



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA QUÍMICA
CURSO DE GRADUAÇÃO EM ENGENHARIA QUÍMICA



JÚLIA BARBOSA MORAIS

**DESENVOLVIMENTO E ANÁLISE DE ESTRATÉGIAS DE CONTROLE
UTILIZANDO UM DISPOSITIVO DE CONTROLE DE TEMPERATURA E A
PLATAFORMA OPENPLC**

UBERLÂNDIA
2024

JÚLIA BARBOSA MORAIS

**DESENVOLVIMENTO E ANÁLISE DE ESTRATÉGIAS DE CONTROLE
UTILIZANDO UM DISPOSITIVO DE CONTROLE DE TEMPERATURA E A
PLATAFORMA OPENPLC**

Monografia de graduação apresentada à Universidade Federal de Uberlândia como parte dos requisitos necessários para a aprovação na disciplina de Trabalho de Conclusão de Curso do curso de Engenharia Química.

Orientador: Prof. Dr. Thiago Vaz da Costa

**UBERLÂNDIA
2024**

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que estiveram presentes e foram essenciais durante toda a minha jornada e a conclusão deste ciclo tão importante para mim.

Primeiramente, agradeço a Deus por iluminar meu caminho, por me conceder tantas bênçãos e oportunidades, por me fortalecer nos momentos difíceis, e por me cercar de tantas pessoas incríveis ao longo da minha vida.

Aos meus pais, Míriam e Idenir, por não pouparem esforços para que eu pudesse chegar até aqui e por acreditarem nos meus sonhos. Vocês são meu porto seguro e minha maior fonte de inspiração e orgulho. Obrigada por tanto amor, dedicação e compreensão. Aos meus irmãos, Roberta e Gabriel, por estarem sempre ao meu lado, por compartilharem a vida comigo e por serem as melhores pessoas do meu "*Big Three*". A toda a minha família, por todo carinho e incentivo. Nada disso seria possível sem o apoio incondicional de vocês.

Ao meu namorado, Alberto, por ser o melhor companheiro e maior incentivador. Obrigada por acreditar em mim mesmo quando eu duvidava e por me dar forças para continuar. Além de ter me presenteado com uma segunda família, especialmente minha cunhada Maria Eugênia, por estar sempre presente me incentivando e inspirando, mesmo a milhares de quilômetros.

Ao PET-EQ, por ter sido uma experiência transformadora, muito além do acadêmico. Por todos os projetos, minicursos, atividades e oportunidades de poder fazer a diferença e impactar a pessoas. Mais ainda pelas pessoas que permitiu conhecer e amizades que eu fiz, em especial, Maria Fernanda, Elmis, Ana Luísa, Lucas Netto, Antônio e Gabriel Bonifácio. Obrigada por tornarem a jornada mais leve e divertida, por todas as trocas e muitas risadas na sala do PET. Estendo meu agradecimento aos amigos que dividiram a jornada da graduação comigo, João Vicente, Laís, Júlia Damasceno.

Aos meus professores, minha profunda admiração e gratidão. Obrigada por compartilharem seu conhecimento e sabedoria e por contribuírem tanto para minha formação. Um agradecimento especial aos professores: Thiago, Marisa, Danylo e Fábio, por me inspirarem inúmeras vezes e acreditarem no meu potencial. Principalmente, meu orientador Thiago Costa, deixo todo meu agradecimento por todos ensinamentos, orientação, paciência ao longo do desenvolvimento deste projeto.

Por fim, meu agradecimento a todos que estiveram, de alguma forma, presentes na minha trajetória. Cada um que contribuiu para o meu crescimento, tanto profissional quanto pessoal, guardarei esses momentos e aprendizados com muito carinho.

RESUMO

A automação e o controle de processos industriais são fundamentais para o aprimoramento das operações, tornando-as cada vez mais eficientes e seguras. Nesse contexto, os Controladores Lógicos Programáveis se destacam como dispositivos essenciais para garantir esse controle. No entanto, seu alto custo e complexidade podem restringir seu uso em ambientes acadêmicos e didáticos, tornando necessário a busca por alternativas mais acessíveis para simulação e ensino prático de controle de processos. Desta forma, o presente trabalho tem como objetivo desenvolver um sistema de pequena escala utilizando o microcontrolador Arduino Uno como Controlador Lógico Programável, para a implementação e análise de estratégias de controle de temperatura, buscando proporcionar uma alternativa de baixo custo que simule o quanto for possível o controle em um ambiente industrial. Para tanto, foi empregado o *software* de código aberto *OpenPLC*, para a configuração do microcontrolador na linguagem *Ladder* e para comunicação entre os componentes via protocolo *Modbus TCP/IP*. Além disso, a linguagem de programação *Python* foi utilizada para a programação, monitoramento e supervisão do sistema. No estudo, foram implementadas e comparadas duas abordagens de controle para o sistema MIMO: a técnica clássica Proporcional-Integral-Derivativo (PID) com sintonias feitas utilizando o IMC e ITAE e a técnica avançada de Controle Preditivo Baseado em Modelo (MPC). Os resultados demonstraram que ambas as técnicas tiveram bom desempenho no rastreamento de *setpoints*. O PID, ajustado com o método ITAE, destacou-se pela resposta mais rápida e agressiva, enquanto o ajuste IMC resultou em respostas mais suaves, com menor *overshoot*. Já o controle MPC mostrou-se uma opção com respostas mais suaves e consistentes e boa aderência ao modelo matemático utilizado, com contribuição da simulação para o ajuste e sintonia desse tipo de controle. Dessa forma, a integração do aparato experimental, do *software OpenPLC* e do *Python* mostrou-se uma ferramenta didática acessível e versátil, aproximando o sistema de um ambiente industrial adequado para o ensino prático de controle de processos industriais.

Palavras-chaves: Controle de temperatura, CLP, Arduino, OpenPLC, Python, PID, MPC.

ABSTRACT

Automation and control of industrial processes are fundamental to improving operational efficiency and safety. In this context, Programmable Logic Controllers are essential for achieving effective control. However, their high cost and complexity can limit their use in academic and educational environments, which makes it necessary to seek more affordable alternatives for the simulation and practical teaching of process control. Thus, this paper aims to develop a small-scale plant using the Arduino Uno microcontroller as a Programmable Logic Controller for the implementation and analysis of temperature control strategies, in order to provide a low-cost alternative that simulates industrial control as closely as possible. To this objective, the open-source OpenPLC software was employed for configuring the controller in Ladder Logic and for communication between components via the Modbus TCP/IP protocol. Additionally, the Python programming language was used for system programming, monitoring, and supervision. In the study, two control approaches for the MIMO system were implemented and compared: the classical Proportional-Integral-Derivative (PID) technique with tunings made using the IMC and ITAE methods and the advanced Model Predictive Control (MPC) technique. The results demonstrated that both techniques performed well in tracking setpoints. The PID tuned with the ITAE method stood out for its faster and more aggressive response, while the IMC tuning resulted in smoother responses with less overshoot. On the other hand, MPC control proved to be an option with a smoother and consistent responses and good adherence to the mathematical model used, with the simulation contributing to the tuning of this type of control. Thus, integrating the experimental apparatus, OpenPLC software and Python proved to be an accessible and versatile educational tool, which brings the system closer to an industrial environment suitable for practical teaching of industrial process control.

Keywords: Temperature Control, PLC, Arduino, OpenPLC, Python, PID, MPC.

LISTA DE ILUSTRAÇÕES

Figura 1 – Pirâmide de Automação	15
Figura 2 – Conceito básico para MPC	21
Figura 3 – Partes de um controlador programável modular - <i>Mitsubishi Automation</i>	29
Figura 4 – Diagrama <i>Ladder</i>	30
Figura 5 – Dispositivo TCLab	33
Figura 6 – Aparato Experimental	35
Figura 7 – Arduino UNO	36
Figura 8 – Representação componentes e pinos Arduino UNO	36
Figura 9 – Conjunto: Sensor e Transistor com Dissipador de Calor	38
Figura 10 – Interface do <i>software</i> Open PLC Editor	41
Figura 11 – Configuração por meio da aba ' <i>Slave Devices</i> ' no <i>OpenPLC Runtime</i>	42
Figura 12 – Interface da aba Monitoramento do <i>software OpenPLC Runtime</i>	43
Figura 13 – <i>Scaling</i> das variáveis de saída dos sensores de temperatura	43
Figura 14 – Filtro de média móvel	44
Figura 15 – <i>Scaling</i> das variáveis de entrada dos transistores	45
Figura 16 – Acionamento dos LEDs	45
Figura 17 – Bloco PID <i>OpenPLC</i>	46
Figura 18 – Interface gráfica da Tela de Monitoramento do Sistema do programa em <i>Python</i>	48
Figura 19 – Esquema comunicação serial entre ambientes de desenvolvimento e aplicação	49
Figura 20 – Controle PID <i>Openplc Editor</i>	51
Figura 21 – Parâmetros sintonia QDMC <i>Python</i>	54
Figura 22 – Curva das Temperaturas 1 e 2 em resposta ao degrau da porcentagem do Aquecedor 1 - $G_{11}(s)$ e $G_{21}(s)$	56
Figura 23 – Curva das Temperaturas 1 e 2 em resposta ao degrau da porcentagem do Aquecedor 2 - $G_{21}(s)$ e $G_{22}(s)$	56
Figura 24 – Diagrama de blocos simplificado do Sistema de Controle MIMO com pareamento 1-1/2-2	58
Figura 25 – Teste Controle das Temperaturas 1 e 2 - Problema Servo Sintonia pelo método IMC($\tau_c = 0,25\tau_p$)	59
Figura 26 – Teste Controle das Temperaturas 1 e 2 - Problema Servo Sintonia pelo método IMC($\tau_c = 0,4\tau_p$)	59
Figura 27 – Teste Controle das Temperaturas 1 e 2 - Problema Servo Sintonia pelo método ITAE	60
Figura 28 – Teste Controle das Temperaturas 1 e 2 - Problema Servo Comparativo Sintonias	60
Figura 29 – Comparativo valores medidos e Modelo baseado nos Balanços de Energia do Sistema em resposta a diversas alterações nos Aquecedores	62
Figura 30 – Resposta em malha fechada com o controlador QDMC utilizando os parâmetros da Simulação 1	64

Figura 31 – Resposta em malha fechada com o controlador QDMC utilizando os parâmetros da Simulação 2	64
Figura 32 – Teste Controle das Temperaturas 1 e 2 - Problema Servo Controlador QDMC	65
Figura 33 – Comparativo da resposta do sistema em malha fechada com os parâmetros da simulação e dados experimentais do controle QDMC	65

LISTA DE TABELAS

Tabela 1 – Identificação de Pinos de Entradas e Saídas no Arduino UNO	37
Tabela 2 – Parâmetros Do Modelo Matemático	40
Tabela 3 – Relação e Endereçamento de Variáveis de Entrada, Saída e Memórias	47
Tabela 4 – Parâmetros de sintonia do controle PI determinados por diferentes métodos Malha 1: $T_1 - U_1$	58
Tabela 5 – Parâmetros de sintonia do controle PI determinados por diferentes métodos Malha 2: $T_2 - U_2$	58
Tabela 6 – Índices de Desempenho para cada sintonia do controlador PI	61
Tabela 7 – Parâmetros Estimados do Modelo Matemático	62
Tabela 8 – Parâmetros de sintonia do controle QDMC para Simulação	63
Tabela 9 – Índices de Performance dos Controladores PID e MPC	66

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ADC	<i>Analogic Digital Conversor</i>
APC	<i>Advanced Process Control</i>
AVR	<i>Alf-Veegard-RISC</i>
CLP	Controlador Lógico Programável
DMC	<i>Dynamic Matrix Control</i>
ERP	<i>Enterprise Resource Planning</i>
EWMA	<i>Exponentially Weighted Moving Average</i>
FDB	<i>Function Block Diagram</i>
FOPDT	<i>First Order Plus Dead Time</i>
HMI	<i>Human Machine Interface</i>
IAE	<i>Integral of the Absolute Error</i>
IEC	<i>International Electrotechnical Commission</i>
IL	<i>Instruction List</i>
IMC	<i>Internal Model Control</i>
IoT	<i>Internet of Things</i>
ISA	<i>International Society for Automation</i>
ISE	<i>Integral of the Squared Error</i>
ITAE	<i>Integral of the time-weighted absolute error</i>
LD	<i>Ladder</i>
MIMO	<i>Multiple-Input and Multiple-Output</i>
MPC	<i>Model predictive control</i>
NEMA	<i>National Electrical Manufacturers Association</i>
PC	<i>Personal computer</i>
PI	Proporcionanl-Integral

PID	Proporcional-Integral-Derivativo
POU	<i>Program Organization Units</i>
PWM	<i>Pulse Width Modulation</i>
QDMC	<i>Quadratic Dynamic Matrix Control</i>
QP	<i>Quadratic Program</i>
RGA	<i>Relative Gain Array</i>
RTO	<i>Real-Time Optimization</i>
RTU	<i>Remote Terminal Unit</i>
SCADA	<i>Supervisory Control and Data Acquisition</i>
SDCD	Sistema de Controle Distribuído
SFC	<i>Sequential Flow Chart</i>
ST	<i>Structured Text</i>
TCLab	<i>Temperature Control Lab</i>
TCP	<i>Transmission Control Protocol</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Objetivos	14
2	REVISÃO BIBLIOGRÁFICA E FUNDAMENTAÇÃO TEÓRICA	15
2.1	Controle de processos	15
2.2	Desenvolvimento do Sistema de Controle	16
2.3	Controle Proporcional Integral Derivativo - PID	17
2.4	Controle Preditivo baseado em Modelo - MPC	20
2.5	Controlador Lógico Programável	28
2.5.1	Composição do Hardware do Controlador Lógico Programável	29
2.5.2	Programação dos Controladores Lógicos Programáveis e o uso da Linguagem Ladder	30
2.5.3	Sistemas de Controle SCADA	31
2.5.4	Protocolo MODBUS: Comunicação entre Controladores Lógicos Programáveis	31
2.6	Microcontroladores como alternativa para aprendizado sobre CLPs	32
2.6.1	O uso do Arduino como um Microcontrolador	32
2.7	Projeto TCLab: uma pequena planta com grande potencial para estudo de controle	33
3	MATERIAIS E MÉTODOS	35
3.1	Descrição do aparato experimental	35
3.1.1	Arduino UNO	35
3.1.2	Componentes conectados ao Arduino	37
3.2	Modelo matemático do sistema a partir de equações de balanço de energia	38
3.3	Configuração do dispositivo utilizando a plataforma OpenPLC	40
3.3.1	Aplicação da plataforma OpenPLC	40
3.3.2	Programação do controlador em Ladder	43
3.3.3	Configuração de entradas e saídas analógicas do controlador	46
3.3.4	Seleção do tempo de amostragem para o sistema em malha aberta e em malha fechada	47
3.4	A aplicação da linguagem de programação Python	48
3.5	Sistema em malha aberta	49
3.6	Implementação das malhas de controle	50
3.6.1	PID	51
3.6.1.1	Método de Sintonia IMC	51
3.6.1.2	Método de Sintonia ITAE	52
3.6.2	MPC	52
3.6.3	Métricas de Desempenho dos Controladores	54
4	RESULTADOS E DISCUSSÕES	56
4.1	Sistema em malha aberta	56

4.2	Implementação do Controle PI	58
4.3	Implementação do Controle MPC	62
5	CONCLUSÃO	67
5.1	Sugestões para Trabalhos Futuros	67
	REFERÊNCIAS	69

1 INTRODUÇÃO

Com o avanço tecnológico acelerado das últimas décadas e a crescente demanda por processos mais eficientes e confiáveis, o setor industrial tem se transformado cada vez mais, influenciando diretamente o desenvolvimento das tecnologias de controle de processos e automação industrial. Essas mudanças não apenas vêm modificando a operação do setor, mas também estabelecem novos padrões de desempenho, segurança e qualidade na produção. O controle de processos, que envolve a supervisão e regulação das variáveis operacionais, juntamente com a automação industrial, que utiliza sistemas tecnológicos para executar operações de forma autônoma, têm sido fundamentais para otimizar a eficiência operacional, reduzir custos, melhorar a qualidade dos produtos e garantir a segurança das pessoas e do meio ambiente.

Neste contexto, são empregados diversos tipos de controle no meio industrial. Entre os mais amplamente utilizados estão o controle Proporcional-Integral-Derivativo (PID) e técnicas de controle avançado, como o Controle Preditivo Baseado em Modelo (MPC). O controle PID é uma técnica clássica, amplamente empregada devido à sua simplicidade e eficácia em diversas aplicações (SEBORG et al., 2010). No entanto, esse tipo de controle apresenta limitações evidentes com a evolução da complexidade dos processos, em que variáveis interdependentes e restrições operacionais precisam ser gerenciadas de maneira mais adequada. Neste cenário, o MPC se destaca como uma solução, oferecendo uma abordagem mais flexível e eficaz para o gerenciamento de sistemas dinâmicos complexos, com múltiplas variáveis, especialmente em ambientes industriais que exigem alta precisão e confiabilidade.

Para a implementação dessas técnicas no setor industrial, normalmente são empregados os Controladores Lógicos Programáveis (CLPs), dispositivos responsáveis por automatizar ações de controle e com capacidade de gerenciar processos complexos em tempo real. Os CLPs são dispositivos altamente confiáveis, programáveis para o controle e monitoramento de diversas variáveis críticas e adaptáveis a uma ampla gama de aplicações. Dessa forma, esse equipamento contribuiu para proporcionar às indústrias a capacidade de realizar ajustes de suas operações referentes às novas demandas ou mudanças no processo com maior agilidade e automação.

No âmbito da engenharia química, porém, o uso de CLPs apresenta desafios significativos. Seu custo elevado e a complexidade de instalação e manutenção tornam difícil a implementação em laboratórios e ambientes educacionais. Além disso, a dificuldade em reproduzir uma planta industrial de escala real nesse ambiente, juntamente com as preocupações de segurança, impõe barreiras ao aprendizado prático e à compreensão profunda dos conceitos de controle e automação industrial (WEBB; REIS, 2013; SILVEIRA, 2022).

Diante disso, a utilização de plantas didáticas de pequena escala, de baixo custo e representativas dos processos industriais, surge como uma das soluções para a aplicação prática de estratégias de controle. Para a implementação do controle nessas plantas, o Arduino se destaca como uma alternativa viável, funcionando como um microcontrolador de fácil uso, versátil e com uma plataforma de código aberto. Dessa forma, aplicados em conjunto, é possível empregar protótipos ou plantas mais simples para reproduzir algum fenômeno físico em escala reduzida,

viabilizando o ensino e pesquisa de estratégias de controle na prática (KVALSUND; WINKLER, 2023).

Entre as diversas abordagens de implementação de malhas de controle utilizando a plataforma Arduino, o controle de temperatura é uma das mais comuns. Essa variável é crítica em muitos processos industriais que requerem precisão e confiabilidade. Um exemplo notável é o TCLab (*Temperature Control Lab*), uma planta térmica desenvolvida com Arduino Leonardo, amplamente utilizada como recurso portátil e de baixo custo para auxílio de aprendizagem de conceitos de dinâmica e teoria de controle em diversas universidades. Esse projeto de *hardware* aberto, tem sido aplicado em diversos estudos e demonstrações práticas para fins educativos (PARK et al., 2020; OLIVEIRA; HEDENGREN; ROSSITER, 2020; OLIVEIRA; SOARES; CARDOSO, 2022; GUZMÁN et al., 2022; ROSSITER et al., 2019).

Levando isso em consideração, o estudo de Paparella (2021) aborda também o desenvolvimento de um sistema composto por *hardware* e *software* para promover o aprendizado prático de controle e automação. Com a utilização de *softwares* de código aberto, como o *OpenPLC* para a configuração do Arduino como um controlador lógico programável, e o *ScadaBR* como supervisor, o trabalho ofereceu uma simulação mais realista das funções de controladores industriais, ampliando seu potencial educativo.

O TCLab foi inicialmente implementado na linguagem de programação Arduino, baseada em C/C++. Esta linguagem é amplamente empregada no desenvolvimento de protótipos e sistemas embarcados devido à sua simplicidade e flexibilidade. Porém, apesar de sua eficácia em aplicações de automação básica e projetos de pequena escala, essa linguagem não é a solução que mais se aproxima do setor industrial, quando se trata de controle de processos, onde normalmente é empregada as linguagens de programação de CLPs, como o *Ladder*.

Diante dessas circunstâncias, este trabalho propõe o desenvolvimento de um dispositivo de pequena escala, inspirado no TCLab, utilizando o Arduino Uno para implementação de malhas de controle. Para isso, assim como no estudo de Paparella (2021), foi aplicado a plataforma *OpenPLC* para o desenvolvimento e configuração do Arduino como CLP e a linguagem *Python* para monitoramento do sistema em tempo real. Além disso, foram exploradas duas técnicas de controle, o PID e controle avançado MPC. Essa abordagem busca oferecer uma solução de baixo custo e alta versatilidade, facilitando a compreensão e aplicação de estratégias de controle industrial, ao mesmo tempo em que contribui para o avanço do ensino de controle de processos industriais.

1.1 Objetivos

O objetivo principal deste trabalho consiste em utilizar ferramentas de automação de código aberto implementadas em um dispositivo de baixo custo para aplicações de controle utilizando um microcontrolador Arduino como um Controlador Lógico Programável. Integrado à plataforma *OpenPLC* e utilizando a linguagem de programação *Python*, o projeto busca obter um sistema representativo de sistemas de controle industrial, com o intuito de oferecer ferramenta didática para a implementação e análise de estratégias de controle.

Os objetivos específicos são:

1. Adaptar o uso do TcLab em aplicações de controle de temperatura, a partir do uso da linguagem *Ladder* segundo a Norma IEC 61131-3, de modo a estabelecer um ambiente em que o Arduino seja programado e configurado como um Controlador Lógico Programável a partir da plataforma *OpenPLC*;
2. Implementar e avaliar diferentes técnicas de controle, incluindo o controle Proporcional-Integral-Derivativo (PID) com diferentes métodos de sintonia e o Controle Avançado baseado em Modelo (MPC), para comparar a eficiência e a precisão de cada método no contexto da planta desenvolvida;
3. Utilizar a linguagem de programação *Python* para coleta de dados, monitoramento do sistema em tempo real e implementação do controle MPC.

