ESCOLA TÉCNICA ESTADUAL FREDERICO GUILHERME SCHMIDT

TÉCNICO EM ELETROTÉCNICA

MIPI - MONITORAMENTO INTELIGENTE DE PROCESSOS INDUSTRIAIS

ALAN RODRIGUES DOS SANTOS

ALESSANDRA CLARA CHAVES

ANDRYA DOS SANTOS PALINSKI

SÃO LEOPOLDO

2025

ALAN RODRIGUES DOS SANTOS

ALESSANDRA CLARA CHAVES

ANDRYA DOS SANTOS PALINSKI

MIPI - MONITORAMENTO INTELIGENTE DE PROCESSOS INDUSTRIAIS

Projeto de Trabalho de Conclusão de Curso Técnico apresentado ao Curso de Eletrotécnica da Escola Técnica Estadual Frederico Guilherme Schmidt como requisito para aprovação nas disciplinas do curso sob orientação do professor Adriano Santos e coorientação do professor André Vigano de Oliveira.

SÃO LEOPOLDO

2025

RESUMO

Este projeto apresenta o desenvolvimento de um protótipo de monitoramento inteligente de processos industriais utilizando sensores integrados à plataforma Arduino. O objetivo é propor uma solução acessível e eficiente para pequenas e médias empresas, possibilitando a coleta e análise de parâmetros elétricos e térmicos em tempo real. A justificativa da pesquisa está na necessidade de aumentar a segurança operacional, prevenir falhas e reduzir custos com manutenção, ao mesmo tempo em que se busca maior eficiência produtiva. A metodologia adotada foi de caráter quantitativo e descritivo, iniciando-se com uma revisão exploratória sobre sistemas de monitoramento, seguida da construção do protótipo. O sistema foi implementado com Arduino Uno, sensor de temperatura LM35, sensor de corrente e tensão INA219 e módulo de ventilação, compondo uma estrutura de baixo custo e fácil replicação. Os dados são apresentados por meio de uma interface de *software* dedicada, permitindo melhor visualização e análise das medições. Os testes realizados demonstraram que o protótipo é capaz de registrar e processar dados de tensão, corrente, potência e temperatura com confiabilidade, além de realizar o controle automático e manual da velocidade do ventilador conforme as condições de operação. Conclui-se que a proposta representa uma alternativa viável de modernização centrada nos conceitos da Indústria 4.0, fornecendo às empresas recursos de monitoramento em tempo real de maneira simples, precisa e econômica.

Palavras-chave: Monitoramento industrial; Arduino; Sensores; Automação; Indústria 4.0.

ABSTRACT

This project presents the development of an intelligent monitoring prototype for industrial processes using sensors integrated with the Arduino platform. The objective is to propose an accessible and efficient solution for small and medium-sized enterprises, enabling the collection and analysis of electrical and thermal parameters in real time. The justification for this research lies in the need to increase operational safety, prevent failures, and reduce maintenance costs, while also seeking greater production efficiency. The adopted methodology was quantitative and descriptive, beginning with an exploratory review of monitoring systems, followed by the construction of the prototype. The system was implemented using an Arduino Uno, an LM35 temperature sensor, an INA219 current and voltage sensor, and a ventilation module, forming a low-cost and easily replicable structure. The data are displayed through a dedicated software interface, allowing better visualization and analysis of the measurements. The tests carried out demonstrated that the prototype is capable of reliably recording and processing voltage, current, power, and temperature data, in addition to performing automatic and manual control of fan speed according to operating conditions. It is concluded that the proposal represents a viable modernization alternative aligned with the concepts of Industry 4.0, providing companies with real-time monitoring resources in a simple, accurate, and economical manner.

Keywords: Industrial monitoring; Arduino; Sensors; Automation; Industry 4.0.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

V	Volts (Tensão)
W	Watts (Potência)
CC	Corrente contínua

SUMÁRIO

1. INTRODUÇÃO

Na indústria, o monitoramento inteligente de máquinas e processos automáticos é fundamental para assegurar a integridade das máquinas, prevenir falhas e otimizar o funcionamento e a precisão dos equipamentos. Esse tipo de acompanhamento é aplicado em fábricas automobilísticas, alimentícias e petroquímicas, onde o sistema de controle é conectado às máquinas, e através de sensores integrados à plataforma digital de vigilância, onde será possível coletar os dados necessários para a avaliação, como temperatura, consumo elétrico e desempenho mecânico em tempo real. Esses dados são transmitidos e verificados por *softwares*, que contribuem nas tomadas de decisão, na prevenção de acidentes e na redução de custos industriais, como a manutenção de equipamentos. O monitoramento pode ser realizado por meio de conexões via cabo ou por *Wi-Fi*, e conforme a infraestrutura disponível na empresa.

Dessa forma, o projeto tem como objetivo o desenvolvimento de um protótipo de sistema de supervisão inteligente, utilizando sensores integrados à plataforma Arduino. A proposta é realizar a coleta de dados como temperatura, corrente elétrica e tensão elétrica utilizando uma tecnologia de baixo custo, eficiente e acessível, não apenas para grandes empresas, mas também para pequenas e médias empresas.

O protótipo será construído com o uso de uma placa Arduino Uno como controladora central, conectada a sensores específicos: LM35 para medição de temperatura, e sensores de ventilação para análise da refrigeração.

Os dados coletados serão processados pelo arduino e enviados para uma interface digital, que irá apresentar as informações sobre a tensão, corrente e temperatura do mini cooler. Os dados serão apresentados tanto de forma numérica quanto em gráfico, juntamente com opções para ligar e controlar a velocidade do ventilador. O sistema será alimentado por uma fonte de 12V e projetado para operar em ambientes industriais. Dito isso, esperamos desenvolver uma alternativa viável e acessível ao monitoramento industrial inteligente, contribuindo com soluções tecnológicas práticas para pequenas e médias empresas.

1.1 TEMA E SUA DELIMITAÇÃO

Sistema para monitorar equipamentos na indústria, fazendo a supervisão desses maquinários e seu funcionamento e desempenho, precavendo acidentes e possíveis perdas no setor de produção, visando a eficiência, baixo custo e acessibilidade às empresas.

1.2 PROBLEMA

É possível desenvolver um sistema de monitoramento para aparelhos no setor industrial, que garanta a coleta de dados desses equipamentos e previna possíveis acidentes?

1.3 OBJETIVOS

1.3.1 Objetivo geral

Desenvolver um sistema integrado de monitoramento inteligente de processos industriais, utilizando sensores de temperatura, corrente, ventilação e Arduino, para capturar dados operacionais no ambiente industrial.

1.3.2 Objetivo específico

- Projetar e implementar um sistema de monitoramento inteligente utilizando a plataforma Arduino, capaz de coletar e processar dados em tempo real.
- Analisar e desenvolver a lógica de programação necessária para o funcionamento do sistema de monitoramento, garantindo a integração eficaz dos componentes e a tomada de decisões precisas.

- Construir a plataforma de monitoramento, integrando os sensores e componentes necessários para a coleta de dados e o funcionamento do sistema.
- Realizar a conexão da plataforma de monitoramento ao Arduino, garantindo a comunicação eficaz entre os componentes e o processamento dos dados coletados.

1.4 JUSTIFICATIVA

O ambiente industrial exige uma série de controles rigorosos sobre diversos parâmetros para garantir a eficiência da produção e a segurança dos trabalhadores. Sensores de temperatura são essenciais para evitar superaquecimento de máquinas e processos; sensores de corrente elétrica são fundamentais para monitorar o consumo e o desempenho elétrico, evitando sobrecargas e falhas; e sensores de ventilação, essenciais para garantir a circulação de ar e refrigeração.

Embora já existam sistemas que realizam esse tipo de coleta via cabos, como *Ethernet/IP* ou *PROFINET*, sendo uma tecnologia que utiliza o padrão *Ethernet* para comunicação em ambientes industriais, oferecendo alta velocidade e flexibilidade, embora apresentem como principal limitação a curta distância entre as máquinas e a central de controle. Já os sistemas via *Wi-Fi*, como *MindSphere* da *Siemens,* é uma plataforma de Internet Industrial das Coisas *(IIoT)* em nuvem que permite conectar máquinas, plantas e sistemas de forma a obter dados para otimizar operações, criar produtos de melhor qualidade e implementar novos modelos de negócio. Embora ofereçam maior liberdade de posicionamento e amplitude de atividade, exigem uma infraestrutura mais cara, organizada e complexa de parte da empresa. A proposta deste trabalho busca oferecer um sistema de monitoramento intermediário entre essas duas soluções, que permita monitoramento a média e longa distância, mantendo baixo custo e eficiência comparável às tecnologias convencionais, além de uma disponibilidade maior para empresas de pequeno e médio porte.

O equipamento de monitoramento inteligente será aplicado em diversas indústrias, por meio de comunicação digital em tempo real, proporcionando benefícios significativos para as empresas. A partir disso, é possível identificar

problemas antes que eles causem falhas e paradas de produção. Além disso, a coleta de dados possibilita o ajuste dos processos para melhorar a eficiência e reduzir custos operacionais, otimizando o funcionamento das empresas. O monitoramento de parâmetros críticos, como temperatura e corrente elétrica, também contribui para a prevenção de acidentes e garantia da segurança dos trabalhadores.

Ademais, a manutenção preventiva e a otimização dos processos permitem reduzir os custos de manutenção e operação, melhorando a rentabilidade das empresas. Dessa forma, o equipamento de monitoramento inteligente pode ser uma ferramenta valiosa para as empresas que buscam melhorar a eficiência, reduzir custos e garantir a segurança dos trabalhadores. Sua aplicação pode ser feita em diversas indústrias, como as fábricas automobilísticas, indústria alimentícia, petroquímica, entre outras.

2. ESTADO DA ARTE

O avanço das tecnologias aplicadas à automação industrial e ao monitoramento inteligente de processos tem proporcionado soluções cada vez mais eficazes para otimizar o controle, a análise e a manutenção de sistemas produtivos. A conectividade entre máquinas, sensores e plataformas digitais, possibilitada pela Internet das Coisas Industrial (IIoT), é hoje um dos pilares da Indústria 4.0. Nesse contexto, é fundamental compreender o estado atual das tecnologias disponíveis, os dispositivos e sistemas já desenvolvidos, de forma a identificar referências, limitações e oportunidades de inovação que embasam o desenvolvimento do projeto proposto.

Tabela 1 - Estado da arte

Pesquisa	Autoria	Ano de publicação
Siemens MindSphere	Siemens	2017
Modbus RTU (RS-485)	Modicon	1979
Profibus	Johan Sartwish Wilman	1987
Tractian	Igor Marinelli	2019

Fonte: Os autores

2.1 TECNOLOGIAS DE CONEXÃO SEM FIO (WIRELESS)

2.1.1 Siemens MindSphere

O Siemens MindSphere (2017) é uma plataforma de Internet das Coisas (IoT) industrial baseada em nuvem, com foco na conectividade de dispositivos físicos (edge devices) para análise e otimização de processos industriais. Seu diferencial está na capacidade de coletar e processar dados de máquinas e sensores em tempo

real, fornecendo painéis de análise para manutenção preditiva, eficiência energética e tomada de decisão estratégica. O sistema é aberto e compatível com diversos protocolos e APIs, o que amplia sua aplicabilidade em diferentes setores industriais.

2.1.2 AWS IoT

O AWS IoT, da Amazon, é um conjunto robusto de serviços gerenciados para conectar dispositivos à nuvem, com destaque para escalabilidade e confiabilidade. Ele permite a coleta, armazenamento e análise de dados de sensores em ambientes industriais exigentes. Empresas como a Volkswagen e Pentair utilizam essa tecnologia para otimizar processos e melhorar a qualidade dos produtos, com suporte a inteligência artificial e aprendizado de máquina.

2.2 TECNOLOGIAS DE CONEXÃO COM FIO

2.2.1 Modbus RTU (RS-485)

O Modbus RTU é um protocolo de comunicação serial amplamente utilizado na indústria desde 1979. Opera com barramentos RS-485 ou RS-232, permitindo a comunicação entre dispositivos mestre e escravo de forma simples, estável e com baixo custo. Seu uso é comum em controle de processos, supervisão de máquinas e coleta de dados em ambientes industriais robustos. Ainda hoje, é uma solução confiável para sistemas de automação que não exigem altas taxas de transferência.

2.2.2 Profibus

Desenvolvido nos anos 1980, o Profibus é um protocolo digital de campo (fieldbus) projetado para permitir a comunicação entre diversos dispositivos, como controladores lógicos programáveis (CLPs), sensores e atuadores. O Profibus-DP é ideal para o chão de fábrica, com alta velocidade de transmissão, enquanto o Profibus-PA é voltado à automação de processos, permitindo alimentação e comunicação por um único cabo. Sua padronização e confiabilidade o tornaram um dos protocolos mais difundidos em automação.

2.3 TECNOLOGIA BRASILEIRA RELACIONADA: TRACTIAN

A Tractian é uma startup brasileira que desenvolve soluções de manutenção preditiva e monitoramento inteligente de máquinas industriais. Seus sensores sem fio capturam dados como vibração, temperatura e consumo de energia, que são enviados para a nuvem e analisados por algoritmos inteligentes. A plataforma permite a visualização em tempo real e o diagnóstico de falhas iminentes, ajudando empresas a evitarem paradas não planejadas. Sua aplicação prática demonstra como sensoriamento e conectividade inteligente podem transformar a gestão industrial.

