobreposição entre as caixas, tornando a avaliação mais robusta.

A fórmula do mAP é dada por:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

- **mAP@0.5:** Média da precisão calculada considerando um limiar de sobreposição de 50%. Ou seja, as predições são consideradas corretas se a área de interseção entre a caixa predita e a caixa real for superior a 50%.
- **mAP@0.5:0.95:** Média da precisão calculada considerando diferentes limiares de sobreposição, variando de 0.5 a 0.95, com incrementos de 0.05. Essa métrica fornece uma avaliação mais rigorosa da performance do modelo, considerando diferentes limiares de sobreposição.

2.9.5 TAXA DE FALSOS POSITIVOS (FALSE POSITIVE RATE - FPR)

A taxa de falsos positivos mede a probabilidade de o modelo classificar incorretamente um objeto que não existe. Ela é dada pela relação entre o número de falsos positivos e o número total de objetos negativos (ou seja, onde não há objeto):

$$FPR = \frac{FP}{FP + TN}$$

Onde:

- TN (True Negatives): Verdadeiros Negativos, ou seja, o número de vezes em que o modelo corretamente identificou a ausência de um objeto.

2.9.6 IOU (INTERSECTION OVER UNION)

O IoU é uma métrica utilizada para avaliar a sobreposição entre as caixas delimitadas preditas e as caixas reais. O IoU é calculado pela razão entre a área de interseção das caixas e a área de união das mesmas:

$$IoU = \frac{\text{Área de Interseção}}{\text{Área de União}}$$

Quanto maior o IoU, mais precisa é a detecção do modelo. Em tarefas de detecção de objetos, um IoU superior a 0.5 é frequentemente considerado uma detecção válida.

2.10 DATASETS PARA TREINAMENTO E VALIDAÇÃO

A seguir, destacam-se alguns dos principais datasets utilizados no campo da visão computacional, com ênfase em suas características e aplicações.

2.10.1 KITTI VISION BENCHMARK SUITE

O KITTI é amplamente reconhecido como um dos mais relevantes datasets para tarefas relacionadas à visão computacional em veículos autônomos. Desenvolvido por Geiger, Lenz e Urtasun (2012), ele apresenta imagens capturadas por câmeras embarcadas em veículos, cobrindo cenários urbanos, rodoviários e rurais. Além disso, o KITTI oferece anotações detalhadas para detecção de objetos, rastreamento, segmentação semântica e odometria visual.

2.10.2 COCO (COMMON OBJECTS IN CONTEXT)

O COCO (Common Objects in Context) é um dataset versátil e amplamente utilizado, especialmente para tarefas de detecção de objetos e segmentação de imagens. Embora não seja focado exclusivamente no monitoramento de tráfego, ele inclui classes de veículos como carros e caminhões, sendo útil para aplicações que demandam diversidade de cenários e objetos. Lin et al. (2014) destacam

que o COCO fornece anotações em bounding boxes e máscaras segmentadas, atendendo a uma ampla gama de tarefas de visão computacional.

2.10.3 MOTCHALLENGE

Desenvolvido para avaliação de algoritmos de rastreamento de múltiplos objetos (MOT), o MOTChallenge é um benchmark consolidado na área. Desenvolvido por Dendorfer et al. (2021), ele inclui vídeos em cenários desafiadores, como tráfego intenso e condições de iluminação variáveis. O conjunto de dados fornece anotações detalhadas por frame, com IDs únicos para cada objeto, o que o torna essencial para avaliação de rastreamento.

2.10.4 MIO-TCD (TRAFFIC CAMERA DATASET)

O MIO-TCD (Traffic Camera Dataset) é um conjunto de dados projetado especificamente para aplicações de visão computacional no contexto de monitoramento de tráfego. A versão utilizada neste trabalho, o MIO-TCD Localization, é composta por aproximadamente 11 mil imagens urbanas e rodoviárias, anotadas com bounding boxes para diferentes categorias de veículos, como carros, caminhões, ônibus, motocicletas e bicicletas. Um dos diferenciais deste dataset é o ponto de vista elevado das câmeras, muitas vezes instaladas em semáforos ou postes de sinalização, o que o torna especialmente útil para treinamentos de detecção de veículos em semáforos e interseções urbanas. Essa perspectiva simula condições reais de sistemas de detecção, alinhando-se diretamente com os objetivos deste trabalho. Conforme descrito por Luo et al. (2018), o MIO-TCD foi desenvolvido com o intuito de fomentar pesquisas em classificação e detecção de veículos em cenários reais de trânsito.

2.11 CONSIDERAÇÕES FINAIS

Com base na revisão da literatura e nos conceitos abordados neste capítulo, fica evidente a importância da visão computacional na detecção e rastreamento de veículos em ambientes urbanos. O avanço das redes neurais convolucionais e dos algoritmos de aprendizado profundo permitiu a criação de sistemas cada vez mais precisos e eficientes para essa finalidade.

A visão computacional tem-se consolidado como uma ferramenta fundamental para a automação e otimização de sistemas de monitoramento de tráfego, possibilitando a detecção e o rastreamento de veículos em tempo real. O presente capítulo abordou os principais conceitos teóricos relacionados a essa área, destacando o processamento digital de imagens, redes neurais convolucionais e algoritmos

avançados, como YOLO e StrongSORT, que desempenham um papel essencial na identificação eficiente de objetos em ambientes dinâmicos.

Apesar dos avanços significativos nessa área, desafios como o processamento em tempo real, a necessidade de hardware de alto desempenho e as dificuldades impostas por variações ambientais ainda representam obstáculos à implementação prática desses sistemas em larga escala. No entanto, as tendências atuais indicam um crescimento contínuo no desenvolvimento de modelos mais eficientes e otimizados, reforçando a importância de pesquisas que aprimorem a precisão e a robustez desses algoritmos.

Dessa forma, a fundamentação teórica apresentada neste capítulo serve como base para a modelagem proposta. A partir desses conceitos, a próxima etapa consiste na definição da metodologia a ser empregada, visando validar as abordagens discutidas.

3 METODOLOGIA

A primeira etapa do desenvolvimento deste trabalho consistiu na realização de uma revisão bibliográfica, com o objetivo de fundamentar teoricamente a proposta e compreender as abordagens utilizadas na literatura sobre visão computacional. Foram analisados artigos científicos, dissertações e livros sobre processamento digital de imagens, redes neurais convolucionais e algoritmos de rastreamento. Esta revisão permitiu a identificação de ferramentas relevantes e metodologias utilizadas em sistemas de detecção e rastreamento de veículos, possibilitando a formulação de um modelo adequado e uma estruturação do processo metodológico.

3.1 DEFINIÇÃO DOS REQUISITOS DE FUNCIONAMENTO

Nesta etapa, definiram-se os objetivos e os requisitos essenciais para o funcionamento adequado do modelo. Foram estabelecidos os seguintes aspectos fundamentais:

- Detectar veículos em vídeo de forma confiável, com base em imagens obtidas de câmeras ou bases de dados públicas.
- Conseguir reconhecer e acompanhar o movimento de veículos entre quadros de um vídeo.
- Manter um tempo de resposta aceitável, sem exigir computadores muito potentes, considerando o uso em aplicações práticas futuras.

Esses critérios foram definidos com base nos objetivos do projeto, nas ferramentas disponíveis e nas necessidades de aplicação do sistema em contextos reais.

3.2 AQUISIÇÃO DE IMAGENS

A obtenção de imagens e vídeos foi essencial para o treinamento e validação do sistema de detecção e rastreamento de veículos. Foram utilizadas bases de dados públicas amplamente reconhecidas na área de visão computacional, como o MIO-TCD e o COCO, que fornecem imagens e vídeos anotados com informações sobre objetos em cena, incluindo veículos de diferentes tipos, tamanhos e posições. Essas bases possibilitaram a realização de treinos com conteúdo já estruturado, facilitando a evolução e avaliação da acurácia da detecção.

Além dessas bases, também foram utilizados vídeos da plataforma Pixabay, que oferece conteúdo gratuito e isento de royalties. Os vídeos escolhidos apresentam cenas de tráfego urbano capturadas por câmeras fixas, com diferentes ângulos de visão e variações de iluminação. Essa diversidade de condições visuais foi importante para testar a robustez do sistema em contextos mais próximos da aplicação real.

A combinação entre bases anotadas e vídeos livres permitiu equilibrar praticidade e realismo na validação do modelo, possibilitando testar o funcionamento do sistema tanto em imagens controladas quanto em cenas reais não rotuladas.

3.3 PREPARAÇÃO DOS DADOS

A preparação dos dados foi realizada com o objetivo de garantir a compatibilidade com a arquitetura YOLOv5. Inicialmente, os conjuntos COCO e MIO-TCD foram convertidos para o formato de anotações utilizado pelo YOLO, que requer coordenadas normalizadas. Essa conversão foi feita por meio de um script desenvolvido em Python, com filtragem das anotações para manter apenas as instâncias da classe “Car”.

Após a conversão, os dados foram divididos em subconjuntos de treino e validação na proporção de 80/20, organizados em diretórios hierárquicos específicos e referenciados por um arquivo de configuração YAML, contendo os nomes das classes e os caminhos relativos para imagens e rótulos.

3.4 PRÉ-PROCESSAMENTO DE IMAGENS

O pré-processamento de imagens é uma etapa crucial para garantir que os dados de entrada estejam adequadamente estruturados e compatíveis com a arquitetura da rede YOLOv5. Embora grande parte desse processo seja automatizada, ele inclui transformações essenciais para o bom funcionamento do modelo de detecção.

No contexto deste trabalho, o carregamento e preparo das imagens foram gerenciados pelo componente `DataLoader`, responsável por organizar os dados em lotes (batches) e aplicar, em tempo real, as transformações necessárias para alimentar a rede neural.

Esse processo inclui a leitura dos arquivos de imagem no formato RGB (três canais), o redimensionamento para a dimensão padrão de 640×640 pixels e a normalização dos valores de pixel para a faixa $[0, 1]$, por meio da divisão dos valores originais (0 a 255) por 255. Esse procedimento tem como objetivo melhorar a estabilidade do treinamento, assegurando que todos os valores de entrada estejam na mesma escala.

Além disso, o `DataLoader` é responsável por aplicar técnicas de aumento de dados (data augmentation) em tempo real. Essas técnicas visam aumentar a variabilidade do conjunto de dados, o que pode ajudar a reduzir o risco de overfitting. No caso deste projeto, as técnicas de aumento de dados incluem:

- **Mosaico (mosaic):** Combinação aleatória de quatro imagens em uma única composição.
- **Flip horizontal (fliplr):** Inversão horizontal de imagens com probabilidade de 50%.
- **Transformações de cor (hsv_h, hsv_s, hsv_v):** Variações no matiz, saturação e brilho da imagem:
 - **hsv_h:** Variação de 1,5% no matiz.
 - **hsv_s:** Variação de 70% na saturação.
 - **hsv_v:** Variação de 40% no brilho.
- **Translação e escala (translate, scale):** Deslocamentos e redimensionamentos leves nas imagens:
 - **translate:** Deslocamento de até 10% nas imagens.
 - **scale:** Redimensionamento entre 90% e 110% do tamanho original.
- **Mixup:** Combinação de duas imagens, com 10% de chance de aplicação.
- **Copy-Paste:** Técnica de copiar segmentos de uma imagem e colá-los em outra, com 10% de chance de aplicação.