2 REVISÃO BIBLIOGRÁFICA E FUNDAMENTAÇÃO TEÓRICA

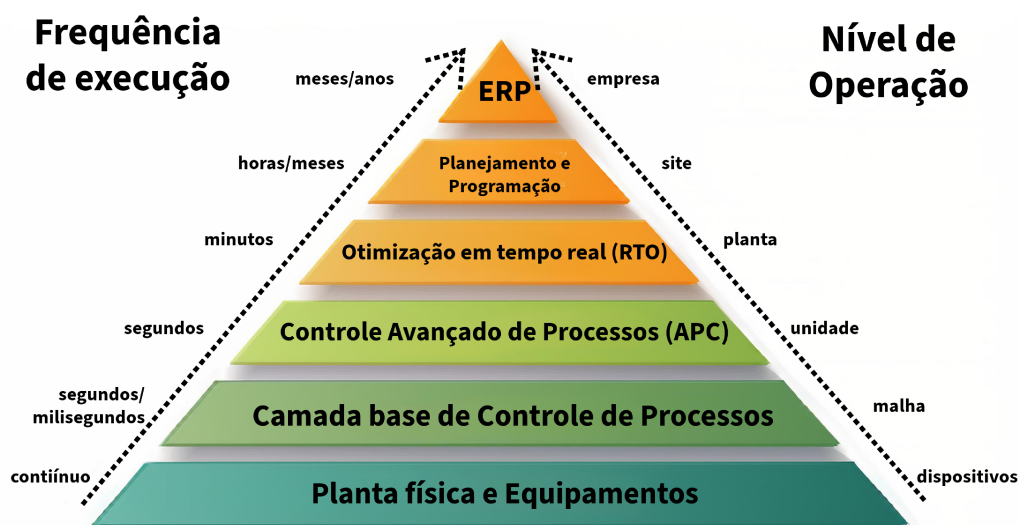
2.1 Controle de processos

Controle de processos é a área de Controle e Automação que se concentra na teoria e tecnologia envolvida na regulação e monitoramento de operações para garantir que processos industriais permaneçam o mais próximo possível do seu estado ideal. Sua função é assegurar que o sistema apresente uma dinâmica desejável e responda de forma adequada em estado estacionário, através de medidas imediatas para corrigir desvios e anormalidades. Por meio do controle, alcançam-se objetivos cruciais como segurança e eficiência operacional, redução de custos de produção, além do cumprimento de requisitos ambientais e de qualidade do produto. (SEBORG et al., 2010).

Para cumprir tais objetivos operacionais, uma solução de controle de processo deve satisfazer os seguintes requisitos operacionais: em condições normais, entregar o produto correto com a qualidade e quantidade adequadas; em caso de interrupções no processo, desviar temporariamente do objetivo operacional, mas manter-se dentro dos limites operacionais; e durante mudanças operacionais, assegurar uma transição rápida e segura de uma condição para outra (NIU; XIAO, 2022).

De acordo com Niu e Xiao (2022), o controle de processos pode ser entendido como uma hierarquia, frequentemente referida como Pirâmide de Automação, subdividida em camadas, como mostrado na Figura 1. Cada nível da pirâmide serve como alicerce para as camadas superiores, que realizam funções mais avançadas. Ao mesmo tempo, cada camada subjacente depende das anteriores para operar adequadamente. À medida que se aproxima do topo, as aplicações possuem um escopo mais amplo e são executadas com menor frequência.

Figura 1 – Pirâmide de Automação



Fonte: Adaptado de Niu e Xiao (2022)

1. **Planta física e Equipamentos:** A base da pirâmide é composta pela operação da planta e seus dispositivos, como sensores, transmissores e atuadores, que fornecem o conjunto de informações básicas das variáveis de processo e os meios para determinar o controle;
2. **Camada base de Controle de Processos:** Nesta camada, predominam controladores PID, implementados em sistemas de Controle Lógico Programável (CLP) ou Sistemas de Controle Distribuído (SDCD). Esses controladores realizam tarefas de controle em tempo real, para manter a operação da planta nos pontos operacionais ideais e dentro das limitações operacionais estabelecidas;
3. **Controle Avançados de Processos (APC):** Geralmente, envolve sistemas mais complexos de controle PID, bem como a utilização de Controle Preditivo Baseado em Modelo (MPC). Esta camada foca na operação estável de uma área de processo mais ampla;
4. **Otimização de tempo real (RTO):** Visa otimizar a lucratividade de uma unidade operacional ou de toda a planta através de uma combinação de modelos matemáticos detalhados, funções objetivas econômicas específicas e acesso em tempo real aos sistemas de controle;
5. **Planejamento e Programação:** Essa camada fornece informações gerenciais e de apoio à decisão, voltadas para o planejamento e a programação de médio prazo. Inclui aspectos como produção, compras, estoque e expedição;
6. **Planejamento de Recursos Empresariais (ERP):** No topo da pirâmide, está o planejamento de recursos empresariais, que abrange a gestão corporativa da cadeia de suprimentos. Estabelece objetivos operacionais de longo prazo para melhorar o controle e a otimização das operações.

Cada uma dessas camadas interage de forma dinâmica, visando garantir que o processo industrial como um todo opere de maneira eficiente e segura, desde a operação básica dos equipamentos até a gestão estratégica da empresa.

2.2 Desenvolvimento do Sistema de Controle

Segundo Seborg et al. (2010), existem duas perspectivas principais no projeto de controle de sistemas. Na abordagem tradicional, o ajuste do controlador é realizado com base no conhecimento prévio do processo, normalmente após o projeto da planta estar bem definido e a instalação do sistema de controle já ter ocorrido. No entanto, essa abordagem apresenta limitações significativas, pois o projeto e a operacionalidade da planta influenciam diretamente a dinâmica do processo. Como resultado, o sistema de controle pode não ser otimizado, levando a um desempenho subótimo. Neste trabalho, a abordagem tradicional foi adotada para o controle PID.

Uma abordagem mais eficaz é desenvolver primeiramente um modelo dinâmico do processo. Esse modelo permite que o sistema de controle seja projetado de maneira mais precisa,

com uma compreensão clara das interações e das variáveis dinâmicas envolvidas. Essa abordagem baseada em modelo dinâmico foi a escolhida neste trabalho para o desenvolvimento do controle avançado MPC, permitindo um *design* adaptado às condições reais do processo.

Os principais passos necessários para o desenvolvimento de um sistema de controle baseado em modelo dinâmico são:

1. Formulação dos objetivos de controle: Esta etapa envolve a fundamentação do propósito operacional particular de cada planta, levando em conta as restrições do processo.
2. Elaboração do modelo dinâmico do processo: O modelo dinâmico pode ser fundamentado teoricamente em princípios físicos e químicos, como as leis de conservação de massa e energia, ou pode ser desenvolvido empiricamente utilizando dados experimentais. Neste último caso, a validação é crucial para garantir que ele reflita com precisão o comportamento do processo.
3. Projeto e sintonia do controlador: Nesta fase, é desenvolvida uma estratégia de controle apropriada que satisfaça os objetivos de forma eficiente, respeitando as restrições do processo.
4. Testes em simulação: Antes da implementação no processo real, diferentes estratégias de controle podem ser testadas em simulação, para auxiliar a escolha de diferentes estratégias de controle. Além disso, esses testes oferecem estimativas iniciais e permitem refinar as configurações do controlador e avaliar como ele responderá a diferentes cenários operacionais.
5. Implementação no processo e Monitoramento de desempenho: O sistema de controle é então implementado na planta, utilizando as estimativas preliminares da fase de *design* como ponto de partida. Posteriormente, o desempenho do sistema é monitorado continuamente, e ajustes adicionais podem ser feitos para otimizar o funcionamento.

Essa abordagem centrada no desenvolvimento de um modelo dinâmico, permite um controle mais preciso e que se adapta melhor às necessidades e condições operacionais da planta. Adicionalmente, promove uma maior flexibilidade e capacidade de resposta à mudanças ou perturbações inesperadas no processo.

2.3 Controle Proporcional Integral Derivativo - PID

O Controle Proporcional Integral Derivativo (PID) é uma das técnicas de controle mais populares e empregadas em sistemas de controle automático, especialmente em ambientes industriais. Sua popularidade deve-se à combinação de simplicidade, flexibilidade e eficácia, o que permite sua aplicação em uma vasta gama de processos industriais. O PID opera em uma estrutura de *feedback*, onde a ação de controle é ajustada continuamente para minimizar a diferença, ou erro, entre a variável controlada e seu valor de referência (*setpoint*).

Um controlador PID é composto por três modos de controle: o proporcional, o integral e o derivativo. A ação proporcional é diretamente dependente do erro, ajustando-se de forma proporcional a ele. Isso significa que quanto maior o erro, maior será a correção aplicada. Essa característica ajuda a acelerar a resposta do sistema, mas o controle proporcional sozinho não consegue eliminar completamente o erro de estado estacionário, conhecido como *offset*.

Para lidar com o *offset*, o componente integral é adicionado ao controlador. A ação integral considera a soma acumulada dos erros ao longo do tempo, aplicando uma correção que visa eliminar o erro residual que o controle proporcional não pode resolver. No entanto, o uso excessivo do componente integral pode introduzir instabilidade ao sistema, especialmente em processos dinâmicos rápidos.

O terceiro componente, o termo derivativo, corresponde à taxa de mudança do erro, ou seja, a velocidade com que o erro está se alterando. Isso proporciona uma forma de previsão que ajuda a amortecer oscilações e a melhorar a estabilidade do sistema. Assim, o componente derivativo é frequentemente utilizado para suavizar a resposta do controlador e prevenir oscilações indesejadas. Porém, o fator derivativo pode deixar o controlador muito sensível a ruídos de alta frequência.

Matematicamente, a saída de um controlador PID, representada por $u(t)$, é dada por uma combinação ponderada dos três componentes acima, como mostrado na Equação 2.1. Outra forma de expressar a ação de controle é através da sua função de transferência, conforme a formulação ISA (*International Society for Automation*), ilustrada na Equação 2.2.

$$u(t) = K_c \left(e(t) + \frac{1}{\tau_i} \int_0^t e(t) dt + \tau_d \frac{de(t)}{dt} \right) \quad (2.1)$$

$$G_{PID}(s) = \frac{U(s)}{E(s)} = K_c \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad (2.2)$$

onde K_c é o ganho proporcional do controlador, τ_i a constante de tempo integral, τ_d a constante de tempo derivativo, e o erro, t o tempo e s é a variável de Laplace.

Em termos de comportamento, existem dois tipos principais de ação de controle: direta e reversa. Um controlador opera em modo direto quando um aumento na saída do controlador resulta em um aumento correspondente na saída da planta. Neste caso, o sinal de entrada do controlador, correspondente ao erro da variável controlada em relação ao *setpoint*, representado usualmente por $E(s)$ é dado pela Equação 2.3. Por outro lado, se um aumento na saída do controlador provoca uma diminuição na saída da planta, o controlador opera em modo reverso, com $E(s)$ representado pela Equação 2.4. A escolha entre ação direta ou reversa depende do tipo de processo a ser controlado e dos objetivos específicos de controle (SEBORG et al., 2010).

$$E(s) = R(s) - Y(s) \quad (2.3)$$

$$E(s) = Y(s) - R(s) \quad (2.4)$$

sendo $R(s)$ o sinal do *setpoint* e $Y(s)$ o sinal da variável controlada.

Com o avanço da tecnologia, os controladores PID foram adaptados para operar em sistemas digitais. Na versão digital, o controlador PID é discretizado com base em um tempo de amostragem das variáveis do processo. Isso permite uma implementação flexível em micro-controladores, Controladores Lógicos Programáveis (CLPs) e Sistemas de Controle Distribuído (SDCD), que são comuns em ambientes industriais modernos.

Duas das formas mais comuns de implementar o controlador PID digital são através do algoritmo da posição e do algoritmo da velocidade. O algoritmo da posição calcula a saída do controlador em termos absolutos. Conforme a Equação 2.5, a saída $u(k)$ em um instante de amostragem k é determinada pela soma dos componentes proporcional, integral e derivativa. Isso significa que a saída do controlador é atualizada com base na posição absoluta da variável de controle em relação ao *setpoint* atual (SEBORG et al., 2010).

$$u(k) = u_{ee} + K_c \left[e_k + \frac{\Delta t}{\tau_i} \sum_{n=1}^k e_n + \frac{\tau_d}{\Delta t} (e_k - e_{k-1}) \right] \quad (2.5)$$

Este método é direto e intuitivo, mas pode ser suscetível a dificuldades de implementação do algoritmo de controle e problemas de *windup* na ação integral, especialmente em sistemas com saturação na saída do atuador. O problema de *windup* acontece quando a saída do controlador atinge seus limites máximos ou mínimos, porém a ação integral continua somando o erro, acumulando erro de forma desproporcional. Isso causa uma saturação do atuador e um comportamento excessivamente lento ou instável na recuperação do sistema (SEBORG et al., 2010).

O algoritmo da velocidade, demonstrado pela Equação 2.6, por outro lado, calcula a mudança incremental na saída do controlador entre dois instantes de amostragem consecutivos. Em vez de calcular a posição absoluta, ele foca na variação da saída do controlador, o que pode ser vantajoso para evitar o problema de *windup* e melhorar a resposta do sistema em casos onde a saída do atuador está saturada. O algoritmo da velocidade é especialmente útil em sistemas onde pequenas alterações na variável de controle são necessárias para manter a estabilidade do processo (SEBORG et al., 2010).

$$\Delta u_k = K_c \left[(e_k - e_{k-1}) + \frac{\Delta t}{\tau_i} e_k + \frac{\tau_d}{\Delta t} (e_k - 2e_{k-1} + e_{k-2}) \right] \quad (2.6)$$

A escolha entre o algoritmo da posição e o algoritmo da velocidade depende das características do processo e dos requisitos de controle. Em muitos casos, a implementação digital de controladores PID modernos incorpora técnicas adicionais, como *anti-windup* e filtros para suavizar a resposta derivativa, melhorando ainda mais o desempenho do controlador.

Embora o PID tenha sido a técnica dominante de controle ao longo de décadas, ele enfrenta desafios significativos, especialmente na sintonia de sistemas complexos e na gestão de sistemas com múltiplas entradas e múltiplas saídas (MIMO, na sigla em inglês) que operam

sob restrições rigorosas. Esses desafios frequentemente resultam em desempenho ineficiente, o que impulsionou a busca por controladores avançados. Uma das respostas a esses desafios é o Controle Preditivo Baseado em Modelo (MPC), que oferece uma abordagem para lidar com a complexidade e as interações entre múltiplas variáveis, melhorando o desempenho do sistema de controle em ambientes industriais modernos (NIU; XIAO, 2022).

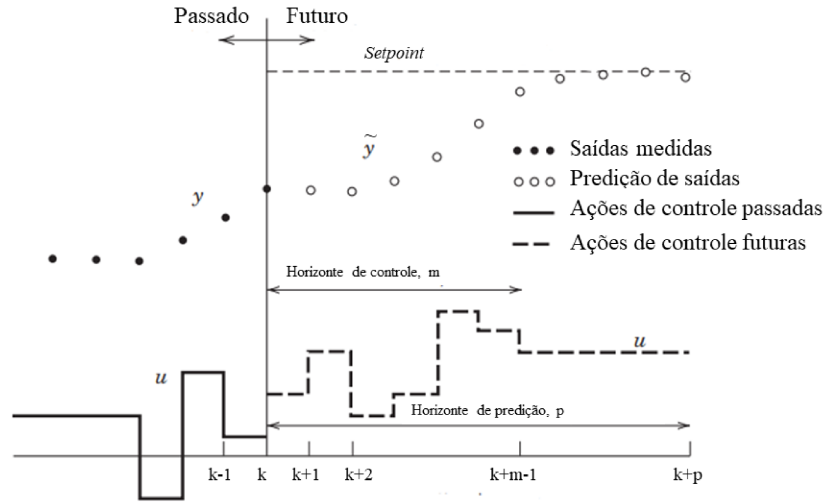
2.4 Controle Preditivo baseado em Modelo - MPC

O Controle Preditivo Baseado em Modelo (*Model Predictive Control - MPC*) é uma abordagem avançada de controle que utiliza o modelo do processo para prever o comportamento futuro das saídas de um sistema. Desenvolvido na década de 1970, o MPC surgiu em resposta aos desafios crescentes enfrentados pelas indústrias de processos, que precisavam operar sob condições cada vez mais exigentes. A complexidade inerente ao controle de processos multivariáveis, aliada a restrições operacionais rígidas, tornou imperativo a adoção de técnicas de controle mais sofisticadas e otimizadas. O MPC destaca-se pela sua capacidade de antecipar e ajustar as ações de controle com base em previsões futuras realizadas a partir do modelo da planta, o que é essencial para garantir tanto a eficiência quanto a segurança operacional em ambientes complexos (CAMACHO; ALBA, 2013).

O MPC utiliza um modelo dinâmico do sistema para prever a resposta futura do processo. Com base nessa previsão, o controlador resolve um problema de otimização em tempo real, frequentemente apresentado na forma de um problema de programação quadrática ou linear. A solução desse problema fornece a sequência ideal de ações de controle, minimizando uma função de custo que considera tanto a previsão da saída do processo quanto o esforço de controle necessário, penalizando também o uso excessivo das ações de controle.

De acordo com a Figura 2, a cada instante de tempo k , é realizada a predição da variável controlada ao longo do horizonte de predição p , com base nos valores dessa variável nos instantes anteriores. O controlador gera um vetor contendo variações do sinal de controle, $\Delta u(k)$ no horizonte de controle m , que minimizam o erro entre as saídas preditas e o *setpoint*, além de penalizar o esforço de controle. Porém, apenas o primeiro elemento dessa sequência é aplicado ao sistema, enquanto os demais são descartados e recalculados no próximo ciclo de controle, com base nas novas previsões e condições do sistema (ROSSITER, 2017).

Figura 2 – Conceito básico para MPC



Fonte: Adaptado de Seborg et al.(2010)

Uma das primeiras implementações práticas do MPC foi o algoritmo *Dynamic Matrix Control* (DMC), desenvolvido na década de 1970 pela *Shell Oil Company* para otimizar o controle de processos industriais complexos. O DMC continua sendo amplamente empregado, contudo, possui limitações quando aplicado em sistemas com interações não lineares e restrições rigorosas. Para superar as limitações relacionadas as restrições, foi incorporado o *Quadratic Dynamic Matrix Control* (QDMC), que acrescenta um termo quadrático à função objetivo. Assim, o QDMC permite uma melhor consideração das restrições operacionais, melhorando a estabilidade do controle e a capacidade de otimização do processo. A seguir, apresenta-se o desenvolvimento do controlador QDMC conforme descrito em Morari et al. (2002).

Considerando um sistema MIMO que tem n_y saídas, tal que $y_l \in \mathbb{R}^{n_y}$ e v_j entradas, tal que $v_j \in \mathbb{R}^{n_v}$. Sendo assim, o vetor de saída é definido pela Equação 2.7 e o vetor de entrada pela Equação 2.8.

$$y(k-1) = \left[y_1(k-1), \dots, y_{n_y}(k-1) \right]^T \quad (2.7)$$

$$\Delta v(k-1) = \left[\Delta v_1(k-1), \dots, \Delta u_{n_v}(k-1) \right]^T \quad (2.8)$$

O modelo de resposta ao degrau (do inglês, *Step Response Model*), é gerado aplicando-se degraus unitário a um processo que está em estado estacionário. Os coeficientes do modelo correspondem aos valores de saída em cada momento de tempo. Assim, forma-se a Matriz de coeficientes de resposta ao degrau conforme a Equação 2.9.

$$S_i = \begin{bmatrix} s_{1,1,i} & s_{1,2,i} & \cdots & s_{1,n_u,i} \\ s_{2,1,i} & & & \\ \vdots & & & \\ s_{n_y,1,i} & s_{n_y,2,i} & \cdots & s_{n_y,n_u,i} \end{bmatrix} \quad (2.9)$$

onde $s_{l,j,i}$ é o i ésimo coeficiente de resposta ao degrau relacionado a entrada j com a saída l . A matriz de resposta ao degrau S , definida na Equação 2.10 é uma descrição completa de como uma determinada entrada afeta a saída do sistema considerando n intervalos de tempo. Essa matriz de dimensão $(n_y \cdot n) \times n_v$ é obtida concatenando as matrizes de coeficiente de resposta ao degrau.

$$S = [S_1^T, S_2^T, \dots, S_n^T]^T \quad (2.10)$$

Considerando um sistema, em que as variáveis de entrada, podem ser divididas em variáveis manipuladas u e variáveis de perturbação d , então o vetor de entrada pode ser definido conforme a Equação 2.11 e a Matriz de Resposta ao degrau de acordo com a Equação 2.12.

$$\Delta v(k) = \begin{bmatrix} \Delta u(k) \\ \Delta d(k) \end{bmatrix} \quad (2.11)$$

$$S = \begin{bmatrix} S^u & S^d \end{bmatrix} \quad (2.12)$$

O sistema de múltiplas saídas no instante k é definido como a Equação 2.13.

$$\tilde{Y}(k) = [\tilde{y}_0^T(k), \tilde{y}_1^T(k), \dots, \tilde{y}_{n-1}^T(k)]^T \quad (2.13)$$

em que $\tilde{y}_i(k)$ corresponde à:

$$\tilde{y}_i(k) = y(k+i) \text{ para } \Delta v(k+j) = 0; j \geq 0 \quad (2.14)$$

Ao adicionar o efeito da mudança das entradas Δv no tempo k , obtem-se:

$$\tilde{y}_0(k) = \tilde{y}_1(k-1) + S_1 \Delta v(k-1) \quad (2.15)$$

$$\tilde{y}_1(k) = \tilde{y}_2(k-1) + S_2 \Delta v(k-1) \quad (2.16)$$

$$\vdots$$

$$\tilde{y}_{n-1}(k) = \tilde{y}_n(k-1) + S_n \Delta v(k-1) \quad (2.17)$$

e substituindo as variáveis de entradas por variáveis manipuladas e de perturbação:

$$\tilde{y}_0(k) = \tilde{y}_1(k-1) + S_1^u \Delta u(k-1) + S_1^d \Delta d(k-1) \quad (2.18)$$

$$\tilde{y}_1(k) = \tilde{y}_2(k-1) + S_2^u \Delta u(k-1) + S_2^d \Delta d(k-1) \quad (2.19)$$

$$\vdots$$

$$\tilde{y}_{n-1}(k) = \tilde{y}_n(k-1) + S_n^u \Delta u(k-1) + S_n^d \Delta d(k-1) \quad (2.20)$$

Assim, têm-se a relação recursiva do modelo de resposta ao degrau do sistema multi-variável, que foi proposto na formulação original do *Dynamic Matrix Control* demonstrada na Equação 2.21.

$$\tilde{Y}(k) = \tilde{M}\tilde{Y}(k-1) + S\Delta v(k-1) = \tilde{M}\tilde{Y}(k-1) + S^u \Delta u(k-1) + S^d \Delta d(k-1) \quad (2.21)$$

onde a matriz $\tilde{M}$ é definida como

$$\tilde{M} = \begin{bmatrix} 0 & I & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & I & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & I \\ 0 & 0 & 0 & \cdots & \cdots & 0 & I \end{bmatrix}_n \quad (2.22)$$

sendo a matriz identidade I de dimensão $n_y \times n_y$.

