Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

NATHÁLIA ARTHUR BRUNET MONTEIRO

MONITORAMENTO E CONTROLE DE TEMPERATURA DE UMA ESTUFA UTILIZANDO O CONCEITO DE SENSOR VIRTUAL

Campina Grande, Paraíba
Julho de 2014

NATHÁLIA ARTHUR BRUNET MONTEIRO

MONITORAMENTO E CONTROLE DE TEMPERATURA DE UMA ESTUFA UTILIZANDO O CONCEITO DE SENSOR VIRTUAL

Dissertação de Mestrado submetida à Coordenação de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica

Professor José Sérgio da Rocha Neto, D. Sc.
Orientador

Professor Jaidilson Jó da Silva, D. Sc
Orientador

Campina Grande, Paraíba
Julho de 2014

M775m Monteiro, Nathália Arthur Brunet.
Monitoramento e controle de temperatura de uma estufa utilizando o conceito de sensor virtual / Nathália Arthur Brunet Monteiro. - Campina Grande, 2014.
61 f.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2014.
"Orientação: Prof. Dr. José Sérgio da Rocha Neto, Prof. Dr. Jaidilson Jó da Silva".
Referências.

1. Controle de Temperatura. 2. Sensor Virtual. 3. Modelos. 4. Estufa Térmica. 5. Dissertação - Engenharia Elétrica. I. Rocha Neto, José Sérgio da. II. Silva, Jaidilson Jó da. III. Universidade Federal de Campina Grande - Campina Grande (PB). IV. Título

CDU 621.317(043)

"MONITORAMENTO E CONTROLE DE TEMPERATURA DE UMA ESTUFA UTILIZANDO O CONCEITO DE SENSOR VIRTUAL "

NATHÁLIA ARTHUR BRUNET MONTEIRO

DISSERTAÇÃO APROVADA EM 28/07/2014

JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG
Orientador(a)

JAIDILSON JÓ DA SILVA, D.Sc., UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc., UFCG
Examinador(a)

SAULO OLIVEIRA DORNELLAS LUIZ, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

Agradecimentos

Agradeço aos meus pais e irmãos pelo apoio, compreensão, ajuda, e, em especial, por todo carinho ao longo deste percurso.

Aos professores Jaidilson e José Sérgio pela orientação e ajuda, sem a qual este trabalho não seria realizado.

Aos meus amigos, pela cumplicidade, ajuda e amizade.

E a todos que contribuíram de forma direta e indireta para realização deste trabalho.

Resumo

Grandezas e fenômenos físicos, como por exemplo, pressão, volume, diferença de potencial, resistência, dentre outras, dependem da temperatura, o que a torna um parâmetro de grande relevância. Atualmente é uma das variáveis mais usadas na indústria de controle de processos nos seus mais diversos segmentos. O controle automático tem desempenhado um papel fundamental no avanço da engenharia e da ciência, além de ser de grande importância e parte integrante dos processos industriais e de produção. Atualmente, sensores virtuais têm sido utilizados nas indústrias para fazer com que o sistema físico atenda às especificações de desempenho previamente estabelecidas com sucesso. Desta forma, neste trabalho, o sistema em questão é uma estufa térmica, em que se utiliza o conceito de sensor virtual para a realização do monitoramento e controle de temperatura desta estufa, baseado em duas vertentes: medições, onde foram realizadas várias repetições de testes para obter uma relação entre a tensão que é aplicada na resistência no interior da estufa para seu aquecimento e a temperatura medida por meio do sensor, tendo assim uma referência de que valor de temperatura esperar de acordo com a tensão aplicada e com a informação do tempo de variação desse valor de tensão aplicada; e modelagem e controle, onde é apresentada a avaliação dos métodos SIMC (Skogestad - *Internal Model Control*) e CHR (Chien, Hrones e Reswick) como estratégias de controle para o modelo dinâmico obtido para a planta em estudo, no caso, sistemas de primeira e segunda ordem com atraso. A aquisição dos valores referentes à temperatura é realizada utilizando uma placa de aquisição de dados, e a implementação da rotina para o cálculo dos controladores e a interface gráfica são realizadas por meio do *software* MATLAB.

Palavras-chave: Controle de Temperatura, Sensor Virtual, Modelos, Estufa Térmica.