Os hiperparâmetros responsáveis pelas transformações de entrada foram ajustados conforme as recomendações da documentação oficial do YOLOv5. O valor de mosaic foi mantido em 1.0, assegurando que o mosaico seja sempre aplicado, enquanto o Mixup e Copy-Paste foram ativados com 10% de probabilidade, buscando aumentar a diversidade das amostras. As transformações no espaço HSV foram ampliadas para uma maior variação nas cores e brilhos das imagens, com o objetivo de melhorar a generalização do modelo.

Essas configurações contribuíram para aumentar a robustez do modelo, preservando a integridade estrutural das imagens e ao mesmo tempo ampliando a variabilidade dos dados.

3.4.1 TREINAMENTO DA REDE YOLO

O treinamento do modelo YOLOv5 foi realizado em duas etapas principais. Na primeira, utilizou-se um subconjunto filtrado do dataset COCO, contendo apenas anotações da classe “car”. Essa abordagem simplificada permitiu focar inicialmente em um único tipo de objeto, reduzindo a complexidade do treinamento e permitindo uma análise mais objetiva do comportamento da rede. Essa configuração também facilitou o monitoramento das métricas de desempenho e o ajuste dos hiperparâmetros do processo de treinamento, tais como a taxa de aprendizado, o número de épocas e o tamanho do batch. Esses hiperparâmetros, definidos manualmente, influenciam diretamente na velocidade de convergência e na capacidade do modelo de generalizar corretamente.

A partir dos pesos obtidos nesse primeiro experimento, foi conduzido um segundo treinamento com o conjunto MIO-TCD, também restrito à classe “car”, utilizando a técnica de transferência de aprendizado. A reutilização dos pesos treinados no COCO proporcionou ao modelo uma base de representações visuais previamente aprendidas, permitindo um processo de convergência mais rápido e robusto ao longo das 150 épocas de ajuste com o MIO-TCD. Durante essa fase, as métricas de precisão, recall e mAP foram monitoradas continuamente.

O treinamento foi realizado em uma estação equipada com GPU NVIDIA RTX 2080 Ti (11 GB) e 32 GB de RAM, usando o otimizador SGD, taxa de aprendizado inicial (lr_0) de 0,01 e batch size de 16. A etapa inicial em COCO foi composta por 66 épocas, com duração média de aproximadamente 12 h por época (≈ 792 h no total). Em seguida, o refinamento em MIO-TCD envolveu 150 épocas, com cerca de 1 h por época (≈ 150 h no total), totalizando aproximadamente 942 h de treinamento contínuo, ou 39.25 dias corridos.

3.5 DETECÇÃO DE VEÍCULOS

A etapa de detecção consistiu na aplicação do modelo YOLOv5, previamente treinado com diferentes conjuntos de dados: COCO e MIO-TCD. Cada um desses datasets contribuiu com características específicas que reforçaram a capacidade de generalização do modelo. O COCO foi utilizado inicialmente para estabelecer uma base robusta de reconhecimento de objetos e o MIO-TCD forneceu um conjunto mais próximo do ponto de vista das câmeras de tráfego fixas, semelhante ao cenário analisado neste trabalho.

O modelo foi executado de forma offline, fazendo a detecção sobre vídeos previamente gravados, quadro a quadro. Os frames foram redimensionados e ajustados ao padrão do modelo (640×640 pixels), utilizando o esquema de letterbox para preservar a proporção original das imagens. Em seguida, cada imagem foi passada pela rede neural, gerando um conjunto de predições contendo as bounding boxes e respectivos scores de confiança.

As detecções foram armazenadas em arquivos no formato YOLO, com as coordenadas normalizadas para cada objeto identificado. Para garantir a qualidade dos dados, foi aplicado um filtro baseado no score de confiança, descartando detecções abaixo de um limiar definido empiricamente. Essa saída foi utilizada posteriormente como entrada para o módulo de rastreamento.

A abordagem modular separando detecção e rastreamento permitiu uma análise mais controlada dos resultados, além de facilitar ajustes em cada etapa de forma independente. Essa estratégia também tornou possível a comparação entre os diferentes modelos treinados com cada base de dados, observando suas respectivas performances qualitativas sobre os mesmos vídeos de teste.

3.6 RASTREAMENTO DE VEÍCULOS

Para implementar essa funcionalidade, foi adotado o repositório BoxMot (BROS-TRÖM, 2022), em sua versão 8.0, que integra o detector YOLOv5 ao algoritmo de rastreamento StrongSORT. Essa escolha se deu em função da compatibilidade direta com modelos treinados em YOLOv5 e do bom desempenho do StrongSORT em ambientes urbanos densos, com múltiplos objetos e visibilidade intermitente.

Entretanto, como a maioria dos conjuntos de dados utilizados para detecção não oferece suporte nativo à avaliação quantitativa de rastreamento, por conta de terem imagens atemporais e sem identificação por ID, foi necessário estruturar um ambiente de validação específico. Para isso, empregou-se o conjunto KITTI Tracking, cuja sequência foi selecionada por conter um cenário urbano representativo e anotações completas de identidade por objeto. Como esse conjunto é originalmente disponibilizado em imagens estáticas, foi realizada a conversão das imagens em um único arquivo de vídeo contínuo, respeitando a ordem cronológica e a resolução nativa de 1242 × 374 pixels.

As anotações do KITTI foram convertidas para o formato de entrada utilizado pelo YOLOv5, seguindo a estrutura [classe, x, y, w, h] com coordenadas normalizadas. Em seguida, aplicou-se um filtro para manter exclusivamente os objetos rotulados como Car, descartando demais categorias (como pedestres, ciclistas e vans), de modo a alinhar o processo de detecção e rastreamento com o escopo

definido neste trabalho. Embora o rastreamento em si fosse funcional mesmo com modelos treinados em outros domínios, optou-se por realizar um processo de fine-tuning com os dados convertidos da sequência selecionada, com o objetivo exclusivo de garantir que a avaliação quantitativa do rastreamento não fosse prejudicada por falhas de detecção causadas por diferença de domínio ou perspectiva. Essa medida buscou assegurar que as métricas reportadas refletissem, de forma mais justa, a capacidade do StrongSORT em manter identidades ao longo do tempo, e não fossem afetadas por limitações pontuais do detector.

A seguir, descreve-se o pipeline completo utilizado no processo de rastreamento:

1. **Leitura do vídeo:** O sistema carrega o vídeo de entrada a partir do caminho definido nos argumentos de execução. Caso o parâmetro especifique uma câmera ou um fluxo online, o sistema alterna automaticamente para o modo de captura em tempo real.
2. **Pré-processamento dos quadros:** Cada quadro do vídeo é redimensionado para a resolução especificada (neste caso, 1242×374 pixels), normalizado e convertido para o formato de entrada exigido pela arquitetura do YOLOv5. Essa etapa inclui a padronização dos valores de pixel para o intervalo $[0, 1]$ e a conversão para tensor em GPU.
3. **Inferência com YOLOv5:** O modelo processa cada quadro e retorna uma lista de detecções com as caixas delimitadoras no formato $[x1, y1, x2, y2, \text{confiança}, \text{classe}]$, representando as coordenadas dos objetos detectados, o nível de confiança da predição e o índice da classe correspondente. Como o treinamento foi realizado exclusivamente para a classe Car, todas as detecções retornam com esse rótulo.
4. **Supressão de não-máximos (NMS):** Após a inferência, é aplicada a técnica de supressão de não-máximos (Non-Maximum Suppression), utilizando os valores de confidence threshold e IoU threshold definidos no momento da execução. Essa etapa garante que, em caso de múltiplas detecções sobrepostas de um mesmo objeto, apenas a mais confiável seja mantida.
5. **Conversão das detecções:** As caixas delimitadoras resultantes do NMS são redimensionadas de volta para a escala original do quadro e convertidas para o formato $[x, y, w, h]$, necessário para o funcionamento do algoritmo de rastreamento. As informações de confiança e classe são mantidas.
6. **Rastreamento com StrongSORT:** As detecções convertidas são encaminhadas para o algoritmo StrongSORT, que aplica uma associação baseada em predição de movimento (filtro de Kalman) e em similaridade visual (Re-Identification). Cada objeto rastreado recebe um identificador único, o qual é preservado ao longo dos quadros subsequentes, mesmo em situações de oclusão parcial ou temporária.
7. **Visualização e exportação dos resultados:** A cada quadro, os resultados são sobrepostos à imagem original por meio de anotações visuais contendo as caixas delimitadoras, identificadores e, opcionalmente, o nível de confiança. O sistema permite a exportação simultânea dos resultados em dois formatos: um vídeo com as anotações e arquivos texto no padrão MOT Challenge, compatíveis com ferramentas de avaliação quantitativa de rastreamento.

3.7 SIMULAÇÃO DE CONTROLE DE TRÁFEGO COM SEMÁFORO INTELIGENTE (SUMO)

Com o objetivo de demonstrar a aplicabilidade dos dados obtidos pelo sistema de detecção e rastreamento desenvolvido neste trabalho, foi realizada uma simulação no Simulation of Urban Mobility (SUMO). O SUMO foi escolhido por ser uma plataforma de código aberto amplamente empregada em pesquisas de tráfego urbano e por disponibilizar a interface TraCI, que permite controlar tempos semafóricos em tempo real, via Python.

A simulação teve como foco a análise do impacto que um sistema de detecção de veículos poderia ter na lógica de controle de semáforos em um cruzamento urbano. Embora a integração direta entre a saída do YOLOv5 e o simulador SUMO não tenha sido realizada, uma vez que o modelo foi treinado para imagens reais e o SUMO gera imagens simuladas com características visuais distintas, a lógica de atuação sobre os semáforos foi construída com base nos dados que seriam extraídos por um sistema real de detecção. Dessa forma, buscou-se simular sua aplicação prática no ambiente urbano modelado.

Modelou-se um cruzamento de quatro abordagens bidirecionais (Figura 7) e limite de velocidade de 50 km/h sem faixas exclusivas para conversão; com um fluxo de 100 veículos para cada sentido e direção

Figura 7 – Geometria do cruzamento bidirecional modelado no SUMO (4 abordagens, 1 faixa por sentido)



Fonte: Elaboração do autor

Com as seguintes medidas

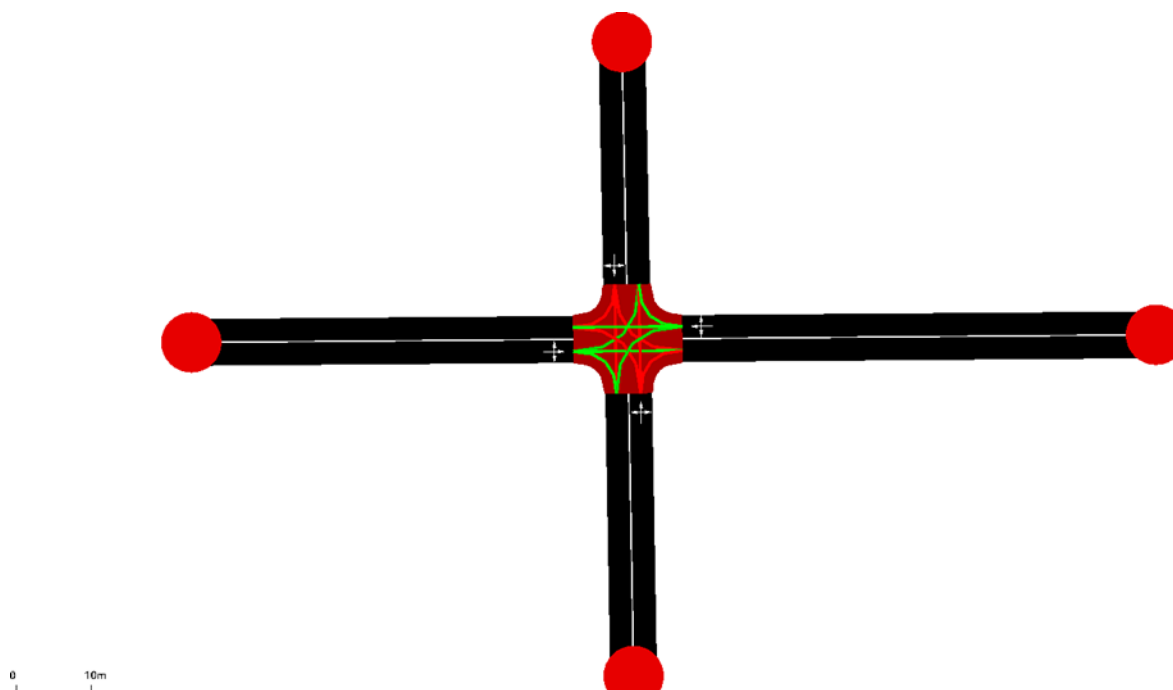
Tabela 1 – Comprimento das aproximações do cruzamento