2.4 COMPARAÇÃO ENTRE TECNOLOGIAS E CONTRIBUIÇÃO DO PROJETO

Tabela 2 - Comparação de tecnologias

Tecnologia/ Plataforma	**Tipo de Conexão**	**Nível de Aplicação**	**Recursos**	**Limitações**
Siemens MindSphere	Sem fio (nuvem)	Supervisão e decisão	Análise de dados, dashboards	Dependência de internet
Modbus RTU (RS-485)	Com fio (RS-485)	Aquisição local	Estável e simples	Baixa velocidade
AWS IoT	Sem fio (nuvem)	Alta escala e integração	IA, análise em tempo real	Custo e complexidade
Tractian	Sem fio (nacional)	Preditiva	Instalação fácil, uso prático	Limitada a sensores proprietários

Fonte: Os autores

Este projeto propõe o desenvolvimento de uma solução de monitoramento inteligente de processos industriais utilizando sensores e conectividade com Arduino,

com base em tecnologias de acesso aberto e fácil integração. Seu diferencial está na simplicidade, baixo custo e possibilidade de personalização, voltada para contextos educacionais e de pequena e média indústria. A proposta visa unir recursos acessíveis com conceitos da Indústria 4.0, fomentando inovação e aprendizado

3. FUNDAMENTAÇÃO TEÓRICA

3.1. TECNOLOGIA ARDUINO

A tecnologia Arduino consiste em uma plataforma de prototipagem eletrônica de código aberto, desenvolvida para facilitar a criação de projetos interativos e sistemas embarcados. Ela é composta por placas com microcontroladores (principalmente da família AVR, como o ATmega328P) e um ambiente de desenvolvimento integrado (IDE) que permite programar em uma linguagem derivada de C/C++. Seu diferencial está na facilidade de uso, no baixo custo e na vasta comunidade de desenvolvedores, o que a torna acessível para estudantes, pesquisadores e profissionais da área de tecnologia e automação.

Conforme descrito por UsinaInfo (2024), a plataforma oferece suporte a uma ampla variedade de sensores e módulos de comunicação (como *Wi-Fi, Bluetooth, LoRa*, entre outros), possibilitando aplicações em monitoramento remoto, controle de processos industriais, robótica, automação residencial e coleta de dados. Além disso, sua arquitetura modular e expansível facilita a integração com outras tecnologias emergentes, como *IoT* e Indústria 4.0. Nas pequenas e médias indústrias, o Arduino tem se destacado como alternativa viável para o desenvolvimento de sistemas inteligentes personalizados, que aliam praticidade e eficiência.

3.2. MICROCONTROLADORES

Os microcontroladores são componentes fundamentais em sistemas embarcados, pois integram em um único circuito a CPU (unidade central de processamento), memória (RAM e ROM), periféricos de entrada/saída e interfaces de comunicação. Eles são projetados para executar tarefas específicas com eficiência, precisão e economia de energia, sendo largamente utilizados em aplicações que requerem respostas em tempo real.

De acordo com a Victor Vision (2024), microcontroladores como o ESP32, Arduino Uno, Mega e Nano vêm sendo amplamente empregados em projetos que envolvem automação industrial, monitoramento ambiental, robótica e IoT, pela sua capacidade de controlar sensores, processar dados e atuar de forma autônoma. Seu uso em ambientes industriais tem crescido devido à versatilidade, escalabilidade e compatibilidade com bibliotecas e protocolos de comunicação industrial, como Modbus, MQTT e Ethernet. O uso de microcontroladores permite minimizar custos, ao mesmo tempo em que promove otimização e controle preciso de processos industriais.

3.3 SISTEMAS INTEGRADOS

Sistemas integrados referem-se à união de diversos componentes tecnológicos — como sensores, microcontroladores, *softwares* de gestão e interfaces homem-máquina (*IHM*) — que operam de forma sincronizada para desempenhar funções específicas em tempo real. Eles são essenciais para o funcionamento da automação moderna, especialmente em ambientes industriais que requerem conectividade, rastreabilidade e eficiência operacional.

Segundo a TOTVS (2024), a adoção de sistemas integrados permite centralizar e padronizar informações, promovendo melhoria na comunicação entre setores, agilidade na tomada de decisões, e redução de falhas humanas. Além disso, conforme Mendes e Filho (2023), esses sistemas viabilizam a integração de dados da produção com sistemas de gestão (ERP, MES e SCM), permitindo que empresas operem com maior controle, flexibilidade e inteligência operacional.

No contexto de projetos com Arduino e sensores, os sistemas integrados garantem a fluidez entre a coleta de dados, o processamento das informações e a visualização em tempo real, sendo peça-chave na implementação de soluções personalizadas de monitoramento inteligente.

3.4 TECNOLOGIAS DE MONITORAMENTO INDUSTRIAL

As tecnologias de monitoramento industrial têm passado por grandes transformações com a incorporação de sensores inteligentes, sistemas embarcados

e conectividade em rede. Elas possibilitam o acompanhamento contínuo de variáveis críticas de processo (como temperatura, vibração, corrente, umidade, entre outros), com o objetivo de diagnosticar falhas precocemente, prever manutenções e otimizar o desempenho operacional.

De acordo com o SENAI-ES (2024), o monitoramento da eficiência operacional por meio de sensores conectados a sistemas como SCADA, IoT e dashboards web contribui diretamente para a melhoria contínua, segurança do trabalho e sustentabilidade industrial. Além disso, com o avanço da Internet das Coisas, torna-se possível enviar dados em tempo real para plataformas na nuvem, facilitando o acesso remoto e o controle distribuído.

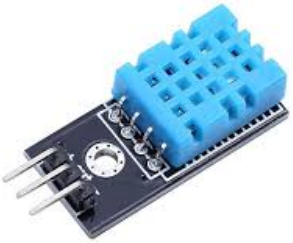
A IBM (2024) destaca o papel da IIoT (Internet das Coisas Industrial) na construção de ambientes industriais inteligentes, onde máquinas, sensores e softwares atuam em rede para garantir eficiência, previsibilidade e autonomia operacional. No caso de aplicações com Arduino, essas tecnologias têm permitido a criação de soluções customizadas para pequenas e médias empresas, com alto impacto e baixo investimento.

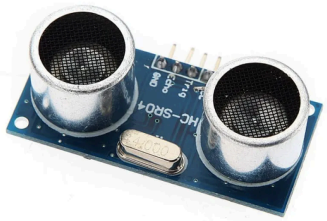
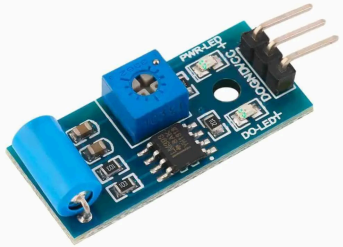
3.5 SENSORES E COLETA DE DADOS

Os sensores industriais são responsáveis por captar variáveis físicas, químicas ou ambientais e pela conversão desses dados em sinais que possam ser interpretados por microcontroladores ou sistemas computacionais. Eles são essenciais para a automação e o monitoramento em tempo real de processos industriais.

Segundo a EGA (2024), os sensores inteligentes vão além da simples medição — eles realizam autodiagnóstico, calibração automática, comunicação digital e até análise pré-processada, aumentando a confiabilidade do sistema e reduzindo a necessidade de intervenção humana. Dentre os sensores mais comuns utilizados em projetos com Arduino, destacam-se: LM35, DHT11 e DHT22 – Medição de temperatura e umidade; ACS712, INA219 – Medição de corrente elétrica; HC-SR04 – Medição de distância por ultrassom; SW-420 – Detecção de vibração; MQ-2, MQ-135 – Medição de gases e qualidade do ar.

Tabela 3 - Sensores e suas funções.

Sensores	Função	Fonte de imagem
	O Sensor de Temperatura LM35 é um dispositivo eletrônico utilizado para medir a temperatura ambiente com alta precisão. Com uma faixa de operação que vai de -55°C a +150°C, o LM35DZ oferece uma ampla gama de aplicações em diversos ambientes.	Baú da tecnologia.
	O Sensor de Umidade e Temperatura DHT11 opera com base em um sensor capacitivo para medir a umidade relativa do ar e um termistor para medir a temperatura ambiente. Ele possui uma membrana porosa que permite que a umidade do ar entre em contato com o sensor capacitivo, alterando sua capacitância de acordo com a umidade.	Auto core Robotica.
	O ACS712 funciona com o princípio do efeito *Hall*, onde uma corrente elétrica que flui através de um condutor gera um campo magnético proporcional à sua intensidade. Este campo magnético é detectado por uma célula de *Hall* integrada, que o converte numa tensão de saída linear.	Usinainfo.

	O Sensor Ultrassônico HC-SR04 é um dispositivo de detecção de distância compacto e eficiente, capaz de medir com precisão objetos em um intervalo de 2 centímetros a 4 metros.	Eletrodex.
	O SW-420 é um módulo sensor de vibração digital que, através de um mecanismo interno (semelhante a um interruptor de inclinação ou "*tilt switch*"), detecta movimentos e vibrações. Ele não mede a intensidade da vibração, mas sim se ela ultrapassa um limite de sensibilidade ajustável.	Casa da Robótica.
	O sensor MQ-2 é um sensor de gás comumente usado para detectar a presença de gases inflamáveis e fumaça. Ele funciona através da variação da resistência elétrica do sensor em função da concentração do gás detectado.	Piscaled.

Fonte: Os autores

A coleta de dados pode ser feita por meio de leitura periódica, armazenamento local (cartões SD) ou envio remoto (por *Wi-Fi, Bluetooth, LoRa ou MQTT*). Esses dados são então utilizados para análise de desempenho, diagnóstico de falhas e controle em tempo real, elevando o nível de automação e inteligência do processo industrial.

3.6 SOFTWARE

Software é o conjunto de instruções, dados ou programas que possibilita a operação de dispositivos computacionais e a realização de tarefas específicas. Ele representa a parte lógica de um sistema computacional, sendo o responsável por orientar o hardware a executar funções determinadas. Conforme destacado pela Lenovo (2024), o *software* pode ser entendido como o cérebro por trás do funcionamento de dispositivos digitais, desempenhando papel central na transformação de comandos lógicos em ações práticas.

Os *softwares* são classificados em diferentes categorias, de acordo com sua finalidade e aplicação:

- *Software* de sistema: responsável pela gestão dos recursos computacionais e pela interface entre o hardware e o usuário. Engloba os sistemas operacionais, como *Windows*, *macOS* e *Linux*, que garantem o funcionamento básico do computador e o gerenciamento de processos e arquivos.
- *Software* aplicativo: projetado para executar tarefas específicas, como edição de textos, navegação na internet, criação de planilhas, simulações e jogos. Esses softwares atendem diretamente às necessidades do usuário final.
- *Software* de programação: refere-se a ferramentas utilizadas por desenvolvedores para criar outros softwares, como compiladores, interpretadores, editores de código-fonte e depuradores. São fundamentais para o desenvolvimento de sistemas personalizados.
- *Software* de controle: utilizado para comandar e monitorar o funcionamento de máquinas, robôs e dispositivos industriais. Essa categoria é essencial em sistemas de automação e supervisão industrial, permitindo o controle preciso e em tempo real de processos produtivos.

No contexto da indústria, o software assume papel estratégico ao viabilizar o controle de processos, a automação de tarefas, a coleta e análise de dados operacionais e a comunicação entre sistemas. A utilização de softwares personalizados permite adaptar soluções às necessidades específicas de cada linha de produção, promovendo maior eficiência, integração e rastreabilidade. A

digitalização dos processos industriais, viabilizada pelo uso intensivo de software, é um dos pilares da Indústria 4.0 e da transformação digital no setor produtivo.

3.7 PROGRAMAÇÃO

Programação é o processo de criação de códigos que instruem dispositivos eletrônicos, como computadores e microcontroladores, a realizar tarefas específicas. Ela é realizada por meio de linguagens de programação como C, Python, Java, entre outras. O código-fonte resultante é interpretado ou compilado por um processador, que executa as instruções definidas.

De acordo com a Universidade da Tecnologia (2024), uma linguagem de programação define-se como um conjunto estruturado de regras sintáticas e semânticas utilizadas para escrever algoritmos e desenvolver aplicações computacionais. Essas linguagens são responsáveis por estabelecer uma ponte entre o raciocínio humano e a lógica que pode ser processada pelas máquinas, permitindo o controle preciso de dispositivos e sistemas.

Em sistemas embarcados, como os que utilizam a plataforma Arduino, a programação é essencial para o controle de sensores, atuadores e interfaces de comunicação. Através da codificação, é possível desenvolver algoritmos para automação de processos, monitoramento de variáveis e resposta a eventos em tempo real. Além disso, o uso de linguagens específicas, como a linguagem C++ adaptada para o ambiente Arduino, possibilita um controle direto do hardware com baixo consumo de recursos computacionais. A programação, portanto, constitui o elo vital entre o hardware físico e as funcionalidades inteligentes esperadas no contexto da automação e da Indústria 4.0.

3.8 INTERNET DAS COISAS (IoT)

A Internet das Coisas (IoT) é um paradigma tecnológico que promove a conexão entre objetos físicos e a internet, possibilitando a coleta, o

compartilhamento e a análise de dados em tempo real entre dispositivos, sistemas e usuários. Essa rede de dispositivos interconectados utiliza sensores, atuadores, microcontroladores e softwares para formar ambientes inteligentes e responsivos, capazes de operar com autonomia e adaptabilidade.