A Equação 2.21 reflete o efeito da mudança de entradas $\Delta v(k-1)$ no futuro do sistema assumindo que não há mais alterações de entrada. Considerando a predição da saída ao longo de p intervalos de tempo, ou seja, para um horizonte de predição H_p e considerando um horizonte de controle H_c de tamanho m , têm-se que:

$$\begin{aligned} \mathcal{Y}(k+1|k) &= \begin{bmatrix} \tilde{y}_1(k) \\ \tilde{y}_2(k) \\ \tilde{y}_3(k) \\ \vdots \\ \tilde{y}_p(k) \end{bmatrix} + \begin{bmatrix} S_1^d \\ S_2^d \\ S_3^d \\ \vdots \\ S_p^d \end{bmatrix} \Delta d(k) + \begin{bmatrix} y_m(k) - \tilde{y}_0(k) \\ y_m(k) - \tilde{y}_1(k) \\ y_m(k) - \tilde{y}_2(k) \\ \vdots \\ y_m(k) - \tilde{y}_{p-1}(k) \end{bmatrix} \\ &+ \begin{bmatrix} S_1^u & 0 & 0 & \cdots & 0 \\ S_1^u & S_2^u & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ S_m^u & S_{m-1}^u & S_{m-2}^u & \cdots & S_m^u \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ S_p^u & S_{p-1}^u & S_{p-2}^u & \cdots & S_{p-m+1}^u \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \Delta u(k+2|k) \\ \vdots \\ \Delta u(k+p-1|k) \end{bmatrix} \end{aligned} \quad (2.23)$$

em $y_m(k)$ representa o valor da saída conforme realmente medido na planta. Definindo cada uma das matrizes como:

- $\mathcal{Y}(k+1|k)$ o vetor de predição das saídas:

$$\mathcal{Y}(k+1|k) = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+p|k) \end{bmatrix} \quad (2.24)$$

- $\mathcal{S}^u$ a matriz dinâmica:

$$\mathcal{S}^u = \begin{bmatrix} S_1^u & 0 & 0 & \cdots & 0 \\ S_1^u & S_2^u & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ S_m^u & S_{m-1}^u & S_{m-2}^u & \cdots & S_m^1 \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ S_p^u & S_{p-1}^u & S_{p-2}^u & \cdots & S_{p-m+1}^1 \end{bmatrix} \quad (2.25)$$

- o vetor $\mathcal{S}^d$, que multiplicado por $\Delta d(k)$ corresponde ao termo *feedforward*, uma vez que é considerado o efeito das perturbações:

$$\mathcal{S}^d = \begin{bmatrix} S_1^d \\ S_2^d \\ S_3^d \\ \vdots \\ S_p^d \end{bmatrix} \quad (2.26)$$

- $\mathcal{I}_p$ o vetor identidade de tamanho p , que ao ser multiplicado por $(y_m(k) - \tilde{y}_0(k))$ corresponde ao termo *feedback*:

$$\mathcal{I}_p = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix}_p \quad (2.27)$$

- $\Delta \mathcal{U}(k)$ o vetor de movimentos de controle atuais e futuros:

$$\Delta \mathcal{U}(k) = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \Delta u(k+2|k) \\ \vdots \\ \vdots \\ \Delta u(k+p-1|k) \end{bmatrix} \quad (2.28)$$

• e a matriz $\mathcal{M}$:

$$\mathcal{M} = \begin{cases} \begin{bmatrix} 0 & I & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & I & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & I & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & I \end{bmatrix}_p & \text{para } p < n \\ \begin{bmatrix} 0 & I & 0 & \dots & 0 & 0 \\ 0 & 0 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I & 0 \\ 0 & 0 & 0 & \dots & 0 & I \end{bmatrix}_p & \text{para } p \geq n \end{cases} \quad (2.29)$$

têm-se que a predição da saída é denotada de acordo com a Equação 2.30, onde os três primeiros termos são definidos por ações de controle passadas: $\tilde{Y}(k)$ e $\tilde{y}_0(k)$, e medidas atuais: $y_m(k)$ e $\Delta d(k)$ e o último termo descreve o efeito de movimentos futuros de variáveis manipuladas $\Delta \mathcal{U}(k)$, sendo $\mathcal{S}^u$ a matriz dinâmica.

$$\mathcal{Y}(k+1|k) = \mathcal{M}\tilde{Y}(k) + \mathcal{S}^d \Delta d(k) + \mathcal{J}_p(y_m(k) - \tilde{y}_0(k)) + \mathcal{S}^u \Delta \mathcal{U}(k) \quad (2.30)$$

O vetor erro, que consiste na diferença entre a trajetória do *setpoint* e as predições futuras é obtido pela Equação 2.31.

$$E_p(k+1|k) = \mathcal{R}(k+1) - \mathcal{M}\tilde{Y}(k) + \mathcal{S}^d \Delta d(k) + \mathcal{J}_p(y_m(k) - \tilde{y}_0(k)) \quad (2.31)$$

em que o vetor de trajetória do *setpoint* é dado por

$$\mathcal{R}(k+1) = \begin{bmatrix} r(k+1|k) \\ r(k+2|k) \\ \vdots \\ r(k+p|k) \end{bmatrix} \quad (2.32)$$

A função objetivo no QDMC é formulada de modo a minimizar os desvios entre as previsões geradas pelo modelo e o *setpoint* desejado ao mesmo tempo em que penaliza movimentos excessivos na variável de controle. Essa função geralmente é expressa como uma forma quadrática, que pondera tanto os erros de previsão quanto a magnitude das ações de controle, e pode ser escrita segundo a Equação 2.33. O critério quadrático penaliza grandes desvios proporcionalmente mais do que os menores, de modo que, na média, a saída permanece próximo de sua trajetória de referência.

$$\begin{aligned} \min_{\Delta \mathcal{U}(k)} \quad & \{ \|Q_y[\mathcal{Y}(k+1|k) - \mathcal{R}(k+1)]\|^2 + \|Q_u \Delta \mathcal{U}(k)\|^2 \} \\ \text{s.t.} \quad & C^u \Delta \mathcal{U} \leq C(k+1|k) \end{aligned} \quad (2.33)$$

onde Q_u e Q_y correspondem as matrizes diagonais de peso de entradas e saídas, respectivamente.

$$Q_u = \text{diag} \{Q_{u1}, \dots, Q_{um}\} \quad (2.34)$$

$$Q_y = \text{diag} \{Q_{y1}, \dots, Q_{yp}\} \quad (2.35)$$

Além disso, é importante mencionar que vetor de solução do DMC não apenas inclui as ações atuais a serem implementadas, mas também as ações para os próximos m intervalos de tempo. O vetor de restrições presente na Equação 2.33 escrito em termos da variável objetivo, incorpora as restrições das variáveis manipuladas, restrições de taxas variáveis manipuladas e restrições de variáveis de saída, sendo:

$$C^u = \begin{bmatrix} I_L \\ -I_L \\ I \\ -I \\ \mathcal{S}^u \\ -\mathcal{S}^u \end{bmatrix}. \quad (2.36)$$

$$C(k+1|k) = \begin{bmatrix} u_{\max}(0) - u(k-1) \\ \vdots \\ u_{\max}(m-1) - u(k-1) \\ u(k-1) - u_{\min}(0) \\ \vdots \\ u(k-1) - u_{\min}(m-1) \\ \Delta u_{\max}(0) \\ \vdots \\ \Delta u_{\max}(m-1) \\ \Delta u_{\max}(0) \\ \vdots \\ \Delta u_{\max}(m-1) \\ -(\mathcal{M}\tilde{Y}(k) + S^d \Delta d(k) + \mathcal{J}_p(y_m(k) - \tilde{y}_0(k))) + \mathcal{Y}_{\max} \\ \mathcal{M}\tilde{Y}(k) + S^d \Delta d(k) + \mathcal{J}_p(y_m(k) - \tilde{y}_0(k)) - \mathcal{Y}_{\min} \end{bmatrix} \quad (2.37)$$

sendo

$$I_L = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix} \quad (2.38)$$

e $\mathcal{Y}_{\min}$ e $\mathcal{Y}_{\max}$ representam vetores de restrições de saídas ao longo do horizonte de comprimento p :

$$\mathcal{Y}_{\min} = \begin{bmatrix} y_{\min}(1) \\ y_{\min}(2) \\ \vdots \\ y_{\min}(p) \end{bmatrix} ; \quad \mathcal{Y}_{\max} = \begin{bmatrix} y_{\max}(1) \\ y_{\max}(2) \\ \vdots \\ y_{\max}(p) \end{bmatrix} \quad (2.39)$$

Por fim, substituindo a equação de predição na função objetivo, conforme a Equação 2.40.

$$\min_{\Delta \mathcal{U}(k)} \quad \frac{1}{2} \Delta \mathcal{U}(k)^T H^u \Delta \mathcal{U}(k) - \mathcal{G}(k+1|k)^T \Delta \mathcal{U}(k) \quad (2.40)$$

$$\text{s.t.} \quad C^u \Delta \mathcal{U} \leq C(k+1|k) \quad (2.41)$$

em que a Matriz Hessiana é dada por

$$H^u = 2\mathcal{J}^{uT} Q_y^T Q_y \mathcal{J}^u + Q_u^T Q_u \quad (2.42)$$

e o vetor gradiente:

$$\mathcal{G}(k+1 | k) = 2\mathcal{J}^{uT} Q_y^T Q_y E_p(k+1 | k). \quad (2.43)$$

A implementação do QDMC é realizada por meio da estratégia do horizonte móvel. Isso significa que a função objetivo é resolvida a cada ciclo de execução do controlador, permitindo que o algoritmo de controle seja ajustado conforme necessário para lidar com situações inesperadas. Além disso, essa abordagem possibilita a modificação dos parâmetros de peso na função objetivo, adaptando o controle de maneira mais eficaz às condições do sistema.

2.5 Controlador Lógico Programável

O Controlador Lógico Programável (CLP) é definido de maneiras ligeiramente diferentes por diversas associações e normas técnicas. Segundo a norma IEC 61131-1, o CLP é descrito como “um sistema eletrônico operando digitalmente, projetado para uso em um ambiente industrial”. De acordo com a National Electrical Manufacturers Association (NEMA), trata-se de um “aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções para implementações específicas, como lógica, sequenciamento, temporização, contagem e aritmética, controlando diversos tipos de máquinas e processos por meio de módulos de entrada e saída”. Já a Associação Brasileira de Normas Técnicas (ABNT) define o CLP como um “equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais”.

Resumidamente, o CLP é composto por uma parte física e uma eletrônica digital, com memória capaz de armazenar instruções definidas pelo usuário e automatizar de forma eficiente e confiável ações em processos industriais que envolvem controle sequencial, sincronização de processos e elementos auxiliares. Ademais, possuem um ambiente de desenvolvimento para a escrita de programas e suportam protocolos de comunicação industrial, possibilitando supervisão e aquisição de dados. Por isso, são amplamente utilizados na instrumentação e em sistemas automatizados nas indústrias químicas, de manufatura e de processos (ALPHONSUS; ABDULLAH, 2016; ALVES; MORRIS, 2018; CAMACHO; ALBA, 2013).

Originalmente, os CLPs foram desenvolvidos para substituir os sistemas de controle sequencial baseados em relés, que utilizavam dispositivos elétricos e interruptores para controlar processos. Esses sistemas antigos exigiam uma fiação fixa, o que tornava as mudanças nos processos complexas e onerosas. Além disso, apresentavam alta propensão a falhas e elevados custos de manutenção e substituição (GREEFF; GHOSHAL, 2004).

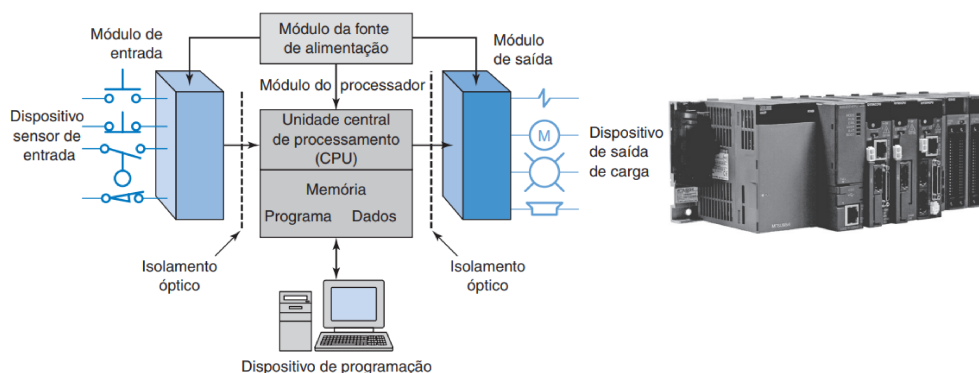
Diante dessas limitações, esses controladores foram projetados com alguns requisitos importantes: ser resistente à ambientes agressivos, como fábricas com umidade, poeira, vibração e ruído elétrico; permitir modificações flexíveis nos circuitos; utilizar uma linguagem de programação familiar, como *Ladder Diagram* (LD), que é amplamente conhecida por eletricitistas e técnicos; e permitir que a fiação de campo fosse conectada diretamente aos terminais de entrada/saída do controlador. Ao longo do tempo, os CLPs evoluíram, permitindo o controle de processos mais complexos e de maior velocidade, mantendo-se compactos e eficientes

(ALPHONSUS; ABDULLAH, 2016).

2.5.1 Composição do Hardware do Controlador Lógico Programável

Em relação ao seu *hardware*, um CLP é composto basicamente por uma Unidade de Processamento Central (CPU), uma fonte de alimentação, dispositivos de programação e os módulos de entradas e saídas (E/S), que podem ser analógicas, digitais e especiais. Em CLPs compactos, todos esses componentes são frequentemente integrados em um único chassi, enquanto sistemas maiores adotam uma arquitetura modular, com possibilidade de expansão conforme necessário, como mostrado na Figura 3 (PETRUZELLA, 2011).

Figura 3 – Partes de um controlador programável modular - *Mitsubishi Automation*



Fonte: Petruzella (2011)

A CPU é responsável por interpretar, coordenar e executar todas as ações do sistema de controle programável. Este módulo normalmente é composto por um microprocessador, uma memória programável para armazenamento do programa personalizado pelo usuário e uma memória temporária para guardar os dados usados durante a execução (ALPHONSUS; ABDULLAH, 2016). Dessa forma, a CPU faz a comunicação entre as entradas e saídas, tanto internas quanto externas, oferecendo interfaces para programação, monitoramento e controle.

De modo geral, para execução de um programa, o microprocessador segue um ciclo contínuo chamado "ciclo de varredura" que consiste em três etapas: leitura das entradas físicas, execução do programa lógico e atualização das saídas físicas (ALVES; MORRIS, 2018). Cada ciclo de varredura ocorre em intervalos de amostragem, caracterizando um controle discreto. Uma característica essencial de um CLP é a capacidade de realizar esse ciclo em um tempo de resposta reduzido, garantindo agilidade no controle do processo (TASCA; FREITAS; WAGNER, 2020).

O sistema de entradas e saídas (E/S) atua como a interface de conexão entre os dispositivos de campo e o CLP, ajustando os diferentes sinais que chegam desses dispositivos ou que são enviados para eles. Entre os dispositivos de entrada, encontram-se os sensores e botões de comando, que se conectam aos terminais de entrada. Já os dispositivos de saída, podem ser

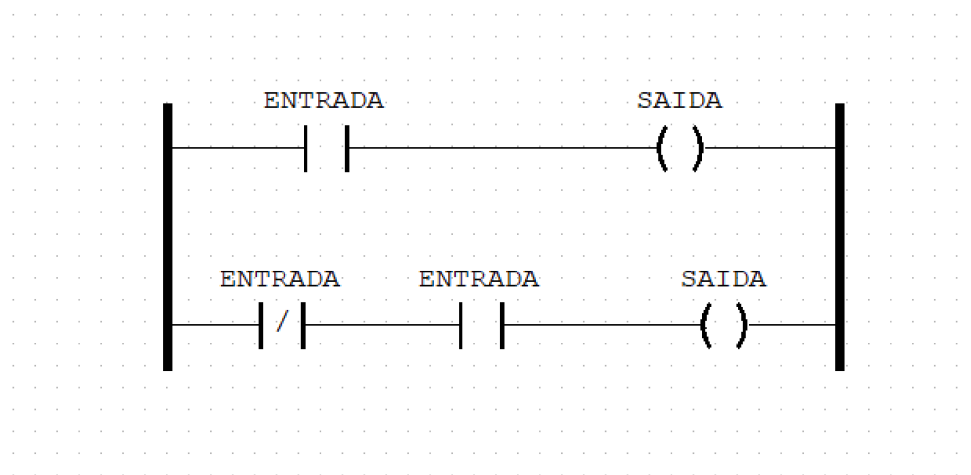
pequenos motores, válvulas solenoides etc, e são ligados aos terminais de saída (PETRUZELLA, 2011).

Por fim, o dispositivo de programação tem como objetivo inserir o programa na memória do processador. Quando conectados ao CLP, esses dispositivos possibilitam tanto a programação quanto o monitoramento do sistema. Além disso, eles são essenciais para verificar e solucionar defeitos nos equipamentos e modificar programas existentes. O dispositivo de programação mais amplamente utilizado é o computador pessoal (PC), devido à sua versatilidade e facilidade de uso (PETRUZELLA, 2011).

2.5.2 Programação dos Controladores Lógicos Programáveis e o uso da Linguagem Ladder

A primeira linguagem de programação adotada para os Controladores Lógicos Programáveis foi a linguagem em forma de diagrama *Ladder*, devido a sua popularização entre os profissionais da automação e seu uso nos antigos sistemas relés. Sendo assim, ela foi escolhida para atenuar a transição entre esses dois sistemas. Nessa linguagem gráfica, as entradas são representadas por contatos elétricos, as saídas por bobinas de relés, e as conexões lógicas por fios, se assemelhando aos circuitos de comutação dos antigos esquemas de sistemas de relés, como é apresentado na Figura 4.

Figura 4 – Diagrama *Ladder*



Fonte: Autoria própria (2024)

Apesar da Linguagem *Ladder* ser a primeira utilizada e tenha se tornado a mais comum para CLPs, ela não é a única. Com a crescente variedade de equipamentos e a necessidade de maior flexibilidade na automação industrial, foi criado um padrão internacional pela *International Electrotechnical Commission* (IEC) estabelecido na norma IEC 61131, para controladores lógicos programáveis. Esta norma aborda a documentação, instalação, comunicação e define cinco tipos de linguagens de programação dos CLPs.

A terceira parte da norma aborda sobre essas linguagens, que podem ser gráficas ou textuais, e são elas: Texto Estruturado (*Structured Text* - ST), Lista de Instruções (*Instruction List*

- IL), Linguagem *Ladder* (*Ladder* - LD), Diagrama de Bloco de Função (*Function Block Diagram* - FDB) e Diagrama de Fluxo e Sequenciamento Gráfico de Funções (*Sequential Flow Chart* - SFC). Essas diferentes opções foram desenvolvidas para atender à diversidade de sistemas e necessidades de automação, proporcionando flexibilidade na programação de CLPs.

2.5.3 Sistemas de Controle SCADA

Após a criação dos primeiros Controladores Lógicos Programáveis, houve a necessidade de estabelecer comunicação com sistemas de Entrada/Saída (E/S) remotos e, posteriormente, com outros CLPs. Essa comunicação permitiu que os CLPs controlassem dispositivos à distância e transmitissem dados ao sistema de Controle de Supervisão e Aquisição de Dados (SCADA).

O SCADA é um sistema de gerenciamento, monitoramento e registro de alarmes e dados, automatizando o controle de processos interligados aos CLPs. Além disso, pode coletar dados importantes sobre a planta de processo, o que contribuiu para a minimização do tempo de inatividade, aumento da produtividade e melhoria da qualidade segurança e precisão dos registros (GREEFF; GHOSH, 2004). Dessa forma, o *software* do sistema SCADA, armazena informações no histórico de dados e as exibe em uma Interface Homem-Máquina (*Human Machine Interface* - HMI), que permite o monitoramento centralizado das informações digitais de *status* e controle do sistema (UPADHYAY; SAMPALLI, 2020).

2.5.4 Protocolo MODBUS: Comunicação entre Controladores Lógicos Programáveis

O *Modbus* foi um dos primeiros protocolos de comunicação entre CLPs, derivado da arquitetura *Master/Slave* e introduzido pela fabricante de equipamentos *Modicon* (atual *Schneider Electric*). Esse protocolo rapidamente se popularizou e se tornou um padrão industrial amplamente utilizado em sistemas SCADA devido à sua disponibilidade aberta e à ausência de taxas de licenciamento para implementação. Além disso, sua capacidade de operar em diversos meios de comunicação, como linhas telefônicas, *Ethernet*, rádio e satélite, o torna ideal para grandes sistemas SCADA (ALVES; MORRIS, 2018).

O protocolo *Modbus* é definido como *Master/Slave*, no qual um dispositivo que opera como *Master* consulta um ou mais dispositivos operando como escravos (*Slaves*). A comunicação de dados é sempre iniciada pelo *Master*, que tem o controle integral do fluxo de dados, iniciando todas as interações e determinando a sequência das operações. Já os dispositivos *Slaves* não respondem voluntariamente, apenas quando solicitados, garantindo uma comunicação ordenada e previsível.

Entre as variantes do *Modbus*, as mais comumente utilizadas são *Modbus RTU* e *Modbus TCP* (Protocolo de Controle de Transmissão). O primeiro tipo é usado para transferir dados através de interfaces seriais com fio, enquanto o segundo é projetado para uso em redes *Ethernet* e na *Internet*. Independente da variante, a configuração inclui um *Modbus Master*, que é tipicamente um *software* rodando em um computador e que atua como host para acessar os *Modbus Slaves*

(ELAMANOV et al., 2024).

2.6 Microcontroladores como alternativa para aprendizado sobre CLPs

Controladores Lógicos Programáveis fazem parte de um mercado moderno de controladores analógicos configurados digitalmente, caracterizado por sua alta tecnologia e alto custo. Além disso, existem taxas de licenciamento associadas aos programas de simulação desses controladores, o que podem representar limitações de uso. Como alternativa mais acessível para aprender sobre controle e automação de forma simples e econômica, os microcontroladores têm se mostrado uma opção viável (KVALSUND; WINKLER, 2023).

Um microcontrolador é um sistema integrado e microprocessado voltado para aplicações de controle, contendo diversos periféricos internos no mesmo circuito. Isso permite que o dispositivo interaja com o ambiente e controle atuadores, incluindo, além da unidade central de processamento, periféricos como memórias, dispositivos PWM e conversores analógico-digitais (PAPARELLA, 2021). Ao contrário de um microprocessador, um microcontrolador possui praticamente tudo o que é necessário para operar de forma independente, sendo suficiente conectar suas interfaces ao meio e programá-lo conforme desejado (SILVA; STEVAN, 2015).

Atualmente, há uma ampla variedade de tipos e marcas de microcontroladores disponíveis, e um exemplo popular e acessível é o Arduino. Este microcontrolador tem se tornado uma tendência de ferramenta para desenvolver diferentes aplicações nas áreas de automação e engenharia (MÁRQUEZ-VERA et al., 2023).

2.6.1 O uso do Arduino como um Microcontrolador

O Arduino é uma plataforma eletrônica de código aberto de placa única que oferece *software* e *hardware* de uso simplificado. Suas placas possuem entradas analógicas e digitais, possibilitando a leitura de diversos tipos de entradas, como sensores e botões, bem como a execução de saídas, como a ativação de motores e o acendimento de *LEDs*. Os comandos são enviados à placa por meio rotinas programadas a partir do *software* ARDUINO IDE (ARDUINO, 2024b).

Originalmente, essa plataforma surgiu como uma ferramenta simples para permitir prototipagem rápida, especialmente voltada para estudantes sem experiência em programação e eletrônica. Porém, seu público-alvo foi se diversificando e sendo cada mais abrangente, envolvendo criadores, programadores, artistas e *designers*. Essa abrangência se justifica por diversos fatores, dentre eles, seu baixo custo, *software* e *hardware* extensíveis e de código aberto e um ambiente de programação simples e claro (ARDUINO, 2024b).

À medida que ganhou popularidade, o Arduino evoluiu expandindo sua gama de produtos, desde placas básicas de 8 *bits* até soluções avançadas para *Internet* das coisas (*IoT*), *wearables*, impressão 3D, etc. (ARDUINO, 2024b). Atualmente, existem várias versões distintas de placas Arduino, que se diferenciam principalmente pelo número de canais de entrada e saída, pelo

formato, memória e pelo tipo de microprocessador utilizado.

Por apresentar *hardware* livre, possibilitou que fornecedores desenvolvessem placas periféricas, conhecidas como *shields*, para atender a suas características padronizadas. O uso dessas placas nos Arduinos diminui problemas de conexão relacionados a maus contatos ou erros, o que facilitou ainda mais a elaboração de projetos e aplicações da plataforma.