Sentido Leste–Oeste		Sentido Norte–Sul	
Faixa	L (m)	Faixa	L (m)
Leste → Semáf.	62.73	Sul → Semáf.	37.38
Semáf. → Leste	62.73	Semáf. → Sul	37.37
Oeste → Semáf.	50.56	Norte → Semáf.	32.06
Semáf. → Oeste	50.56	Semáf. → Norte	32.06

Foram injetados 1.200 veículos do tipo passenger (com 5 m de comprimento), distribuídos de forma uniforme ao longo de 3600 s. Isso corresponde a uma taxa média de 400 veh/h por eixo (100 veh/h em cada aproximação), volume moderado o suficiente para evitar a saturação do cruzamento e, assim, evidenciar diferenças entre os dois regimes de controle.

Foram então comparados dois regimes de sinalização:

- **Controle Fixo:** tempo de semáforo constante com offset de 0 (Alternância Rígida).



Fonte: Elaboração do autor

Figura 8 – Configuração controle fixo do Semáforo

Tabela 2 – Detalhamento das fases do semáforo fixo

Fase	Descrição	Duração (s)
0	Verde para leste-oeste (reta e conversão)	42,00
1	Amarelo para leste-oeste	3,00
2	Verde para norte-sul (reta e conversão)	42,00
3	Amarelo para norte-sul	3,00
Total de ciclo		90,00

Nota: A configuração de semáforo fixo apresentada na Figura 8 e tabela 2 segue o padrão de planos de tempo fixo comum em interseções urbanas, com duração total de ciclo de 90s (verde 42s, amarelo 3s, verde oposto 42s, amarelo 3s), conforme diretrizes de sinalização viária. Utilizada como referência para quantificar os benefícios do controle adaptativo baseado em visão computacional.

- **Controle Adaptativo (Max-Pressure)** — implementado em Python–TraCI. Fundamenta-se no princípio de pressão de interseção (VARAIYA, 2013):

$$\Delta q_{ij}(t) = q_i(t) - q_j(t), \quad (3.1)$$

onde q_i é o comprimento de fila na via i atualmente verde, e q_j na via j atualmente vermelha. Para cada fase possível p , que agrupa movimentos (i, j) , define-se sua pressão total

$$P_p(t) = \sum_{(i,j) \in p} \Delta q_{ij}(t). \quad (3.2)$$

A cada janela $\Delta t = 0.5$ s, escolhe-se a fase que maximiza a pressão:

$$p^*(t) = \arg \max_p P_p(t). \quad (3.3)$$

Na prática, o script executa em cada Δt :

1. Leitura do comprimento de fila de cada eixo via `getJamLengthVehicle` $\times 5$ m;
2. Calcula Δq pela Eq. (3.1) e P_p pela Eq. (3.2);
3. aplica os critérios da Tabela 3 para determinar se encerra a fase verde corrente.

Tabela 3 – Parâmetros do algoritmo Max-Pressure

Parâmetro	Valor
Janela de decisão (Δt)	0.5 s
Mínimo de verde (GREEN_MIN)	7 s
Máximo de verde (GREEN_MAX)	40 s
Limite de pressão (PRESS_TH)	5 m

Fonte: Elaboração do autor.

A fase verde é encerrada 2 s após:

- (a) $\Delta q > \text{PRESS_TH}$ e $t_{\text{verde}} \geq \text{GREEN_MIN}$; ou
- (b) $t_{\text{verde}} = \text{GREEN_MAX}$.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos com a aplicação do modelo YOLOv5 para detecção de veículos, associado ao algoritmo Strong SORT para rastreamento. Também é discutida uma simulação complementar realizada no SUMO (Simulation of Urban Mobility), utilizada para demonstrar o potencial de aplicação de um sistema de controle de tráfego inteligente com detecção e rastreamento de veículos.

4.1 RESULTADOS

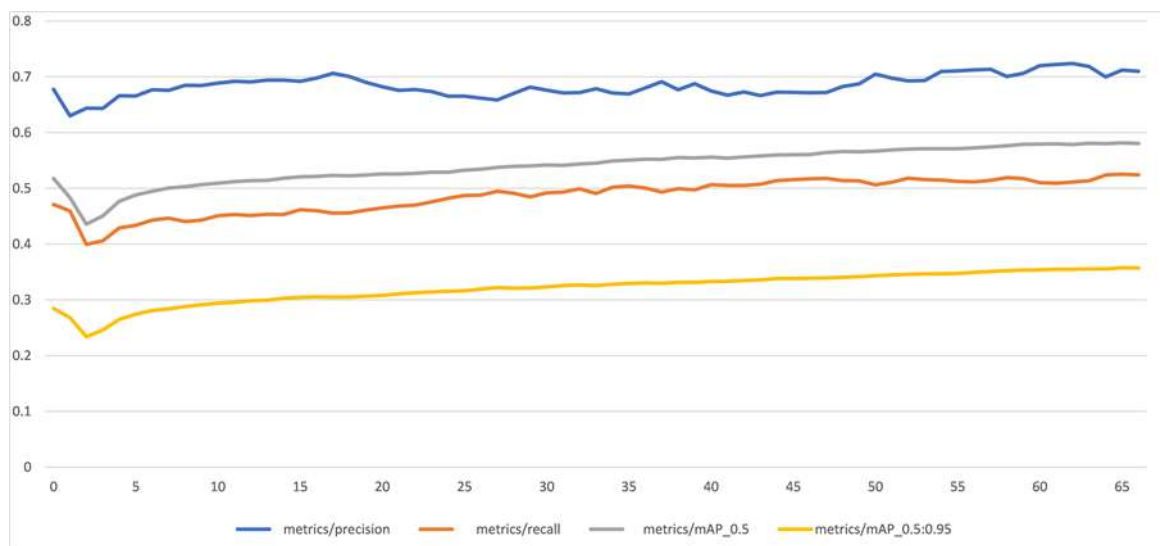
4.1.1 RESULTADOS DO TREINAMENTO

O processo de treinamento do modelo YOLOv5 foi conduzido em duas etapas principais, com o objetivo de construir um detector de veículos eficiente para ambientes urbanos.

Na primeira etapa, foi utilizado o dataset COCO, filtrado para conter exclusivamente a classe “car”. O treinamento foi iniciado com base nos pesos pré-treinados do modelo yolov5s.pt, fornecido pela própria equipe do YOLOv5, e ajustado ao longo de 66 épocas, até que as métricas de desempenho apresentassem estabilização e convergência. Essa fase teve como finalidade especializar o modelo em detectar veículos de forma geral, aproveitando a diversidade de contextos e perspectivas oferecida pelo COCO.

A Figura 9 apresenta a evolução das principais métricas ao longo do treinamento com o dataset COCO. Observa-se que o modelo alcançou rapidamente um desempenho consistente, mantendo bons níveis de precisão e revocação durante as 66 épocas.

Figura 9 – Convergência das Métricas de Avaliação Durante o Treinamento com o COCO

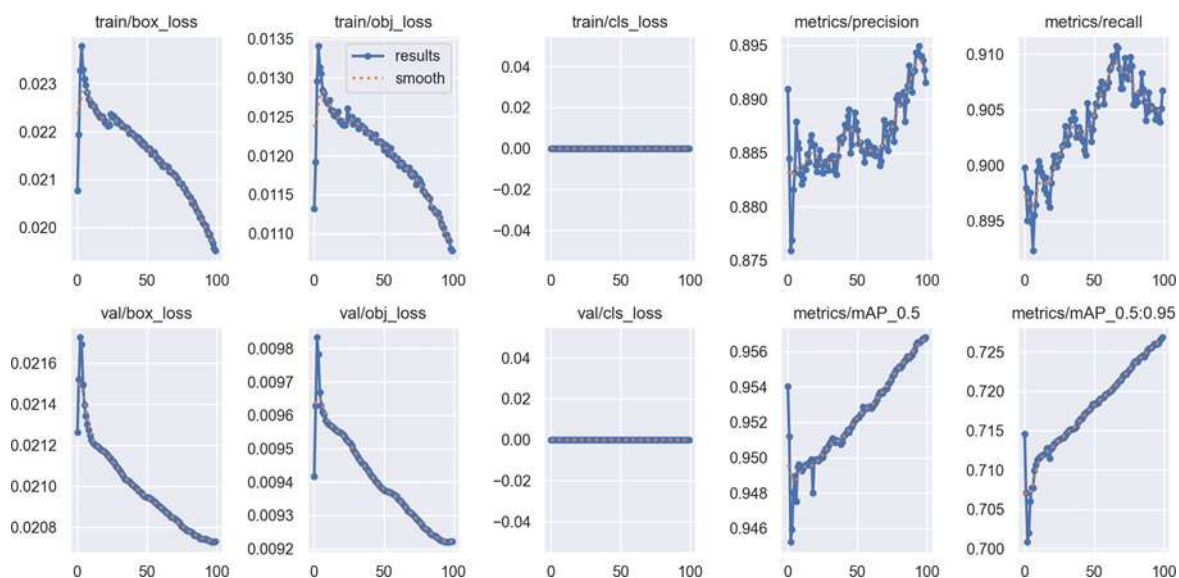


Fonte: Elaboração do autor

Em seguida, o modelo foi refinado com o dataset MIO-TCD Localization, composto por imagens capturadas por câmeras fixas posicionadas em postes e semáforos. O processo de refinamento foi conduzido ao longo de 150 épocas, divididas em duas rodadas sucessivas (50 e 100 épocas, respectivamente), utilizando como base os pesos obtidos na etapa anterior. O objetivo foi adaptar progressivamente a rede ao tipo específico de cenário urbano presente em aplicações reais de monitoramento de tráfego, mantendo a resolução de entrada de 640×640 pixels e o tamanho de lote de 16 imagens.

Durante esse processo, o modelo apresentou convergência estável, com redução contínua das funções de perda. A Figura 10, apresenta a evolução das principais métricas ao longo das últimas 100 épocas de treinamento.

Figura 10 – Evolução das métricas durante o treinamento do YOLOv5 com o dataset MIO-TCD



Fonte: Gerado pelo autor por meio do repositório YOLOv5

As curvas de perda (box_loss e obj_loss) demonstram uma queda progressiva nos conjuntos de treino e validação. A perda de classificação (cls_loss) permaneceu nula, conforme esperado, já que apenas uma classe foi utilizada em ambas as etapas.

As métricas de avaliação também indicaram desempenho significativo:

- **Precisão (Precision):** aproximadamente 90%.
- **Revocação (Recall):** cerca de 91%.
- **mAP@0.5:** atingiu 95,6%.
- **mAP@0.5:0.95:** alcançou 72,6%.