Abstract

Quantities and physical phenomena, such as pressure, volume, potential difference, the phenomenon of boiling, electric current and resistance, among others, depend on temperature, which makes it a parameter of great relevance. It is one of the variables most used in different segments of the process control industry. Automatic control of process has performed a fundamental role in the advancement of engineering and science, and is very important and integral part of the industrial and production process. Currently, soft sensors have been used by industries to make with which physical system meets performance specifications previously established successfully. Thus, in this work, the system in question is a thermal conservatory, which uses the concept of soft sensors to perform the monitoring and control of temperature of this thermal conservatory, based on two aspects: Measurements, where several repetitions of tests were performed to obtain a relationship of the voltage that is applied in the resistance to warm the interior of the thermal conservatory and the temperature measured by the sensor, having thus a reference of which temperature value expected according to the applied voltage and with the variation of the time information of this value of the voltage applied; and Modeling and Control, where is presented the evaluation of SIMC (Skogestad - Internal Model Control) and CHR (Chien, Hrones and Reswick) methods as a control strategy for dynamic model obtained of the plant in study, in this case, first and second order systems with delay. The acquisition of temperature values is performed by the data acquisition board, and the implementation of the routine for the calculation of the controllers and the graphical interface are performed in the MATLAB software.

Keywords: Temperature Control, Soft Sensor, Models, Thermal Conservatory.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADA	Ambiente de Desenvolvimento de Aplicativos
CHR	*Chien, Hrones e Reswick*
DAQ	*Data Acquisition* (Aquisição de Dados)
ETA	Estação de Tratamento de Água
GUIDE	*Graphical User Interface Development Environment*
IAE	*Integral Absolute Error* (Integral do Erro Absoluto)
IMC	*Internal Model Control*
OC	Oxigênio Consumido
P	Proporcional
PCA	*Principal Component Analysis* (Análise de Componentes Principais)
PD	Proporcional Derivativo
pH	Potencial Hidrogeniônico
PLS	*Partial Least Squares* (Mínimos Quadrados Parciais)
PI	Proporcional Integral
PID	Proporcional Integral Derivativo
PVC	Policloreto de Vinila
SIMC	*Skogestad - Internal Model Control*
SISO	*Single Input-Single Output* (Entrada Única-Saída Única)

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Muitas Grandezas e fenômenos físicos, como por exemplo, pressão, volume, diferença de potencial, fenômeno da ebulição, resistência, corrente elétrica, dentre outras, dependem da temperatura, o que a torna um parâmetro de grande relevância. Atualmente é uma das variáveis mais usadas na indústria de controle de processos nos seus mais diversos segmentos [1].

O controle automático tem desempenhado um papel fundamental no avanço da Engenharia e da Ciência, além de ser de grande importância e parte integrante dos processos industriais e de produção. Por exemplo, é essencial no projeto de sistemas de piloto automático na indústria aeroespacial e no projeto de carros na indústria automotiva, como também em operações industriais como o controle de pressão, de temperatura, de umidade, de viscosidade e de vazão nos processos industriais [2].

Um sistema a controlar pode ser parte de um equipamento ou apenas um conjunto de componentes de um equipamento, que funcione de maneira integrada, com o objetivo de realizar determinada operação [2].

Sistemas mais complexos, como plantas de processamento industrial, são em geral fortemente instrumentados com um grande número de sensores. O objetivo principal dos sensores é adquirir dados para o processo de monitoramento e controle das plantas [3]. O conceito de utilização de sensores virtuais visa à modelagem matemática de processos com enfoque na predição de propriedades, a partir de medições disponíveis de outras variáveis da planta [4]. No contexto do processo industrial, esses modelos preditivos são chamados sensores virtuais (*soft sensors*). Este termo é uma combinação das palavras "*software*", pois os modelos são geralmente um conjunto de rotinas de *software*, e "sensores", pois os modelos estão estimando grandezas semelhantes às dos sensores reais [3].

Os sensores virtuais possuem uma grande área de aplicação, pois atuam auxiliando na monitoração, controle e otimização de processos em geral, fornecendo medições mais precisas, mais rápidas e mais confiáveis a um custo mais baixo tanto para desenvolvimento e implantação como para manutenção. Além disso, deve-se ressaltar que estes podem atuar substituindo sensores reais ou trabalhando em conjunto com estes e auxiliando na monitoração e controle de falhas e manutenção preventiva [4].

Soares [5] apresenta um caso de utilização de sensores virtuais para estimação em tempo real da temperatura dos fornos eletrolíticos e a simulação do balanço térmico dos mesmos a partir desse tipo de sensor. As medidas de temperaturas são feitas em intervalos muito longos de tempo e por instrumentação cara, dada à natureza corrosiva do forno. A proposta é oferecer alternativas para obter a temperatura usando a ideia de técnicas de inferência. Assim, de forma geral, o objetivo é a estimação do valor de temperatura atual e futuro com base na situação atual do forno, para que o controle da química de banho seja melhorado.