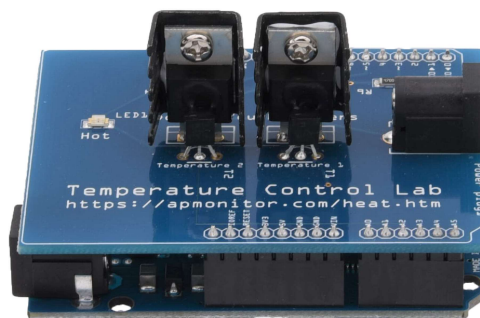
O *hardware* do Arduino se divide nas famílias Clássica, Nano, MKR e Mega. Entre os mais populares estão o UNO, Leonardo, Mega 2560, amplamente utilizados no desenvolvimento de plantas didáticas para implantação de estratégias de controle. Entre as diversas aplicações estão situações práticas com o propósito de ensino de controle, experimentos em laboratório e ferramentas de pesquisa.

Para este trabalho foi escolhido a placa Arduino UNO integrado com o microcontrolador ATMEL ATMEGA328P, uma vez que trata-se de uma das plataformas mais conhecidas que permite ligação com um computador. Esse microcontrolador é baseado na arquitetura aprimorada do processador RISC da AVR (*Alf-Veegard-RISC*) e possui uma memória de programa *Flash* de 32 kb, superior em comparação a outros microcontroladores AVR com a mesma quantidade de pinos.

2.7 Projeto TCLab: uma pequena planta com grande potencial para estudo de controle

O projeto *Temperature Control Lab* (TCLab) para desenvolvimento de controle *feedback*, desenvolvido com um Arduino Leonardo, exemplifica a aplicação prática e de baixo custo do Arduino envolvendo controle e automação. O TCLab é composto basicamente por dois transistores funcionando como elementos aquecedores e dois sensores de temperatura acoplados na placa Arduino, Figura 5. Seu funcionamento envolve o envio de um sinal de controle para um ou dois transistores que aquecem até o *setpoint* da temperatura desejada. A energia térmica é transferida para os sensores acoplados, que medem as temperaturas. Esses dados são então registrados no computador ao qual a placa Arduino está conectada (HEDENGREN, 2019).

Figura 5 – Dispositivo TCLab



Fonte: Hedengren (2019)

O projeto ainda permite a programação utilizando diferentes linguagens e ambientes de programação, como *Python*, *MATLAB* e *Simulink* e disponibiliza *scripts* para experimentos

de controle, identificação e modelagem da planta, além de aplicações em ciência de dados e modelos de controle, entre outros (FONSECA, 2022).

Diversos trabalhos utilizaram o TCLab como uma ferramenta eficaz de estudo e pesquisa nas áreas de controle e dinâmica de processos. O artigo de Park et al. (2020) explora o uso do TCLab como um *benchmark* experimental para análise comparativa de métodos de modelagem e controle, como a otimização de controladores PID e controle preditivo baseado em modelo (MPC), usando diferentes abordagens de modelagem. No estudo, os algoritmos de controle e a interface foram desenvolvidos utilizando os *softwares* MATLAB, Simulink e a linguagem de programação em Python, obtendo resultados que destacam o potencial da plataforma como *benchmark* de *hardware* para modelagem transitória e métodos de controle clássico e avançado.

Complementando esse enfoque, Oliveira, Hedengren e Rossiter (2020) demonstraram como o TCLab pode proporcionar uma experiência de aprendizado acessível e prática para estudantes de graduação da Universidade de Trás-os-Montes e Alto Douro (UTAD), em Portugal, da Universidade Brigham Young, nos Estados Unidos, e a Universidade de Sheffield, no Reino Unido. Através de experimentos didáticos envolvendo os modos de controle *on-off* e PID, o dispositivo contribuiu para que os alunos compreendessem conceitos complexos de controle digital em um contexto de aplicação realista e portátil.

De maneira semelhante, Paparella (2021) adaptou um módulo com Arduino, baseado no princípio do TCLab, além de utilizar os *softwares open-source* OpenPLC e ScadaBR, para a simulação de configuração de um Controlador Lógico Programável. Com isso, apresentou uma proposta mais próxima da realidade industrial, ao incluir a configuração do Arduino como um CPL em programação Ladder no estudo.

Este estudo foi inspirado no dispositivo experimental proposto pela APMonitor para o *Temperature Control Lab* (TCLab), conforme descrito no trabalho de Park et al. (2020) para implementação dos controles PID e MPC. Além disso, também incorporou a abordagem de Paparella (2021), ao incluir a configuração do dispositivo como um Controlador Lógico Programável.

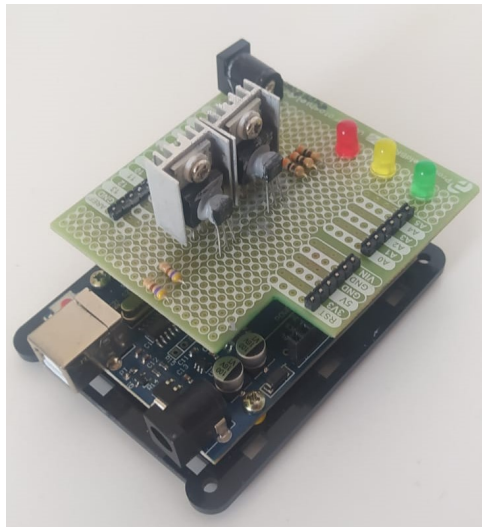
3 MATERIAIS E MÉTODOS

Com base no modelo do TCLab, o protótipo deste trabalho foi desenvolvido com o objetivo de simular uma planta de pequena escala, em que a saída de calor foi produzido dois transistores e transferido para os sensores de temperatura, representando um sistema *Multiple-Input and Multiple-Output* (MIMO). Para tanto, foi utilizado um Arduino Uno operando como CLP com o uso da plataforma *OpenPLC*. A coleta de dados, monitoramento do sistema e a implementação do controle MPC foram realizadas por meio da linguagem de programação em *Python*.

3.1 Descrição do aparato experimental

O dispositivo experimental utilizado é composto por uma placa *Protoshield*, que foi conectada a uma placa Arduino Uno. Na placa, foram conectados dois transistores como elementos aquecedores, dois sensores de temperatura de circuito integrado e três LEDs conforme mostrado na Figura 6.

Figura 6 – Aparato Experimental

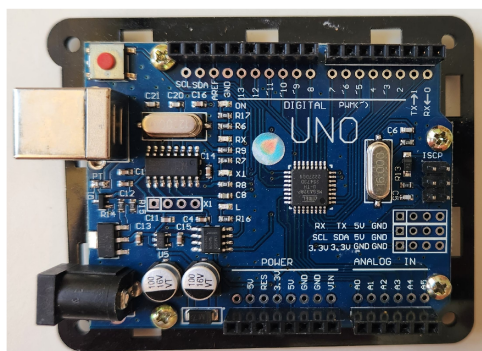


Fonte: Autoria própria (2024)

3.1.1 Arduino UNO

O *hardware* do Arduino Uno, Figura 7, conta com um microcontrolador ATMEL AT-MEGA328P. A placa inclui 6 canais analógicos, ligados a um Conversor Analógico para Digital (ADC) de 10 *bits* utilizados para medir a voltagem dos sensores de temperatura em 1024 níveis discretos (2^{10}). Possui também 14 pinos, que podem ser usados como entradas/saídas digitais, dos quais alguns podem ser usados como saídas de Modulação por Largura de Pulso (PWM) de 8 *bits* com 256 níveis (2^8).

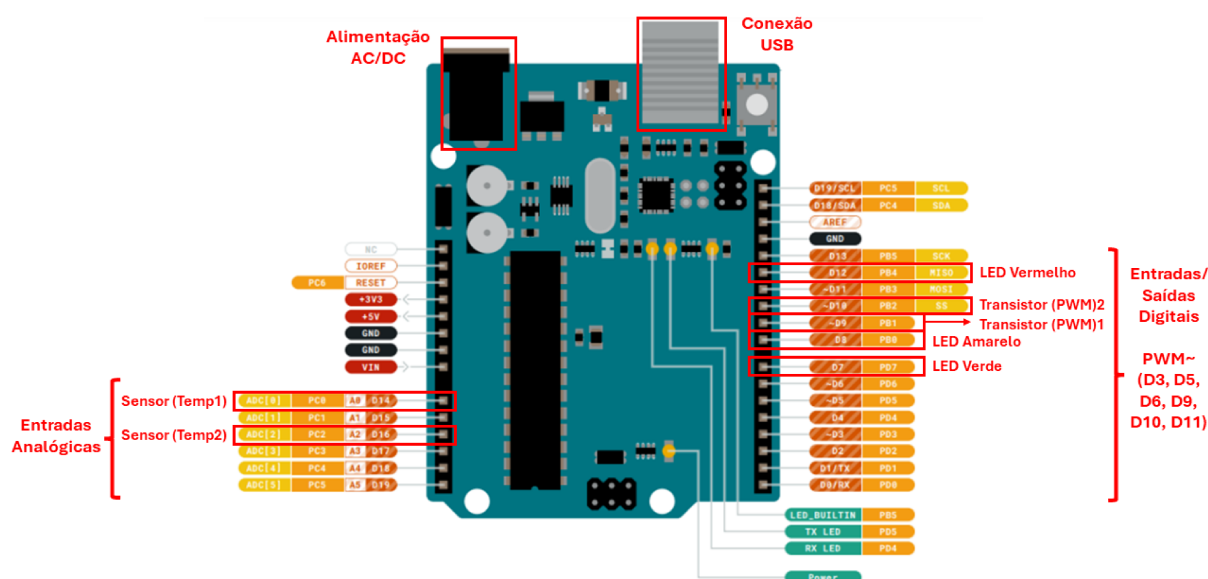
Figura 7 – Arduino UNO



Fonte: Autoria própria (2024)

Os principais componentes e pinos do Arduino Uno são representados na Figura 8. Os pinos analógicos, identificados como A0 a A5, recebem sinais analógicos na faixa de 0 a 5V, e convertem esses sinais em valores digitais. Esses valores digitais podem então ser processados por um computador através da conexão USB.

Figura 8 – Representação componentes e pinos Arduino UNO



Fonte: Adaptado de Arduino (2024a)

Os 14 pinos digitais, identificados como D0 a D13, podem atuar como entradas ou saídas digitais. Esses pinos podem ser utilizados tanto para controlar o estado de um circuito digital, quanto para receber entradas digitais de dispositivos externos. Desses 14 pinos digitais, 6 pinos de saídas digitais (D3, D5, D6, D9, D10 e D11), são disponibilizados para serem configurados como saídas PWM, ou seja, Modulação por Largura de Pulso, que simulam um sinal analógico. São nominados assim pois produzem um sinal analógico em formato de onda quadrada, variando

de 0 a 5V, com a frequência modulada. Para alimentar um circuito externo, existem os pinos de 5V, que oferecem 5V de energia, e o pino 3.3V. Além desses, existem três pinos de aterramento, nomeados como GND.

Neste trabalho, foram utilizados dois pinos analógicos: o Sensor de Temperatura 1 foi conectado ao pino A0 e o Sensor de Temperatura 2 conectado ao pino A2. Em relação aos pinos digitais, 3 foram usados como saídas digitais, para acionamento dos LEDs: o pino D7 foi usado para ligar o LED Verde, o D8 para ligar o LED amarelo e o D12 para o LED Vermelho. Já os transistores 1 e 2 foram conectados aos pinos D9 e D10, respectivamente, em que o sinal PWM é enviado e o calor gerado depende do valor transmitido, sendo o aquecimento proporcional a intensidade do sinal enviado ao transistor. Na Tabela 1 é apresentada a relação de entradas e saídas do Arduino UNO utilizadas para a montagem do dispositivo.

Tabela 1 – Identificação de Pinos de Entradas e Saídas no Arduino UNO

Pino	Tipo	Descrição
A0	Analog Input (AI)	Sensor de Temperatura 1
A2	Analog Input (AI)	Sensor de Temperatura 2
D9	Analog Output (AO)	Transistor (PWM) 1
D10	Analog Output (AO)	Transistor (PWM) 2
D7	Digital Output (DO)	LED Verde
D8	Digital Output (DO)	LED Amarelo
D12	Digital Output (DO)	LED Vermelho
5V	Power	Fornecimento de 5V energia
GND	Power	Aterramento

Fonte: Autoria Própria (2024)

3.1.2 Componentes conectados ao Arduino

O aquecimento gerado pelo circuito foi proporcionado a partir de dois transistores bipolares NPN TIP31, que atuam como elementos aquecedores. Esses aquecedores transistorizados foram alimentados por uma fonte de alimentação de 5V/2A.

Para melhor dissipação do calor durante o aquecimento do transistor, adaptou-se um dissipador de calor com as dimensões adequadas para o transistor.

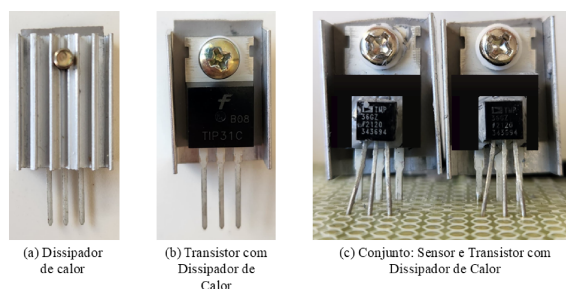
Dois sensores de temperatura termistores TMP36GZ foram usados para capturar e realizar a leitura de temperatura do aparato. De acordo com a sua folha de dados ou *Datasheet*, este sensor não necessita de calibração e tem uma tensão de saída (mV) que é linearmente proporcional à temperatura medida de acordo com a Equação 3.1. A precisão típica do sensor é de $\pm 1^\circ\text{C}$ em temperatura ambiente (25°C) e $\pm 2^\circ\text{C}$ na faixa de operação de -40°C a 150°C (ANALOGDEVICES, 2002).

$$T(^{\circ}\text{C}) = 0,1V(\text{mV}) - 50 \quad (3.1)$$

em que T representa a temperatura em $^{\circ}\text{C}$ e V a tensão de saída em mV .

Para manter em contato o sensor de temperatura e o transistor foi usado uma pasta térmica Prata GD900. O conjunto sensor, transistor e dissipador de calor pode ser visualizado na Figura 9.

Figura 9 – Conjunto: Sensor e Transistor com Dissipador de Calor



Fonte: Autoria própria (2024)

A sinalização de eventos e alarmes foi realizada por meio de 3 LEDs, um verde, um amarelo e um vermelho. O LED Verde efoi utilizado para indicação de funcionamento da placa, o LED Amarelo como alarme de temperatura alta, acima de 50°C e o LED Vermelho como alarme de temperatura muito alta, acima de 80°C.

3.2 Modelo matemático do sistema a partir de equações de balanço de energia

De acordo com Park et al. (2020), para o TCLab, a resposta do sistema às variações nos elementos aquecedores é bem representada por um modelo de parâmetros concentrados com convecção, condução e radiação térmica. Esse modelo é uma simplificação, em que a temperatura é descrita de forma concentrada, desconsiderando a distribuição de temperatura ao longo das coordenadas espaciais do dissipador. Neste trabalho, utilizou-se o modelo matemático com as mesmas hipóteses apresentadas em Park et al. (2020) para testes de simulação do Controle Preditivo baseado em Modelo (MPC).

Para tal modelo fenomenológico, assumiu-se que tanto os aquecedores (T_{H1} e T_{H2}) quanto os sensores de temperatura (T_{C1} e T_{C2}) têm uma temperatura uniforme. Além disso, os sensores de temperatura (T_{C1} e T_{C2}) possuem massa e área superficial pequenas e as mudanças de temperatura são descritas pela condução de calor proveniente dos aquecedores.

Com base nas equações de balanço de energia apresentadas nas Equações 3.2 a 3.5, obteve-se a relação entre as variações das temperaturas dos elementos aquecedores e dos sensores e a temperatura detectada por cada componente do sistema.

$$\frac{dT_{H1}(t)}{dt} = \frac{1}{mc_p}(Q_{C\infty 1} + Q_{R\infty 1} + Q_{C12} + Q_{R12} + \alpha_1 Q_1) \quad (3.2)$$

$$\frac{dT_{H2}(t)}{dt} = \frac{1}{mc_p}(Q_{\infty 2} + Q_{R\infty 2} - Q_{C12} - Q_{R12} + \alpha_2 Q_2) \quad (3.3)$$

$$\frac{dT_{C1}(t)}{dt} = \frac{1}{\tau_c}(T_{H1} - T_{C1}) \quad (3.4)$$

$$\frac{dT_{C2}(t)}{dt} = \frac{1}{\tau_c}(T_{H2} - T_{C2}) \quad (3.5)$$

em que, pela Lei do Resfriamento de Newton, a taxa de transferência de calor entre os elementos aquecedores e o ambiente é dada pelas Equações 3.6 e 3.7. A taxa de transferência de calor por convecção do aquecedor 1 para o aquecedor 2, é dada pela Equação 3.8. Enquanto as Equações 3.9 e 3.10 correspondem a transferência de calor por radiação entre os elementos aquecedores e o ambiente e a Equação 3.11 entre os dois aquecedores 1 e 2.

$$Q_{C\infty 1} = UA(T_{\infty} - T_{H1}) \quad (3.6)$$

$$Q_{C\infty 2} = UA(T_{\infty} - T_{H2}) \quad (3.7)$$

$$Q_{C12} = U_S A_S (T_{H2} - T_{H1}) \quad (3.8)$$

$$Q_{R\infty 1} = \varepsilon \sigma A (T_{\infty}^4 - T_{H1}^4) \quad (3.9)$$

$$Q_{R\infty 2} = \varepsilon \sigma A (T_{\infty}^4 - T_{H2}^4) \quad (3.10)$$

$$Q_{R12} = \varepsilon \sigma A (T_{H2}^4 - T_{H1}^4) \quad (3.11)$$

A resposta da temperatura em relação ao tempo dos dois sensores de temperatura ocorre principalmente por convecção, uma vez que a massa e a área superficial dos sensores são pequenas quando comparadas aos aquecedores. A variável τ_c das Equações 3.4 e 3.5 é proveniente da versão discreta da Lei de Fick, conforme a Equação 3.12.

$$\tau_c = m_s c_{ps} \frac{\Delta x}{k_c A_{cond}} \quad (3.12)$$

em que m_s corresponde a massa do sensor, c_{ps} a capacidade térmica do sensor, k_c a condutividade térmica da pasta térmica e x a largura da pasta térmica. Tais parâmetros foram combinados ao único parâmetro τ_c e cujo o valor foi estimado a partir de dados coletados do dispositivo.

Os parâmetros do modelo foram adotados com base nos valores apresentados em Park et al. (2020) e estão dispostos na Tabela 2. Para a estimativa dos parâmetros U , U_s , α_1 , α_2 e τ_c foi realizado um teste com diversas alterações em degrau para os dois aquecedores e utilizando o pacote *Gekko* (BEAL et al., 2018) do *Python*.

Tabela 2 – Parâmetros Do Modelo Matemático

Parâmetros	Valor
Temperatura Inicial (T_0)	22°C
Temperatura Ambiente (T_∞)	22°C
Calor específico (c_p)	500 J/(kg K)
Área Superficial entre os Aquecedores (A_s)	$2 \times 10^{-4} m^2$
Área Superficial desconsiderando A_s (A)	$1 \times 10^{-3} m^2$
Massa (m)	0,004 kg
Emissividade (ϵ)	0,9
Constante de <i>Stefan Boltzmann</i> (σ)	$5,67 \times 10^{-8} W/(m^2 K^4)$

Fonte: Autoria Própria (2024)

3.3 Configuração do dispositivo utilizando a plataforma OpenPLC

Neste trabalho, a programação do CLP foi feita a partir do *software OpenPLC*, compatível com a norma IEC 61131-3. Na sua maior parte, foi desenvolvido programas em linguagem *Ladder* (LD) e blocos de função (FDB), com alguns usos da linguagem texto estruturado (ST) para implementação de filtros.

A linguagem *Ladder* é uma das mais disseminadas e utilizadas pela indústria para a programação de Controladores Lógicos Programáveis (ROQUE, 2014). Esse fato somado a boa representação gráfica e a facilidade de se obter familiaridade com o ambiente, fundamentaram a escolha por essa linguagem. Para a implementação de filtros, o texto estruturado foi utilizado devido à sua flexibilidade com operadores matemáticos e funções. E os blocos de funções complementaram a utilização da programação com auxílio de funções pré-definidas existentes no *software OpenPLC*.

3.3.1 Aplicação da plataforma OpenPLC

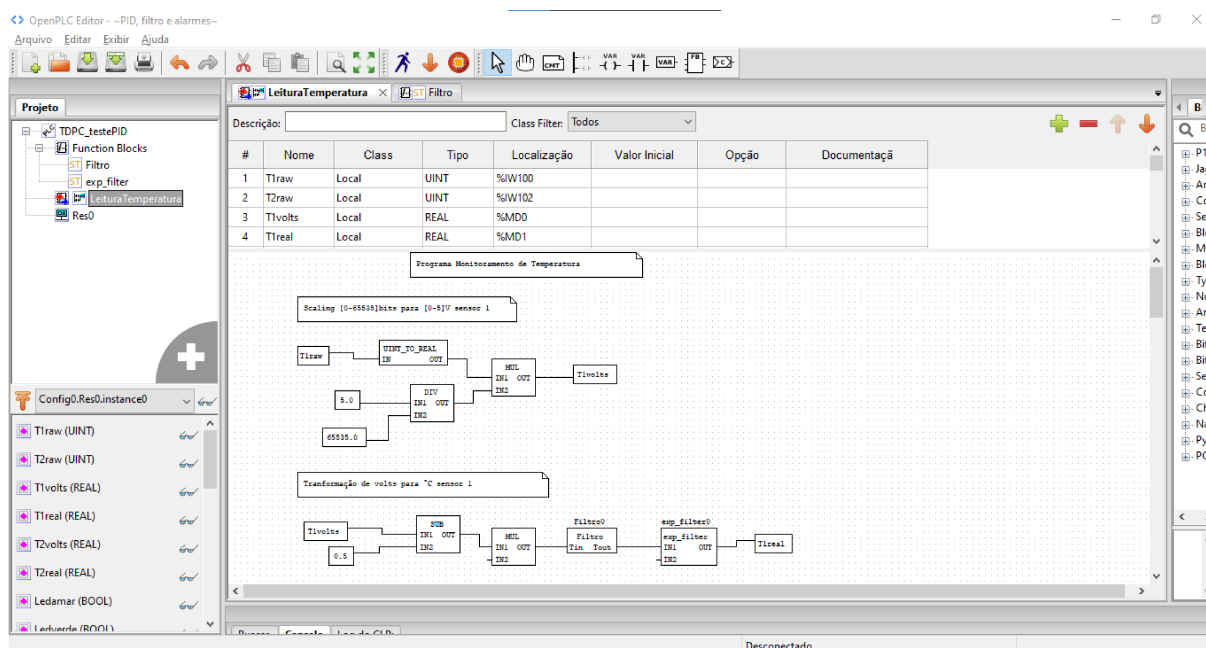
A programação do CLP e a comunicação de acordo com o protocolo *Modbus/TCP* foram feitas através do *software OpenPLC*, em que o Arduino foi usado como dispositivo escravo (*slave device*) por meio da comunicação *Modbus*. O uso de *OpenPLC* se justifica por ser um software de código aberto, permitindo uma implementação de baixo custo para a planta de controle de temperatura.