Esses resultados mostram que o modelo YOLOv5 foi capaz de se adaptar progressivamente aos diferentes domínios dos dados utilizados, resultando em um detector eficiente e com alta capacidade de generalização para a tarefa de detecção de veículos.

4.1.2 RESULTADOS DA VALIDAÇÃO DO MODELO

Após o término do treinamento, o modelo foi avaliado utilizando o conjunto de validação do dataset MIO-TCD Localization, correspondente a aproximadamente 20% das imagens originalmente convertidas e anotadas. Essa etapa teve como objetivo verificar a capacidade de generalização do detector em dados não utilizados durante o treinamento, simulando um cenário mais próximo da aplicação real.

A avaliação foi realizada com o script val.py, fornecido pelo repositório YOLOv5, utilizando os pesos finais obtidos após as 150 épocas de refinamento. O conjunto de validação incluiu 11.000 imagens, das quais 23.071 instâncias da classe “car” foram corretamente identificadas, como indicado na Tabela 4.

Tabela 4 – Resultados das métricas de desempenho na validação com o dataset MIO-TCD

Métrica	Valor	Descrição
Número de imagens avaliadas	11.000	Total de imagens do conjunto de validação
Instâncias anotadas	23.071	Total de objetos da classe ‘car’ presentes nas anotações de validação
Precisão (Precision)	89,2%	Proporção de verdadeiros positivos entre as detecções
Revocação (Recall)	90,6%	Proporção de verdadeiros positivos entre as ocorrências reais
mAP@0.5	95,7%	Média da precisão com IoU mínimo de 50%
mAP@0.5:0.95	72,7%	Média da precisão com IoUs variando de 50% a 95%
Tempo de inferência	2,1 ms	Tempo médio para processamento por imagem (forward pass)
Tempo de NMS	0,9 ms	Tempo médio da supressão de não-máximos por imagem

Fonte: Resultados obtidos pelo autor com a ferramenta de validação do YOLOv5

Além das métricas apresentadas, calculou-se o F_1 -score — a média harmônica entre Precisão (89,2 %) e Recall (90,6 %):

$$F_1 = 2 \times \frac{0,892 \times 0,906}{0,892 + 0,906} \approx 0,899 \text{ (89,9 \%)}.$$

Para ilustrar visualmente a qualidade das detecções realizadas, a Figura 11 apresenta uma imagem do conjunto de validação com os resultados gerados pelo modelo treinado. As caixas delimitadoras foram corretamente posicionadas sobre os veículos, demonstrando a capacidade do detector em identificar com precisão a classe “car” em um cenário real. Ao lado de cada rótulo “car”, informa um valor numérico referente a confiança da detecção, indicando a probabilidade atribuída pelo modelo da presença de um veículo na caixa delimitadoras desenhada.

Figura 11 – Detecção de veículos realizada pelo modelo treinado



Fonte: Elaboração do autor

4.1.3 RESULTADOS DO RASTREAMENTO

O rastreamento de veículos representa uma etapa essencial para manter a identidade dos objetos detectados ao longo de sequências de vídeo, possibilitando a análise de trajetórias, tempo de permanência e contagem precisa, sem duplicidades. Diferentemente da tarefa de detecção, o rastreamento exige conjuntos de dados com continuidade temporal, o que inviabiliza a aplicação do MIO-TCD Localization, utilizado anteriormente, por se tratar de imagens estáticas sem relação sequencial.

Para permitir a avaliação quantitativa do rastreamento, foi utilizada a sequência 0002 do conjunto de dados KITTI Tracking, que apresenta uma cena urbana contínua, composta por aproximadamente 233 quadros e 15 veículos distintos. As anotações originais foram convertidas para o formato MOT Challenge, de modo a permitir a comparação com os resultados gerados pelo sistema de rastreamento.

O pipeline utilizado foi composto pelo algoritmo StrongSORT integrado ao YOLOv5, utilizando pesos previamente ajustados por fine-tuning com as imagens da própria sequência de validação. É

importante destacar que esse ajuste não foi necessário para o funcionamento do rastreamento em si, mas teve como finalidade assegurar que as métricas avaliadas refletissem com maior fidelidade o desempenho do rastreador, evitando distorções causadas por erros sistemáticos do detector em relação à perspectiva ou ao domínio visual da cena.

O rastreamento foi executado mantendo a resolução original de entrada (1242×374 pixels), e considerando exclusivamente a classe Car, tanto nas detecções quanto na avaliação. As métricas foram obtidas por meio da biblioteca motmetrics, a partir da comparação entre as predições e o ground truth convertido. A Tabela 5 apresenta os principais indicadores quantitativos obtidos:

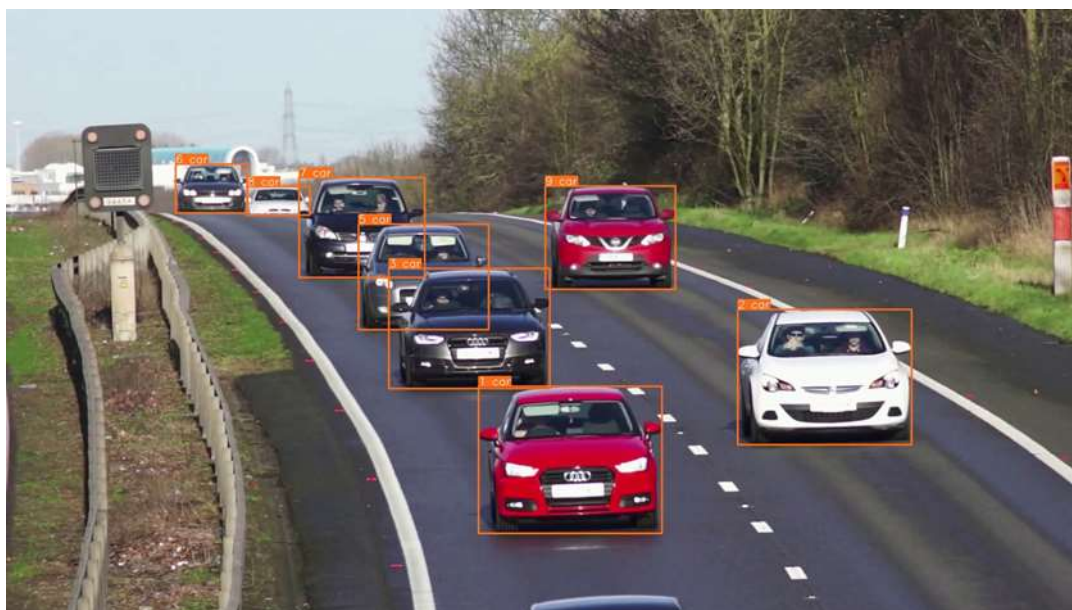
Tabela 5 – Métricas de rastreamento obtidas com o modelo YOLOv5 + StrongSORT

Métrica	Valor	Descrição
ID Switches	25	Número de trocas de identidade
Tracks Fragmentados (FM)	57	Número de interrupções no rastreamento
Taxa de Rastreamento (MOTA)	61,6%	Avaliação combinada de FPs, FNs e ID switches
Precisão de Rastreamento (MOTP)	0,246	Precisão média da sobreposição caixa-a-caixa
IDF1	57,2%	Média harmônica entre IDP e IDR
IDP	66,3%	Precisão de identidade (ID Precision)
IDR	50,3%	Revocação de identidade (ID Recall)
FP	61	Falsos Positivos totais
FN	310	Falsos Negativos totais
GT	15	Total de objetos únicos anotados (Ground Truth)
MT	5	Objetos muito rastreados (cobertos >80%)
PT	8	Objetos parcialmente rastreados (20–80%)
ML	2	Objetos perdidos (cobertura <20%)

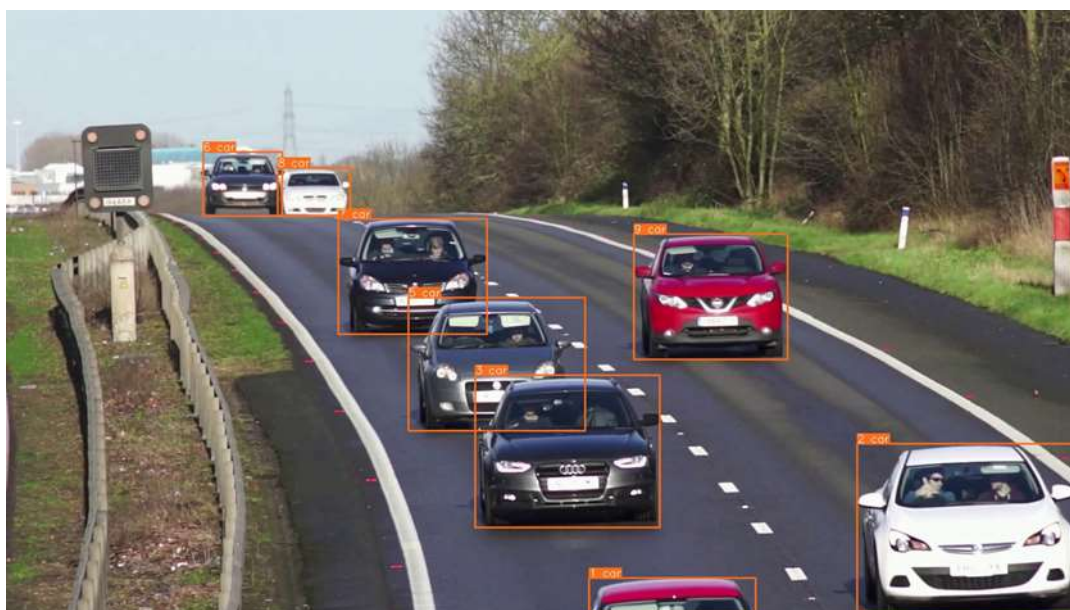
Fonte: Elaboração do autor (motmetrics).

A Figura 12 apresenta dois quadros extraídos de momentos distintos de um vídeo de teste, demonstrando o rastreamento realizado pelo sistema YOLOv5 + StrongSORT. Em ambas as imagens, os veículos são identificados por caixas delimitadoras com rótulos numéricos únicos, que se mantêm consistentes entre os quadros. Essa persistência de identificadores ao longo do tempo demonstra a capacidade do algoritmo de rastrear corretamente os objetos, mesmo em presença de movimento e mudanças de posição.

Figura 12 – Exemplo de rastreamento com YOLOv5 + StrongSORT



(a) Quadro inicial com rastreamento



(b) Quadro posterior com IDs preservados

4.1.4 RESULTADOS DA SIMULAÇÃO EM SUMO

Após a execução dos dois cenários de simulação, calcularam-se, para o conjunto de 1.200 veículos, quatro indicadores: duração média da viagem, tempo médio de espera no semáforo, emissão total de CO₂ e consumo total de combustível. Esses resultados estão reunidos na Tabela 6, permitindo uma comparação direta entre o ciclo fixo e o controle adaptativo.