Choi [6] realizou o desenvolvimento de técnicas de sensores virtuais em uma planta de tratamento de águas residuárias. Foi proposta uma rede neural híbrida com um sensor virtual para inferir os parâmetros de qualidade das águas residuárias, o que inclui um método de estatística convencional de análise de componentes principais (PCA - *Principal Component Analysis*) para o pré-processamento dos dados. A técnica de redes neurais artificiais prevê valores de parâmetros não medidos utilizando a correlação entre os parâmetros medidos e o parâmetro alvo. Quando existe uma correlação entre os dados medidos, é possível reduzir a sua dimensão para a previsão de parâmetros não medidos usando PCA. A técnica de redes neurais artificiais mostra um aumento da capacidade de previsão e reduz o problema de super ajuste de redes neurais. O resultado mostra que esta técnica pode ser usada para extrair informação de dados com ruído e para descrever a não-linearidade de processos de tratamento de águas residuárias complexas.

Atkinson [7] descreve um sistema preditivo para diagnóstico *On-Board* (Sistema de autodiagnóstico e notificação desenvolvido para veículos automotores) e controle de motores empregando um sensor virtual baseado em rede neural. O sistema é capaz de prever em tempo real a potência do motor, o consumo do combustível e as emissões, utilizando medições dos parâmetros do motor através dos ciclos de operação transitórios do mesmo. O sistema consiste de um modelo preditivo do motor que é executado em um microprocessador, em paralelo com o motor, em tempo real, tendo sinais de entrada a partir dos mesmos sensores como o próprio motor.

Pode-se observar a utilidade de se monitorar e controlar processos em geral e ver que o uso dos sensores virtuais já é de indiscutível importância para a realização do monitoramento e controle de processos, obtendo resultados de forma mais rápida e mais confiável, melhorando o desempenho do processo como um todo.

1.2 OBJETIVOS

Visto que já são muitos os casos onde os sensores virtuais são utilizados para o monitoramento, controle e otimização de processos em geral, com medições mais precisas, mais rápidas e confiáveis; e que a temperatura é uma das grandezas físicas mais medidas em processos industriais, sendo relacionada ao conforto, segurança e qualidade, o objetivo principal nesta dissertação é utilizar o conceito de sensor virtual para realizar o monitoramento e controle de temperatura de uma estufa térmica utilizando duas vertentes: medições, para o monitoramento do processo, e modelagem e controle, para o controle da temperatura no interior da estufa.

Para o monitoramento do processo, utilizar um conjunto de medições para obter uma relação entre a tensão, que é aplicada na resistência no interior da estufa para seu aquecimento, e a temperatura medida por meio do sensor real, tendo assim uma referência de que valor de temperatura esperar de acordo com determinada tensão aplicada e da informação do tempo de variação desse valor de tensão aplicada;

Para o controle do processo, desenvolver a modelagem e controle da estufa e avaliar os métodos SIMC (Skogestad - *Internal Model Control*) e CHR (Chien, Hrones e Reswick) como estratégias de controle para o modelo dinâmico, obtido para a planta em estudo, no caso, sistemas de primeira e segunda ordem com atraso.

1.3 METODOLOGIA

Inicialmente, para a etapa de monitoramento, um conjunto de medições é realizado, com o intuito de se obter uma relação dos valores de tensão aplicada na resistência e de temperatura no interior da estufa. Após a obtenção deste conjunto de medições é realizado o tratamento estatístico, onde são calculados a média aritmética e o desvio padrão dos valores para se obter uma faixa de confiança.

Para a etapa de controle do processo, antes de projetar os controladores é necessário fazer a modelagem e identificação do sistema, para isso, com os dados obtidos nos testes foi realizada a geração dos modelos de primeira e segunda ordem com atraso de transporte, utilizando a ferramenta *ident* do MATLAB. Com os modelos determinados, o próximo passo foi a validação dos mesmos, onde foram utilizados testes diferentes para gerar o modelo e para fazer a comparação na validação.

Uma vez realizada a identificação do sistema, determinando e validando os modelos, a etapa seguinte foi o projeto dos controladores para atuar na estufa, onde foram calculados os parâmetros k_p (Ganho Proporcional), T_i (Constante de Tempo Integral) e T_{td} (Constante de Tempo Derivativa)

dos controladores utilizando os métodos de sintonia propostos por *Skogestad* (SIMC) e por *Chien, Hrones e Reswick* (CHR). Para a avaliação de desempenho e robustez dos sistemas de controle em malha fechada com os controladores projetados foram simuladas as respostas ao aplicar uma variação no sinal de referência e uma perturbação de carga, ambas do tipo degrau unitário e calculados os índices de desempenho M_s (Valor de pico da função de sensibilidade) e *IAE* (Integral do Erro Absoluto).

Por fim, foram criadas as interfaces gráficas para uma melhor interação com o usuário e, realizados os experimentos de monitoramento e controle com os controladores projetados atuando na estufa.