O *OpenPLC* é um *software* para desenvolvimento de programas, que pode ser executado

em plataformas como computadores com *Linux* e *Windows*. Além disso, é compatível com a norma IEC 61131-3, permitindo a programação de computadores e controladores como CLPs conforme essa norma. O *softPLC* ou *OpenPLC* oferece suporte às plataformas como *Arduino*, *Raspberry Pi*, *UniPi Industrial Platform* e permite a utilização de dispositivos escravos. Ele usa apenas o protocolo de comunicação *Modbus/TCP* para comunicação e oferece uma interface de usuário intuitiva e fácil de usar. Com isso, oferece a possibilidade de ler e escrever dados no CLP escravo por meio da comunicação *Modbus*, além de permitir a integração com Sistemas de Supervisão e Aquisição de Dados (SCADA). O projeto se divide em duas partes essenciais: o *OpenPLC Runtime* e o *OpenPLC Editor*.

O *OpenPLC Editor* é a plataforma de desenvolvimento usada para criar, compilar e fazer *upload* dos programas para o *OpenPLC Runtime*. Ele inclui uma interface gráfica que facilita a criação de projetos de CLP com várias Unidades de Organização de Programa (POUs). Ele ainda conta com um módulo responsável por compilar as linguagens gráficas (*Ladder*, Diagrama de Bloco de Função e Sequenciamento Gráfico de Funções) em código baseado em Texto Estruturado (ST). Para esse projeto, o desenvolvimento dos programas em *Ladder* foram feitos diretamente no *Editor*, conforme é ilustrado na Figura 10.

Figura 10 – Interface do *software* Open PLC Editor



Fonte: Autoria Própria (2024)

No contexto de sistemas de controle de processos, após o desenvolvimento dos programas, o *OpenPLC* pode ser utilizado de duas formas principais: embarcando o código diretamente no microcontrolador, de modo que ele funcione como um controlador programável; ou gerando um arquivo que é executado no *Open PLC Runtime*, que por sua vez é instalado em um computador. Nesta última configuração, o computador assume o papel de CLP, enquanto o microcontrolador

atua como interface de entradas e saídas. Neste trabalho, foi aplicado somente a última configuração, de forma que os arquivos gerados no *OpenPLC Editor* foram posteriormente compilados e usados no *OpenPLC Runtime*, responsável pela sua execução.

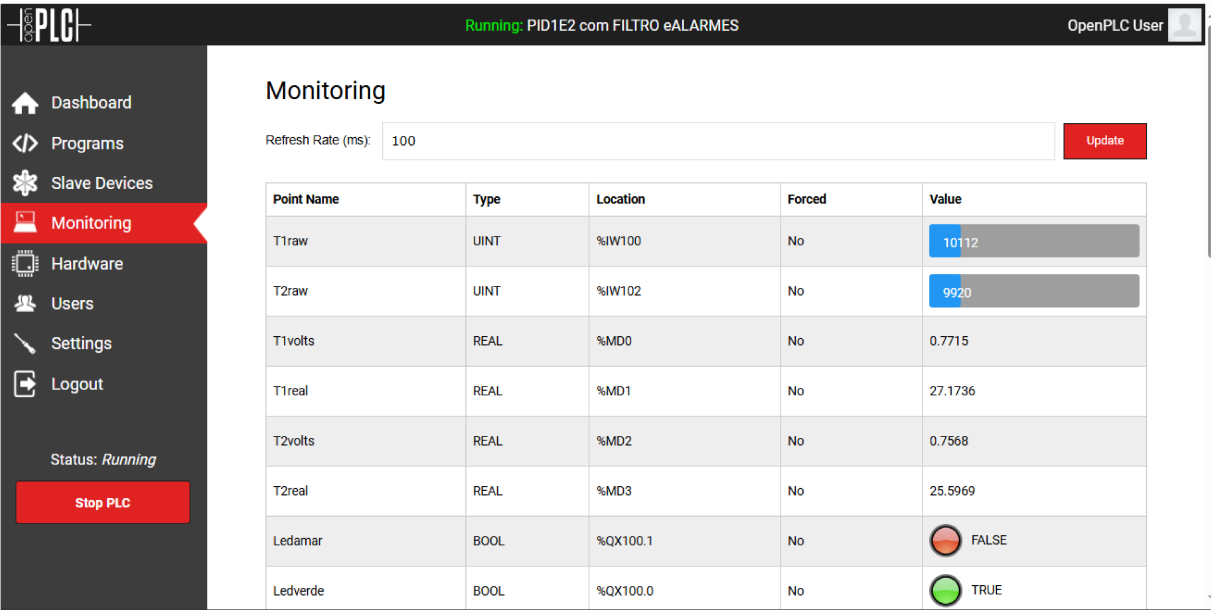
Dessa forma, para executar os arquivos gerados pelo *OpenPLC Editor*, foi instalado e acessado o servidor *web* que permite configurar e monitorar o *OpenPLC Runtime*. O *OpenPLC Runtime* foi instalado e configurado em um *notebook desktop*, em que o Arduino foi conectado como dispositivo escravo *Modbus*. As configurações para adicionar o Arduino como um dispositivo escravo foram feitas conforme a documentação disponibilizada no site do projeto (ALVES, 2017), através da página *Slave Device*, Figura 11. O *upload* dos arquivos baseados em texto estruturado, com extensão '.st' foram feitos através da aba *Programs*. Após isso, o sistema foi ativado através do botão *Start PLC*. A seção de Monitoramento do *Runtime* é mostrada na Figura 12, que permite visualizar os valores das variáveis do programa PLC.

Figura 11 – Configuração por meio da aba 'Slave Devices' no *OpenPLC Runtime*

The screenshot displays the 'Edit slave device' interface in the OpenPLC Runtime web application. The left sidebar contains navigation links: Dashboard, Programs, Slave Devices (highlighted), Monitoring, Hardware, Users, Settings, and Logout. The main content area is divided into two columns. The left column contains configuration fields for the slave device: Device Name (text input: 'Arduino UNO TDPC'), Device Type (dropdown: 'Arduino Uno'), Slave ID (text input: '0'), COM Port (dropdown: 'COM10'), Baud Rate (text input: '115200'), Parity (dropdown: 'None'), Data Bits (text input: '8'), Stop Bits (text input: '1'), and Transmission Pause (text input: '0'). The right column contains sections for Modbus registers: Discrete Inputs (%IX100.0), Coils (%QX100.0), Input Registers (%IW100), Holding Registers - Read (%IW100), and Holding Registers - Write (%QW100). Each section has 'Start Address' and 'Size' input fields. At the bottom right, there are 'Save device' and 'Delete device' buttons. The bottom status bar shows 'Status: Running' and a 'Stop PLC' button.

Fonte: Autoria Própria (2024)

Figura 12 – Interface da aba Monitoramento do *software OpenPLC Runtime*

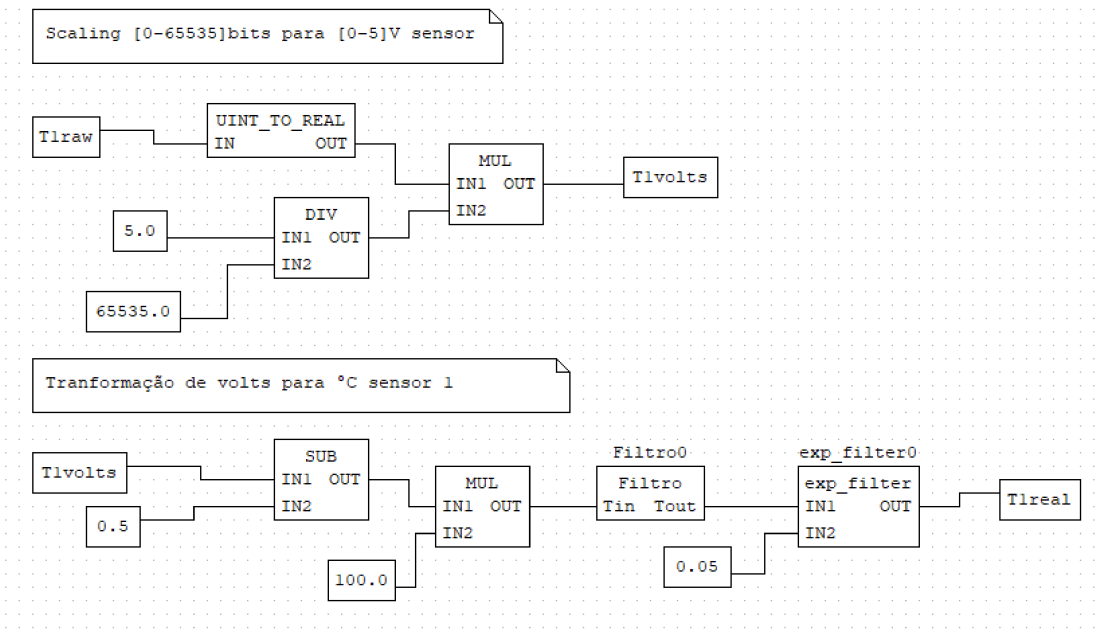


Fonte: Autoria Própria (2024)

3.3.2 Programação do controlador em Ladder

Para o monitoramento do dispositivo de controle de temperatura, os sinais dos sensores foram convertidos de valores inteiros para valores reais com o significado físico de interesse, a Temperatura na escala Celsius (°C). Assim, para fazer o *scaling* das variáveis, utilizaram-se blocos de função disponibilizados no *software*, conforme demonstrado na Figura 13.

Figura 13 – *Scaling* das variáveis de saída dos sensores de temperatura



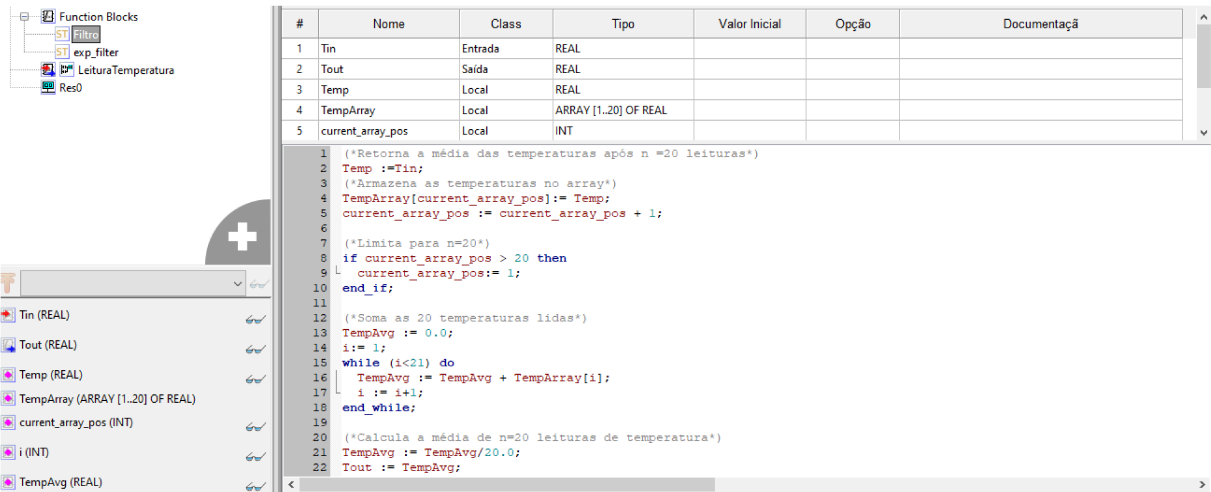
Fonte: Autoria Própria (2024)

De acordo com a Figura 13, primeiramente foi inserido um bloco para converter a variável do tipo inteira para uma do tipo real. Essa nova variável foi usada em sequência para um bloco de multiplicação que realizou a operação pelo coeficiente de conversão de *bits* para *Volts*. Para finalizar o processo de conversão, a então variável de temperatura em *Volts* foi inserida em um bloco de subtração e depois em um bloco de multiplicação, de acordo com a Equação 3.1. Esse procedimento foi desenvolvido para cada um dos dois sensores.

Para essa última transformação de *Volts* para °C, foram usados dois filtros digitais, um filtro de média móvel e um filtro exponencial de primeira ordem (também denominado de filtro de média móvel exponencialmente ponderada, *Exponentially Weighted Moving Average* - EWMA), com o objetivo de filtrar o sinal proveniente dos transmissores e tentar reduzir os efeitos de ruídos de medição.

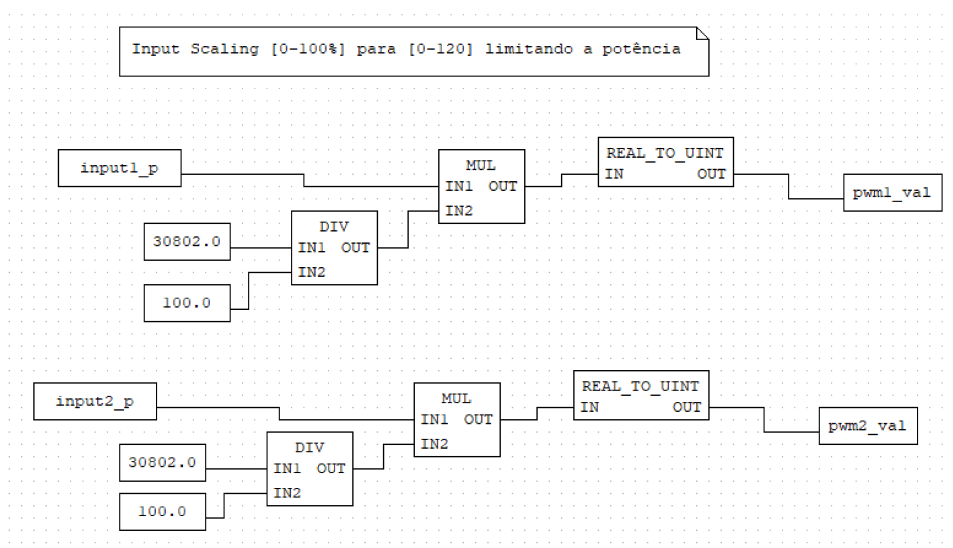
Neste caso, é crucial levar em conta ruídos que são classificados como sinais não correlacionados de média zero (ruído branco) ao projetar a malha de controle. Isso ajuda a evitar que as ações de controle sejam comprometidas por sinais de alta frequência, os quais geram demandas desnecessárias no atuador, pois esses ruídos não podem ser controlados (COSTA, 2014). A implementação desses filtros foi feita utilizando a linguagem de texto estruturado, conforme mostrado na Figura 14, pela facilidade de compreensão e por permitir o uso de operadores matemáticos.

Figura 14 – Filtro de média móvel



Fonte: Autoria Própria (2024)

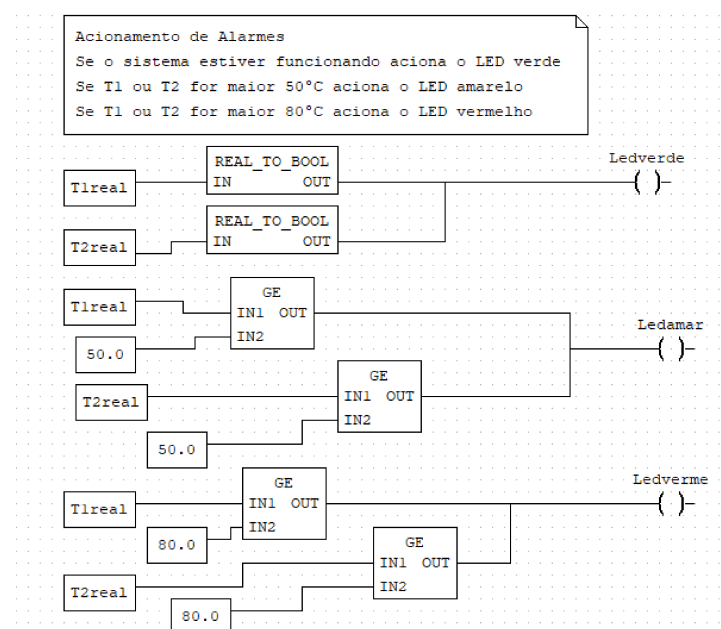
De forma semelhante ao *scaling* feito para os sensores de temperatura, também foi realizado um *unscaling* para as saídas PWM do Arduino, conforme a Figura 15 de modo a converter a variável de entrada dos transistores do tipo real (em percentual 0-100%) para uma do tipo inteira, limitando a sua potência.

Figura 15 – *Scaling* das variáveis de entrada dos transistores

Fonte: Autoria Própria (2024)

Para o acionamento dos LEDs, como alarmes de temperaturas, foi realizada a conversão das variáveis reais para variáveis booleanas, como mostrado na Figura 16. Para os LEDs amarelo e vermelho, foi inserido um bloco para comparação das temperaturas em relação a referência do acionamento do alarme, cuja suas entradas são variáveis reais, e sua saída uma variável booleana.

Figura 16 – Acionamento dos LEDs

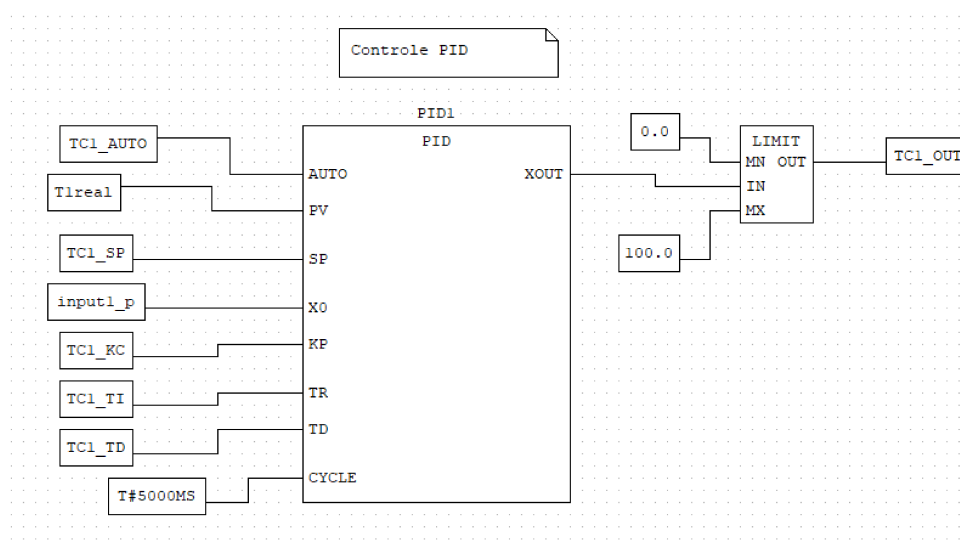


Fonte: Autoria Própria (2024)

As malhas de controle de temperatura foram implementadas a partir de blocos de função PID disponíveis no *Open PLC Editor*. Assim, para a configuração de cada controlador, utilizou-se um bloco PID (Figura 17), em que os parâmetros de sintonia foram inseridos e as variáveis

correspondentes a cada estrutura do bloco foram ajustadas. É importante ressaltar que este bloco tem ação reversa, uma vez que o sinal de erro é calculado de acordo com a Equação 2.4. Dessa forma, foram adicionados ao bloco valores negativos de ganho, para implementação da ação direta. Como existem duas malhas de temperatura, dois blocos PID diferentes foram utilizados: um para a malha de temperatura T_1 e outro para a malha de temperatura T_2 .

Figura 17 – Bloco PID *OpenPLC*



Fonte: Autoria Própria (2024)

3.3.3 Configuração de entradas e saídas analógicas do controlador

A interação do CLP com o sistema foi feita a partir de entradas e saídas digitais e analógicas. As variáveis conectadas aos módulos de entradas/saídas e comunicação são associadas a um endereço de CLP. O *OpenPLC* adota a nomenclatura IEC 61131-3 para identificar locais de entrada, saída e memória. O endereçamento desses locais é realizado por meio de sequências de caracteres específicas, em que o mapeamento entre o PLC e entradas/saídas físicas varia conforme o dispositivo utilizado.

De maneira geral, as sequências são formadas por um sinal de porcentagem %, um prefixo de local (sendo I para entrada, Q para saída e M para memória), um prefixo de tamanho (X para *bit*, B para *byte*, W para *palavra*, D para *palavra dupla* e L para *palavra longa*) e um ou mais números naturais. Para o Arduino UNO, foi realizado conforme disponibilizado na documentação do *OpenPLC* (ARDUINO, 2024b). Todas as variáveis utilizadas na programação do CLP, assim como seu endereçamento, podem ser visualizadas na Tabela 3.

Tabela 3 – Relação e Endereçamento de Variáveis de Entrada, Saída e Memórias

Nome	Tipo de variável	Endereço	Descrição
T1raw	UINT	%IW100	Sinal Temperatura 1 em bits
T2raw	UINT	%IW102	Sinal Temperatura 2 em bits
T1volts	REAL	%MD0	Sinal Temperatura 1 em Volts
T2volts	REAL	%MD2	Sinal Temperatura 2 em Volts
T1real	REAL	%MD1	Sinal Temperatura 1 em °C
T2real	REAL	%MD3	Sinal Temperatura 2 em °C
Ledverde	BOOL	%QX100.0	LED Verde
Ledamar	BOOL	%QX100.1	LED Amarelo
Ledverme	BOOL	%QX100.2	LED Vermelho
Input1_p	REAL	%MD4	Transistor - PWM1 em percentagem
Input2_p	REAL	%MD5	Transistor - PWM2 em percentagem
PWM1_val	UINT	%QW100	Transistor - PWM1 em potência
PWM2_val	UINT	%QW101	Transistor - PWM2 em potência
TC1_AUTO	BOOL	%QX100.5	Acionamento do PID da T1
TC1_SP	REAL	%MD6	Setpoint para Temperatura 1
TC1_KC	REAL	%MD7	K_c do PID da Temperatura 1
TC1_TI	REAL	%MD8	τ_i do PID para T1
TC1_TD	REAL	%MD9	τ_d do PID para T1
TC1_OUT	REAL	%MD10	Temperatura de Saída do PID1
TC2_AUTO	BOOL	%QX100.6	Acionamento do PID da T2
TC2_SP	REAL	%MD11	Setpoint para Temperatura 2
TC2_KC	REAL	%MD12	K_c do PID para T2
TC2_TI	REAL	%MD13	τ_i do PID para T2
TC2_TD	REAL	%MD14	τ_d do PID para T2
TC2_OUT	REAL	%MD15	Temperatura de Saída do PID2

Fonte: Autoria Própria (2024)

3.3.4 Seleção do tempo de amostragem para o sistema em malha aberta e em malha fechada

A estrutura de controle e aquisição de dados foi realizada de forma discreta e os cálculos e ações de controle foram realizados em intervalos de amostragem. Dessa forma, os sinais contínuos provenientes da planta são discretizados e processados. Levando isso em consideração, a rotina computacional operou de forma cíclica e repetitiva enquanto a comunicação *Modbus* estava ativa e o sistema funcionando.