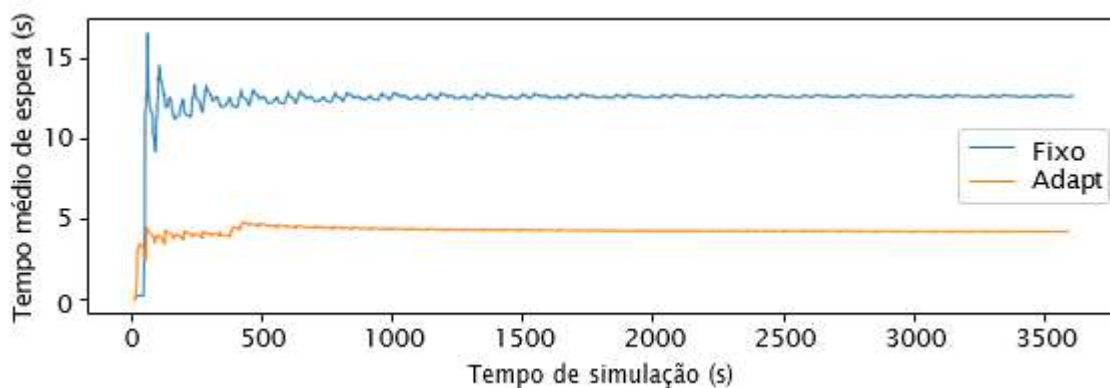
Tabela 6 – Indicadores médios por cenário de controle (1.200 veículos)

Regime	Duração média (s)	Tempo médio de espera (s)	CO ₂ (kg)	Combustível (L)
Ciclo fixo	31.54	12.75	95.57	41.87
Adaptativo (MP)	20.40	4.26	77.38	33.90
Redução (%)	35.3	66.6	19.1	19.0

Fonte: Elaboração do autor (simulações no SUMO).

A Figura 13 mostra a evolução do tempo médio de espera ao longo da simulação, enquanto a Figura 14 apresenta a distribuição dos tempos de viagem individuais em classes de 5 s. Esses gráficos complementam a tabela ao exibir a dinâmica temporal e a dispersão dos resultados.

Figura 13 – Progressão do tempo médio de espera (ciclo fixo × adaptativo)



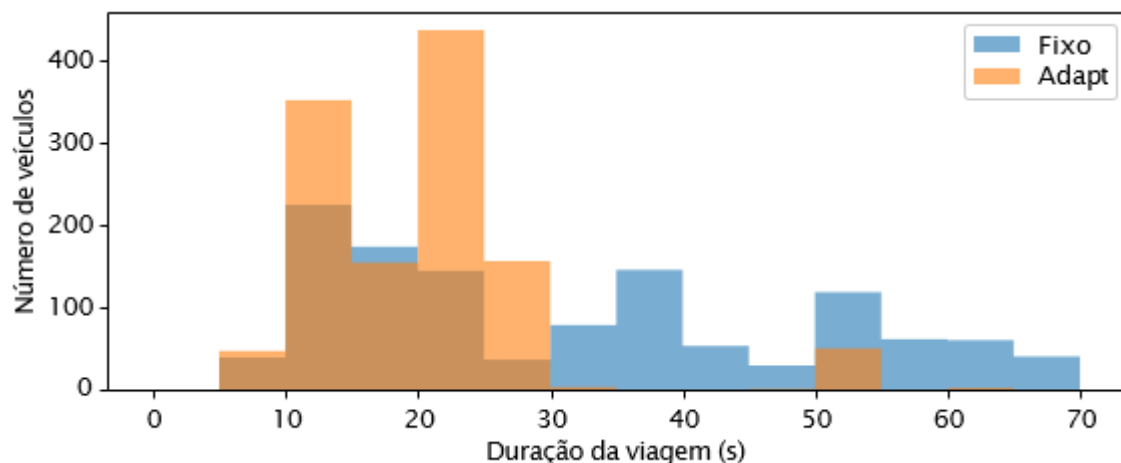
Fonte: Elaboração do autor

4.2 DISCUSSÃO

4.2.1 DESEMPENHO DO TREINAMENTO INICIAL

O modelo YOLOv5 passou por 66 épocas de pré-treinamento no subconjunto do COCO contendo apenas a classe Carro. A Figura 9 exibe curvas de perda que decrescem de forma regular, indicando convergência estável. Entre a época 0 e a época 66 observou-se um aumento de **+3,22** pontos percentuais (p.p.) em Precisão (de 67,75 para 70,98%), **+5,34** p.p. em Recall (de 47,09 para 52,43%) e **+6,25** p.p. em mAP@0.5 (de 51,80 para 58,04%). Esses pesos finais fornecem uma inicialização robusta para as etapas de especialização subsequentes.

Figura 14 – Histograma dos tempos de viagem (ciclo fixo x adaptativo)



Fonte: Elaboração do autor

4.2.2 REFINAMENTO POR TRANSFERÊNCIA DE APRENDIZADO

Partindo dos pesos obtidos após as 66 épocas no COCO, executou-se um fine-tuning de 150 épocas no conjunto MIO–TCD Localization, cujas imagens reproduzem o ponto de vista de câmeras fixas em postes de semáforo. O detector passou de Precision = 70,98 %, Recall = 52,43 %, mAP@0.5 = 58,04 % para Precision = 89,2 %, Recall = 90,6 %, mAP@0.5 = 95,7 % e mAP@0.5:0.95 = 72,7 %. Em termos absolutos, isso representa um ganho de +18,22 pontos percentuais (p.p.) em precision, +38,17 p.p. em recall e +37,66 p.p. em mAP@0.5.

Análise das causas e implicações

- Aumento expressivo de recall: o salto de 52,4% para 90,6% indica que o modelo passou a detectar quase todas as ocorrências de veículos, reduzindo falsos negativos típicos do COCO (por conter principalmente imagens de ruas capturadas de perto).
- Aumento da Precisão: a elevação de 71,0% para 89,2% confirma menor taxa de falsos positivos, mas em proporção menor que o aumento de recall. Algumas detecções incorretas ainda ocorrem, principalmente em condições de iluminação adversa.
- mAP@0.5:0.95 = 72,7%: o ganho em limiares altos de IoU indica caixas mais bem localizadas; isso se deve, em parte, à resolução padronizada de 720×480 do MIO–TCD e ao maior número de exemplos de veículos parcialmente ocluídos, que ajudam o modelo a aprender a ajustar as bordas com mais precisão.

- Transferência eficiente: mesmo com 150 épocas, a perda de validação continuou caindo sem sinais de sobreajuste, possivelmente porque o conjunto de treinamento continha variedade de climas e horários, mantendo a capacidade de generalização.
- Impacto no pipeline: Com um tempo médio medido de 3,6 ms para inferência e 1,7 ms para NMS (0,2 ms de pré-processamento), totalizando 5,5 ms por imagem em 640×640 com uma GPU RTX 2060. A taxa efetiva é de aproximadamente 182 FPS, se mostrando suficiente para aplicações em tempo real.
- F1-score: média harmônica entre Precision e Recall subiu de 60,5 % (treinamento COCO) para 89,9 % (refinamento MIO-TCD), refletindo a melhoria equilibrada.

Em síntese, o ajuste no MIO-TCD não apenas elevou drasticamente a cobertura do detector, mas também melhorou a fidelidade das caixas de detecção, mantendo uma velocidade de inferência suficiente. Esses ganhos são fundamentais para abastecer o rastreador StrongSORT com entradas de alta qualidade e, por consequência, reduzir ID switches nas etapas posteriores.

4.2.3 ANÁLISE DO RASTREAMENTO

Após as 150 épocas no MIO-TCD, foi feito um refinamento extra de 100 épocas no KITTI (1242×374 px) para ajustar a posição das caixas e lidar com a mudança de perspectiva desse conjunto, captado por câmeras em veículos autônomos. Avaliado na sequência KITTI_0002, o pipeline YOLOv5 + StrongSORT obteve **MOTA = 61,6%, IDF1 = 57,2%** e registrou **25 trocas de ID**. O **MOTP = 0,246** mostra que a sobreposição média entre caixas preditas e anotações de referência ainda é baixa.

Pontos positivos

- Consistência de identidade: a redução de falsos negativos ajuda a manter os IDs ativos por mais tempo, refletindo no IDF1 de 57,2%.
- Velocidade preservada: o acréscimo de 100 épocas não aumentou o tempo médio de inferência, mantendo a operação em tempo real.

Limitações atuais

- Fragmentações: ainda ocorrem 57 fragmentações em oclusões prolongadas, o que diminui o número de trilhas “Muito Rastreadas” (MT).
- Aderência das caixas: o MOTP baixo indica desalinhamento, possivelmente por diferenças de proporção entre o KITTI e o pré-processamento em 640 px.
- ReID genérico: sem ajuste específico para o KITTI, o extrator pode confundir veículos semelhantes, limitando o IDF1.

Em síntese, o modelo apresenta rastreamento consistente e operação em tempo real, mas o alinhamento das caixas e o tratamento de oclusões seguem como principais desafios para melhorar o IDF1.

4.2.4 DISCUSSÃO DA SIMULAÇÃO NO SUMO

Os resultados da Tabela 6 mostram reduções expressivas quando se passa do ciclo fixo para o controle adaptativo. A duração média da viagem caiu de 31,53s para 20,40s (-35,3%) e o tempo médio de espera no semáforo de 12,74s para 4,26s (-66,5%). Além disso, indica que as emissões totais de CO₂ foram de 95,6kg para 77,4kg (-19%) e o consumo de combustível de 41,87L para 33,9L (-19%).

4.2.4.1 LEITURA DETALHADA DOS GRÁFICOS

- Figura 13 – curva de espera. As linhas começam juntas e se separam por volta de 400 s, indicando que o MP reage rapidamente ao desequilíbrio inicial de filas. A partir de 1 000 s a curva do ciclo fixo estabiliza acima de 10 s, enquanto a adaptativa oscila abaixo de 5 s.
- Figura 14 – histograma de viagens. No ciclo fixo observa-se um número expressivo de viagens com duração superior a 45s, chegando a 70s nos casos mais longos. No regime adaptativo quase todas as viagens ficam abaixo de 35s, com pico de frequência entre 20 e 25s, o que indica uma variabilidade menor.

4.2.4.2 INTERPRETAÇÃO DOS GANHOS

1. Verde não desperdiçado – O algoritmo encerra fases de baixa demanda logo após o verde mínimo, redistribuindo segundos ociosos para outras faixas com fila.
2. Troca antes de obstruir o cruzamento – Quando a diferença de filas ultrapassa o limiar, o controlador antecipa a mudança de fase; assim, os últimos veículos ainda conseguem atravessar sem ficar parados dentro da interseção, evitando que o fluxo perpendicular fique travado no ciclo seguinte.
3. Menos paradas e emissões – Ciclos mais curtos reduziram a média de paradas por veículo de 0,88 para 0,71 (-19 %); com o motor em marcha lenta menos tempo, as emissões de CO₂ caíram de 95,7 kg para 77,6 kg e o consumo de combustível de 41,9 L para 34,0 L (-19 %).

4.2.4.3 ASPECTOS AINDA NÃO EXPLORADOS

- **Escopo:** o estudo contempla apenas um cruzamento isolado com demanda estática. Avaliações em rede, bem como cenários de hora-pico e vale, permanecem para trabalhos futuros.

- **Contagem idealizada:** o controlador foi alimentado por filas exatas geradas pelo SUMO. A integração com contagens por visão computacional exigirá tratar erros de detecção e oclusões
- **Condições ambientais dinâmicas:** fatores como chuva, neblina ou variação de iluminação não foram simulados, mas podem impactar tanto a qualidade das detecções quanto o desempenho do controlador.

4.2.4.4 IMPLICAÇÕES PARA AUTOMAÇÃO DE TRÁFEGO

O controlador adaptativo em tempo real, alimentado por sensores de fluxo, reduziu o tempo de espera em quase dois terços, encurtou a viagem média em um terço e cortou aproximadamente um quinto das emissões de CO₂ e do consumo de combustível, tudo isso sem aumentar a duração total dos ciclos. Esses resultados indicam que sistemas de automação baseados em dados locais podem elevar simultaneamente a eficiência operacional e a sustentabilidade de cruzamentos semelhantes, dispensando coordenação centralizada e servindo como etapa incremental rumo a redes inteligentes de controle de tráfego.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 CONCLUSÃO

Este trabalho apresentou um pipeline completo de visão computacional para detecção e rastreamento de veículos, integrado a uma lógica adaptativa de controle semafórico em ambiente simulado. Os objetivos gerais, detectar e contar veículos em tempo real para subsidiar a automação de semáforos, foram atendidos.