1.4 ORGANIZAÇÃO DO TEXTO

Esta dissertação está organizada da seguinte forma:

Neste capítulo é realizada uma breve introdução, sendo apresentados a motivação, objetivos e metodologia da dissertação.

No Capítulo 2, é apresentada uma revisão bibliográfica da dissertação, apontando algumas aplicações de sensores virtuais e uma fundamentação teórica sobre sensores virtuais, controle de temperatura, modelagem e identificação de sistemas.

No Capítulo 3 é realizada a descrição da estufa e da plataforma de testes da dissertação.

No Capítulo 4 são apresentados os experimentos realizados e os resultados obtidos. Neste Capítulo são apresentadas as medições realizadas, a identificação dos modelos matemáticos da planta e o projeto dos controladores para atuar na estufa, comentando sobre os métodos SIMC (Skogestad - *Internal Model Control*) e CHR (Chien, Hrones e Reswick), os índices de desempenho utilizados na classificação do melhor controlador, as interfaces gráficas criadas para uma melhor interação com o usuário e, por fim, as simulações e os resultados obtidos.

As conclusões gerais em relação ao trabalho realizado e também as sugestões para trabalhos futuros são apresentadas no Capítulo 5.

Nos ANEXOS A e B, são apresentadas as rotinas implementadas para o cálculo dos controladores pelos métodos SIMC e CHR, respectivamente.

Nos ANEXOS C e D, são apresentadas, respectivamente, as rotinas de monitoramento e controle do processo, ambas com a utilização do GUIDE *(Graphical User Interface Development Environment)* para criação da interface gráfica.

2 REVISÃO BIBLIOGRÁFICA E FUNDAMENTAÇÃO TEÓRICA

2.1 REVISÃO BIBLIOGRÁFICA

O conceito de utilização de sensores virtuais visa à modelagem matemática de processos com enfoque na predição de propriedades, a partir de medições disponíveis de outras variáveis da planta [4]. No contexto do processo industrial, esses modelos preditivos são chamados sensores virtuais (*soft sensors*). Este termo é uma combinação das palavras "*software*", pois os modelos são geralmente um conjunto de rotinas de *software*, e "sensores", pois os modelos estão representando informações semelhantes aos dos sensores reais [3].

Na Figura 2.1, o funcionamento de um sensor virtual é representado de forma simplificada: as variáveis Y, que não são medidas diretamente, são calculadas com base nas variáveis X, que estão disponíveis em um banco de dados histórico do processo; o algoritmo consulta estes dados e realiza cálculos para gerar uma estimativa da variável Y. Obviamente, serão necessárias medidas reais dessa variável para validar as medidas indiretas.