Segundo a Oracle (2024), a IoT transforma dados brutos em informações estratégicas por meio da análise contínua de sensores, permitindo que sistemas tomem decisões mais precisas e rápidas. A AWS (2024), por sua vez, destaca que a IoT não se limita à simples conectividade — ela envolve também a gerência de dispositivos, segurança, inteligência artificial e armazenamento em nuvem, viabilizando soluções escaláveis e eficientes.

No contexto industrial, a IoT desempenha papel central na chamada Indústria 4.0, permitindo a comunicação máquina a máquina (M2M), a integração com sistemas de gestão (como ERP, MES e SCADA), além da automação e controle de processos complexos. Essa conectividade resulta na otimização da produção, redução de falhas, aumento da segurança operacional e manutenção preditiva, antecipando falhas com base em dados históricos e padrões de comportamento.

Plataformas embarcadas como Arduino, quando integradas a redes IoT, permitem o desenvolvimento de sistemas de monitoramento remoto, controle de variáveis ambientais e operação em tempo real, tornando-se uma alternativa viável e econômica para aplicações industriais e educacionais.

3.9 INDÚSTRIA 4.0

A Indústria 4.0, também denominada de Quarta Revolução Industrial, representa a convergência entre o mundo físico e digital por meio da digitalização e automação inteligente dos processos produtivos. Seu conceito baseia-se na incorporação de tecnologias como sistemas ciberfísicos, inteligência artificial, robótica, *Big Data*, Internet das Coisas (IoT), computação em nuvem e análise

avançada de dados, permitindo a criação de fábricas inteligentes *(smart factories)* capazes de operar de forma autônoma e interconectada.

De acordo com o Sebrae (2024), o termo surgiu oficialmente na Alemanha em 2011 como parte de uma estratégia nacional de modernização da indústria. Desde então, sua adoção tem se expandido globalmente, influenciando diretamente a forma como os produtos são fabricados, distribuídos e otimizados. Já segundo a TOTVS (2024), a Indústria 4.0 promove eficiência energética, redução de custos, rastreabilidade e personalização em massa, além de permitir maior agilidade na tomada de decisões por meio do uso intensivo de dados e sensores.

A SAP (2024) ressalta que essa nova era industrial transforma toda a cadeia de suprimentos, com integração entre chão de fábrica e sistemas de gestão (ERP, MES, SCM), melhorando a produtividade e a previsibilidade operacional. Nesse contexto, dispositivos embarcados como Arduino, sensores e controladores programáveis desempenham papel central ao possibilitar o monitoramento em tempo real, a automação de processos e a execução de tarefas com base em algoritmos inteligentes.

A adoção da Indústria 4.0 representa, portanto, um diferencial estratégico e competitivo, promovendo sustentabilidade, inovação e resiliência frente às mudanças do mercado global.

4. METODOLOGIA

Neste projeto será utilizado método descritivo, buscando o objetivo de desenvolver um dispositivo de monitoramento inteligente de processos industriais com sensores e tecnologia Arduino.

A fase inicial do projeto é uma pesquisa exploratória, onde serão analisadas as tecnologias de monitoramento inteligentes utilizadas em diversas indústrias. Também serão realizadas pesquisas em artigos científicos, patentes e estudos de caso anteriores, como o *Tractian*, a fim de identificar os desafios e as oportunidades no desenvolvimento de um dispositivo de monitoramento inteligente. Além disso, investigaremos as propriedades dos sensores que serão empregados, como o LM35 ou o DHT11, avaliando sua aplicabilidade em ambientes industriais.

Este projeto adota ainda uma abordagem exploratória e explicativa, com o objetivo de compreender as tecnologias já existentes no setor de automação e monitoramento, ao mesmo tempo em que busca propor inovações acessíveis e de baixo custo. O foco é investigar os conceitos já estabelecidos e, paralelamente, explorar como o protótipo pode aprimorar esses padrões.

Será realizada também uma pesquisa aplicada, utilizando o método quantitativo. Nesse método, serão coletados dados referentes a variáveis como temperatura, corrente e tensão elétrica.

Com base nessas informações, iniciaremos o desenvolvimento de um protótipo funcional. Nesta etapa, serão selecionados e integrados os componentes necessários, como a placa Arduino Uno, sensores de temperatura e corrente, módulos de tensão e sistema de ventilação. O protótipo será projetado para validar a viabilidade do monitoramento em tempo real, documentando todas as etapas da construção, os desafios encontrados e as soluções implementadas.

Após a montagem, serão realizados testes experimentais para avaliar a funcionalidade e o desempenho do sistema. Os testes incluirão a medição de

parâmetros industriais simulados, a análise da precisão dos sensores e a confiabilidade da coleta e transmissão de dados. Também serão observados os limites operacionais do protótipo, como a estabilidade da interface, a capacidade de resposta e a eficiência energética.

4.1 TIPO DE PESQUISA

Os dados coletados durante os testes serão analisados quantitativamente e qualitativamente. No aspecto quantitativo, serão avaliados o consumo energético, a variação de temperatura, a estabilidade do sistema e a eficiência da ventilação. No aspecto qualitativo, será analisada a usabilidade do protótipo com base no *feedback* de profissionais da área industrial.

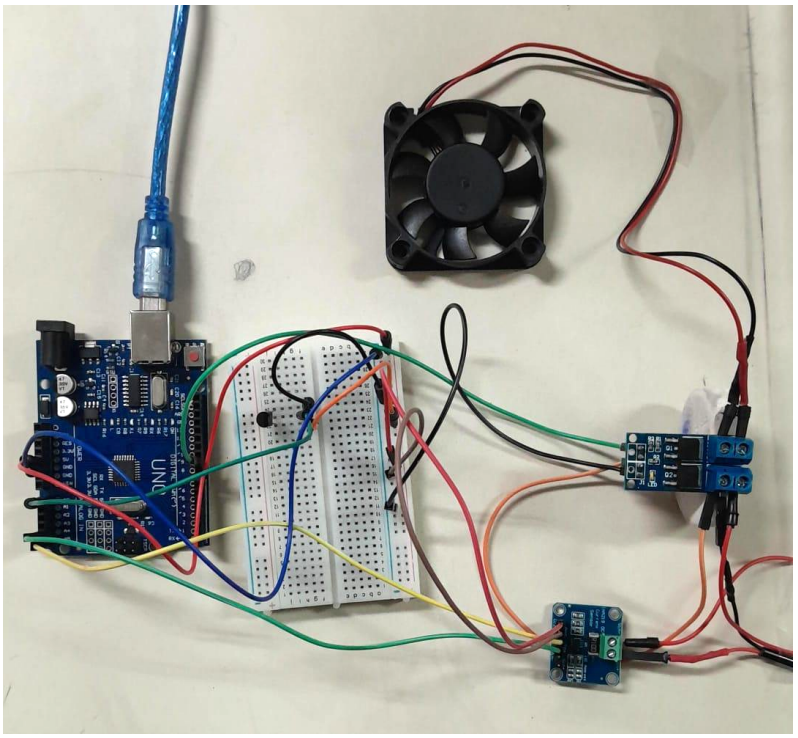
Com base nessas análises, o sistema poderá ser ajustado, se necessário, e recomendações serão apresentadas para futuras implementações em escala maior.

Por fim, todo o processo será documentado em relatório técnico, incluindo a revisão bibliográfica, o desenvolvimento do protótipo, os testes realizados, os resultados obtidos e as conclusões. Também será elaborada uma apresentação visual para demonstrar o funcionamento do sistema e divulgar os resultados à banca avaliadora e demais interessados.

4.3 ESQUEMA ELÉTRICO

A montagem do protótipo inicia-se com a preparação dos componentes e da bancada. Devem ser separados o Arduino Uno, o sensor de temperatura LM35, o módulo INA219, o driver MOSFET, a fonte de 12 volts, o cooler, a protoboard e fios jumper. Antes de qualquer ligação, é importante garantir que todas as fontes de alimentação estejam desligadas e que o circuito possua um ponto comum de aterramento (GND compartilhado entre a fonte, o driver e o Arduino), evitando possíveis danos aos componentes.

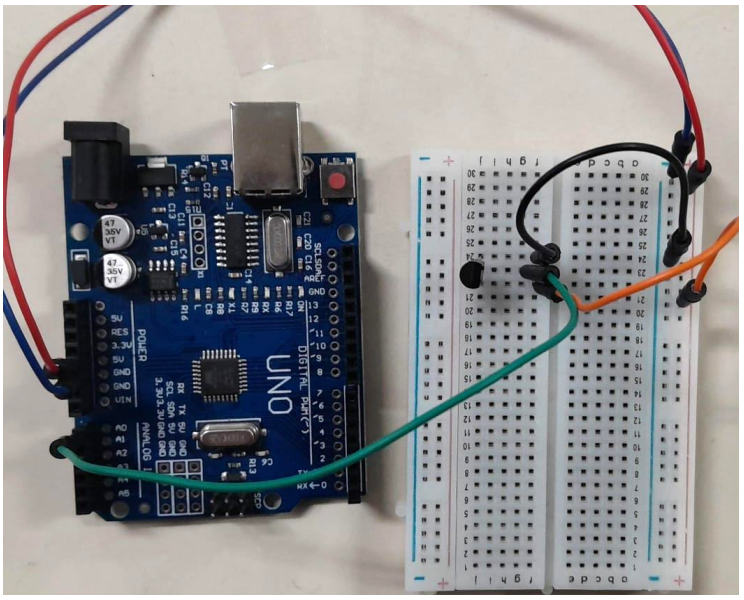
Imagem 1 - Esquema elétrico de base do protótipo.



Fonte: Autores.

A montagem começa pelo sensor LM35. Com o sensor posicionado de frente, o pino 1 (à esquerda) deve ser conectado ao terminal de 5V do Arduino, garantindo a alimentação do sensor. O pino central (pino 2), responsável pela saída de tensão proporcional à temperatura, deve ser ligado à entrada analógica A0 do Arduino. Por fim, o pino 3 (à direita) é conectado ao GND do Arduino, completando o circuito do sensor de temperatura.

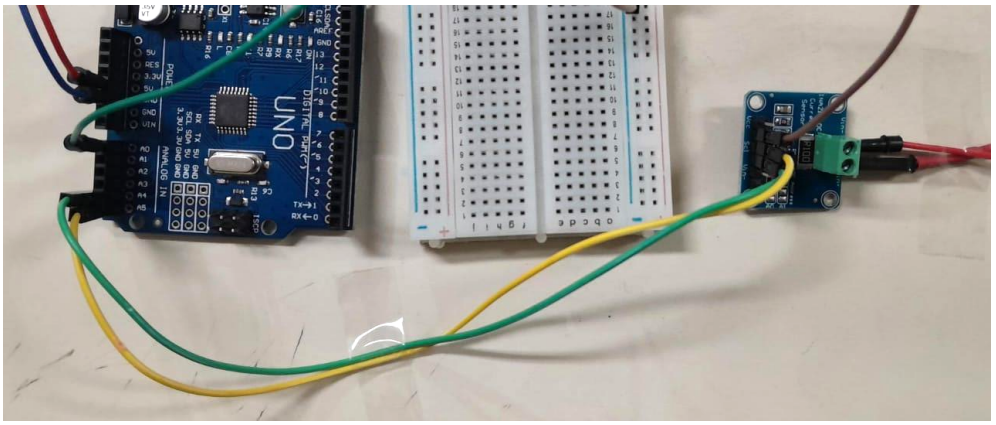
Imagem 2 - Início das conexões.



Fonte: Autores.

Em seguida, realiza-se a ligação do módulo INA219, responsável pela medição de tensão e corrente do sistema. O pino VCC do INA219 é conectado ao 5V do Arduino, enquanto o pino GND é ligado ao GND comum da protoboard. Os pinos de comunicação I2C — SDA e SCL — são conectados respectivamente aos pinos A4 e A5 do Arduino Uno. O terminal VIN+ do INA219 deve ser ligado ao polo positivo da fonte de 12V, e o terminal VIN− é conectado ao pino VIN+ do driver MOSFET, de forma que o INA219 fique em série com o circuito de alimentação do cooler, permitindo a medição de corrente e tensão com precisão.

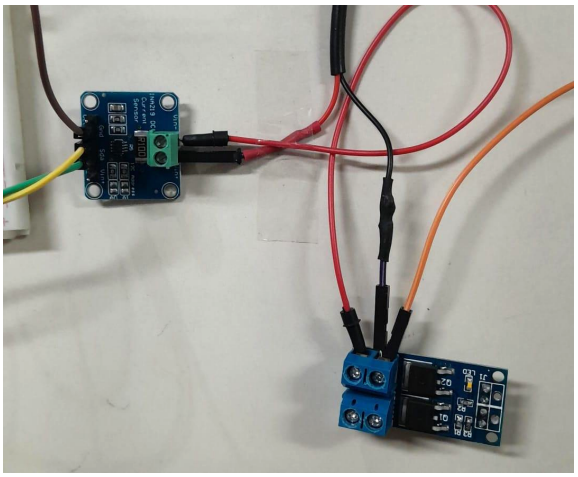
Imagem 3 - Conexões no display.