No detector YOLOv5, o treinamento inicial no COCO filtrado para a classe car e o refinamento de 150 épocas no MIO-TCD levaram a indicadores de alto desempenho: Precisão $\approx 90\%$, Recall $\approx 91\%$, mAP@0.5 = 95,6% e mAP@0.5:0.95 = 72,6%. A validação em imagens não vistas confirmou a robustez com precisão de 89,2% e revocação de 90,6%.

O rastreamento com StrongSORT, executado na sequência KITTI_0002, atingiu MOTA = 61,6%, IDF1 = 57,2 % e apenas 25 trocas de identidade em 15 objetos, preservando a identidade de cinco veículos por mais de 80% dos quadros.

Na simulação SUMO, o controlador adaptativo reduziu a duração média da viagem de 31,53 s para 20,40 s e o tempo médio de espera de 12,74s para 4,26s, sem aumentar o ciclo total. Esses resultados

mostram que a combinação detecção + rastreamento fornece dados suficientes para melhorar significativamente a eficiência de um cruzamento mesmo com lógica local.

Limitações persistem: MOTP de 0,246 indica necessidade de caixas mais aderentes; 57 fragmentações em oclusões prolongadas; a simulação usou contagens perfeitas, não a saída real do detector. Ainda assim, os ganhos observados demonstram a viabilidade de sistemas de automação de tráfego baseados apenas em visão.

5.2 TRABALHOS FUTUROS

- **Ajuste fino do StrongSORT** - Treinar o extrator ReID em recortes KITTI e Cityscapes para elevar o IDF1 além de 65%.
- **Integração visão–SUMO offline.** Utilizar vídeos reais de câmeras de tráfego para extrair contagens com YOLOv5 + StrongSORT e, em seguida, alimentar essas contagens no SUMO como entrada estática, avaliando como o ruído de detecção impacta pode afetar a automação.
- **Escalonamento de rede** - Replicar o experimento em uma malha 3 × 3 de cruzamentos (Totalizando 9 junções), avaliando propagação de filas e formação de “onda verde”.
- **Condições adversas** - Realizar fine-tuning adicional do detector e do ReID com imagens de cenários variados (noite, chuva, neblina, sombras fortes), além de usar técnicas de data augmentation específicas (brilho, contraste, ruído), para garantir que o pipeline funcione de forma confiável em condições extremas e não vistas durante o treinamento.
- **Implantação embarcada** - Integrar o pipeline em um computador de bordo de baixo consumo, por exemplo Jetson Nano, Jetson Xavier NX ou Raspberry Pi 4, visando processamento em tempo real, respeitando o limite de potência típico desses dispositivos (10–15W).
- **Avaliação ambiental aprofundada.** Ampliar a análise ambiental já realizada na comparação entre o ciclo fixo e o controlador adaptativo, incluindo emissões de CO, NOx e material particulado e relacionando esses valores ao desempenho do pipeline de detecção e rastreamento. Calcular taxas de emissão por veículo e por quilômetro, comparar cenários de pico e de baixo fluxo e avaliar variações na composição da frota (porcentagens de veículos elétricos ou híbridos). Por fim, traduzir as reduções de poluentes em indicadores de sustentabilidade, como toneladas de CO2 evitadas, créditos de carbono ou número equivalente

de árvores plantadas, reforçando o impacto prático da integração de visão computacional na automação semafórica.

Em resumo, o sistema proposto comprova que soluções baseadas em visão computacional podem servir como blocos fundamentais na automação do tráfego urbano, oferecendo ganhos tangíveis de desempenho mesmo antes da etapa de validação em campo.

REFERÊNCIAS

- BROSTRÖM, M. Real-time multi-camera multi-object tracker using YOLOv5 and StrongSORT with OSNet. 2022. <https://github.com/mikel-brostrom/Yolov5_StrongSORT_OSNet>. Citado na página 46.
- CANNY, J. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8, n. 6, p. 679–698, 1986. Citado na página 24.
- DENDORFER, P. et al. Motchallenge: A benchmark for single-camera multiple target tracking. International Journal of Computer Vision, v. 129, p. 1–37, 04 2021. Citado na página 40.
- DONG, X.; CAPPUCCIO, M. Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions. 2023. Citado na página 35.
- DU, Y. et al. Strongsort: Make deepsort great again. IEEE Transactions on Multimedia, v. 25, p. 8725–8737, 2023. Citado na página 31.
- DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. Commun. ACM, Association for Computing Machinery, New York, NY, USA, v. 15, n. 1, p. 11–15, jan. 1972. ISSN 0001-0782. Disponível em:<<https://doi.org/10.1145/361237.361242>>. Citado na página 24.
- GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). [S.l.: s.n.], 2012. p. 3354–3361. Citado 2 vezes nas páginas 22 e 39.
- GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. 4. ed. Hoboken, NJ: Pearson, 2018. Citado 4 vezes nas páginas 19, 20, 21 e 26.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. Cambridge: MIT Press, 2016. Citado na página 26.
- HAO, W. et al. The role of activation function in cnn. In: 2020 2nd International Conference on Information Technology and Computer Application (ITCA). [S.l.: s.n.], 2020. p. 429–432. Citado na página 26.
- HE, K. et al. Mask r-cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV). [S.l.: s.n.], 2017. p. 2980–2988. Citado na página 21.
- Instituto de Pesquisa Econômica Aplicada. Relatório sobre Mobilidade Urbana e Desenvolvimento Sustentável. 2021. Disponível em: <<https://www.ipea.gov.br>>. Citado na página 16.
- KAEHLER, A.; BRADSKI, G. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. [S.l.]: O'Reilly Media, 2016. Citado na página 29.
- KALMAN, R. E. A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering, v. 82, n. 1, p. 35–45, 1960. Citado na página 30.

KARPATHY, A.; JOHNSON, J.; LI, F.-F. Convolutional Neural Networks. s.d. Acesso em: 13 jan. 2025. Disponível em: <<https://cs231n.github.io/convolutional-networks/>>. Citado na página 27.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, p. 1097–1105, 2012. Citado 3 vezes nas páginas 22, 23 e 25.

KUSTIJA, J.; NUR, A. Scats (sydney coordinated adaptive traffic system) as a solution to overcome traffic congestion in big cities. *International Journal of Research and Applied Technology*, v. 3, p. 1–14, 05 2023. Citado na página 34.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 23.

LIN, T. Y. et al. Microsoft coco: Common objects in context. In: *European Conference on Computer Vision (ECCV)*. [s.n.], 2014. Disponível em: <<https://cocodataset.org>>. Citado na página 39.

LOWE, D. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. [S.l.: s.n.], 1999. v. 2, p. 1150–1157 vol.2. Citado na página 25.

LUO, Z. et al. Mio-tcd: A new benchmark dataset for vehicle classification and localization.

IEEE Transactions on Image Processing, v. 27, n. 10, p. 5129–5141, 2018. Citado 2 vezes nas páginas 40 e 74.

MEINHARDT, T. et al. Trackformer: Multi-object tracking with transformers. *CoRR*, abs/2101.02702, 2021. Disponível em: <<https://arxiv.org/abs/2101.02702>>. Citado na página 30.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, USA, v. 22, n. 10, p. 1345–1359, out. 2010. ISSN 1041-4347. Disponível em: <<https://doi.org/10.1109/TKDE.2009.191>>. Citado na página 23.

REDIN, R. F. et al. Sistema de semáforo inteligente baseado em visão computacional para otimização de fluxo. 2019. Acesso em: 2 jan. 2025. Disponível em: <<https://online.unisc.br/acadnet/anais/index.php/mostraextensaounisc/article/viewFile/23982/1192614415>>. Citado na página 17.

REDMON, J.; FARHADI, A. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. Citado 4 vezes nas páginas 16, 22, 27 e 28.

REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2016. Disponível em: <<https://arxiv.org/abs/1506.01497>>. Citado na página 22.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. Disponível em: <<https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf>>. Citado na página 24.

ROY, A. M.; BHADURI, J. A deep learning enabled multi-class plant disease detection model based on computer vision. *AI*, v. 2, n. 3, p. 413–428, 2021. ISSN 2673-2688.

Disponível em: <<https://www.mdpi.com/2673-2688/2/3/26>>. Citado na página 37.

STAUFFER, C.; GRIMSON, W. Adaptive background mixture models for real-time tracking. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. [S.l.: s.n.], 1999. v. 2, p. 246–252 Vol. 2. Citado na página 24.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. New York: Springer, 2022. Citado na página 20.

Transport for NSW. SCATS Core. s.d. Acesso em: 15 ago. 2025. Disponível em: <<https://www.transport.nsw.gov.au/system/files/media/documents/2022/>

SCATS-Core-brochure-Final-web-spreads_0.pdf>. Citado 2 vezes nas páginas 17 e 34.

United Nations. *World Urbanization Prospects: The 2018 Revision*. United Nations, 2018. Disponível em: <<https://www.un.org/>>. Citado na página 16.

VARAIYA, P. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, v. 36, p. 177–195, 2013. Citado na página 50.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple Online and Realtime Tracking with a Deep Association Metric. 2017. Disponível em: <<https://arxiv.org/abs/1703.07402>>. Citado 2 vezes nas páginas 30 e 31.

APÊNDICES

APÊNDICE A – Script de controle adaptativo em Python usando TraCI

O código a seguir implementa o algoritmo de controle Max-Pressure sobre o cruzamento modelado, utilizando a interface TraCI do SUMO. A lógica avalia o número de veículos parados em cada eixo (Norte-Sul e Leste-Oeste) e ajusta a duração dos sinais de acordo com o diferencial de pressão.