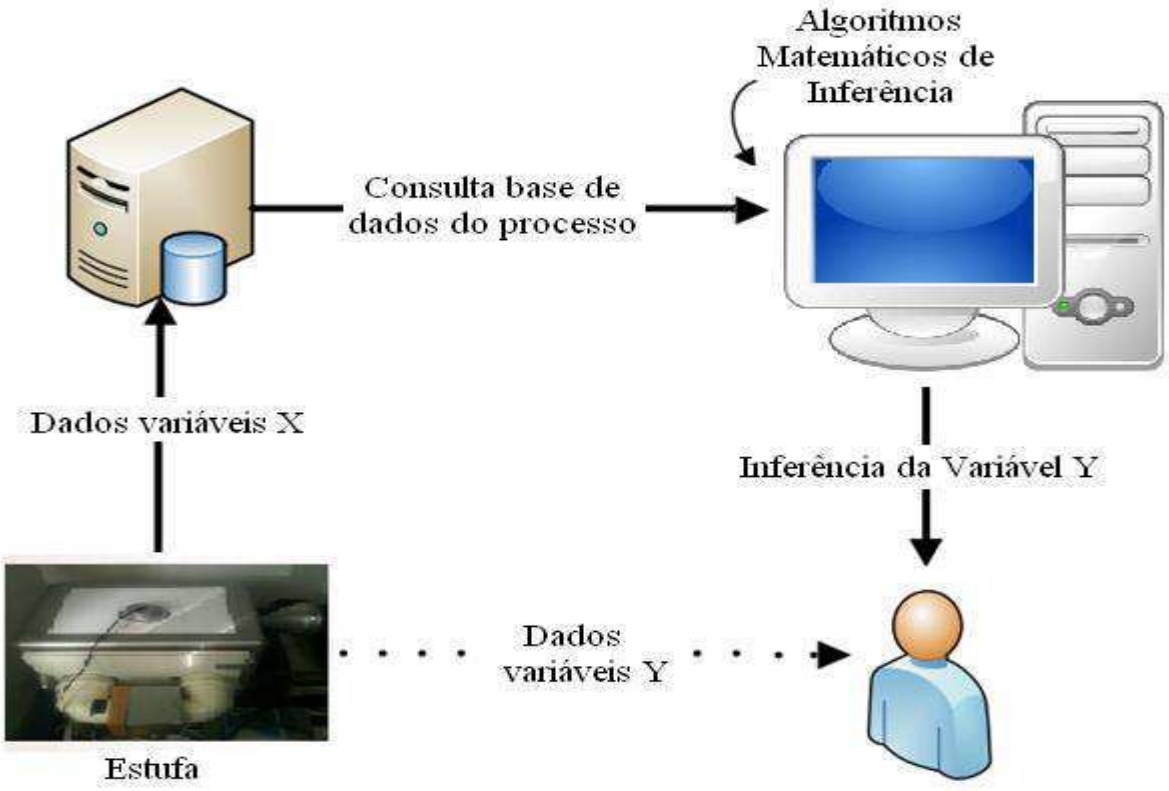


Figura 2.1 – Diagrama do Processo de Funcionamento do Sensor Virtual.

Aplicações de sensores virtuais estão aumentando com sua crescente popularidade. Isso é devido principalmente ao seu custo relativamente baixo em comparação com os sensores reais, quando se trata de grandes processos industriais, onde as plantas de processamento são em geral fortemente instrumentadas, com um grande número de sensores. Existem várias aplicações de sensores virtuais, particularmente na indústria, onde os sensores reais necessitam de aquisição, instalação, operação e custos de manutenção. Sensores virtuais estão se tornando ferramentas de rotina não só como uma fonte informativa para operadores das salas de controle, mas também no controle inferencial de circuitos fechados e esquemas adaptativos [8].

Muitas vezes é impraticável ter processos completamente monitorizados, para isso é necessário a construção de plantas fortemente instrumentadas por um custo elevado. Por esta razão, sensores virtuais provaram ser uma alternativa valiosa a esta solução. O monitoramento de processos possibilita o diagnóstico de problemas, isto é particularmente útil em processos descontínuos (processos divididos em etapas). Em um processo industrial, por exemplo, o diagnóstico de problemas reduz o número de produtos rejeitados, permite a detecção de falhas antes que o produto em questão esteja concluído, e ajuda a prevenir falhas nos produtos subsequentes [9].

Amazouz [10] apresentou outro exemplo do uso de sensores virtuais em monitoramento de processos. O sensor virtual desenvolvido foi baseado em técnicas estatísticas multivariadas, análise de componentes principais (PCA - *Principal Component Analysis*) e mínimos quadrados parciais (PLS - *Partial Least Quares*) para monitorar a secagem da madeira serrada (processamento em lote). O estudo comprovou a aplicabilidade dos métodos de estatística multivariada como uma ferramenta de mineração de dados para desempenho de monitoramento. O método mostrou ser bastante útil para detectar anormalidades e diagnosticar falhas em processos. Segundo os autores, a implementação *online* de tal ferramenta poderia aumentar a produtividade, poupar energia e planejar a manutenção. Uma base de dados de falhas e diagnósticos conhecida e um sistema para diagnóstico automático de falhas estão em desenvolvimento.

Reda [11], para monitorar as temperaturas e as localizações de *hot spots* (pontos aquecidos) em processadores reais, propôs uma técnica computacional de sensor virtual que utiliza as medições dos sensores reais para calcular de forma otimizada as temperaturas onde não há sensores embutidos. A técnica com sensor virtual pode melhorar a resolução de monitoramento térmico e contornar restrições de projeto sobre a colocação do sensor.

Zanata [4] desenvolveu um sensor virtual, com base em redes neurais artificiais para estimar, em tempo real, a composição dos produtos no topo de uma coluna de destilação multicomponente com condensador parcial, a partir de informações do tipo temperaturas e pressões em diversos pontos da coluna e vazões de entrada, de saída e de reciclo. Uma grande contribuição do trabalho foi o estudo realizado sobre os principais erros que podem ocorrer neste tipo de sensores

que são raramente tratados em publicações científicas. É também proposta uma metodologia para detecção e correção destes erros que foram encontrados e que afetam o comportamento do sensor, alterando sua precisão e capacidade de ser utilizado em um controle inferencial da planta.

Thanayankizil [12] descreve um sistema de gerenciamento de energia (chamado *softgreen*) para poupar energia em construções comerciais, com base na informação recolhida a partir de fontes de contextos pré-existentes. É realizado um perfil dos espaços do escritório, e detectada a área onde a ocupação pode ser realizada por sensores virtuais. Também foi implementado um modelo genérico de cargas elétricas baseado nos níveis de ocupações detectadas, que permite o cálculo da economia de energia.