Na seleção do período de amostragem para a implementação do controle, foi determinada de modo a garantir que a taxa fosse suficientemente representativa do comportamento dinâmico do sistema, evitando a perda de informações sobre a dinâmica do sistema. De acordo com Seborg et al. 2010, o período de amostragem ideal deve ser de 10 a 20 vezes menor que a constante de tempo dominante do processo, garantindo uma resposta precisa e capturando a dinâmica essencial do sistema. No entanto, um período de amostragem muito curto pode resultar em superamostragem, o que aumenta o custo computacional, especialmente no caso do controlador QDMC, onde as matrizes geradas pelo problema de otimização se tornariam excessivamente

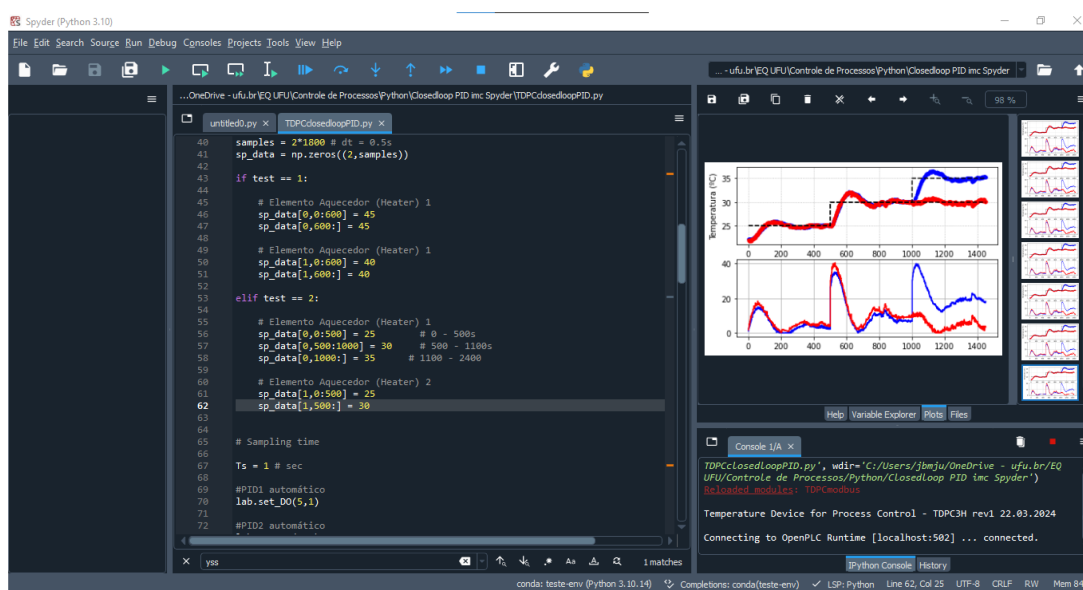
grandes e complexas.

Dessa forma, foi selecionado um período de amostragem de 1 segundo para o sistema em malha aberta e de 5 segundos para os testes em malha fechada com ambos os controladores, balanceando a necessidade de representar a dinâmica do sistema com a eficiência computacional. Esse intervalo está dentro das recomendações de Seborg et al. 2010, e garante que o sistema não seja superamostrado, evitando sobrecargas desnecessárias no processamento e mantendo o controle eficaz.

3.4 A aplicação da linguagem de programação Python

A supervisão, monitoramento e armazenamento dos dados do sistema, assim como a implementação de rotinas do controle avançado foram feitas através de uma comunicação serial. Os programas usando a linguagem *Python* foram desenvolvidos no *Spyder*, possibilitando gerar gráficos das temperaturas e sinais da variável manipulada simultaneamente a ocorrência do processo (Figura 18), além de registrar os dados em um arquivo do tipo '.csv' e implementar funcionalidades relacionadas à interface de supervisão e controle, com base na programação orientada a objetos.

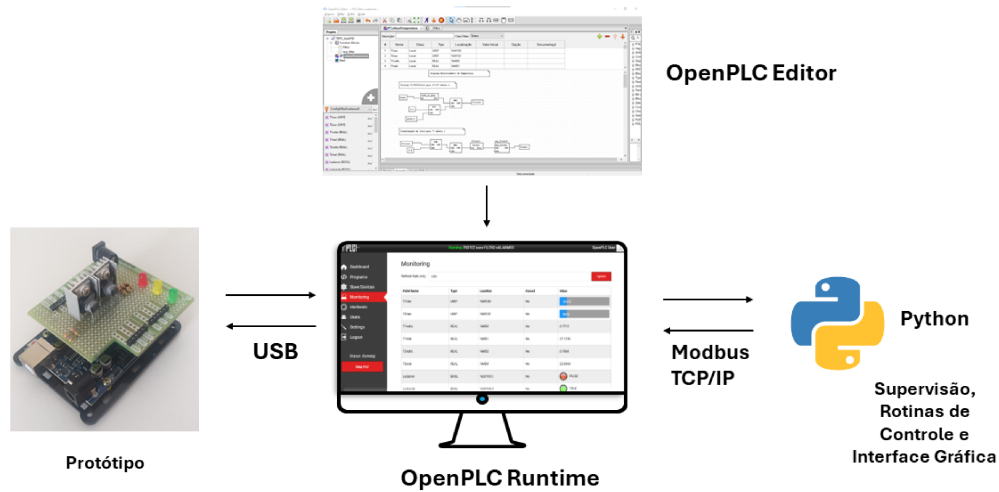
Figura 18 – Interface gráfica da Tela de Monitoramento do Sistema do programa em *Python*



Fonte: Autoria Própria (2024)

Para tanto, o envio de informações ao sistema foram feitos por meio de comunicação *Modbus TCP/IP* utilizando uma biblioteca chamada *PyModbus*, adaptada para este trabalho de acordo com o endereçamento *Modbus*, conforme a documentação do *OpenPLC* (ALVES, 2017). O servidor por sua vez se comunicou com o *OpenPLC Runtime* em um computador, conectado via *USB* ao dispositivo experimental, conforme o esquema apresentado na Figura 19.

Figura 19 – Esquema comunicação serial entre ambientes de desenvolvimento e aplicação



Fonte: Autoria Própria (2024)

3.5 Sistema em malha aberta

De acordo com Seborg et al. (2010), como mencionado na Seção 2.2, o segundo passo para o desenvolvimento do sistema de controle consiste na elaboração do modelo dinâmico do processo. Neste trabalho, para a implementação do controle PID adotou-se o modelo de processo simplificado, baseado em funções de transferência desenvolvidas por dados experimentais por meio de testes em malha aberta.

Tendo em vista que tal sistema possui duas variáveis controladas (Temperatura 1 e Temperatura 2) e duas variáveis manipuladas (Aquecedor 1 e Aquecedor 2), ele é caracterizado como sendo multivariável (MIMO) e sua representação matricial é indicada em 3.13.

$$\begin{bmatrix} T_1(s) \\ T_2(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \quad (3.13)$$

sendo $G_{11}(s)$ a função de transferência em malha aberta que representa a relação entre a Temperatura 1 (T_1) e o Aquecedor 1 (U_1); $G_{12}(s)$ a função de transferência em malha aberta que representa a relação entre a Temperatura 1 (T_1) e o Aquecedor 2 (U_2); $G_{21}(s)$ a função de transferência em malha aberta que representa a relação entre a Temperatura 2 (T_2) e o Aquecedor 1 (U_1) e $G_{22}(s)$ a função de transferência em malha aberta que representa a relação entre a Temperatura 2 (T_2) e o Aquecedor 2 (U_2).

Levando isso em consideração, foram realizados dois testes diferentes aplicando uma alteração degrau com o sistema em malha aberta para avaliar a influência do aquecimento de cada aquecedor tem sob as temperaturas dos sensores para a determinação das duas malhas de controle, com uma amostragem configurada para ocorrer a cada 1 segundo. No primeiro teste, o valor do Aquecedor 2 foi mantida constante ao longo de todo o período ($U_2 = 0\%$), enquanto uma variação em degrau de 20% foi aplicada no Aquecedor 1 (U_1). Isso permitiu avaliar o efeito isolado do calor gerado pelo elemento aquecedor 1 exerce sobre as variáveis de temperatura. De

maneira semelhante, um segundo teste foi realizado, o valor do elemento aquecedor 1 ($U_1 = 0\%$) constante, mas aplicando uma variação em degrau de 20% no Aquecedor 2 (U_2), para análise do efeito isolado o aquecimento do Aquecedor 2 nas temperaturas.

A partir do recorte de dados referentes ao degrau de 0% e 20%, obteve-se os parâmetros necessários para a determinação das funções de transferência. Elas foram obtidas levando em consideração que o sistema pode ser aproximado de um sistema de 1ª ordem com tempo morto (*First Order Plus Dead Time* - FOPDT) a partir do site PIDTuner (2024), conforme a Equação 3.14 definida no domínio de Laplace.

$$G(s) = \frac{K_p}{\tau_p s + 1} e^{-\theta s} \quad (3.14)$$

3.6 Implementação das malhas de controle

Quanto aos métodos de controle adotados neste trabalho, o Controle PID foi implementado utilizando a linguagem de programação *Ladder* diretamente no ambiente *OpenPLC Editor*. Portanto, neste trabalho assume-se uma configuração semelhante ao ambiente industrial, no qual o PID integra a Camada Base de Controle de Processos na Pirâmide de Automação, conforme descrito por Niu e Xiao (2022).

O Controle Preditivo Baseado em Modelo (MPC), por sua vez, foi desenvolvido e executado utilizando a linguagem *Python*, aproveitando a flexibilidade e a capacidade avançada de processamento dessa linguagem para lidar com a complexidade do modelo preditivo. Por isso, o MPC é executado no Computador de Processos, fora do controlador, o que é típico desse tipo de controle, que se posiciona na Camada de Controle Avançado (APC), acima da Camada Base na Hierarquia de controle.

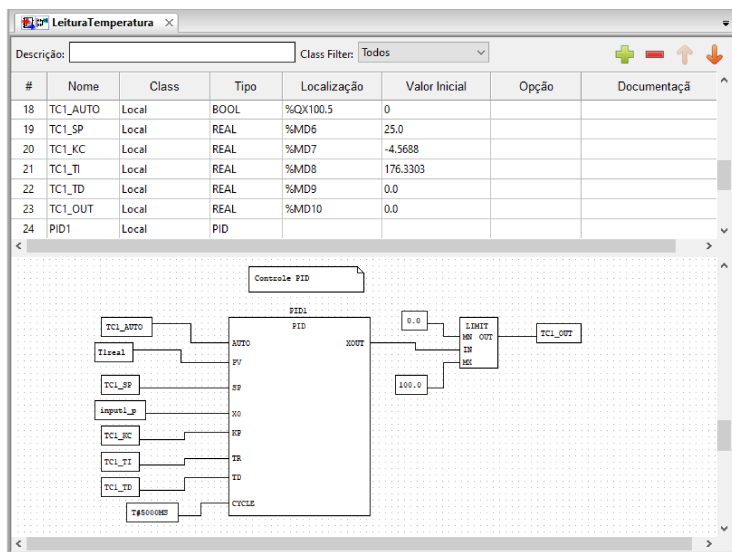
A etapa de sintonia dos controladores consiste na determinação dos parâmetros adequados para a faixa de estabilidade do processo em malha fechada. Idealmente, além disso, o sistema de controle deveria minimizar os efeitos de perturbações; permitir a transição rápida e suave a mudanças de *setpoint*. Além disso, evitar ação de controle excessiva e ser um sistema de controle robusto. Porém não é possível obter todos esses critérios simultaneamente, por isso deve-se buscar um equilíbrio entre robustez e desempenho. Robustez se relaciona com um controle satisfatório em uma ampla faixa de condições de processos, mesmo diante de certa imprecisão do modelo. Enquanto desempenho envolve a capacidade de reação forma rápida com pouca ou nenhuma oscilação (SEBORG et al., 2010).

Por fim, foram utilizados critérios de desempenho de sintonia IAE (*Integral of the Absolute Error*), ISE (*Integral of the Squared Error*) e ITAE (*Integral of Time Weighted Absolute Error*) para valiação da performance de cada método de sintonia.

3.6.1 PID

A implementação do controle clássico PID foi feita utilizando o bloco de função disponível no *OpenPLC Editor*. Como esse bloco não possui filtro derivativo, o controlador selecionado foi o proporcional integral (PI). Dessa forma, os valores dos parâmetros K_c e τ_i , assim como os limites de saturação de saída entre 0 e 100, foram adicionados no programa em *Ladder*, como mostrado na Figura 20. Já as alterações dos *setpoints* e acionamento dos PIDs foram realizados através da linguagem *Python*.

Figura 20 – Controle PID *Openplc Editor*



Fonte: Autoria Própria (2024)

A sintonia do controlador PI foi feita utilizando dois métodos muito conhecidos: *Internal Model Control* (IMC) - desenvolvido por Morari et al. (1986) - e *Integral of the time-weighted absolute error* (ITAE) - desenvolvido por Smith e Corripio (1997). Dessa forma, dispondo das funções de transferência em malha aberta, foram aplicados os métodos IMC e ITAE (servo).

3.6.1.1 Método de Sintonia IMC

A sintonia do controlador pelo Método IMC foi determinada conforme as relações descritas por Seborg et al. (2010), considerando que o modelo da função de transferência em malha aberta se assemelha a um sistema de 1ª ordem com tempo morto. Neste método, é escolhido o quão rápido (f) se deseja que o sistema em malha fechada seja em comparação ao sistema em malha aberta, através da definição da constante de tempo em malha fechada (τ_c), conforme apresentado na Equação 3.15. As demais variáveis presentes nas Equações 3.16 e 3.17 para um controlador PI foram obtidas diretamente das funções de transferência, conforme a Equação 3.14.

$$\tau_c = \frac{\tau_p}{f} \quad (3.15)$$

$$K_c = \frac{1}{K_p} \frac{\tau_p}{\tau_c + \theta} \quad (3.16)$$

$$\tau_i = \tau_p \quad (3.17)$$

3.6.1.2 Método de Sintonia ITAE

O método de sintonia ITAE (*Integral of Time Weighted Absolute Error*), é uma técnica que visa minimizar o valor da integral do erro absoluto ponderado pelo tempo. Sua aplicação envolve dois cenários distintos: quando envolve o isolamento dos efeitos do rastreamento de *setpoint* (controle servo), e quando deseja-se a análise efeitos relacionados à rejeição de perturbações (controle regulador) (SEBORG et al., 2010). Neste trabalho, utilizou-se apenas o controle do tipo servo.

Para a determinação dos parâmetros do controlador PI, (K_c e τ_i) utilizando o método ITAE Servo, foram usadas as Equações 3.18 e 3.19 e constantes dispostas em Sebor et al. (2010). As demais variáveis foram extraídas das funções de transferência em malha aberta.

Para o método proporcional: $A = 0,5867$ e $B = -0,916$

$$K_c = \frac{1}{K_p} A \left(\frac{\theta}{\tau_p} \right)^B \quad (3.18)$$

Para o método integral: $A = 1,030$ e $B = -0,165$

$$\tau_i = \frac{\tau_p}{A + B \left(\frac{\theta}{\tau_p} \right)} \quad (3.19)$$

3.6.2 MPC

A primeira etapa determinante no desenvolvimento do MPC é a obtenção do modelo do sistema, que foi utilizado para obter as previsões sobre o comportamento do futuro da planta. Utilizando as equações de balanço propostas por Park et. al. (2020) e os parâmetros físicos, foi feito um teste com alterações em degrau para o elemento aquecedor 1 e o elemento aquecedor 2 para a estimativa dos parâmetros U , U_s , α_1 , α_2 e τ_c e validação do modelo.

Posteriormente, foi realizada a sintonia do MPC, a partir de simulações utilizando o modelo não linear conforme discutido no capítulo 3.2. A sintonia foi feita através do ajuste dos parâmetros: o horizonte de predição (P), o horizonte de controle (H_c) e os pesos Q_u e Q_y . O horizonte de predição (H_p) determina em quantos instantes de tempo futuros a resposta da variável controlada será predita pelo modelo da planta. Esse horizonte é crucial para a antecipação das futuras necessidades do sistema e permite um planejamento adequado das ações de controle.

De acordo com Camacho e Alba (2013), é aconselhável selecionar um horizonte de predição que seja compatível com o tempo de acomodação do sistema em malha aberta.

O horizonte de controle (H_c) refere-se ao número de ações de controle que serão aplicadas para garantir que a variável atinja a referência dentro do horizonte de predição. Esse parâmetro deve satisfazer a condição $1 \leq H_c \leq H_p$. À medida que o horizonte de controle aumenta, a carga computacional do controlador também se eleva. Por essa razão, valores menores de H_p tendem a tornar o controlador menos agressivo, resultando em uma resposta mais lenta e menor sensibilidade a ruídos. Além disso, a diferença $H_p - H_c$ deve ser suficientemente ampla para garantir que as últimas variações no controle se reflitam dentro do horizonte de predição.

Dessa forma, a função objetivo é baseada em previsões de saída ao longo de um horizonte de previsão de H_p passos de tempo é minimizada por uma seleção de movimentos de variáveis manipuladas ao longo de um horizonte de controle de H_c movimentos de controle (MORARI; LEE; GARCÍA, 2002; CAMACHO; ALBA, 2013).

As matrizes de pesos Q_u e Q_y afetam a função objetivo do MPC. O peso Q_y está relacionado à penalização do desvio das saídas em comparação a suas trajetórias de *setpoints*. Valores maiores de Q_y proporcionam uma resposta mais rápida, mas podem levar a maior instabilidade. Já o peso Q_u está vinculado ao custo da taxa de variação das entradas. Assim, valores maiores de Q_u resultam em uma resposta mais suave, minimizando as oscilações, mas podem tornar o sistema mais lento para reagir a mudanças. Com base nisso, o ajuste adequado desses parâmetros e pesos é fundamental para garantir um equilíbrio entre a rapidez da resposta e a estabilidade do sistema (CAMACHO; ALBA, 2013).

Para o controle QDMC, foram fornecidos os parâmetros de sintonia definidos como entradas de uma biblioteca implementada para esse trabalho, responsável por calcular a saída dos controladores a cada instante de tempo, como ilustrado na Figura 21. Para a matriz S , os coeficientes de resposta ao degrau foram obtidos experimentalmente, através do primeiro Teste em malha aberta.

Figura 21 – Parâmetros sintonia QDMC *Python*

```

# DMC configuration
ny = 2
nu = 2

w = np.zeros((ny,1))
y = np.zeros((ny,1))
uk = np.zeros((nu,1))

data1 = np.loadtxt("lab_data202514.csv",delimiter=";",skiprows=1)
data2 = np.loadtxt("lab_data193602.csv",delimiter=";",skiprows=1)

S11 = data1[100:800:5,1]-23
S21 = data1[100:800:5,2]-23

S12 = data2[100:800:5,1]-23
S22 = data2[100:800:5,2]-22

S = np.array([[S11/20,S12/20],
              [S21/20,S22/20]])

yss = np.array([22,22])
uss = np.array([1,1])

Hp = 60
Hc = 10

Qy = np.diag([1,1])
Qu = np.diag([1,1])

dumax = np.array([+10,+10])
dumin = np.array([-10,-10])

umax = np.array([100,100]) - uss
umin = np.array([0,0]) - uss

dmc = DMC_controller(S,Hp,Hc,Qy,Qu,dumax,dumin,umax,umin)

```

Fonte: Autoria Própria (2024)

3.6.3 Métricas de Desempenho dos Controladores

Os critérios de desempenho são indicadores importantes para avaliação de algoritmos de controle. Entre os mais utilizados, destacam-se as integrais do erro: IAE (*Integral of Absolute Error*), ISE (*Integral of Squared Error*), e ITAE (*Integral of Time-weighted Absolute Error*). Essas métricas foram usadas para avaliar a qualidade da resposta do sistema utilizando as diferentes técnicas de sintonia do controle PID e o controle MPC.

O IAE representa a integral do valor absoluto do erro ao longo do tempo, conforme a Equação 3.20. Essa métrica dá igual importância a desvios grandes e pequenos, sem considerar a duração dos erros.

$$\text{IAE} = \int_0^{\infty} |e(t)| dt \approx \sum_{i=0}^n |e_k| \Delta t \quad (3.20)$$

O ISE, por sua vez, está relacionado a integral do erro elevado ao quadrado ao longo do tempo, conforme a Equação 3.21. Diferentemente do IAE, o ISE atribui maior penalização a erros maiores, uma vez que o erro é elevado ao quadrado.

$$\text{ISE} = \int_0^{\infty} e^2(t) dt \approx \sum_{i=0}^n e_k^2 \Delta t \quad (3.21)$$

Por fim, o ITAE pondera o erro absoluto pelo tempo, como mostrado na Equação 3.22. Esse critério penaliza mais fortemente os erros que persistem por um longo período, enfatizando a importância de uma rápida correção do sistema.

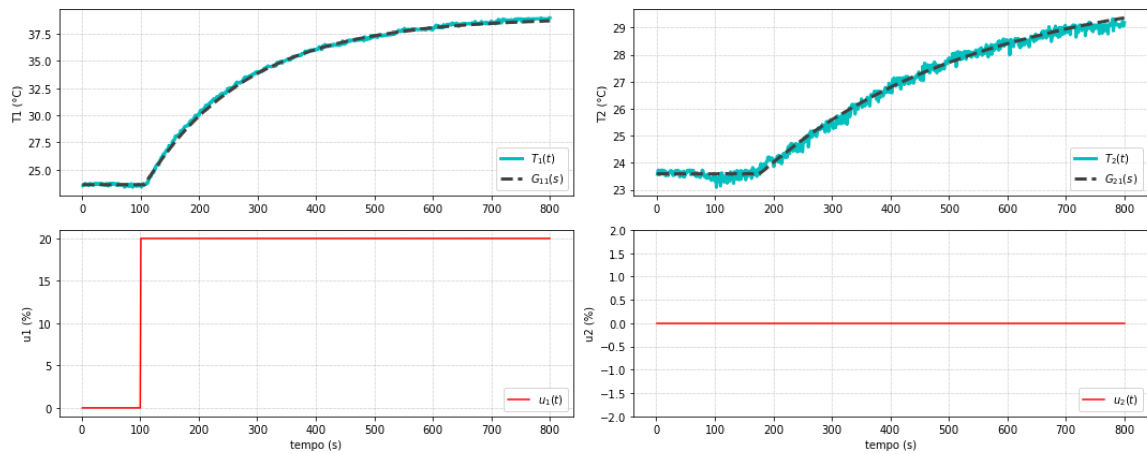
$$\text{ITAE} = \int_0^{\infty} t|e(t)|dt \approx \sum_{i=0}^n t_k|e_k|\Delta t \quad (3.22)$$

4 RESULTADOS E DISCUSSÕES

4.1 Sistema em malha aberta

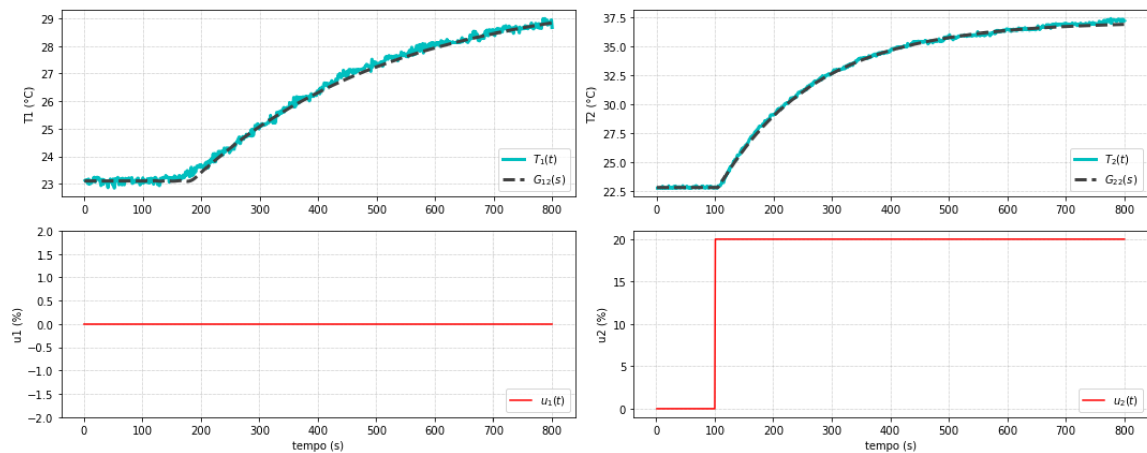
Os resultados experimentais obtidos ao aplicar uma variação em degrau na porcentagem do elemento aquecedor 1, juntamente com as respostas obtidas pelas respectivas funções de transferência são apresentados na Figura 22. Enquanto na Figura 23 são mostrados os resultados da alteração no valor do elemento aquecedor 2, incluindo a simulação ajustada pelas funções de transferência correspondentes.