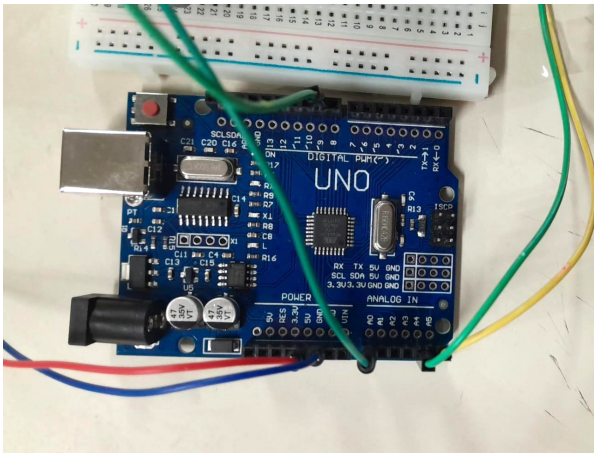


Fonte: Autores.

Com o INA219 devidamente ligado, passa-se à conexão do driver MOSFET, que controla o acionamento do cooler por meio de um sinal PWM enviado pelo Arduino. O pino VIN+ do driver deve ser conectado ao terminal VIN− do INA219, enquanto o pino VIN− do driver é ligado ao GND da fonte de 12V, junto ao segundo terminal GND do próprio driver (GND2), garantindo o retorno da corrente. O pino OUT+ do driver é conectado ao terminal positivo do cooler, e o OUT− é ligado ao terminal negativo do mesmo. O pino GND1 do driver deve ser interligado ao GND do Arduino, assegurando a referência de terra comum. Por fim, o pino PWM do driver é conectado ao pino digital 9 do Arduino, que é capaz de gerar o sinal modulador necessário para o controle da velocidade do cooler.

Imagem 4 e 5 - Conexões no Driver mosfet.

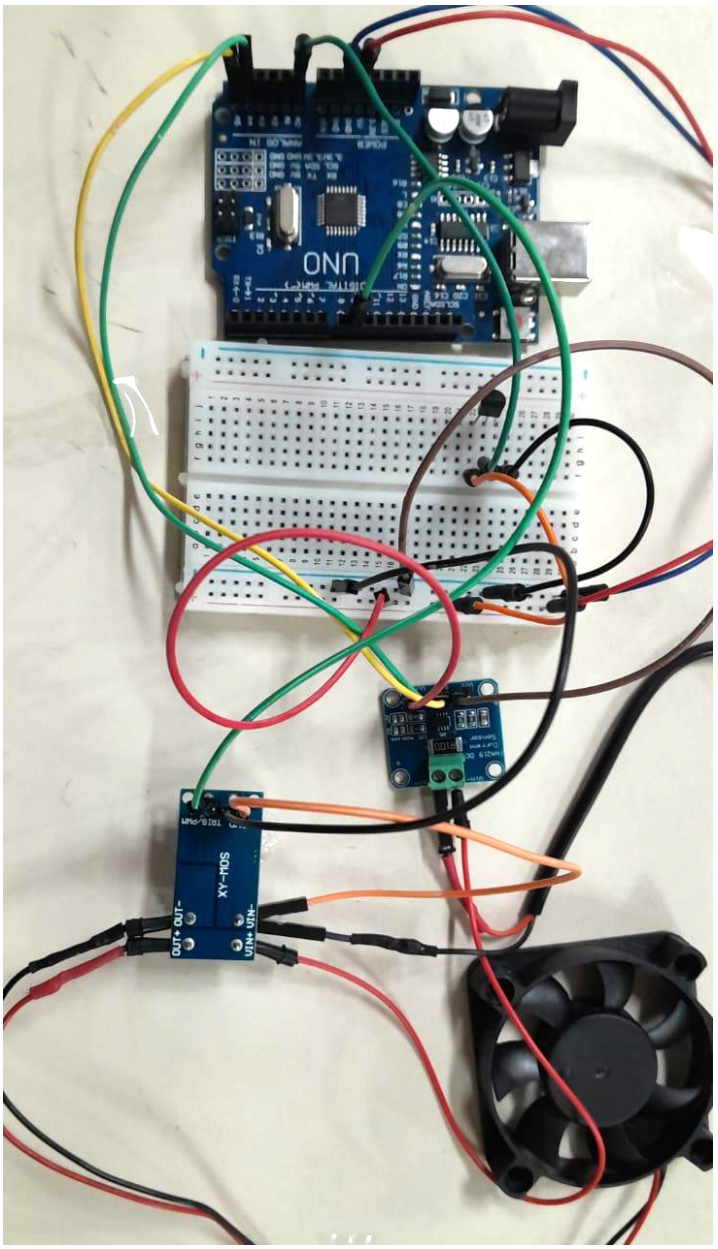




Fonte: Autores.

A fonte de 12V deve ter seu terminal positivo ligado ao VIN+ do INA219, enquanto o terminal negativo (GND) é conectado ao VIN− do driver MOSFET, completando o circuito de potência. Dessa forma, a corrente elétrica proveniente da fonte passa primeiro pelo INA219, depois pelo driver e, por fim, pelo cooler. Assim, o INA219 consegue monitorar a tensão e corrente consumidas pelo sistema em tempo real.

Imagem 6 - Últimas conexões dos sensores.

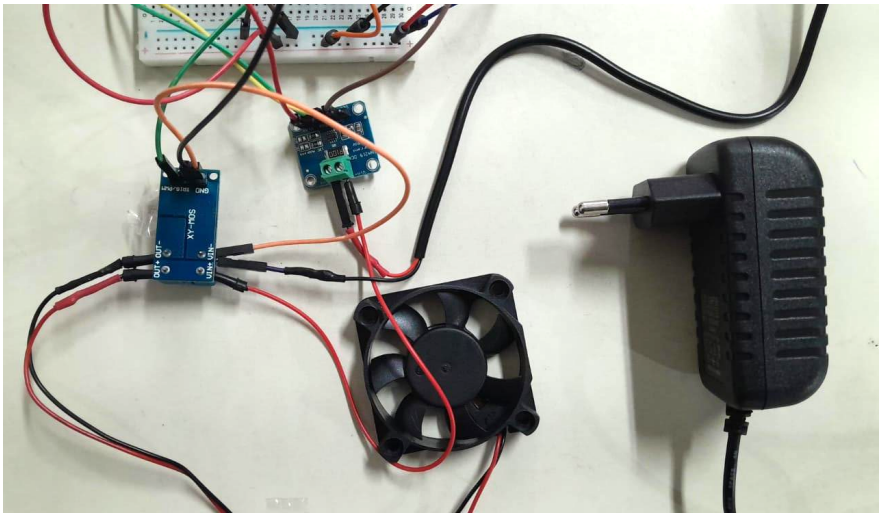


Fonte: Autores.

As ligações finais no Arduino incluem a alimentação principal do protótipo: o pino de 5V do Arduino é conectado ao barramento positivo da protoboard, distribuindo energia ao LM35 e ao INA219, e o pino GND é ligado ao barramento de terra comum. O pino A0 recebe o sinal analógico do LM35, enquanto os pinos A4 e

A5 são utilizados para a comunicação I2C com o INA219. O pino digital 9 é responsável pelo envio do sinal PWM ao driver MOSFET, controlando a velocidade do cooler de acordo com a leitura de temperatura.

Imagem 7 - Últimas conexões dos sensores.



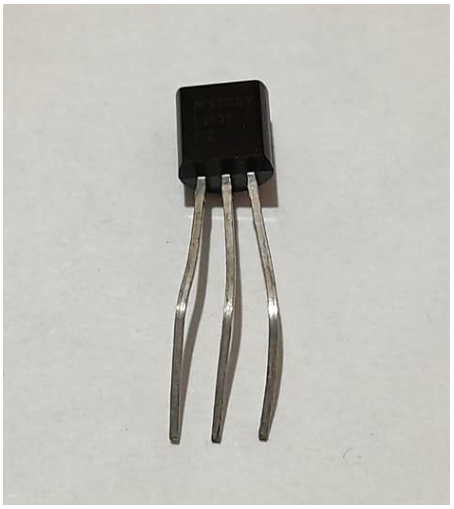
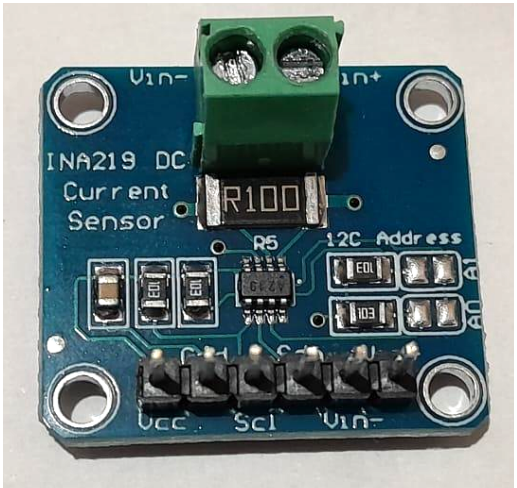
Fonte: Autores.

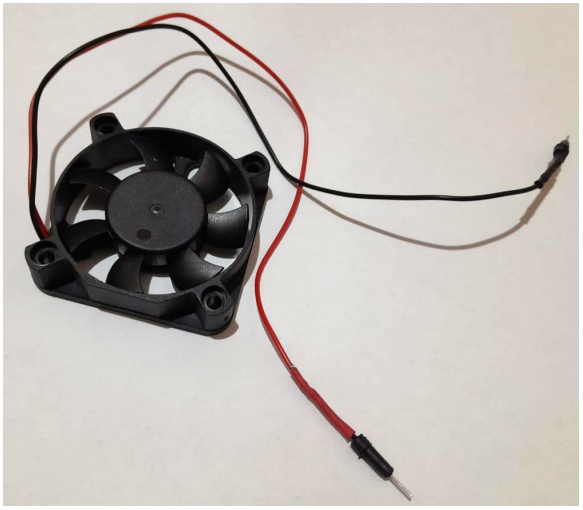
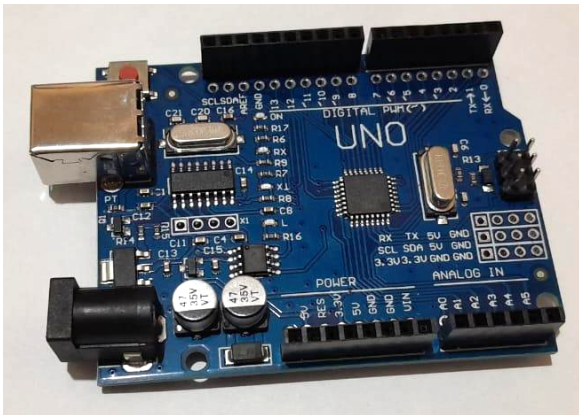
Antes de ligar a alimentação, deve-se conferir cuidadosamente todas as conexões, garantindo que não há fios invertidos ou curtos-circuitos. Recomenda-se verificar a continuidade dos GNDs com um multímetro, confirmar a polaridade dos componentes e assegurar que o cooler está corretamente conectado com o positivo em OUT+ e o negativo em OUT−. Após a conferência, o sistema pode ser energizado primeiramente via USB para testes iniciais, verificando a leitura do LM35 e a comunicação I2C do INA219. Em seguida, a fonte de 12V pode ser ligada, e o controle PWM testado com duty cycles baixos, observando o funcionamento do cooler.

Seguindo corretamente essas etapas, obtém-se um protótipo funcional capaz de medir temperatura com o LM35, monitorar tensão e corrente com o INA219, e controlar a velocidade do cooler através do driver MOSFET, tudo comandado pelo Arduino Uno de forma segura e eficiente.

4.4 Descrição da montagem dos componentes do protótipo.

Tabela 4 - Componentes do protótipo

	Lm35: Pino 1 (esquerda) é ligado no 5V do arduino. Pino 2 (meio) é conectado no pino A0 do arduino. Pino 3 (direita) é ligado no GND do arduino.
	Fonte 12V: Positivo é ligado no VIN+ do INA219. GND é conectado no VIN- no driver mosfet
	INA219: VCC é ligado no 5V do arduino. GND é conectado ao GND do arduino. SDA é ligado no pino A4 do arduino. SCL é conectado no pino A5 do arduino. VIN+ é ligado no positivo da fonte 12V. VIN- é conectado ao VIN+ do mosfet.