Júnior [13] desenvolveu um sensor virtual aplicando modelos de inferência da quantidade de oxigênio consumido (OC) no processo de pré-dosagem de oxidantes de uma estação de tratamento de água (ETA), correspondendo a sensores de medição implementados em *software*, que possuem a capacidade de estimar, em tempo real, o oxigênio consumido, a partir de medidas de pH (Potencial Hidrogeniônico) e turbidez da água em tratamento.

Santos [14] propôs um sensor virtual capaz de estimar o índice de conforto térmico em vários pontos de um ambiente hipotético. O modelo do sensor implementa os principais fenômenos físicos observados na operação de um sistema de condicionamento de ar, de forma que as variáveis físicas obtidas por meio da simulação deste modelo apresentam precisão satisfatória quando comparadas com as variáveis que seriam efetivamente medidas em uma situação real. Neste trabalho também foi discutida a aplicação do sensor virtual para que seu uso seja viável em instalações de condicionamento de ar de pequeno porte, por exemplo, uso doméstico, onde não há a possibilidade de se dispor de sistemas de sensoriamento mais complexos.

No capítulo seguinte é apresentada uma fundamentação teórica sobre pontos e aspectos importantes ao trabalho.

2.2 FUNDAMENTAÇÃO TEÓRICA

2.2.1 SENSORES VIRTUAIS

Um sensor pode ser definido como sendo um dispositivo que responde a um estímulo físico, como, por exemplo, calor, luz, som, pressão, dentre outros, e transmite como resultado um sinal proporcional à quantidade medida. Os sensores surgem ante a necessidade de quantificar os fenômenos físicos na natureza para a geração de informação ou para controle de processos, que o ser humano, em alguns casos, percebe através dos seus próprios sentidos de forma qualitativa [15].

Sensores virtuais são algoritmos matemáticos implementados em software e executados por meio de um sistema microcontrolado, capazes de estimar, em tempo real, variáveis de interesse não medidas, a partir de outras variáveis disponíveis medidas no processo [16].

Já é bastante antiga a utilização do conceito de sensores virtuais e está ligada a modelagem matemática de processos com abordagem na predição de propriedades a partir de dados da planta. Os primeiros sensores virtuais surgiram a partir de sistemas de controle indiretos, nos quais a variável a ser controlada (primária) era controlada através do comportamento de outras variáveis (secundárias). Um dos primeiros sensores desenvolvidos é o estimador inferencial de *Brosilow*, apresentado no final da década de 70 [4].

A ideia principal em torno da utilização de um sensor virtual é a existência de situações industriais nas quais haja algum impedimento ao uso de sensores reais. Esse impedimento pode ser causado, por exemplo, por fatores como inexistência do sensor propriamente dito para uma variável específica, alto custo, imprecisão proibitiva à utilização desejada, tempo de resposta muito alto para aplicações em tempo real (controle) e impossibilidade de acesso ao local necessário para a instalação do sensor [17].

O sensor virtual pode ser considerado como o resultado da intersecção das técnicas de modelagem e identificação de sistemas e da tecnologia de instrumentos inteligentes, instrumentos que, aliado a sistemas digitais como microprocessadores ou microcontroladores, modificam seu comportamento, manipulando computacionalmente as informações medidas para melhor se adaptar à coleta e manipulação dos dados de um processo transmitindo-os da melhor forma possível [18].

Essa associação entre sensor e modelo é a ideia fundamental do sensor virtual. Podemos separar essa ideia em duas partes distintas: a parte do sensor real ou conjunto desses sensores, que adquirem variáveis relacionadas com a variável desejada e a parte do sensor virtual, o modelo desta relação, geralmente implementado em *software*, capaz de fornecer, por meio de simulação, uma estimativa da variável desejada [18].

Os sensores virtuais possuem um grande número de propriedades atrativas, dentre elas podem ser destacadas as seguintes [13]:

• representam uma alternativa de baixo custo frente aos caros dispositivos tradicionais de medição, permitindo a implementação de mais redes de monitoramento;

• podem trabalhar de forma paralela com sensores reais, dando informações úteis para a detecção de falhas, permitindo assim uma operação mais confiável do processo;

• podem facilmente ser implementados nos *hardware* existentes, e podem retornar os parâmetros do sistema quando se faz necessária alguma modificação;

• são capazes de fazer estimação de dados em tempo real, superando os atrasos introduzidos por meio dos sensores reais lentos, melhorando assim o desempenho das estratégias de controle.