Figura 22 – Curva das Temperaturas 1 e 2 em resposta ao degrau da porcentagem do Aquecedor 1 - $G_{11}(s)$ e $G_{21}(s)$



Fonte: Autoria Própria (2024)

Figura 23 – Curva das Temperaturas 1 e 2 em resposta ao degrau da porcentagem do Aquecedor 2 - $G_{21}(s)$ e $G_{22}(s)$



Fonte: Autoria Própria (2024)

Levando em consideração a aproximação do sistema para um de 1ª ordem com tempo morto, FOPDT, as funções de transferência no domínio de Laplace obtidas utilizando o aplicativo PIDTuner, são dispostas nas Equações 4.1 a 4.4.

$$G_{11}(s) = \frac{T_1(s)}{U_1(s)} = \frac{K_p}{\tau_p s + 1} e^{-\theta s} = \frac{0,77}{176,33s + 1} e^{-6,82s} \quad (4.1)$$

$$G_{21}(s) = \frac{T_2(s)}{U_1(s)} = \frac{K_p}{\tau_p s + 1} e^{-\theta s} = \frac{0,36}{373,90s + 1} e^{-76,49s} \quad (4.2)$$

$$G_{12}(s) = \frac{T_1(s)}{U_2(s)} = \frac{K_p}{\tau_p s + 1} e^{-\theta s} = \frac{0,34}{342,71s + 1} e^{-84,27s} \quad (4.3)$$

$$G_{22}(s) = \frac{T_2(s)}{U_2(s)} = \frac{K_p}{\tau_p s + 1} e^{-\theta s} = \frac{0,72}{167,33s + 1} e^{-5,48s} \quad (4.4)$$

É constatado que para os quatro casos, o ajuste da constante de ganho e da constante de tempo foi apropriado e os resultados foram considerados satisfatórios, conforme pode-se observar na comparação entre os dados e o modelo apresentados nas Figuras 22 e 23. Pode-se observar o comportamento dinâmico da Temperatura 1 em relação ao Aquecedor 1 e da Temperatura 2 em relação a variação do Aquecedor 2 podem ser considerados aproximadamente semelhantes, em função da proximidade das funções de transferência $G_{11}(s)$ e $G_{22}(s)$.

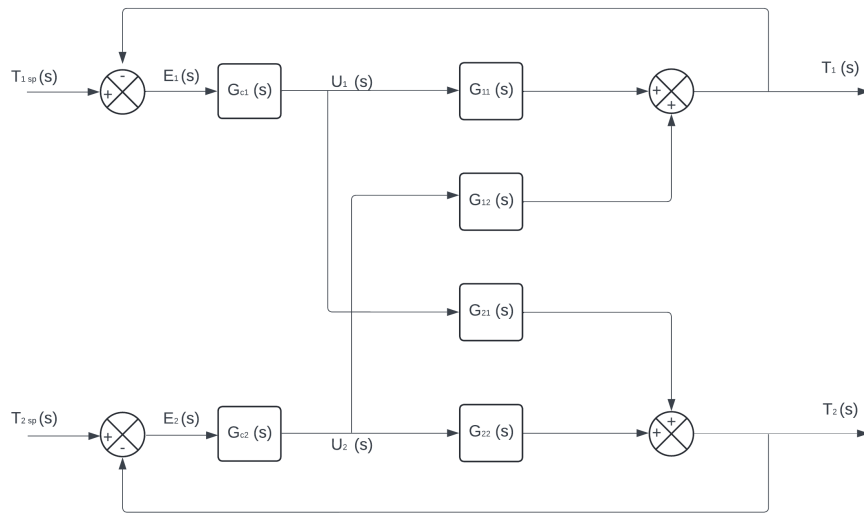
Como esperado e conforme os resultados apresentados, a mudança no Aquecedor 1, embora afete as duas temperaturas, exerce um impacto maior sobre a Temperatura 1, lida pelo sensor que estava em contato com ele. A variação da Temperatura 1 foi de cerca de 15°C, enquanto a Temperatura 2 apresentou uma amplitude de aproximadamente 6°C. De forma semelhante, a alteração na porcentagem do Aquecedor 2 teve um efeito mais pronunciado no aumento da Temperatura 2, com uma variação de cerca de 14°C, em comparação a Temperatura 1, que apresentou uma mudança de aproximadamente 6°C.

Essa análise foi confirmada pela análise da matriz de ganhos relativos (RGA - *Relative Gain Array*), (Equação 4.5), estudada por Bristol (1966). A matriz RGA é usada para determinar quais pares de variáveis de entrada e saída devem ser controlados conjuntamente para minimizar a interação entre as variáveis e melhorar o desempenho do sistema de controle.

$$\Lambda = \begin{bmatrix} 1,2852 & -0,2852 \\ -0,2852 & 1,2852 \end{bmatrix} \quad (4.5)$$

Por isso, o controle de temperatura foi estabelecido de forma que a malha do controle da Temperatura 1 foi ajustada com base nos parâmetros da função de transferência $G_{11}(s)$, enquanto a malha de controle da Temperatura 2 foi ajustada com base nos parâmetros da função de transferência $G_{22}(s)$, conforme o pareamento $T_1 - U_1$ e $T_2 - U_2$, demonstrado no diagrama de blocos na Figura 24.

Figura 24 – Diagrama de blocos simplificado do Sistema de Controle MIMO com pareamento 1-1/2-2



Fonte: Autoria Própria (2024)

4.2 Implementação do Controle PI

Os resultados das configurações dos controladores estão apresentados nas Tabelas 4 e 5, que correspondem a primeira malha de controle da Temperatura 1 e a segunda malha de controle da Temperatura 2, respectivamente.

Tabela 4 – Parâmetros de sintonia do controle PI determinados por diferentes métodos | Malha 1: $T_1 - U_1$

Parâmetro	IMC($\tau_c = 0,25\tau_p$)	IMC($\tau_c = 0,4\tau_p$)	ITAE (servo)
K_c (°C/%)	4,51	2,50	15,04
τ_i (s)	176,33	176,33	172,26

Fonte: Autoria Própria (2024)

Tabela 5 – Parâmetros de sintonia do controle PI determinados por diferentes métodos | Malha 2: $T_2 - U_2$

Parâmetro	IMC($\tau_c = 0,25\tau_p$)	IMC($\tau_c = 0,4\tau_p$)	ITAE (servo)
K_c (°C/%)	4,94	3,23	18,78
τ_i (s)	167,33	167,33	163,31

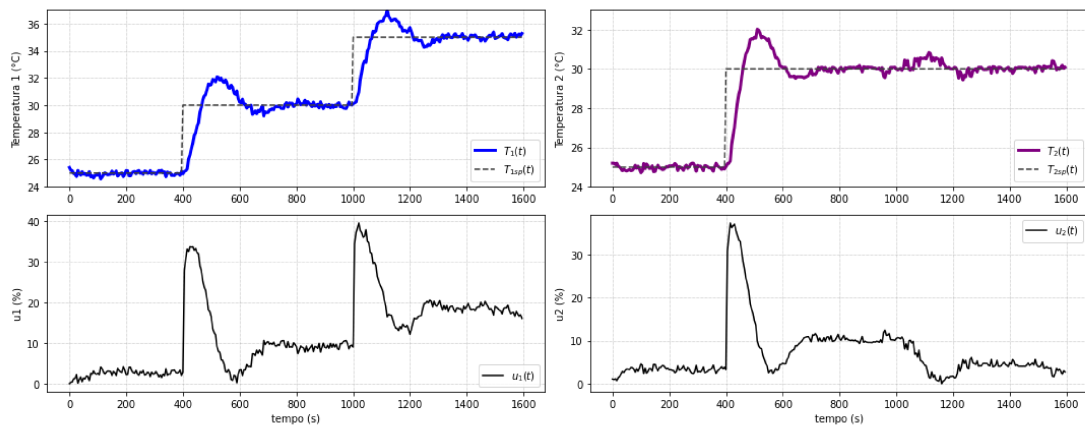
Fonte: Autoria Própria (2024)

Para investigar o comportamento do sistema em malha fechada, avaliar a eficácia do controle implementado e identificar o método de sintonia mais adequado, foram realizados testes em malha fechada com variações degrau nos *setpoints* das duas temperaturas. Isso permitiu uma

análise detalhada do comportamento do sistema sob efeitos do rastreamento do *setpoint*, além de possibilitar a comparação do desempenho dos controladores testados.

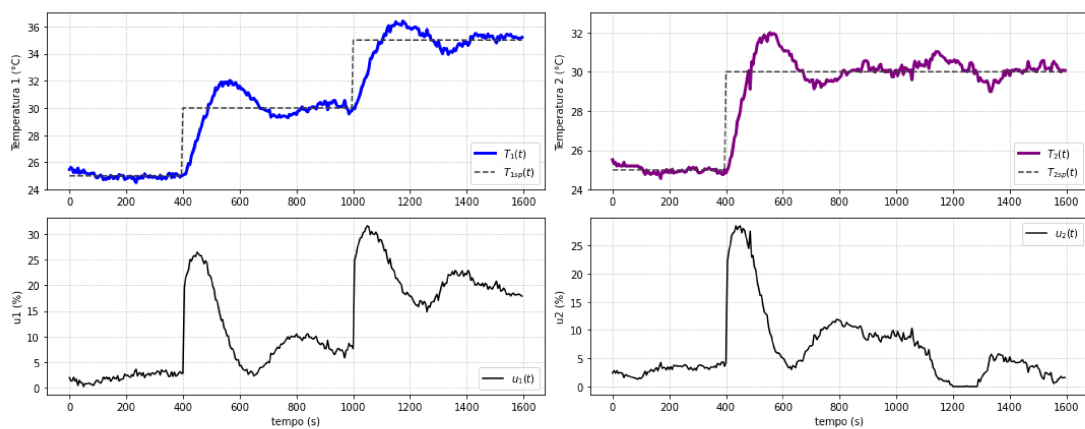
O teste experimental consistiu em aplicar uma alteração degrau nos *setpoints* das duas temperaturas no instante 400 segundos e em seguida aplicar um degrau no *setpoint* apenas da Temperatura 1 no instante de 1000 segundos e manter constante a Temperatura 2. Os resultados obtidos desse teste são apresentados nas Figuras 25, 26 e 27, que correspondem às sintonias realizadas pelos métodos IMC($\tau_c = 0,25\tau_p$), IMC($\tau_c = 0,4\tau_p$) e ITAE (servo), respectivamente. Uma comparação das respostas do sistema, obtidas pelas 3 sintonias de controle, é apresentada no gráfico da Figura 28. Através desse gráfico é possível fazer uma avaliação visual e qualitativa da resposta dinâmica de cada um dos testes em malha fechada.

Figura 25 – Teste Controle das Temperaturas 1 e 2 - Problema Servo | Sintonia pelo método IMC($\tau_c = 0,25\tau_p$)



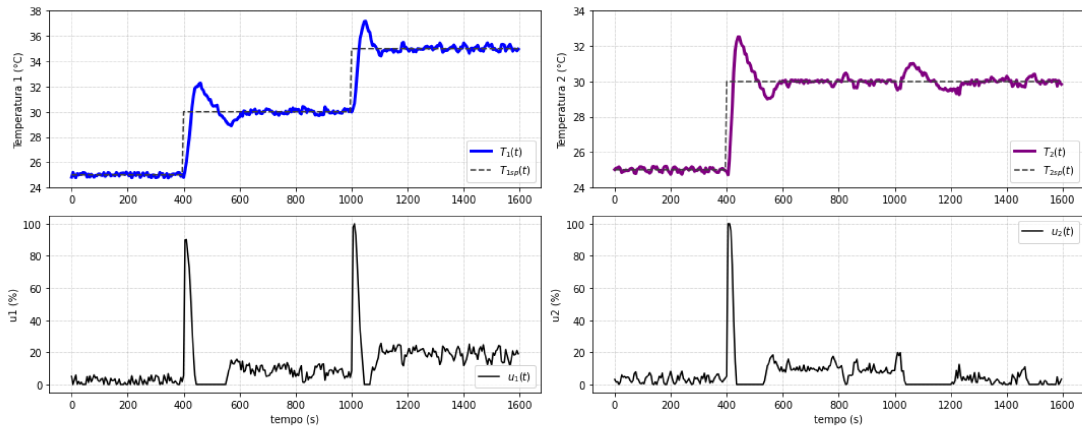
Fonte: Autoria Própria (2024)

Figura 26 – Teste Controle das Temperaturas 1 e 2 - Problema Servo | Sintonia pelo método IMC($\tau_c = 0,4\tau_p$)



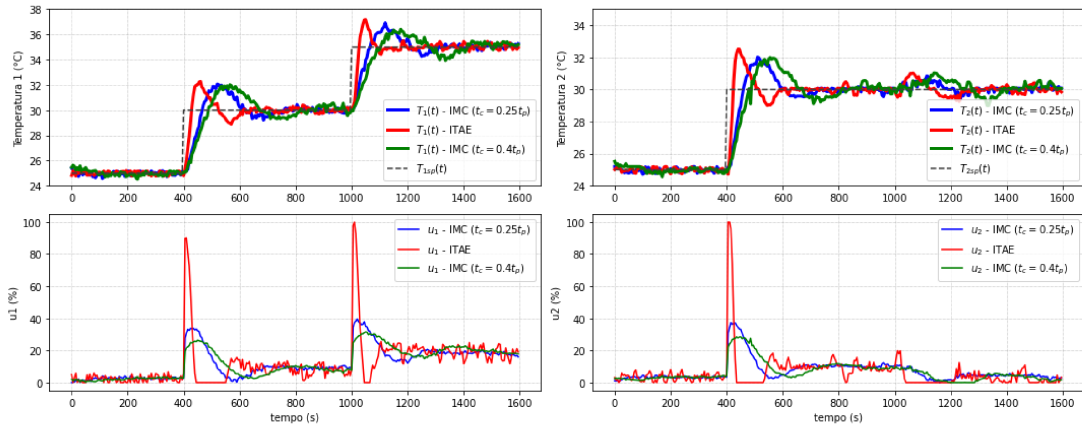
Fonte: Autoria Própria (2024)

Figura 27 – Teste Controle das Temperaturas 1 e 2 - Problema Servo | Sintonia pelo método ITAE



Fonte: Autoria Própria (2024)

Figura 28 – Teste Controle das Temperaturas 1 e 2 - Problema Servo | Comparativo Sintonias



Fonte: Autoria Própria (2024)

É possível observar que, ao aplicar um controlador PI ao sistema (que em malha aberta é de primeira ordem), resulta em um comportamento de malha fechada de segunda ordem. Isso ocorre devido ao efeito do termo integral do controlador, que introduz uma dinâmica adicional e melhora a resposta do sistema ao diminuir o erro estacionário e ajustar a velocidade de resposta.

A análise das Figuras 25 e 27 demonstra que a configuração e sintonia dos dois controladores do sistema foram satisfatórias. Embora as sintonias tenham apresentado um pequeno *overshoot* de aproximadamente 2°C, o sistema foi capaz de ajustar rapidamente as temperaturas ao valor de referência estabelecido. Esse desempenho indica uma resposta rápida e eficaz do sistema ao controle aplicado, o que é desejável em muitos processos térmicos.

Porém, percebe-se pela Figura 26 que, para a sintonia IMC($\tau_c = 0,4\tau_p$), o sistema apresenta uma resposta mais lenta, uma vez que a Temperatura 1 não havia estabilizado totalmente no *setpoint* de 30 °C, antes da aplicação do segundo degrau. Da mesma forma, a Temperatura 2

apresenta pequenas oscilações em torno do seu *setpoint*. Esse comportamento se deve a ganhos insuficientes, que limita a resposta rápida do sistema.

Ao avaliar as variáveis manipuladas, nota-se que o controle baseado no método ITAE (servo) foi mais agressivo entre eles. Esse comportamento agressivo pode ser observado nos picos das variáveis manipuladas, $u_1(\%)$ e $u_2(\%)$, em resposta as mudanças dos *setpoints* e através dos parâmetros utilizados para sua sintonia. A maior amplitude nos movimentos de controle para o controlador PI sintonizado pelo método ITAE justifica-se por sua configuração com maiores valores da constante de ganho (K_c) e menores valores da constante de tempo (τ_i). Entretanto, isso também torna o sistema mais sensível a perturbações, como evidenciado pelos sinais de controle com maior ruído. Além disso, o ITAE apresentou saturação de u_1 e u_2 em 0%, uma vez que a escolha dos *setpoints* favoreceu essa saturação, já que o resfriamento do sistema não é ativo e o bloco PID do *OpenPLC* não possui recurso de *anti-windup*.

Em contraste, os controladores configurado pelo método IMC respondem de maneira mais suave e com um menor *overshoot*, além de apresentar perturbações. Isso pode ser preferível para evitar desgaste ou demanda excessiva das variáveis manipuladas, além de assegurar maior estabilidade, uma vez que sinais de controle menos agressivos tendem a preservar os componentes do sistema. Devido a isso, por alcançar os *setpoints* de forma satisfatória, sem ação de controle excessiva e saturação das variáveis manipuladas, o controle IMC $\tau_c = 0,25\tau_p$ obteve um melhor desempenho.

Adicionalmente, é importante destacar a influência das ações de controle nas variáveis manipuladas. Quando ocorre a segunda alteração de grau da Temperatura 1, resultando em seu aumento, a ação de controle do elemento aquecedor/transistor 2 é ajustada de forma a reduzir sua influência, mantendo a Temperatura 2 constante. Esse ajuste evidencia a capacidade do sistema em fazer ajustes simultâneos em diferentes atuadores conforme as necessidades do processo.

A Tabela 6 fornece uma visão quantitativa sobre o comportamento dos controladores através dos índices de desempenho IAE, ISE e ITAE.

Tabela 6 – Índices de Desempenho para cada sintonia do controlador PI

		IMC($\tau_c = 0,25\tau_p$)	IMC($\tau_c = 0,4\tau_p$)	ITAE (servo)
IAE	T_1	901,62	1150,32	625,33
	T_2	556,67	796,79	490,56
ISE	T_1	1947,47	2409,68	1175,95
	T_2	994,34	1330,43	750,58
ITAE	T_1	684209,78	914782,40	468084,63
	T_2	348797,17	555431,99	341446,49

Fonte: Autoria Própria (2024)

Em termos de índices de desempenho, percebe-se que o método de sintonia ITAE apresentou os menores valores de erros (IAE, ISE e ITAE), em comparação com as duas sintonias usando o método IMC. Por outro lado, o controlador ajustado com o método IMC com $\tau_c = 0,25\tau_p$ apresentou valores intermediários de erro, com uma resposta rápida e um

desempenho mais suave do que o ITAE. Já o IMC com $\tau_c = 0,4\tau_p$, embora menos agressivo, resultou em maiores valores de erro (IAE, ISE e ITAE) em relação aos outros métodos, sugerindo que essa configuração comprometeu a precisão em favor de uma maior estabilidade.

4.3 Implementação do Controle MPC

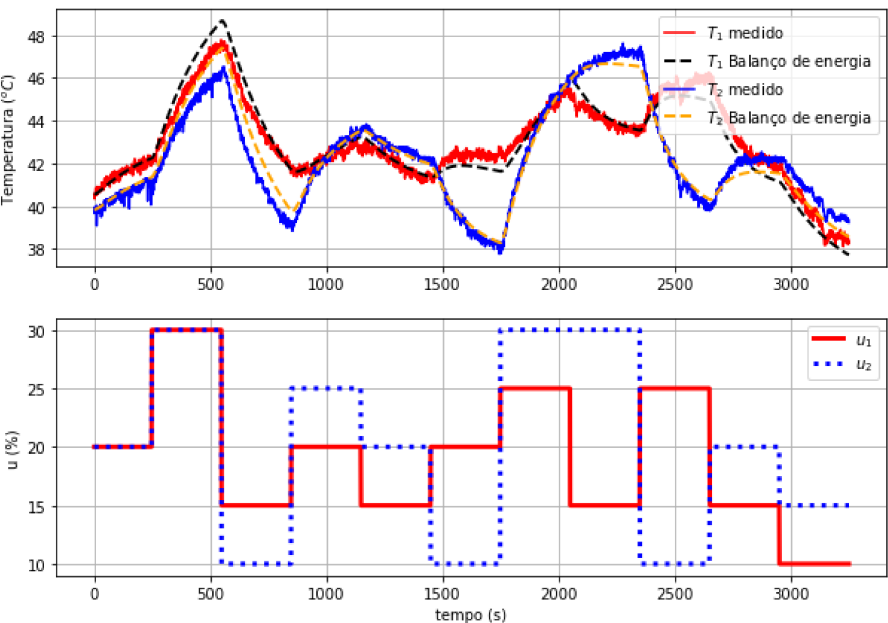
Para a implementação e definição dos parâmetros de sintonia do Controle QDMC, foram realizados testes de simulação utilizando o Modelo não linear da planta descrito na Seção 3.2. Para tanto, foi necessário realizar um teste com diversas alterações em degrau para o Aquecedor 1 e Aquecedor 2 para a estimativa dos parâmetros $U, U_s, \alpha_1, \alpha_2$ e τ_c e validação do modelo. Os valores estimados de tais parâmetros são apresentados na Tabela 7. Um comparativo entre os valores medidos de temperatura e aqueles estimados pelo modelo não linear da planta, frente a diferentes variações nos sinais de controle é apresentado na Figura 29.

Tabela 7 – Parâmetros Estimados do Modelo Matemático

Parâmetros	Valor
Coeficiente de Tranferência de Calor (U)	$1 \text{ W}/(m^2 \text{ K})$
Coeficiente de Tranferência de Calor entre os aquecedores (U_s)	$5 \text{ W}/(m^2 \text{ K})$
Fator de aquecimento (α_1)	$0,00671 \text{ W}/(\%)$
Fator de aquecimento (α_2)	$0,00571 \text{ W}/(\%)$
Constante de Tempo de Condução (τ_c)	$13,02803 \text{ s}$

Fonte: Autoria Própria (2024)

Figura 29 – Comparativo valores medidos e Modelo baseado nos Balanços de Energia do Sistema em resposta a diversas alterações nos Aquecedores



Fonte: Autoria Própria (2024)

De maneira geral, nota-se pelo gráfico da Figura 29 que o modelo não linear da planta é eficaz na previsão das respostas das temperaturas do sistema às variações nos aquecedores. Embora existam pequenas diferenças entre os valores modelados e medidos, possivelmente devido a fatores não contemplados no modelo, como perdas térmicas ou ruído nos sensores, o comportamento geral é bem capturado. Assim, o modelo proposto foi adequado para realizar a simulação do controle QDMC.