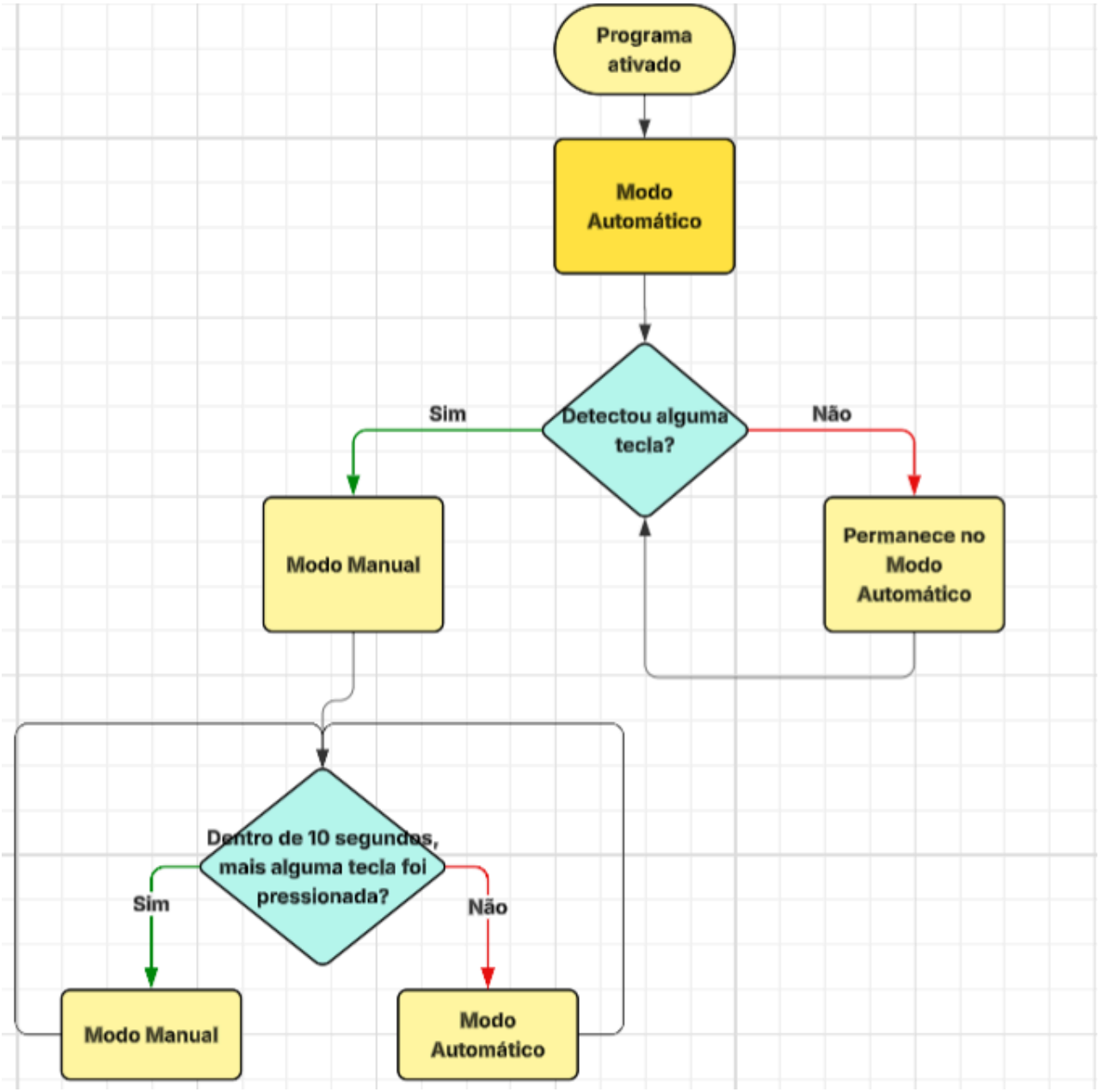
	Driver mosfet: VIN+ é ligado no VIN- do INA219. VIN- é conectado ao GND da fonte juntamente com o segundo GND do próprio mosfet. OUT+ é ligado no positivo do cooler. OUT- é conectado ao negativo do cooler. GND 1 é ligado ao GND do arduino. GND 2 é conectado ao VIN- do próprio mosfet. PWM é ligado no pino 9 (PWM) do arduino.
	Cooler: Positivo é conectado no OUT+ do mosfet. Negativo é ligado no OUT- do mosfet.
	Arduino Uno: 5V é conectado no positivo da protoboard. GND é ligado no GND da protoboard. Pino 9 é conectado ao PWM do mosfet. Pino A0 é ligado no pino 2 do Lm35. Pino A4 é conectado no SDA do INA219. Pino A5 é ligado no SCL do INA219.

Fonte: Os autores

4.5 PROGRAMAÇÃO

Os códigos podem ser observados no ANEXO A (Protótipo) e ANEXO B (Interface).

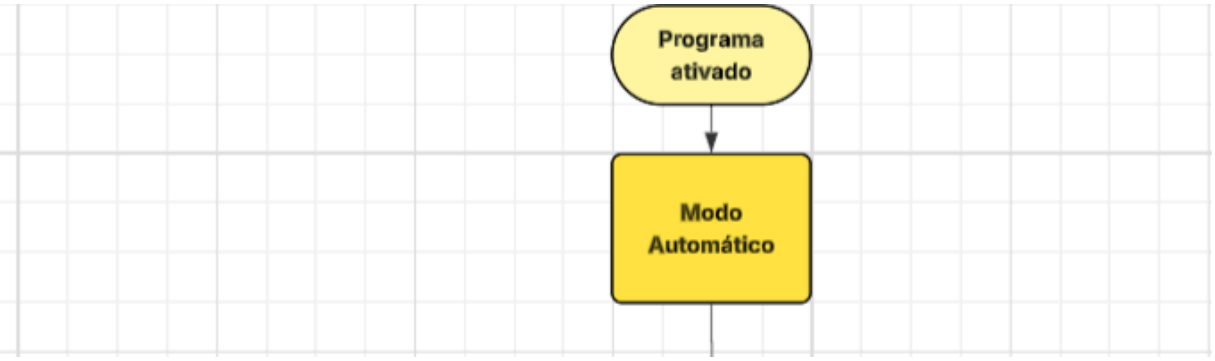
Imagem 8 - Fluxograma do funcionamento geral do sistema inteligente.



Fonte: Autores.

O fluxograma demonstra o funcionamento do programa responsável pelo controle do sistema. O processo se inicia com a ativação do programa, que por padrão entra no Modo Automático.

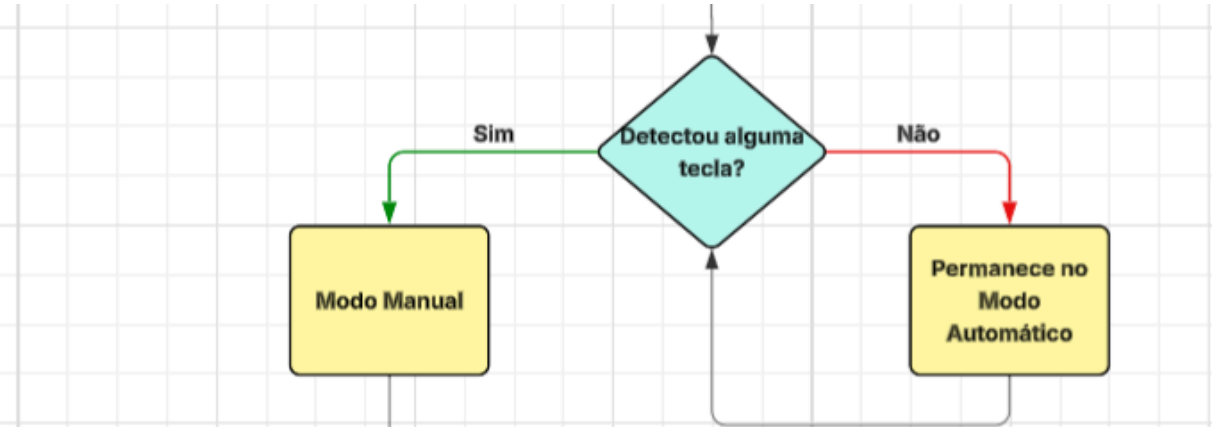
Imagem 9 - Início do programa.



Fonte: Autores.

Após o acionamento do Modo Automático, o sistema verifica se alguma tecla foi pressionada pelo usuário. Caso não haja interação, o programa permanece em modo automático. Por outro lado, caso seja detectada a pressão de uma tecla, o sistema muda para o Modo Manual.

Imagem 10 - Verificação de comando (Manual ou Automático)

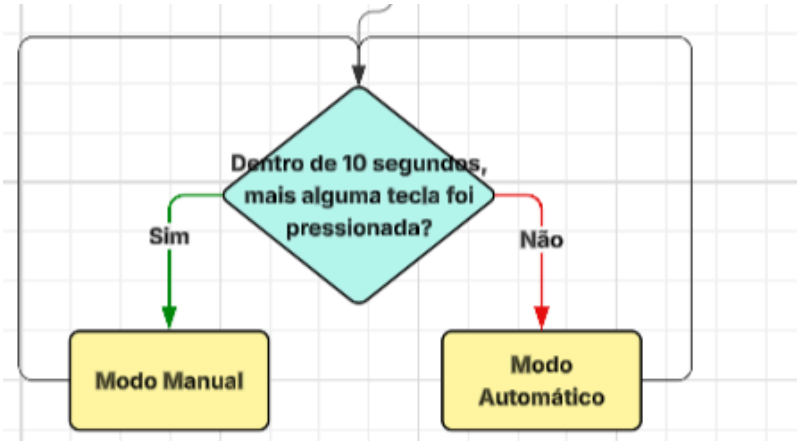


Fonte: Autores.

Uma vez em modo manual, o programa realiza uma nova verificação: se dentro de 10 segundos outra tecla for pressionada, o sistema permanece no modo

manual, garantindo que o usuário mantenha o controle direto da operação. Caso contrário, se nenhuma tecla for pressionada nesse intervalo, o programa retorna automaticamente ao Modo Automático, retomando seu funcionamento padrão.

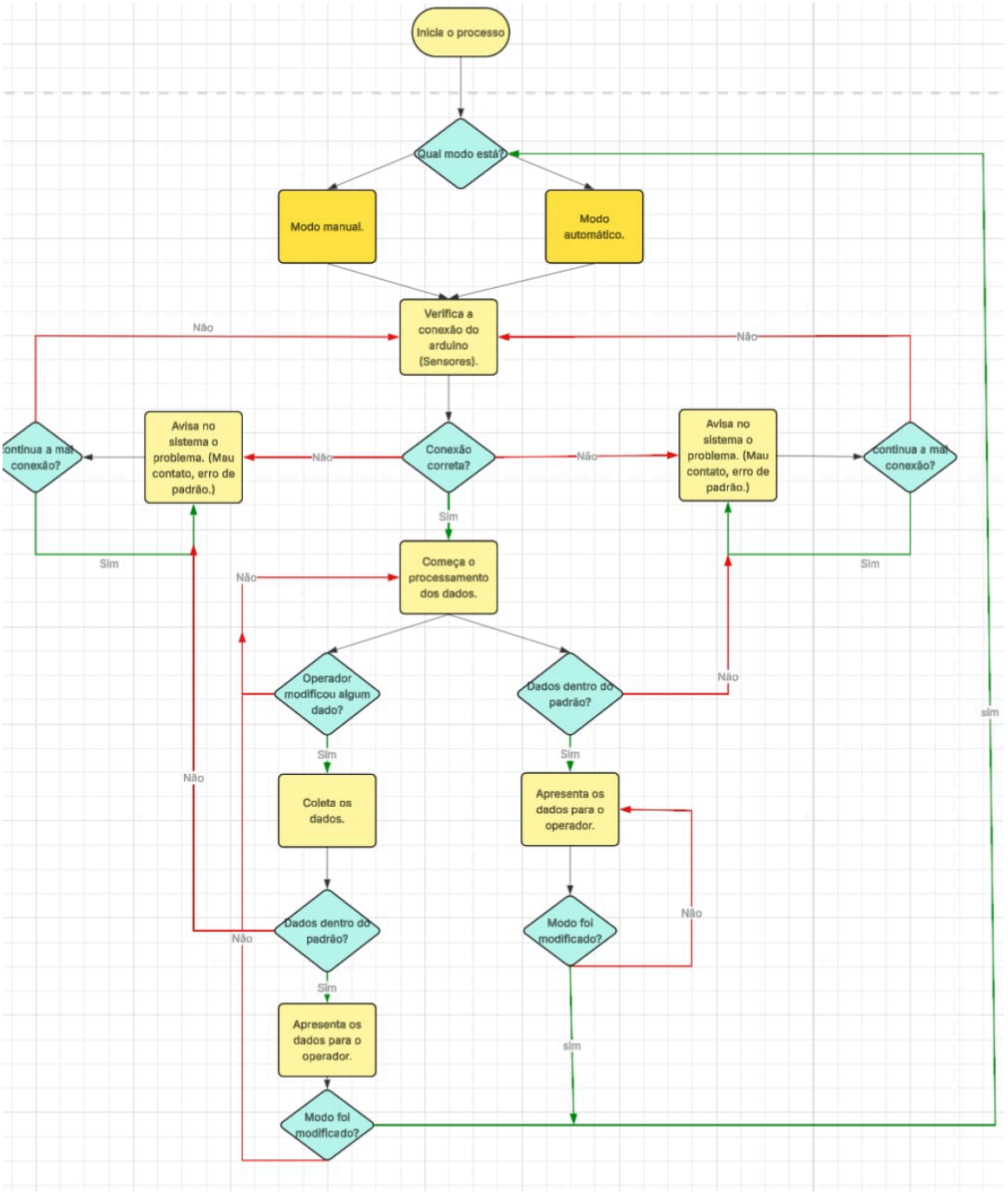
Imagem 11 - Verificação de comando (Retorna ao automático)



Fonte: Autores.

Dessa forma, o fluxograma esclarece que o sistema possui dois modos de operação, sendo capaz de alternar entre eles de forma dinâmica. O Modo Automático garante que o processo siga sem intervenção humana, enquanto o Modo Manual permite ajustes pontuais pelo operador quando necessário. Esse modelo aumenta a flexibilidade do protótipo, garantindo tanto autonomia quanto segurança operacional.

Imagem 12 - Fluxograma do funcionamento da interface do sistema inteligente.