Assim, o sensor virtual é uma boa alternativa em relação ao sensor real, quando as variáveis de entrada (secundárias) podem ser medidas sem problemas e quando a simulação do processo, geralmente responsável pelo fornecimento dos dados para a geração do modelo do sensor implementado em *software*, seja capaz de fornecer a variável desejada [19].

2.2.2 CONTROLE DE TEMPERATURA

O controle automático surgiu com a necessidade de se obter desempenhos cada vez melhores de equipamentos e sistemas industriais, com vistas à obtenção de melhores produtos a custos menores. Desde o controlador centrífugo, desenvolvido para controle de velocidade de máquinas a vapor, os cientistas e engenheiros vêm trabalhando no propósito de desenvolver novas técnicas e equipamentos de controle [20].

Controles tradicionais de sistemas, neste caso, controle de temperatura, são em geral baseados em modelos matemáticos que descrevem o sistema de controle usando equações diferenciais que definem a dinâmica da resposta do sistema para as entradas. Tais controladores são resultados de trabalhos teóricos e práticos desenvolvidos há décadas e são altamente eficazes [11]. Tem desempenhado um papel fundamental no avanço da engenharia e da ciência, além de ser de grande importância e parte integrante dos processos industriais e de produção. Por exemplo, é essencial no projeto de sistemas de piloto automático na indústria aeroespacial e no projeto de carros na indústria automotiva, como também em operações industriais como o controle de pressão, de temperatura, de umidade, de viscosidade e de vazão nos processos industriais [2].

Os avanços no controle automático, na teoria e na prática, vêm produzindo meios para otimizar o desempenho dos sistemas dinâmicos, melhorar a produtividade, diminuir o trabalho árduo de várias rotinas de operações manuais repetitivas, entre outros [2].

Em um sistema de controle, a variável controlada é a grandeza ou condição que precisa ser controlada, como temperatura, pressão ou fluido; e a variável manipulada é a grandeza ou a condição modificada pelo controlador, de modo que afete o valor da variável controlada [2]. Um controlador automático compara o valor real de saída da planta com a entrada de referência (valor desejado), determina o erro e produz um sinal de controle que vai reduzir o erro a zero ou a um valor pequeno e aceitável. A maneira pela qual o controlador automático produz o sinal de controle é chamada de lei de controle. Na Figura 2.2 apresenta-se um diagrama de blocos de um sistema de controle em malha fechada industrial, o qual consiste em um controlador automático, um atuador, uma planta e um sensor (elemento de medida).

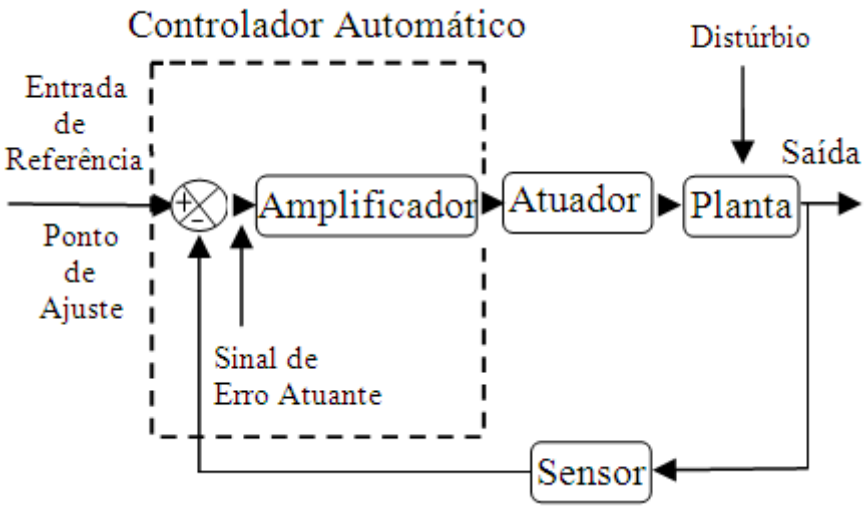


Figura 2.2 – Diagrama de Blocos Geral de um Sistema de Controle em Malha Fechada.

O controlador detecta o sinal de erro atuante e determina o sinal de controle por meio da lei de controle. A saída de um controlador automático alimenta um atuador, que produz o sinal de entrada da planta, de acordo com o sinal de controle, de tal modo que a saída se aproxime do sinal de entrada de referência [2].

O sensor, ou elemento de medição, é um dispositivo que converte a variável de saída em outra variável conveniente, como deslocamento, pressão ou tensão, que pode ser utilizada para comparar a saída ao sinal de entrada de referência. Esse elemento está na malha de realimentação do sistema de malha fechada. O ponto de ajuste do controlador deve ser convertido em um sinal de referência com as mesmas unidades do sinal de realimentação que vem do sensor ou do elemento de medição [2].