Listing A.1 – Script de controle adaptativo usando Max-Pressure

```

1  #!/usr/bin/env python3
2  # -*- coding : utf-8 -*-
3
4  Script de controle adaptativo com base no algoritmo Max-Pressure ,
5  aplicado a um unico cruzamento semaforico .
6
7  Gera os arquivos:
8      - tripinfo_pressure .xml
9      - summary_pressure.xml
10     - emission_pressure .xml
11
12  E exibe no final:
13     - numero total de veiculos
14     - duracao media das viagens
15     - tempo medio de espera (waiting time)
16     - tempo total de simulacao
17     - emissao total de CO2 (em kg)
18     - consumo total de combustivel (em litros)
19
20
21  # -----
22  # 1. Imports
23  import os
24  import sys
25  import xml.etree.ElementTree as ET
26
27  if "SUMO_HOME" not in os.environ :
28      sys.exit(" Erro : a variavel de ambiente SUMO_HOME nao esta definida ."
29      )
30  sys.path.append(os.path.join(os.environ["SUMO_HOME"], "tools"))
31  from sumolib import checkBinary
32  import traci

```

```

33
34 # -----
35 # 2. Configuracoes da simulacao
36 SUMO_CONFIG = r"CAMINHO_DO_SUMOCFG"
37 STEP = 0.1
38 TLS_ID = 'ID_DA_JUNCAO_SIMULADA'
39
40 TRIP_OUT = r"CAMINHO DA PASTA DA SIMULACAO/tripinfo_pressure.xml"
41 SUMMARY_OUT = r"CAMINHO DA PASTA DA SIMULACAO/summary_pressure.xml"
42 EMISSION_OUT = r"CAMINHO DA PASTA DA SIMULACAO/emission_pressure.xml"
43
44 SUMO_CMD = [
45     checkBinary("sumo - gui"),
46     "-c", SUMO_CONFIG,
47     "--stop-length", str(STEP),
48     "--tripinfo-output", TRIP_OUT,
49     "--summary-output", SUMMARY_OUT,
50     "--emission-output", EMISSION_OUT,
51     "--duration-log-statistics", "true"
52 ]
53
54 # -----
55 # 3. Parametros do controlador Max-Pressure
56 GREEN_MIN = 7
57 GREEN_MAX = 40
58 DELTA_T = 0.5
59 PRESS_TH = 2
60
61 IN_EW = ["-E0_0", "E0_0"]
62 IN_NS = ["-E1_0", "E1_0"]
63
64 def media(lista):
65     return sum(lista) / len(lista) if lista else 0.0
66
67 # -----
68 # 4. Execucao da simulacao com controle adaptativo
69 def run():
70     traci.start(SUMO_CMD)
71
72     tempo_total = 0.0
73     acumulador = 0.0
74     tempo_verde = 0.0
75
76     try:
77         while traci.simulation.getMinExpectedNumber() > 0:
78             traci.simulationStep()
79             tempo_total += STEP

```

```

81         acumulador += STEP
82         tempo_verde += STEP
83
84         if acumulador >= DELTA_T:
85             acumulador = 0.0
86
87             fase = traci.trafficlight.getPhase(TLS_ID)
88             fila_ew = sum([traci.lane.getLastStepHaltingNumber(l) for
89                             l in IN_EW])
90             fila_ns = sum([traci.lane.getLastStepHaltingNumber(l) for
91                             l in IN_NS])
92
93             if fase in (0, 1):
94                 pressao = fila_ns - fila_ew
95             else:
96                 pressao = fila_ew - fila_ns
97
98             deve_trocar = pressao > PRESS_TH and tempo_verde >=
99                 GREEN_MIN
100
101             if deve_trocar or tempo_verde >= GREEN_MAX:
102                 traci.trafficlight.setPhaseDuration(TLS_ID, 2)
103                 tempo_verde = 0.0
104
105         finally:
106             traci.close()
107
108     return tempo_total
109
110 # -----
111 # 5. Análise do arquivo de viagens
112 def analisar_tripinfo(path):
113     tree = ET.parse(path)
114     duracoes = [float(x.attrib["duration"]) for x in tree.iter("tripinfo")]
115     esperas = [float(x.attrib["waitingTime"]) for x in tree.iter("tripinfo")]
116     return len(duracoes), media(duracoes), media(esperas)
117
118 # -----
119 # 6. Cálculo de emissões (CO2 e combustível)
120 def analisar_emissoes(path, passo):
121     co2_total = 0.0
122     combustivel_total = 0.0
123
124     for _, elem in ET.iterparse(path, events=["end"]):
125         if elem.tag == "vehicle":
126             co2_total += float(elem.get("CO2", 0.0)) * passo

```

```

102         combustivel_total += float(elem.get("fuel", 0.0)) * passo
103         elem.clear()
104
105     co2_kg = co2_total / 1_000_000
106     combustivel_l = combustivel_total / 740_000
107
108     return co2_kg, combustivel_l
109
110 # -----
111 # 7. Execucao principal
112 if __name__ == "__main__":
113     tempo_simulacao = run()
114
115     total, duracao_med, espera_med = analisar_tripinfo(TRIP_OUT)
116     co2, combustivel = analisar_emissoes(EMISSION_OUT, STEP)
117
118     print("\n===== METRICAS FINAIS (Max-Pressure) =====")
119     print(f"Veiculos simulados      : {total}")
120     print(f"Duracao media (s)          : {duracao_med:.2f}")
121     print(f"Tempo medio de espera       : {espera_med:.2f}")
122     print(f"Tempo total simulacao      : {tempo_simulacao:.2f}")
123     print(f"Emissao total de CO2        : {co2:.2f} kg")
124     print(f"Consumo de combustivel     : {combustivel:.2f} L")
125     print("=====")

```

APÊNDICE B – Exemplo de anotação YOLO para imagem de entrada do dataset MIO-TCD

A Figura 15 exemplifica um quadro extraído do dataset MIO-TCD utilizado. Na Tabela 7 está a respectiva anotação no formato YOLOv5, com bounding boxes associadas à classe carro.

Figura 15 – Imagem anotada do dataset



Fonte: MIO-TCD (LUO et al., 2018).

Tabela 7 – Anotação YOLO correspondente à Figura 15. Cada linha representa um objeto detectado da classe "carro"(Classe 0).

Classe	x_{center}	y_{center}	largura	altura
0	0.871528	0.253125	0.034722	0.039583
0	0.514583	0.433333	0.109722	0.095833
0	0.952778	0.486458	0.069444	0.152083
0	0.866667	0.301042	0.038889	0.052083
0	0.660417	0.283333	0.059722	0.058333
0	0.735417	0.220833	0.031944	0.029167
0	0.765972	0.248958	0.037500	0.039583
0	0.971528	0.337500	0.034722	0.075000
0	0.870139	0.401042	0.059722	0.089583

APÊNDICE C – Arquivos da Simulação no SUMO

C.1 C.1 Arquivo de Configuração da Simulação (sumo2.sumocfg)

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on 2025-07-27 12:09:19 by Eclipse SUMO sumo Version 1.23.1
-->

<sumoConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
,→ xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.x
,→ xsd">

    <input>
        <net-file value="sumo2.net.xml"/>
        <route-files value="sumo2.rou.xml"/>
    </input>
```

C.2 C.2 Arquivo de Rede da Simulação (sumo2.net.xml)

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on 2025-07-27 12:01:36 by Eclipse SUMO netedit Version 1.23.1
<neteditConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
,→ xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/neteditConfigurati
,→ on.xsd">

    <input>
        <!-- Insira o caminho para o arquivo de configuração do SUMO
        ,→ (.sumocfg) -->
        <sumocfg-file value=" - "/>
```

```

11     </input>
12
13     <output>
14         <!-- Insira o caminho e nome do arquivo de resultados gerado -->
15         <output-file value="" />
16     </output>
17
18     <processing>
19         <offset.disable-normalization value="true"/>
20     </processing>
21
22     <junctions>
23         <no-turnarounds value="true"/>
24     </junctions>
25
26     <netedit>
27         <new-network value="lake"/>
28     </netedit>
29
30 </neteditConfiguration>
31 -->
32
33 <net version="1.20" junctionCornerDetail="5" linkTurnSpeed="5.50"
34 ↪ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
35 ↪ xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">
36
37     <location netOffset="0.00,0.00" convBoundary="-64.09,-40.28,63.67,43.61"
38     ↪ origBoundary="10000000000.00,10000000000.00,-10000000000.00,-1000000
39     ↪ 000-0.00"
40     ↪ projParameter="" />
41
42     <edge id=":j4_0" function="internal">
43         <lane id=":j4_0_0" index="0" speed="6.57" length="9.15"
44         ↪ shape="0.93,5.98 -1.55,6.31 -3.34,7.36 -4.44,9.12 -4.84,11.59"/>
45     </edge>
46
47     <edge id=":j4_1" function="internal">
48         <lane id=":j4_1_0" index="0" speed="13.89" length="14.47"
49         ↪ shape="0.93,5.98 -13.54,5.86"/>
50     </edge>
51
52     <edge id=":j4_2" function="internal">
53         <lane id=":j4_2_0" index="0" speed="7.99" length="4.08"
54         ↪ shape="0.93,5.98 -2.93,5.39 -3.08,5.30"/>

```

```

45 </edge>
46 <edge id="j4_12" function="internal">
47   <lane id="j4_12_0" index="0" speed="7.99" length="10.16"
48     ↪ shape="-3.08,5.30 -5.67,3.71 -7.28,0.94 -7.77,-2.94"/>
49 </edge>
50 <edge id="j4_3" function="internal">
51   <lane id="j4_3_0" index="0" speed="6.49" length="9.03"
52     ↪ shape="-4.56,-2.88 -4.26,-0.42 -3.23,1.34 -1.50,2.41 0.95,2.78"/>
53 </edge>
54 <edge id="j4_4" function="internal">
55   <lane id="j4_4_0" index="0" speed="13.89" length="14.47"
56     ↪ shape="-4.56,-2.88 -4.84,11.59"/>
57 </edge>
58 <edge id="j4_5" function="internal">
59   <lane id="j4_5_0" index="0" speed="8.05" length="4.09"
60     ↪ shape="-4.56,-2.88 -5.18,0.97 -5.28,1.13"/>
61 </edge>
62 <edge id="j4_13" function="internal">
63   <lane id="j4_13_0" index="0" speed="8.05" length="10.17"
64     ↪ shape="-5.28,1.13 -6.89,3.71 -9.67,5.34 -13.54,5.86"/>
65 </edge>
66 <edge id="j4_6" function="internal">
67   <lane id="j4_6_0" index="0" speed="6.56" length="9.13"
68     ↪ shape="-13.52,2.66 -11.04,2.33 -9.25,1.28 -8.16,-0.47
69     ↪ -7.77,-2.94"/>
70 </edge>
71 <edge id="j4_7" function="internal">
72   <lane id="j4_7_0" index="0" speed="13.89" length="14.47"
73     ↪ shape="-13.52,2.66 0.95,2.78"/>
74 </edge>
75 <edge id="j4_8" function="internal">
76   <lane id="j4_8_0" index="0" speed="7.98" length="4.07"
77     ↪ shape="-13.52,2.66 -9.65,3.25 -9.51,3.33"/>
78 </edge>
79 <edge id="j4_14" function="internal">
80   <lane id="j4_14_0" index="0" speed="7.98" length="10.16"
81     ↪ shape="-9.51,3.33 -6.92,4.93 -5.31,7.71 -4.84,11.59"/>
82 </edge>
83 <edge id="j4_9" function="internal">
84   <lane id="j4_9_0" index="0" speed="6.48" length="9.03"
85     ↪ shape="-8.03,11.52 -8.34,9.06 -9.35,7.30 -11.09,6.23
86     ↪ -13.54,5.86"/>

```

```

75 </edge>
76 <edge id="J4_10" function="internal">
77   <lane id="J4_10_0" index="0" speed="13.89" length="14.47"
78     ↪ shape="-8.03,11.52 -7.77,-2.94"/>
79 </edge>
80 <edge id="J4_11" function="internal">
81   <lane id="J4_11_0" index="0" speed="8.06" length="4.07"
82     ↪ shape="-8.03,11.52 -7.41,7.67 -7.32,7.53"/>
83 </edge>
84 <edge id="J4_15" function="internal">
85   <lane id="J4_15_0" index="0" speed="8.06" length="10.20"
86     ↪ shape="-7.32,7.53 -5.70,4.93 -2.91,3.30 0.95,2.78"/>
87 </edge>
88 <edge id="E0" from="J1" to="J4" priority="-1">
89   <lane id="E0_0" index="0" speed="13.89" length="62.73"
90     ↪ shape="63.66,6.47 0.93,5.98"/>
91 </edge>
92 <edge id="E00" from="J4" to="J0" priority="-1" spreadType="roadCenter">
93   <lane id="E00_0" index="0" speed="13.89" length="50.56"
94     ↪ shape="-13.54,5.86 -64.10,5.46"/>
95 </edge>
96 <edge id="E1" from="J6" to="J4" priority="-1">
97   <lane id="E1_0" index="0" speed="13.89" length="37.38"
98     ↪ shape="-3.91,-40.25 -4.56,-2.88"/>
99 </edge>
100 <edge id="E10" from="J4" to="J5" priority="-1">
101   <lane id="E10_0" index="0" speed="13.89" length="32.06"
102     ↪ shape="-4.84,11.59 -5.47,43.64"/>
103 </edge>
104 <edge id="E0" from="J0" to="J4" priority="-1" spreadType="roadCenter">
105   <lane id="E0_0" index="0" speed="13.89" length="50.56"
106     ↪ shape="-64.08,2.26 -13.52,2.66"/>
107 </edge>
108 <edge id="E00" from="J4" to="J1" priority="-1">
109   <lane id="E00_0" index="0" speed="13.89" length="62.73"
110     ↪ shape="0.95,2.78 63.68,3.27"/>
111 </edge>
112 <edge id="E1" from="J5" to="J4" priority="-1">
113   <lane id="E1_0" index="0" speed="13.89" length="32.06"
114     ↪ shape="-8.67,43.58 -8.03,11.52"/>
115 </edge>