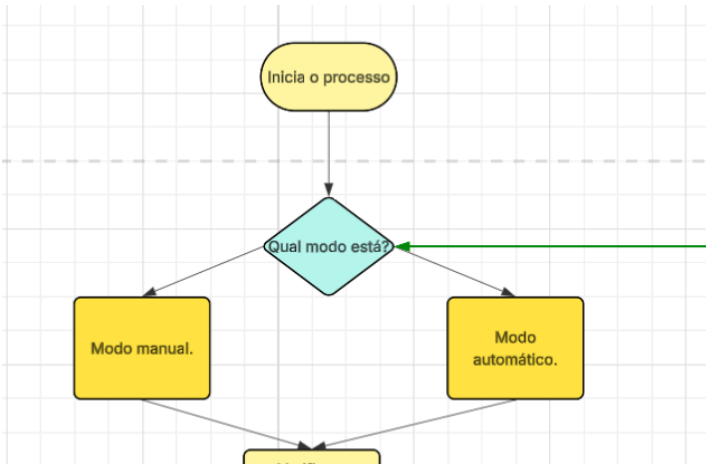


Fonte: Autores.

O fluxograma representa o funcionamento lógico do sistema de controle, que opera em dois modos distintos: manual e automático. O processo inicia com a

ativação do sistema, momento em que é feita a identificação do modo de operação atual.

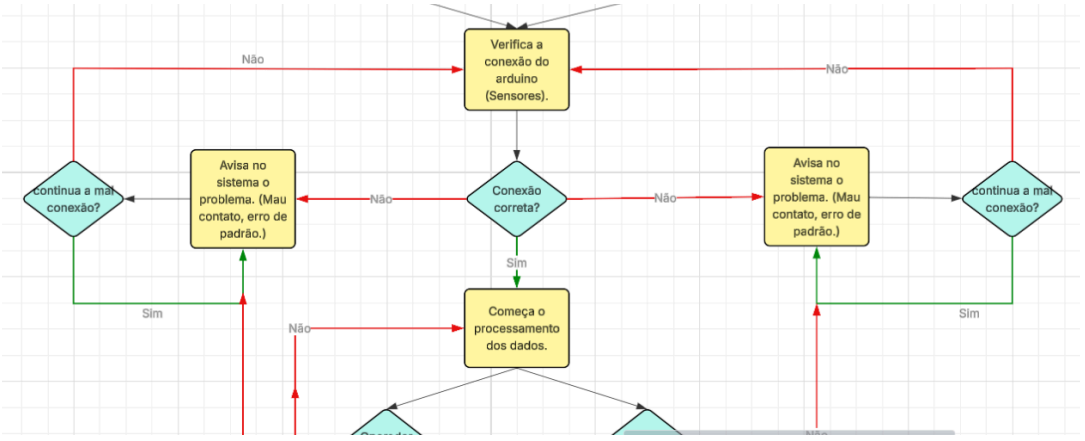
Imagem 13 - Início do programa.



Fonte: Autores.

Assim que o programa é iniciado, ele verifica em qual modo está configurado: manual ou automático. Essa verificação é essencial para determinar como as próximas ações serão executadas e quais verificações deverão ocorrer.

Imagem 14 - Verificação de entradas e saídas.

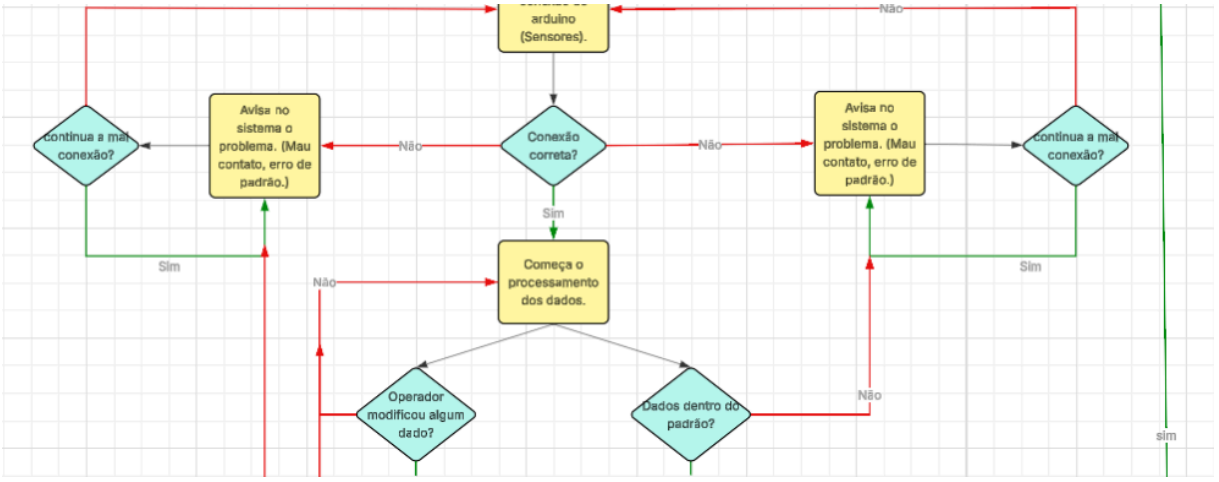


Fonte: Autores.

Caso o sistema esteja em modo automático, ele passa a verificar a conexão dos sensores ao Arduino, que são responsáveis pela coleta de dados do processo.

Se a conexão estiver incorreta (por mau contato ou falha de padrão), o sistema gera um aviso de erro, informando o problema e solicitando verificação. Após isso, o sistema avalia se a má conexão persiste. Caso sim, o processo retorna à etapa de verificação. Se o erro for corrigido, o sistema continua para o processamento dos dados.

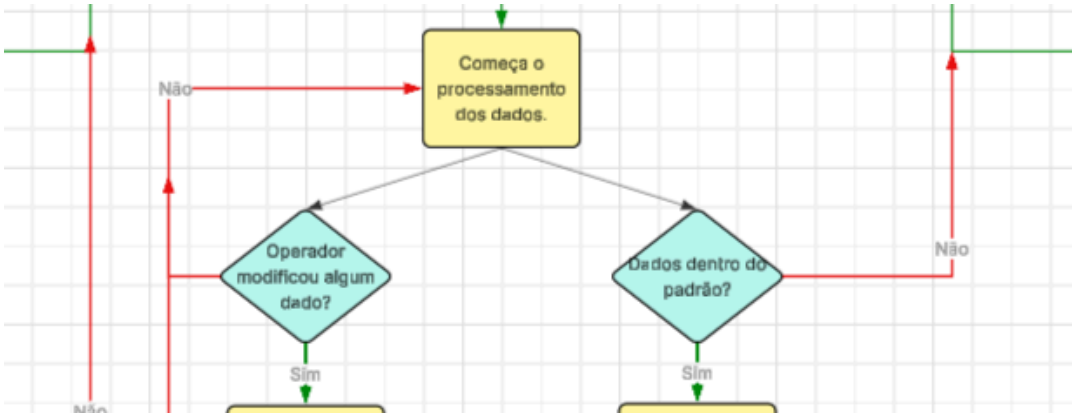
Imagem 15 - Verificação e processamento de dados.



Fonte: Autores.

Com a conexão validada, o sistema inicia o processamento dos dados coletados pelos sensores.

Imagem 16 - Início do processamento dos dados coletados pelos sensores.



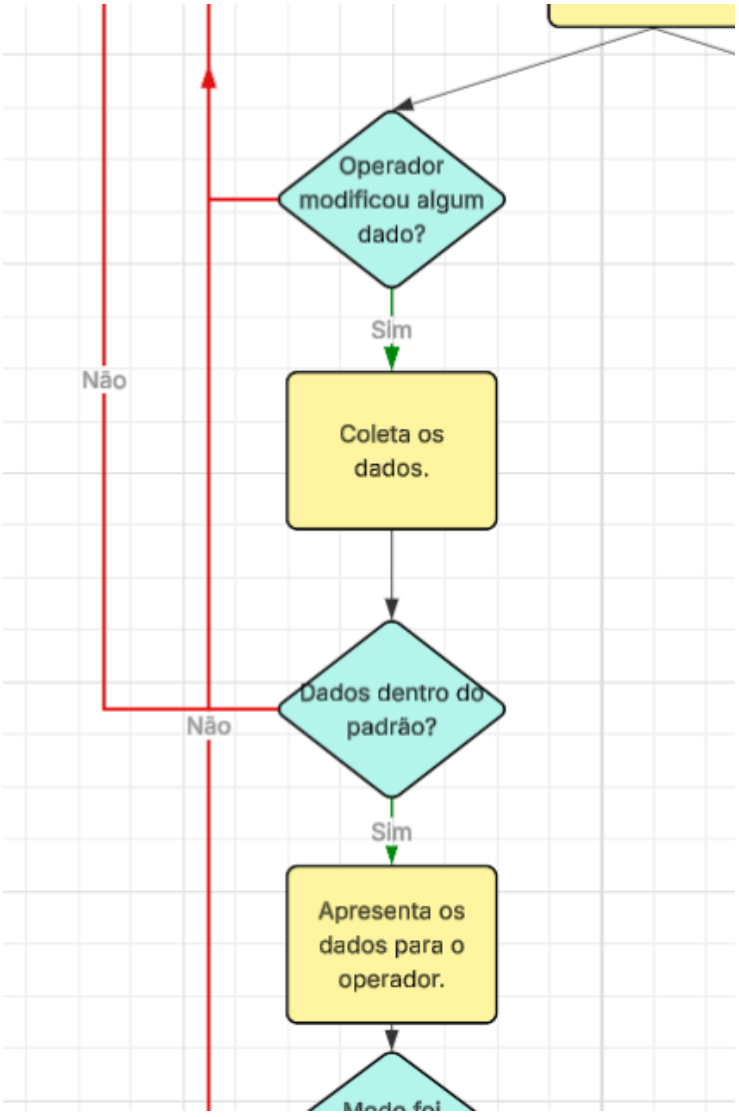
Fonte: Autores.

Em seguida, é verificado se os dados estão dentro dos padrões estabelecidos.

- Se estiverem corretos, os valores são apresentados ao operador para acompanhamento.

- Caso contrário, o sistema retorna à coleta, ajustando as informações conforme o modo de operação.

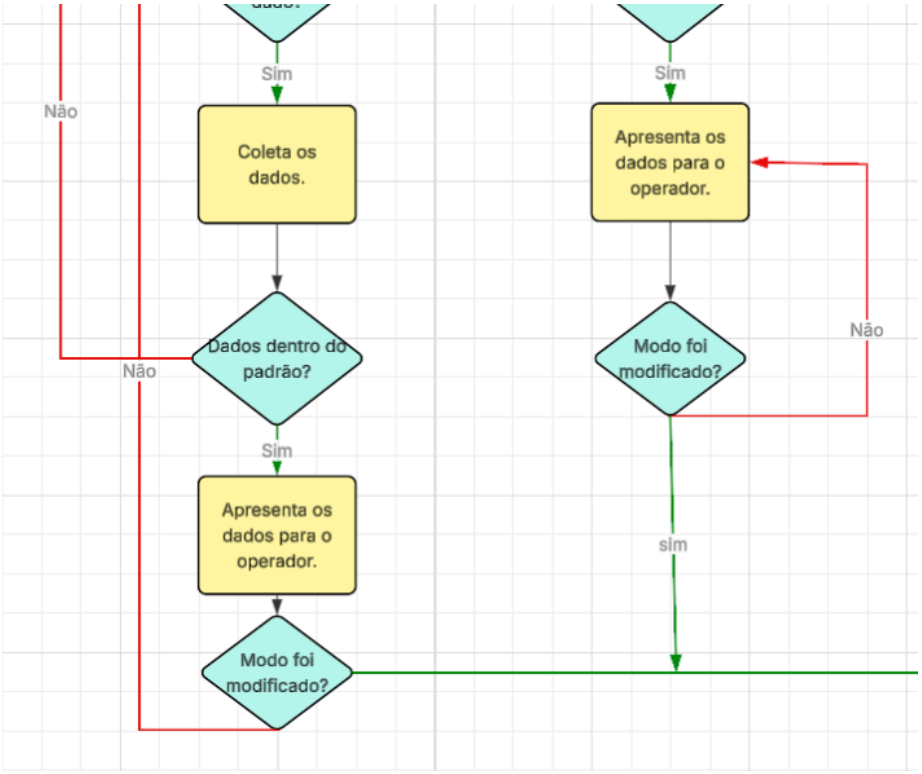
Imagem 17 - Verificação de modificação de dados.



Fonte: Autores.

Se o sistema estiver configurado no modo manual, o operador pode modificar parâmetros diretamente. Ao identificar qualquer modificação, o programa coleta novamente os dados e repete o processo de validação.

Imagem 18 - Coleta, amostra de dados e repetição.



Fonte: Autores.

Se os valores estiverem dentro do padrão, eles são mostrados ao operador, e o sistema verifica se houve alteração do modo, ou seja, se o operador decidiu retornar ao modo automático. Caso não haja mudança, o ciclo de controle continua no modo manual, permitindo novas intervenções sempre que necessário.

Em qualquer momento, se houver falha de conexão ou alteração de modo, o sistema retorna aos pontos de verificação correspondentes, garantindo segurança, estabilidade e integridade dos dados. Esse comportamento cria um loop de controle contínuo, onde a confiabilidade da comunicação e a validação das medições são mantidas em tempo real.