A partir desse modelo matemático, foram obtidos os coeficientes de resposta ao degrau para a definição da matriz de resposta ao degrau S . Em seguida, diversas simulações foram realizadas, variando parâmetros de sintonia como o horizonte de predição (H_p), o horizonte de controle (H_c), as matrizes de pesos Q_y e Q_u , além das restrições em u e Δu . O objetivo foi encontrar uma configuração dos parâmetros que apresentassem um equilíbrio entre a precisão das predições com a estabilidade e robustez do sistema em malha fechada, ao mesmo tempo que se buscava minimizar o custo computacional para a resolução do problema de otimização do QDMC.

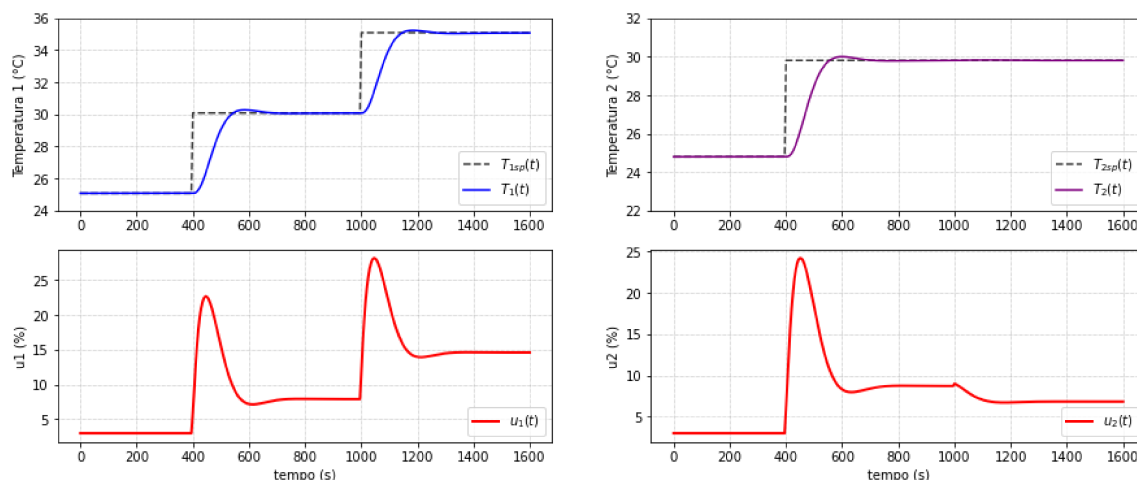
Os dois conjuntos de parâmetros de sintonia do QDMC foram considerados nas simulações estão dispostos na Tabela 8. A primeira simulação foi realizada levando em consideração o tempo dominante do sistema, τ_p , e sua resposta é apresentada na Figura 30. Para este caso, observou-se um elevado custo computacional para a resolução do problema de otimização devido as grandes e complexas matrizes geradas pelo problema de otimização. Para reduzir esse custo, buscou-se reproduzir uma resposta semelhante do sistema utilizando menores valores para o horizonte de predição (H_p) e o horizonte de controle (H_c), conforme definido na Simulação 2 e comprovado na Figura 31. O aumento do peso Q_y não se justificou pois tornava o sistema mais próximo da instabilidade, enquanto o acréscimo de Q_u deixava a resposta mais lenta. Por isso, para a implementação prática do controle QDMC, optou-se por adotar os parâmetros definidos na Simulação 2.

Tabela 8 – Parâmetros de sintonia do controle QDMC para Simulação

Parâmetro	Simulação 1	Simulação 2
H_p	390	60
H_c	100	10
Q_y	Matriz diagonal [1,1]	Matriz diagonal [1,1]
Q_u	Matriz diagonal [1,1]	Matriz diagonal [1,1]
$\Delta u_{\max}$	[10,10]	[10,10]
$\Delta u_{\min}$	[-10,-10]	[-10,-10]
$u_{\max}$	[100,100]	[100,100]
$u_{\min}$	[0,0]	[0,0]

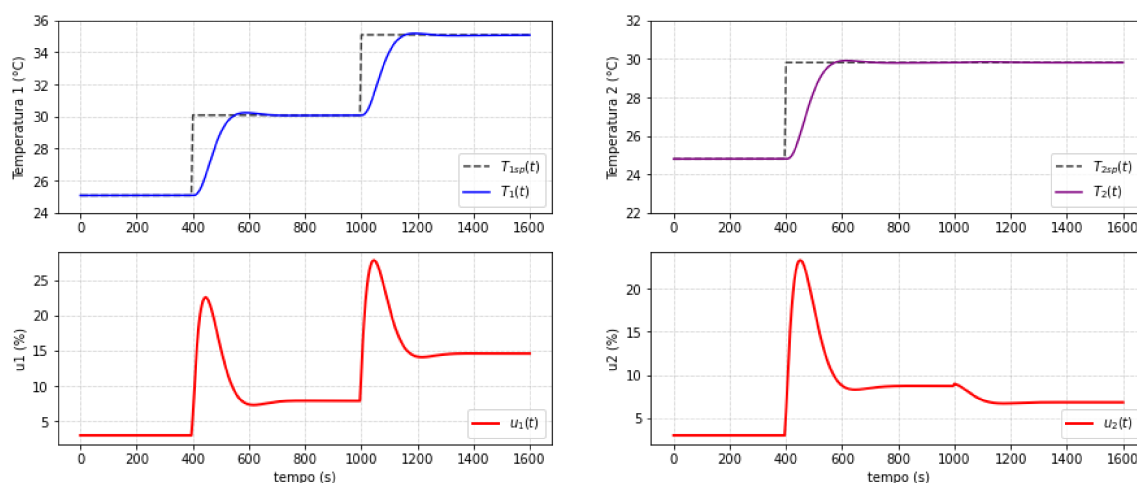
Fonte: Autoria Própria (2024)

Figura 30 – Resposta em malha fechada com o controlador QDMC utilizando os parâmetros da Simulação 1



Fonte: Autoria Própria (2024)

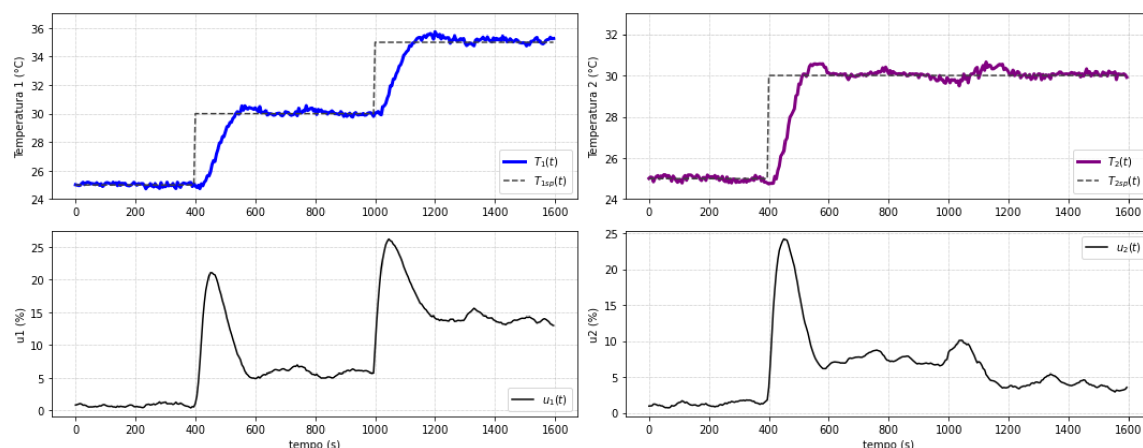
Figura 31 – Resposta em malha fechada com o controlador QDMC utilizando os parâmetros da Simulação 2



Fonte: Autoria Própria (2024)

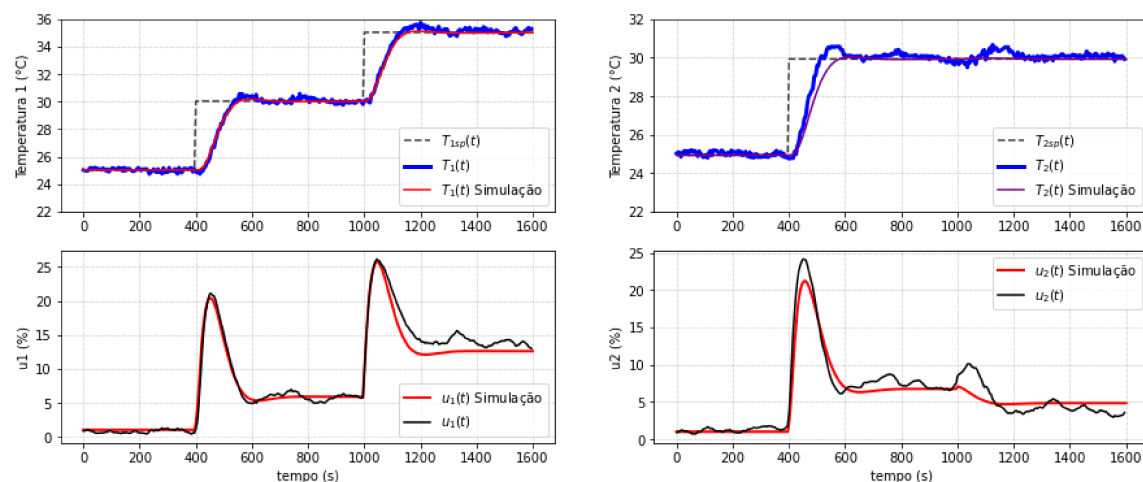
Após a definição da sintonia do controle QDMC, foi realizado um experimento similar ao do controlador PID, com variações em degrau nos *setpoints* das duas temperaturas. Para este teste, foram utilizados os coeficientes de resposta ao degrau obtidos experimentalmente, através do primeiro teste em malha aberta ilustrado nas Figuras 22 e 23. A resposta do controlador QDMC é exposta na Figura 32. Adicionalmente, o gráfico da Figura 33 apresenta a resposta do sistema obtida experimentalmente comparada àquela estimada pela simulação utilizando o modelo baseado no balanço de energia.

Figura 32 – Teste Controle das Temperaturas 1 e 2 - Problema Servo | Controlador QDMC



Fonte: Autoria Própria (2024)

Figura 33 – Comparativo da resposta do sistema em malha fechada com os parâmetros da simulação e dados experimentais do controle QDMC



Fonte: Autoria Própria (2024)

A Figura 32 demonstra que a estratégia e controle avançado QDMC apresenta um bom desempenho em termos de estabilidade, ao mesmo tempo que evita ações abruptas nos atuadores. Não há *overshoot* nem oscilações significativas, o que sugere que o controlador está cumprindo seu papel de otimizar as ações de controle ao longo do tempo. Ambas as temperaturas seguem bem suas referências, com destaque para a Temperatura 1. Embora haja uma pequena oscilação na Temperatura 2 após a transição, provavelmente devido a perturbações e pequenas inconsistências do modelo, mas ainda assim se mantém dentro de limites aceitáveis.

Além disso, a boa sobreposição das curvas experimental e estimada, na Figura 33 indica que a simulação é consistente com os resultados reais, sugerindo que o modelo matemático considerado é uma boa representação do sistema mesmo quando utilizado para simulações em malha fechada. Isso evidencia que a simulação pode ser uma ótima opção para validação do

modelo e definição de adequados parâmetros de sintonia, otimizando o processo em comparação à realização de múltiplos testes experimentais.

Os valores dos índices de desempenho IAE, ISE e ITAE, do MPC em comparação aos diferentes PID's são apresentados na Tabela 9.

Tabela 9 – Índices de Performance dos Controladores PID e MPC

		IMC($\tau_c = 0,25\tau_p$)	IMC($\tau_c = 0,4\tau_p$)	ITAE (servo)	QDMC
IAE	T_1	901,62	1150,32	625,33	964,48
	T_2	556,67	796,79	490,56	564,46
ISE	T_1	1947,47	2409,68	1175,95	2969,78
	T_2	994,33	1330,43	750,58	1471,62
ITAE	T_1	684209,78	914782,40	468084,63	733870,64
	T_2	348797,17	555431,99	341446,49	334748,256

Fonte: Autoria Própria (2024)

A análise dos resultados apresentados na Tabela 9 demonstra que o controlador QDMC exibiu um bom desempenho, proporcionando um rastreamento eficaz dos *setpoints* e mantendo a estabilidade do sistema. Comparado às outras estratégias de controle, o QDMC obteve valores de IAE e ITAE inferiores apenas ao PID sintonizado com $\text{IMC}(\tau_c = 0,4\tau_p)$, porém sem grandes desvios. Além disso, nota-se que os valores de todos os índices se aproximam do $\text{IMC}(\tau_c = 0,25\tau_p)$.

De maneira geral, o controle QDMC mostrou-se eficaz, proporcionando uma resposta estável e suave, especialmente relacionado à Temperatura 1. Embora os dados revelem que o controle da Temperatura 2 ainda possa ser melhorado, a performance geral do QDMC foi particularmente destacada por não apresentar oscilações abruptas ou grandes desvios.

5 CONCLUSÃO

O presente trabalho apresentou o desenvolvimento e análise de controle de temperatura de um sistema MIMO com o uso de um microcontrolador integrado ao *software OpenPLC* e a aplicação da linguagem *Python* permitiram a reprodução de configuração de um CLP em *Ladder*, além da implementação de diferentes estratégias de controle.

A aplicação de diferentes técnicas de controle, como PID e MPC, permitiu a compreensão clara das diferenças entre o controle clássico PID, integrado à Camada Base de Controle de Processos e implementado dentro do controlador, e o MPC, um Controle Avançado de Processos, executado no Computador de Processos. Diante disso, foi possível avaliar os benefícios e limitações de cada abordagem no contexto do dispositivo.

Os resultados obtidos indicam que tanto o PID quanto o QDMC apresentaram desempenhos satisfatórios no rastreamento de *setpoints*. Os métodos de sintonia PID, baseados em IMC e ITAE, demonstraram boa capacidade de ajuste das variáveis controladas. Entre eles, o método ITAE apresentou menores índices de erro (IAE, ISE e ITAE), porém com uma resposta mais agressiva. Já o método IMC proporcionou respostas mais suaves e com menor *overshoot*.

O controle QDMC, por sua vez, mostrou-se uma alternativa que proporciona uma resposta mais suave e consistente com a simulação do controle baseada no modelo da planta. A sobreposição entre as curvas experimentais e simuladas evidenciou a adequação do modelo matemático utilizado, destacando o papel fundamental da simulação para a sintonia e otimização desse controle.

Portanto, pode-se concluir que o dispositivo experimental para o controle de temperatura utilizado neste trabalho representa uma ótima oportunidade de ferramenta didática de baixo custo e alta versatilidade para facilitar a compreensão e aplicação de estratégias de controle de processos industriais. A integração do protótipo com o *OpenPLC* e o *Python* proporciona um ambiente de experimentação que pode ser replicado no contexto de ensino, promovendo um aprendizado prático e eficaz sobre controle de processos.

5.1 Sugestões para Trabalhos Futuros

Para futuros trabalhos, foram identificadas as seguintes sugestões:

- Implementar o desacoplamento das malhas de controle para minimizar a interferência entre elas, assegurando que o controle PID em cada malha seja menos influenciado pelas demais.
- Implementar um controlador PI com recurso *anti-windup* em *Ladder*, de modo que a ação integral seja adequadamente limitado quando as saídas do sistema atingirem seus limites, evitando saturação e melhorando a resposta do controlador.
- Realizar testes com diferentes tipos de perturbações no sistema, proporcionando uma visão mais abrangente sobre o desempenho dos controladores em diferentes cenários.

- Desenvolver uma interface gráfica intuitiva para supervisão e monitoramento dos dados do sistema durante os testes, empregando a linguagem *Python*, de modo a facilitar a visualização e o acompanhamento das variáveis de processo em tempo real.

REFERÊNCIAS

- ALPHONSUS, E. R.; ABDULLAH, M. O. A review on the applications of programmable logic controllers (plcs). *Renewable and Sustainable Energy Reviews*, v. 60, p. 1185–1205, 2016. ISSN 1364-0321. Citado 2 vezes nas páginas 28 e 29.
- ALVES, T. *OpenPLC*. 2017. Disponível em: <<https://autonomylogic.com/docs/>>. Citado 2 vezes nas páginas 42 e 48.
- ALVES, T.; MORRIS, T. Openplc: An iec 61,131–3 compliant open source industrial controller for cyber security research. *Computers Security*, v. 78, p. 364–379, 2018. ISSN 0167-4048. Citado 3 vezes nas páginas 28, 29 e 31.
- ANALOGDEVICES. *Datasheet Sensor de Temperatura TMP35/TMP36/TMP37*. 2002. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Citado na página 37.
- ARDUINO. *Datasheet Arduino UNO*. 2024. Disponível em: <<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>>. Citado na página 36.
- ARDUINO. *What is Arduino?* 2024. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Citado 2 vezes nas páginas 32 e 46.
- BEAL, L.; HILL, D.; MARTIN, R.; HEDENGREN, J. Gekko optimization suite. *Processes*, Multidisciplinary Digital Publishing Institute, v. 6, n. 8, p. 106, 2018. Citado na página 40.
- CAMACHO, E.; ALBA, C. *Model Predictive Control*. [S.l.]: Springer London, 2013. ISBN 9780857293985. Citado 3 vezes nas páginas 20, 28 e 53.
- COSTA, T. V. da. *Estudo e implementação de estruturas de controle reconfigurável aplicado a processos químicos*. Tese (Doutorado) — Universidade Estadual de Campinas, Faculdade de Engenharia Química, Campinas, SP, 2014. Citado na página 44.
- ELAMANOV, S.; SON, H.; FLYNN, B.; YOO, S. K.; DILSHAD, N.; SONG, J. Interworking between modbus and internet of things platform for industrial services. *Digital Communications and Networks*, v. 10, n. 2, p. 461–471, 2024. ISSN 2352-8648. Citado na página 32.
- FONSECA, G. P. d. *Aprendizagem emocional em hardware: aplicação em controle térmico*. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2022. Citado na página 34.
- GREEFF, G.; GHOSHAL, R. *Practical E-Manufacturing and Supply Chain Management*. [S.l.]: Newnes, 2004. (Practical Professional). ISBN 9780080473857. Citado 2 vezes nas páginas 28 e 31.
- GUZMÁN, J.; GARCÍA-MAÑAS, F.; HOYO, ; RAMOS-TEODORO, J.; DONAIRE, J. Use of tclab kits for control engineering curricula at the university of almería*. *IFAC-PapersOnLine*, v. 55, n. 17, p. 362–367, 2022. ISSN 2405-8963. 13th IFAC Symposium on Advances in Control Education ACE 2022. Citado na página 13.
- HEDENGREN, J. D. *APMonitor*. 2019. Disponível em: <<https://apmonitor.com/pdc/>>. Citado na página 33.
- KVALSUND, A. S.; WINKLER, D. Development of an arduino-based, open-control interface for hardware in the loop applications. *HardwareX*, v. 16, p. e00488, 2023. ISSN 2468-0672. Citado 2 vezes nas páginas 13 e 32.

- MORARI, M.; LEE, J. H.; GARCÍA, C. E. *Model Predictive Control*. [S.l.]: Preprint, 2002. ISBN 3540762418. Citado 2 vezes nas páginas 21 e 53.
- MÁRQUEZ-VERA, M.; MARTÍNEZ-QUEZADA, M.; CALDERÓN-SUÁREZ, R.; RODRÍGUEZ, A.; ORTEGA-MENDOZA, R. Microcontrollers programming for control and automation in undergraduate biotechnology engineering education. *Digital Chemical Engineering*, v. 9, p. 100122, 2023. ISSN 2772-5081. Citado na página 32.
- NIU, S.; XIAO, D. *Process Control: Engineering Analyses and Best Practices*. [S.l.]: Springer International Publishing, 2022. ISBN 9783030970673. Citado 3 vezes nas páginas 15, 20 e 50.
- OLIVEIRA, P. B. de M.; SOARES, F.; CARDOSO, A. Pocket-sized portable labs: Control engineering practice made easy in covid-19 pandemic times. *IFAC-PapersOnLine*, v. 55, n. 17, p. 150–155, 2022. ISSN 2405-8963. 13th IFAC Symposium on Advances in Control Education ACE 2022. Citado na página 13.
- OLIVEIRA, P. de M.; HEDENGREN, J. D.; ROSSITER, J. Introducing digital controllers to undergraduate students using the tclab arduino kit. *IFAC-PapersOnLine*, v. 53, n. 2, p. 17524–17529, 2020. ISSN 2405-8963. 21st IFAC World Congress. Citado 2 vezes nas páginas 13 e 34.
- PAPARELLA, L. Sistema programável com ferramenta open-source para ensino de controle e automação. <http://repositorio.utfpr.edu.br/jspui/handle/1/27982>, 2021. Citado 3 vezes nas páginas 13, 32 e 34.
- PARK, J.; MARTIN, R. A.; KELLY, J. D.; HEDENGREN, J. D. Benchmark temperature microcontroller for process dynamics and control. *Computers Chemical Engineering*, v. 135, p. 106736, 2020. ISSN 0098-1354. Citado 5 vezes nas páginas 13, 34, 38, 40 e 52.
- PETRUZELLA, F. D. *Programmable logic controllers*. [S.l.]: McGraw-Hill Higher Education, 2011. Citado 2 vezes nas páginas 29 e 30.
- PIDTUNER. *PIDTuner*. 2024. Disponível em: [<https://pidtuner.com/#/>](https://pidtuner.com/#/). Citado na página 50.
- ROQUE, L. *Automação de processos com linguagem ladder e sistemas supervisórios*. [S.l.]: Grupo Gen - LTC, 2014. ISBN 9788521626824. Citado na página 40.
- ROSSITER, J. *Model-Based Predictive Control: A Practical Approach*. [S.l.]: Springer, 2017. ISBN 9781315272610. Citado na página 20.
- ROSSITER, J.; POPE, S.; JONES, B.; HEDENGREN, J. Evaluation and demonstration of take home laboratory kit. *IFAC-PapersOnLine*, v. 52, n. 9, p. 56–61, 2019. ISSN 2405-8963. 12th IFAC Symposium on Advances in Control Education ACE 2019. Citado na página 13.
- SEBORG, D.; MELLICHAMP, D.; EDGAR, T.; DOYLE, F. *Process Dynamics and Control*. [S.l.]: John Wiley & Sons, 2010. ISBN 9780470128671. Citado 12 vezes nas páginas 12, 15, 16, 18, 19, 21, 47, 48, 49, 50, 51 e 52.
- SILVA, R.; STEVAN, S. *Automação e Instrumentação Industrial com Arduino - Teoria e Projetos*. [S.l.]: Editora Érica | Saraiva, 2015. ISBN 9788536515663. Citado na página 32.
- SILVEIRA, L. A. Estudo de modelagem e simulação da planta didática mtx-lab. *Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Química*, 2022. Citado na página 12.

TASCA, L. C.; FREITAS, E. P. de; WAGNER, F. R. A study on the performance impact of programmable logic controllers based on enhanced architecture and organization. *Microprocessors and Microsystems*, v. 76, p. 103082, 2020. ISSN 0141-9331. Citado na página 29.

UPADHYAY, D.; SAMPALLI, S. Scada (supervisory control and data acquisition) systems: Vulnerability assessment and security recommendations. *Computers Security*, v. 89, p. 101666, 2020. ISSN 0167-4048. Citado na página 31.

WEBB, J.; REIS, R. *Programmable Logic Controllers: Principles and Applications*. [S.l.]: Prentice Hall Education, Career & Technology, 2013. ISBN 9780024249807. Citado na página 12.