```

```

107 <edge id="E10" from="J4" to="J6" priority="1">
108   <lane id="E10_0" index="0" speed="13.89" length="37.37"
109     shape="-7.77,-2.94 -7.11,-40.31"/>
110 </edge>
111
112 <tiLogic id="J4" type="static" programID="0" offset="0">
113   <phase duration="42" state="GGrrrGGrrr"/>
114   <phase duration="3" state="yyyyrrrrrrr"/>
115   <phase duration="42" state="rrrGGrrrGGg"/>
116   <phase duration="3" state="rrrrrrrrrry"/>
117 </tiLogic>
118
119 <junction id="J0" type="dead_end" x="-64.09" y="3.86" inclanes="-E00_0"
120   shape="-64.09,3.86 -64.12,7.06 -64.09,3.86"/>
121 <junction id="J1" type="dead_end" x="63.67" y="4.87" inclanes="E00_0"
122   shape="63.67,4.87 63.70,1.67 63.67,4.87"/>
123 <junction id="J4" type="traffic_light" x="-6.29" y="4.32" inclanes="-E0_0
124   -E1_0 E0_0 E1_0" inclanes=":J4_0_0 :J4_1_0 :J4_12_0 :J4_3_0 :J4_4_0
125   :J4_13_0 :J4_6_0 :J4_7_0 :J4_14_0 :J4_9_0 :J4_10_0 :J4_15_0"
126   shape="0.92,7.58 0.97,1.18 -1.25,0.71 -2.02,0.15 -2.56,-0.63
127   -2.87,-1.63 -2.96,-2.85 -9.37,-2.97 -9.86,-0.71 -10.44,0.07
128   -11.24,0.63 -12.26,0.96 -13.50,1.06 -13.55,7.46 -11.34,7.92
129   -10.57,8.49 -10.03,9.27 -9.72,10.27 -9.63,11.49 -3.24,11.62 -2.74,9.36
130   -2.16,8.57 -1.36,8.01 -0.33,7.68">
131   <request index="0" response="000000000000" flows="000100010000"
132     cont="0"/>
133   <request index="1" response="100000100000" flows="111100110000"
134     cont="0"/>
135   <request index="2" response="100111100000" flows="110111110000"
136     cont="1"/>
137   <request index="3" response="000010000000" flows="100010000000"
138     cont="0"/>
139   <request index="4" response="000110000111" flows="100110000111"
140     cont="0"/>
141   <request index="5" response="1111110000110" flows="111110000110"
142     cont="1"/>
143   <request index="6" response="000000000000" flows="010000000100"
144     cont="0"/>
145   <request index="7" response="100000100000" flows="110000111100"
146     cont="0"/>
147   <request index="8" response="100000100011" flows="110000110111"
148     cont="1"/>

```

```

130     <request index="9" response="000000000010" fees="000000100010"
131     ↪ cont="0"/>
132     <request index="10" response="000111000110" fees="000111100110"
133     ↪ cont="0"/>
134     <request index="11" response="000110011110" fees="000110111110"
135     ↪ cont="1"/>
136 </junction>
137 <junction id="J5" type="dead_end" x="-7.07" y="43.61" inclanes="-E10_0"
138 ↪ inclanes="" shape="-7.07,43.61 -3.87,43.67 -7.07,43.61"/>
139 <junction id="J6" type="dead_end" x="-5.51" y="-40.28" inclanes="E10_0"
140 ↪ inclanes="" shape="-5.51,-40.28 -8.71,-40.34 -5.51,-40.28"/>
141
142 <junction id="J4_12_0" type="internal" x="-3.08" y="5.30"
143 ↪ inclanes="J4_2_0 E0_0" inclanes="J4_4_0 J4_5_0 J4_6_0 J4_7_0
144 ↪ J4_10_0 J4_11_0"/>
145 <junction id="J4_13_0" type="internal" x="-5.28" y="1.13"
146 ↪ inclanes="J4_5_0 E1_0" inclanes="J4_1_0 J4_2_0 J4_7_0 J4_8_0
147 ↪ J4_9_0 J4_10_0"/>
148 <junction id="J4_14_0" type="internal" x="-9.51" y="3.33"
149 ↪ inclanes="J4_8_0 -E0_0" inclanes="J4_0_0 J4_1_0 J4_4_0 J4_5_0
150 ↪ J4_10_0 J4_11_0"/>
151 <junction id="J4_15_0" type="internal" x="-7.32" y="7.53"
152 ↪ inclanes="J4_11_0 -E1_0" inclanes="J4_1_0 J4_2_0 J4_3_0 J4_4_0
153 ↪ J4_7_0 J4_8_0"/>
154
155 <connection from="-E0" to="-E10" fromlane="0" tolane="0" via="J4_0_0"
156 ↪ tl="J4" linkindex="0" dir="r" state="0"/>
157 <connection from="-E0" to="-E00" fromlane="0" tolane="0" via="J4_1_0"
158 ↪ tl="J4" linkindex="1" dir="s" state="0"/>
159 <connection from="-E0" to="E10" fromlane="0" tolane="0" via="J4_2_0"
160 ↪ tl="J4" linkindex="2" dir="l" state="0"/>
161 <connection from="-E1" to="E00" fromlane="0" tolane="0" via="J4_3_0"
162 ↪ tl="J4" linkindex="3" dir="r" state="0"/>
163 <connection from="-E1" to="-E10" fromlane="0" tolane="0" via="J4_4_0"
164 ↪ tl="J4" linkindex="4" dir="s" state="0"/>
165 <connection from="-E1" to="-E00" fromlane="0" tolane="0" via="J4_5_0"
166 ↪ tl="J4" linkindex="5" dir="l" state="0"/>
167 <connection from="E0" to="E10" fromlane="0" tolane="0" via="J4_6_0"
168 ↪ tl="J4" linkindex="6" dir="r" state="0"/>
169 <connection from="E0" to="E00" fromlane="0" tolane="0" via="J4_7_0"
170 ↪ tl="J4" linkindex="7" dir="s" state="0"/>

```

```

150 <connection from="E0" to="E10" fromLane="0" toLane="0" via="j4_8_0"
151   ⌞ tl="j4" linkIndex="8" dir="l" state="a"/>
152 <connection from="E1" to="E00" fromLane="0" toLane="0" via="j4_9_0"
153   ⌞ tl="j4" linkIndex="9" dir="r" state="a"/>
154 <connection from="E1" to="E10" fromLane="0" toLane="0" via="j4_10_0"
155   ⌞ tl="j4" linkIndex="10" dir="a" state="a"/>
156 <connection from="E1" to="E00" fromLane="0" toLane="0" via="j4_11_0"
157   ⌞ tl="j4" linkIndex="11" dir="l" state="a"/>
158
159 <connection from="j4_0" to="E10" fromLane="0" toLane="0" dir="r"
160   ⌞ state="M"/>
161 <connection from="j4_1" to="E00" fromLane="0" toLane="0" dir="a"
162   ⌞ state="M"/>
163 <connection from="j4_2" to="E10" fromLane="0" toLane="0" via="j4_12_0"
164   ⌞ dir="l" state="m"/>
165 <connection from="j4_12" to="E10" fromLane="0" toLane="0" dir="l"
166   ⌞ state="M"/>
167 <connection from="j4_3" to="E00" fromLane="0" toLane="0" dir="r"
168   ⌞ state="M"/>
169 <connection from="j4_4" to="E10" fromLane="0" toLane="0" dir="a"
170   ⌞ state="M"/>
171 <connection from="j4_5" to="E00" fromLane="0" toLane="0"
172   ⌞ via="j4_13_0" dir="l" state="m"/>
173 <connection from="j4_13" to="E00" fromLane="0" toLane="0" dir="l"
174   ⌞ state="M"/>
175 <connection from="j4_6" to="E10" fromLane="0" toLane="0" dir="r"
176   ⌞ state="M"/>
177 <connection from="j4_7" to="E00" fromLane="0" toLane="0" dir="a"
178   ⌞ state="M"/>
179 <connection from="j4_8" to="E10" fromLane="0" toLane="0"
180   ⌞ via="j4_14_0" dir="l" state="m"/>
181 <connection from="j4_14" to="E10" fromLane="0" toLane="0" dir="l"
182   ⌞ state="M"/>
183 <connection from="j4_9" to="E00" fromLane="0" toLane="0" dir="r"
184   ⌞ state="M"/>
185 <connection from="j4_10" to="E10" fromLane="0" toLane="0" dir="a"
186   ⌞ state="M"/>
187 <connection from="j4_11" to="E00" fromLane="0" toLane="0"
188   ⌞ via="j4_15_0" dir="l" state="m"/>
189 <connection from="j4_15" to="E00" fromLane="0" toLane="0" dir="l"
190   ⌞ state="M"/>
191

```

```
170 </net>
```

C.3 Arquivo de Rotas da Simulação (sumo2.rou.xml)

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!-- generated on 2025-07-27 12:09:19 by Eclipse SUMO netedit Version 1.23.1 -->
4
5
6 <routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7   xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
8   <!-- Vehicles, persons and containers (sorted by depart) -->
9   <flow id="e1" begin="0.00" color="red" from="-E0" to="-E10"
10     end="3600.00" vehsPerHour="100"/>
11   <flow id="ew1" begin="0.00" color="red" from="-E0" to="-E00"
12     end="3600.00" vehsPerHour="100"/>
13   <flow id="f_0" begin="0.00" color="red" from="-E0" to="E10"
14     end="3600.00" vehsPerHour="100"/>
15   <flow id="f_1" begin="0.00" color="magenta" from="E1" to="-E00"
16     end="3600.00" vehsPerHour="100"/>
17   <flow id="ne1" begin="0.00" color="magenta" from="E1" to="E00"
18     end="3600.00" vehsPerHour="100"/>
19   <flow id="ns1" begin="0.00" color="magenta" from="E1" to="E10"
20     end="3600.00" vehsPerHour="100"/>
21   <flow id="se1" begin="0.00" color="0,0,128" from="-E1" to="E00"
22     end="3600.00" vehsPerHour="100"/>
23   <flow id="s1" begin="0.00" color="0,0,128" from="-E1" to="-E10"
24     end="3600.00" vehsPerHour="100"/>
25   <flow id="sw1" begin="0.00" color="0,0,128" from="-E1" to="-E00"
26     end="3600.00" vehsPerHour="100"/>
27   <flow id="we1" begin="0.00" from="E0" to="E00" end="3600.00"
28     vehsPerHour="100"/>
29   <flow id="w1" begin="0.00" from="E0" to="-E10" end="3600.00"
30     vehsPerHour="100"/>
31   <flow id="w1" begin="0.00" from="E0" to="E10" end="3600.00"
32     vehsPerHour="100"/>
33 </routes>

```