5. CRONOGRAMA

Tabela 5 – Cronograma

2025	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV
Escolha do tema										
Levantamento de literatura científica										
Introdução										
Tema										
Problema										
Objetivos										
Justificativa										
Estado da Arte										
Fundamentação teórica										
Metodologia										
Cronograma										
Recursos										
Resultados esperados ou parciais										
Referências										
Avaliação do CRC										
Produção do Banner										
27ª Exposchmidt										

6. RECURSOS

Tabela dos componentes e materiais utilizados para a construção do protótipo.

Tabela 6 – Componentes

Material	Valor unitário	Quantidade	Valor total	Fonte	Data
Arduino Uno Dip B0574.	37,86	1	R$ 37,86	WJ - Componentes eletrônicos.	20/05
Sensor De Corrente Dc Ina219 I2c	19,85	1	R$ 19,85	Maker Hero	14/10
Módulo Driver Pwm 5v A 36v 15a 400w	19,01	1	R$ 19,01	Eletronica Alfanit	20/05
Sensor de temperatura (Lm35)	24,40	1	R$ 24,40	Mekanus	20|05
Microventilador Cooler 12V	34,86	1	R$ 34,86	Clicvendas	20/05
Fonte de alimentação 12V	16,10	1	R$ 16,10	LBL Distribuidora	20/05
Jumper fio (40 PCs, 20cm, macho e fêmea)	15,99	1	R$ 15,99	MFL	20/05
Protoboard (400 pontos, Esp32)	12,99	1	R$ 12,99	ARDU robótica e informática LTDA	20/05
				Valor final:	162,07

7. RESULTADOS PARCIAIS

Este projeto almeja alcançar um avanço significativo no mercado industrial vigente, contribuindo com a segurança e o custo-benefício das pequenas e médias empresas por meio do baixo custo da estrutura do protótipo, uma vez que utiliza recursos acessíveis e de fácil implementação.

Por meio da construção e do desenvolvimento de um sistema de monitoramento inteligente baseado em Arduino, foi possível integrar sensores e módulos de controle que permitem a conexão direta com máquinas e a exibição de dados em uma interface acessível, voltada ao acompanhamento de parâmetros críticos, como temperatura, tensão, corrente e potência.

Com a substituição do sensor ACS712 pelo INA219, o sistema passou a apresentar medições mais precisas e estáveis, garantindo maior confiabilidade nos resultados obtidos durante os testes. Além disso, os modos de operação manual e automático foram devidamente implementados e funcionam conforme o esperado, possibilitando tanto o controle supervisionado pelo operador quanto o acionamento autônomo do sistema.

Outro avanço importante foi a aplicação do controle PWM *(Pulse Width Modulation)* no módulo de ventilação, que agora atua de forma proporcional à temperatura medida pelo sensor LM35. Dessa forma, o ventilador ajusta automaticamente sua velocidade conforme o aquecimento detectado, otimizando a dissipação de calor e evitando desperdício de energia.

Do ponto de vista técnico-científico, espera-se validar os conceitos teóricos de monitoramento e controle inteligente em uma aplicação prática de testes de maquinários e acompanhamento de funcionamento, ampliando o conhecimento sobre a adaptação dessas tecnologias em ambientes industriais reais. No âmbito socioeconômico, o projeto contribui para a redução de custos de manutenção e operação, melhoria da eficiência energética e aumento da segurança dos

trabalhadores, demonstrando seu potencial como uma ferramenta valiosa para empresas que buscam modernização e confiabilidade nos processos produtivos.

Portanto, este trabalho demonstra que a utilização de um sistema de monitoramento inteligente, acessível e viável, é capaz de preencher lacunas existentes em pequenas e médias empresas, promovendo segurança, sustentabilidade e eficiência operacional, princípios fundamentais no contexto da Indústria 4.0.

REFERÊNCIAS

ABDI. Agência Brasileira de Desenvolvimento Industrial. Disponível em: https://www.abdi.com.br/. Acesso em: 26 jun. 2025.

ANIMA EDUCAÇÃO. Trabalho de conclusão de curso: sistemas de monitoramento com Arduino. Disponível em: https://repositorio-api.animaeducacao.com.br/server/api/core/bitstreams/379f829e-7919-482b-a6dc-8ebef21b520b/content. Acesso em: 26 jun. 2025.

ARDUINO. Arduino Reference. Disponível em: https://www.arduino.cc/reference/en/. Acesso em: 12 jun. 2025.

AUTOCORE ROBÓTICA. Módulo sensor de umidade e temperatura DHT11. Disponível em: https://www.autocorerobotica.com.br/modulo-sensor-de-umidade-temperatura-dht11. Acesso em: 24 ago. 2025.

AWS. O que é IoT? Amazon Web Services. Disponível em: https://aws.amazon.com/pt/what-is/iot/. Acesso em: 26 jun. 2025.

BAÚ DA ELETRÔNICA. Sensor de temperatura LM35 – Original. Disponível em: https://www.baudaeletronica.com.br/produto/sensor-de-temperatura-lm35-original.html. Acesso em: 24 ago. 2025.

BRASIL ESCOLA. Introdução à programação. Disponível em: https://brasilescola.uol.com.br/informatica/introducao-a-programacao.htm. Acesso em: 03 jun. 2025.

CASA DA ROBÓTICA. Módulo Sensor de Vibração SW-420 – Sensor Digital. Disponível em: https://www.casadarobotica.com/sensores-modulos/sensores/outros/modulo-sensor-de-vibracao-sw-420-sensor-digital. Acesso em: 24 ago. 2025.

DANTAS, Ivanovitch Medeiros. Internet das Coisas: conceitos e aplicações. Porto Alegre: Bookman, 2020.

DEITEL, Paul; DEITEL, Harvey. C: como programar. 8. ed. São Paulo: Pearson, 2016.

EGA. Sensores inteligentes: o que são e como funcionam. Disponível em: https://ega.com.br/sensores-inteligentes/. Acesso em: 16 jul. 2025.

ELETRODEX. Sensor HC-SR04 – Medidor de distância por ultrassom. Disponível em: https://www.eletrodex.net/placasmodulos/modulos/sensores/sensor-hc-sr04-medidor-de-distancia-por-ultrasom. Acesso em: 24 ago. 2025.

EUROPEAN COMMISSION. Digital Strategy: Industry 4.0. Disponível em: https://digital-strategy.ec.europa.eu/. Acesso em: 26 jun. 2025.

GONZÁLEZ, R. C. Sistemas Digitais e Microprocessadores. Pearson.

GSMA. Internet of Things. Disponível em: https://www.gsma.com/iot/. Acesso em: 12 jun. 2025.

IEEE. IEEE Internet of Things Journal. Disponível em: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907. Acesso em: 16 jun. 2025.

IEEE COMPUTER SOCIETY. IEEE Software Magazine. Disponível em: https://www.computer.org/csdl/magazine/so. Acesso em: 16 jun. 2025.

IOT ANALYTICS. Insights and Market Reports. Disponível em: https://iot-analytics.com. Acesso em: 26 jun. 2025.

KAGERMANN, Henning; WAHLSTER, Wolfgang; HELBIG, Johannes. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Frankfurt: Acatech, 2013.

KERNIGHAN, Brian W.; RITCHIE, Dennis M. Linguagem de programação C. 2. ed. Rio de Janeiro: LTC, 2019.

LENOVO. O que é software? Glossário Lenovo. Disponível em: https://www.lenovo.com/br/pt/glossary/what-is-software/. Acesso em: 08 jun. 2025.

MCEWEN, Adrian; CASSIMALLY, Hakim. Designing the Internet of Things. Chichester: Wiley, 2014.

NARDON, A. Automação Industrial: Fundamentos e Aplicações. 2. ed. Rio de Janeiro: LTC, 2019.

ORACLE. O que é Internet das Coisas (IoT). Disponível em: https://www.oracle.com/br/internet-of-things/. Acesso em: 04 jun. 2025.

PISCALED. Sensor de qualidade do ar MQ-135 para gases tóxicos. Disponível em: https://www.piscaled.com.br/sensor-de-qualidade-do-ar-mq-135-para-gases-toxicos. Acesso em: 24 ago. 2025.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

RED HAT. Soluções de automação de software. Disponível em: https://www.redhat.com/. Acesso em: 03 jun. 2025.

SAP. O que é Indústria 4.0? Disponível em: https://www.sap.com/brazil/products/scm/industry-4-0/what-is-industry-4-0.html. Acesso em: 24 jun. 2025.

SCHWAB, Klaus. A quarta revolução industrial. São Paulo: Edipro, 2016.

SCIELO. Gestão da produção e sistemas integrados. Disponível em: https://www.scielo.br/j/gp/a/GmLSKVc7dpRYdBhtbChHDcv/. Acesso em: 22 jul. 2025.

SEBRAE. Quando surgiu a Indústria 4.0. Disponível em: https://sebrae.com.br/sites/PortalSebrae/artigos/quando-surgiu-a-industria-40,4542c009cbce3810VgnVCM100000d701210aRCRD. Acesso em: 16 jul. 2025.

SENAI-ES. Indústria 4.0: o monitoramento da eficiência operacional. Disponível em: https://senaies.com.br/artigo-industria-4-0-o-monitoramento-da-eficiencia-operacional-para-melhoria-continua/. Acesso em: 22 jul. 2025.

SENSOR TIPS. Introduction to Industrial Sensors. Disponível em: https://www.sensortips.com. Acesso em: 6 jul. 2025.

SILVA, J. F. Monitoramento Industrial Inteligente com IoT. Revista Eletrônica Automação, 2021.

SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo: Pearson, 2019.

TECNOLOGIA EXPLICADA. O que é software. Disponível em: https://tecnoblog.net/responde/o-que-e-software/. Acesso em: 26 jun. 2025.

TOTVS. Indústria 4.0: o que é, vantagens e como implementar. Disponível em: https://www.totvs.com/blog/gestao-industrial/industria-4-0/. Acesso em: 25 jun. 2025.

TOTVS. O que é um sistema integrado?. Disponível em: https://www.totvs.com/blog/erp/sistema-integrado/. Acesso em: 22 jul. 2025.

UDESC. Universidade do Estado de Santa Catarina. Diretrizes curriculares para o curso de Engenharia de Produção. Disponível em: https://www.udesc.br/arquivos/ceplan/id_cpmenu/1590/1309_16675007319483_1590.pdf. Acesso em: 26 jun. 2025.

USINAINFO. Sensor de Corrente ACS712 20A AC/DC com Efeito Hall. Disponível em: https://www.usinainfo.com.br/sensor-de-corrente-arduino/sensor-de-corrente-acs712-20a-ac-dc-com-efeito-hall-5701.html. Acesso em: 24 ago. 2025.

W3SCHOOLS. Tutorials for Web Development. Disponível em: https://www.w3schools.com. Acesso em: 04 jun. 2025.

WP AUTOMAÇÃO. Desenvolvimento de projetos de automação com Arduino na indústria. Disponível em: https://www.wpautomacao.com.br/blog/categorias/industrial/desenvolvimento-de-projetos-de-automacao-com-arduino-na-industria. Acesso em: 16 jun. 2025.

ANEXOS

ANEXO A - Código de funcionamento do protótipo base.

```
#include <Wire.h>

#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

const int lm35Pin = A0;   // LM35

const int fanPWM = 9;     // PWM out to MOSFET

int fanSpeed = 0;

bool modoManual = false;

unsigned long tempoUltimoComando = 0;

const unsigned long tempoManual = 10000; // 10s

void setup() {

  Serial.begin(9600);

  while (!Serial) { delay(10); } // opcional em alguns Arduinos

  if (!ina219.begin()) {

    Serial.println("ERR_INA219");

    while (1) delay(1000);

  }

  ina219.setCalibration_32V_2A();

  pinMode(fanPWM, OUTPUT);

  analogWrite(fanPWM, 0);

  Serial.println("READY");

}
```

```
void enviarCSV() {
  // leitura LM35
  float temperatura = analogRead(lm35Pin) * (5.0 / 1023.0) * 100.0;
  // leitura INA219
  float tensao = ina219.getBusVoltage_V();         // V
  float corrente_mA = ina219.getCurrent_mA();      // mA (pode ser
negativo)
  float corrente = corrente_mA / 1000.0;           // A
  float potencia = tensao * corrente;              // W
  int pwmPercent = map(fanSpeed, 0, 255, 0, 100);
  const char* modo = modoManual ? "MAN" : "AUTO";
  // enviar CSV: temperatura,tensao,corrente,potencia,pwm,modo
  Serial.print(temperatura, 1); Serial.print(',');
  Serial.print(tensao, 2);      Serial.print(',');
  Serial.print(corrente, 3);    Serial.print(',');
  Serial.print(potencia, 3);    Serial.print(',');
  Serial.print(pwmPercent);     Serial.print(',');
  Serial.println(modo);
}
void checarComandoSerial() {
  if (Serial.available()) {
    char tecla = Serial.read();
    modoManual = true;
    tempoUltimoComando = millis();
    if (tecla == 'D') fanSpeed = 0;       // Desligado 0%
```

```
    else if (tecla == 'L') fanSpeed = 64; // 25%
    else if (tecla == 'M') fanSpeed = 127;// 50%
    else if (tecla == 'A') fanSpeed = 191;// 75%
    else if (tecla == 'T') fanSpeed = 255;// 100%
    else if (tecla == 'R') modoManual = false; // volta auto
    // opcional: eco para debug
    // Serial.print("CMD:"); Serial.println(tecla);
  }
}
void loop() {
  checarComandoSerial();
  if (modoManual && millis() - tempoUltimoComando > tempoManual) {
    modoManual = false;
  }
  // controle automático simples (se em AUTO)
  if (!modoManual) {
    float temperatura = analogRead(lm35Pin) * (5.0 / 1023.0) * 100.0;
    if (temperatura >= 30) fanSpeed = 255;
    else if (temperatura >= 26) fanSpeed = 191;
    else if (temperatura >= 23) fanSpeed = 127;
    else if (temperatura >= 20) fanSpeed = 64;
    else fanSpeed = 0;
  }
  analogWrite(fanPWM, fanSpeed);
```

```
  // enviar status
  enviarCSV();
  delay(1000); // 1 leitura por segundo
}
```

ANEXO B - Código de funcionamento da interface base.

```
# interface_mipi_atualizado.py
import serial
import serial.tools.list_ports
import threading
import queue
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from collections import deque
import time
import csv
from datetime import datetime
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
BAUDRATE = 9600
```

```
MAX_POINTS = 120

READ_TIMEOUT = 1

class SerialReader(threading.Thread):
    def __init__(self, ser, q, stop_event):
        super().__init__(daemon=True)
        self.ser = ser
        self.q = q
        self.stop_event = stop_event
    def run(self):
        while not self.stop_event.is_set():
            if not self.ser or not self.ser.is_open:
                time.sleep(0.1)
                continue
            try:
                line = self.ser.readline().decode("utf-8", errors="ignore").strip()
                if line:
                    self.q.put(line)
            except Exception:
                break
class MonitorApp:
    def __init__(self, root):
```

```
        self.root = root
        root.title("MIPI - Monitor Arduino")
        root.rowconfigure(0, weight=1)
        root.columnconfigure(0, weight=1)
        self.ser = None
        self.reader = None
        self.reader_stop = threading.Event()
        self.q = queue.Queue()
        self.temps = deque(maxlen=MAX_POINTS)
        self.volts = deque(maxlen=MAX_POINTS)
        self.amps = deque(maxlen=MAX_POINTS)
        self.pows = deque(maxlen=MAX_POINTS)
        self.times = deque(maxlen=MAX_POINTS)
        self.logging = False
        self.logfile = None
        self.csvwriter = None
        self._build_ui()
        self._bind_keys()
        self.root.after(200, self._process_serial_queue)
    def _build_ui(self):
        frm = ttk.Frame(self.root, padding=8)
```

```
        frm.grid(row=0, column=0, sticky="nsew")

        frm.rowconfigure(4, weight=1)

        frm.columnconfigure(0, weight=1)

        # --- Conexão ---

        port_frame = ttk.LabelFrame(frm, text="Conexão")

        port_frame.grid(row=0, column=0, sticky="ew", padx=4, pady=4)

        self.port_cb = ttk.Combobox(port_frame, values=self._list_ports(), width=30)

        self.port_cb.grid(row=0, column=0, padx=4, pady=4)

        ttk.Button(port_frame, text="Atualizar",
command=self._refresh_ports).grid(row=0, column=1, padx=4)

        self.connect_btn = ttk.Button(port_frame, text="Conectar",
command=self._toggle_connect)

        self.connect_btn.grid(row=0, column=2, padx=4)

        # --- Leituras Atuais ---

        display_frame = ttk.LabelFrame(frm, text="Leituras Atuais")

        display_frame.grid(row=1, column=0, sticky="ew", padx=4, pady=4)

        self.temp_lbl = ttk.Label(display_frame, text="Temp: -- °C", font=("Segoe UI",
12))

        self.temp_lbl.grid(row=0, column=0, sticky="w", padx=6, pady=6)

        self.volt_lbl = ttk.Label(display_frame, text="Tensão: -- V", font=("Segoe UI",
12))

        self.volt_lbl.grid(row=0, column=1, sticky="w", padx=6, pady=6)
```

```
        self.amp_lbl = ttk.Label(display_frame, text="Corrente: -- mA", font=("Segoe UI",
12))

        self.amp_lbl.grid(row=0, column=2, sticky="w", padx=6, pady=6)

        self.pow_lbl = ttk.Label(display_frame, text="Potência: -- mW", font=("Segoe
UI", 12))

        self.pow_lbl.grid(row=1, column=0, sticky="w", padx=6)

        self.pwm_lbl = ttk.Label(display_frame, text="PWM: -- %", font=("Segoe UI", 11))

        self.pwm_lbl.grid(row=1, column=1, sticky="w", padx=6)

        self.mode_lbl = ttk.Label(display_frame, text="Modo: --", font=("Segoe UI", 11))

        self.mode_lbl.grid(row=1, column=2, sticky="w", padx=6)

        # --- Controles ---

        ctrl_frame = ttk.LabelFrame(frm, text="Controles")

        ctrl_frame.grid(row=2, column=0, sticky="ew", padx=4, pady=4)

        buttons = [

            ("Desligar (D)", "D"), ("25% (L)", "L"), ("50% (M)", "M"),

            ("75% (A)", "A"), ("100% (T)", "T"), ("Auto (R)", "R")

        ]

        for i, (label, cmd) in enumerate(buttons):

            ttk.Button(ctrl_frame, text=label, command=lambda c=cmd:
self._send_cmd(c)).grid(row=0, column=i, padx=4, pady=4)

        # --- Log ---

        log_frame = ttk.Frame(frm)
```

```
        log_frame.grid(row=3, column=0, sticky="ew", padx=4, pady=4)

        self.log_btn = ttk.Button(log_frame, text="Iniciar Log",
command=self._toggle_logging)

        self.log_btn.grid(row=0, column=0, padx=4)

        ttk.Button(log_frame, text="Salvar CSV Agora",
command=self._save_snapshot).grid(row=0, column=1, padx=4)

        # --- Gráfico ---

        plot_frame = ttk.LabelFrame(frm, text="Gráfico")

        plot_frame.grid(row=4, column=0, sticky="nsew", padx=4, pady=4)

        plot_frame.rowconfigure(0, weight=1)

        plot_frame.columnconfigure(0, weight=1)

        self.fig = Figure(figsize=(10, 5), dpi=100)

        self.ax = self.fig.add_subplot(111)

        self.ax.set_title("Temp / Tensão / Corrente")

        self.line_t, = self.ax.plot([], [], label="Temp (°C)")

        self.line_v, = self.ax.plot([], [], label="Tensão (V)")

        self.line_i, = self.ax.plot([], [], label="Corrente (mA)")

        self.ax.legend(loc="center left", bbox_to_anchor=(1.18, 0.5))

        self.fig.subplots_adjust(left=0.30, right=0.78, top=0.92, bottom=0.12)

        self.ax.grid(True)

        self.canvas = FigureCanvasTkAgg(self.fig, master=plot_frame)
```

```
        widget = self.canvas.get_tk_widget()
        widget.grid(row=0, column=0, sticky="nsew")
        self.canvas.draw_idle()

    def _list_ports(self):
        return [p.device for p in serial.tools.list_ports.comports()]

    def _refresh_ports(self):
        self.port_cb['values'] = self._list_ports()

    def _toggle_connect(self):
        if self.ser and self.ser.is_open:
            self._disconnect()
        else:
            self._connect()

    def _connect(self):
        port = self.port_cb.get().strip()
        if not port:
            messagebox.showwarning("Porta", "Escolha uma porta serial.")
            return
        try:
            self.ser = serial.Serial(port, BAUDRATE, timeout=READ_TIMEOUT)
            time.sleep(1.0)
            self.reader_stop.clear()
```

```
            self.reader = SerialReader(self.ser, self.q, self.reader_stop)
            self.reader.start()
            self.connect_btn.config(text="Desconectar")
            messagebox.showinfo("Conectado", f"Conectado em {port}")
        except Exception as e:
            messagebox.showerror("Erro", f"Não foi possível abrir {port}:\n{e}")
            self.ser = None
    def _disconnect(self):
        try:
            self.reader_stop.set()
            if self.reader and self.reader.is_alive():
                self.reader.join(timeout=1.0)
        except Exception:
            pass
        try:
            if self.ser:
                self.ser.close()
        except Exception:
            pass
        self.ser = None
        self.connect_btn.config(text="Conectar")
```

```
        messagebox.showinfo("Desconectado", "Porta fechada.")
    def _send_cmd(self, cmd):
        if self.ser and self.ser.is_open:
            try:
                self.ser.write(cmd.encode("utf-8"))
            except Exception as e:
                messagebox.showerror("Erro", f"Falha ao enviar comando: {e}")
        else:
            messagebox.showwarning("Não conectado", "Conecte o Arduino antes de
enviar comandos.")
    def _process_serial_queue(self):
        updated = False
        while not self.q.empty():
            line = self.q.get()
            self._handle_line(line)
            updated = True
        if updated:
            self._update_plot()
        self.root.after(200, self._process_serial_queue)
    def _handle_line(self, line):
        try:
```

```
parts = [p.strip() for p in line.split(",")]
if len(parts) < 6:
    return
temp = float(parts[0])
volt = float(parts[1])
amp = float(parts[2])
power = float(parts[3])
pwm = int(float(parts[4]))
modo = parts[5]
# ✅ Conversão para mA e mW
amp_mA = amp * 1000.0
power_mW = power * 1000.0
self.temp_lbl.config(text=f"Temp: {temp:.1f} °C")
self.volt_lbl.config(text=f"Tensão: {volt:.2f} V")
self.amp_lbl.config(text=f"Corrente: {amp_mA:.1f} mA")
self.pow_lbl.config(text=f"Potência: {power_mW:.1f} mW")
self.pwm_lbl.config(text=f"PWM: {pwm} %")
self.mode_lbl.config(text=f"Modo: {modo}")
self.times.append(datetime.now().strftime("%H:%M:%S"))
self.temps.append(temp)
self.volts.append(volt)
```

```
            self.amps.append(amp_mA)
            self.pows.append(power_mW)
            if self.logging and self.csvwriter:
                self.csvwriter.writerow([datetime.now().isoformat(), temp, volt, amp_mA,
power_mW, pwm, modo])
        except Exception as e:
            print("Parse error:", e, "=>", line)
    def _update_plot(self):
        x = list(range(len(self.temps)))
        self.line_t.set_data(x, list(self.temps))
        self.line_v.set_data(x, list(self.volts))
        self.line_i.set_data(x, list(self.amps))
        if x:
            self.ax.set_xlim(0, max(10, x[-1]))
            all_y = list(self.temps) + list(self.volts) + list(self.amps)
            ymin = min(all_y) - 0.1 * abs(min(all_y))
            ymax = max(all_y) + 0.1 * abs(max(all_y))
            if ymin == ymax:
                ymin -= 1
                ymax += 1
            self.ax.set_ylim(ymin, ymax)
```

```
        self.canvas.draw_idle()

    def _toggle_logging(self):

        if not self.logging:

            fname = filedialog.asksaveasfilename(defaultextension=".csv",
filetypes=[("CSV","*.csv")])

            if not fname:

                return

            try:

                f = open(fname, "w", newline="", encoding="utf-8")

                self.logfile = f

                self.csvwriter = csv.writer(f)
self.csvwriter.writerow(["timestamp","temperature_C","voltage_V","current_mA","pow
er_mW","pwm_percent","mode"])

                self.logging = True

                self.log_btn.config(text="Parar Log")

                messagebox.showinfo("Log", "Gravando...")

            except Exception as e:

                messagebox.showerror("Erro", f"Não foi possível criar arquivo:\n{e}")

        else:

            try:

                if self.logfile:

                    self.logfile.close()
```

```
            except Exception:
                pass
            self.logging = False
            self.csvwriter = None
            self.logfile = None
            self.log_btn.config(text="Iniciar Log")
            messagebox.showinfo("Log", "Gravação finalizada.")
    def _save_snapshot(self):
        fname = filedialog.asksaveasfilename(defaultextension=".csv",
filetypes=[("CSV","*.csv")])
        if not fname:
            return
        try:
            with open(fname, "w", newline="", encoding="utf-8") as f:
                w = csv.writer(f)
w.writerow(["timestamp","temperature_C","voltage_V","current_mA","power_mW"])
                for i in range(len(self.times)):
                    w.writerow([self.times[i], self.temps[i], self.volts[i], self.amps[i],
self.pows[i]])
            messagebox.showinfo("Salvo", f"Snapshot salvo em {fname}")
        except Exception as e:
            messagebox.showerror("Erro", f"Falha ao salvar:\n{e}")
```

```
    def _bind_keys(self):
        for key, cmd in [("l", "L"), ("m", "M"), ("a", "A"), ("t", "T"), ("d", "D"), ("r", "R")]:
            self.root.bind(f"<KeyPress-{key}>", lambda e, c=cmd: self._send_cmd(c))
            self.root.bind(f"<KeyPress-{key.upper()}>", lambda e, c=cmd:
self._send_cmd(c))
def on_close(self):
        if messagebox.askokcancel("Sair", "Fechar o programa?"):
            try:
                self._disconnect()
            except Exception:
                pass
            self.root.destroy()
if __name__ == "__main__":
    root = tk.Tk()
    app = MonitorApp(root)
    root.protocol("WM_DELETE_WINDOW", app.on_close)
    root.mainloop()
```