Além das variáveis controlada e manipulada, de interesse direto para o controle do processo, existem outras variáveis que influenciam no processo. Essas variáveis, que afetam o desempenho do processo, podem ser chamadas, de um modo genérico, de distúrbios ou de carga do processo. Como o seu controle direto é difícil, deve-se aprender a conviver com elas e ajustar o sistema para compensar a sua influência [21].

A escolha do tipo de controlador a ser utilizado deve ser decidida com base na natureza da planta e nas condições de operação, incluindo certas considerações, como segurança, disponibilidade, confiabilidade, precisão, peso e tamanho [2].

Diversas técnicas de controle podem ser empregadas em um sistema, mas a mais usada é a do controle PID (combinação das ações de controle Proporcional, Integral e Derivativa). O objetivo é aproveitar as características particulares de cada uma destas ações a fim de se obter uma melhora significativa do comportamento transitório e em regime permanente do sistema controlado [22]. Sua popularidade de deve ao fato de serem facilmente implementáveis, de baixo custo, robustos e

versáteis, com a capacidade de fornecer comportamentos transitório e de regime permanente satisfatórios para uma grande variedade de processos encontrados na indústria [23].

Com relação aos controladores P (Proporcional), PI (Proporcional-Integral), PD (Proporcional-Derivativo) e PID (Proporcional-Integral-Derivativo) temos:

Controlador Proporcional (P)

No controle proporcional (P), o sinal de controle, ou ação do controlador é proporcional ao sinal do erro *e(t)*, ou seja, proporcional à diferença entre o valor ideal e o valor atual da variável controlada. Quanto maior o erro, maior será a ação do controle [24].

O erro é definido na Equação (1):

$$e\,(t) = r(t) - \,y(t) \tag{1}$$

onde $r(t)$ é o sinal de referência e $y(t)$ o sinal medido por meio dos sensores. As representações do controle proporcional, no domínio do tempo e em Laplace, são dadas, respectivamente, pelas equações (2) e (3):

$$u(t) = k_p\, e(t) \tag{2}$$

$$U(s) = k_p\, E(s) \tag{3}$$

onde k_p é o ganho do controlador proporcional.

Controlador Proporcional Integral (PI)

Considerando que a saída do controlador é agora função do erro e da integral do erro, temos um controlador proporcional e integral (PI), em que a atuação é dada pela soma das ações proporcional e integral. A representação no domínio do tempo é dada pela equação (4):

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau \tag{4}$$

onde k_i é o ganho do controlador integral.

Expressando o ganho integral k_i em termos do tempo integral, T_i, dado pela relação entre k_p e k_i, temos a Equação (5):

$$u(t) = k_p e(t) + \frac{k_p}{T_i}\int_0^t e(\tau)d\tau \quad (5)$$

A representação em Laplace é dada pela Equação (6):

$$U(s) = k_p\left(1 + \frac{1}{T_i s}\right)E(s) \quad (6)$$

em que T_i (Tempo integrativo) é o tempo necessário para que a contribuição da ação integral iguale a da ação proporcional quando o erro é um degrau, e é expresso em segundos ou minutos. Quanto maior a ponderação da ação integral, ou seja, quanto maior for $\frac{k_p}{Ti}$ o sistema tende a apresentar comportamento mais oscilatório e apresentar um *overshoot* mais elevado [23].

Proporcional Derivativo (PD)

A ação do controle derivativo é proporcional à variação do erro, isto é, quanto maior for a taxa de variação do erro, ou a velocidade com que o erro varia, maior será a ação derivativa. O controle PD agrupa o controle proporcional, adicionado ao controle derivativo [24].

Sua representação no domínio do tempo e em Laplace é apresentada, respectivamente, nas Equações (7) e (8):

$$u(t) = k_p e(t) + k_p T_{td}\frac{d}{dt}e(t) \quad (7)$$

$$U(s) = k_p(1 + T_{td}s)E(s) \quad (8)$$

em que T_{td} (Tempo derivativo), é o período de tempo antecipado pela ação derivativa relativamente à ação proporcional e é expresso em segundos ou minutos.

Proporcional Integral Derivativo (PID)

Este modo resulta da combinação dos modos proporcional, integral e derivativo. Pode-se afirmar que resulta num compromisso entre as vantagens e desvantagens de um PI e as vantagens de um PD [25]. As representações no domínio do tempo e em Laplace são dadas, respectivamente, pelas Equações (9) e (10):

$$u(t) = k_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_{td}\frac{d}{dt}e(t)] \quad (9)$$

$$U(s) = k_p\left(1 + T_{td}s + \frac{1}{T_i s}\right)E(s) \quad (10)$$

Neste tipo de controlador, o modo integral é usado para eliminar o erro estacionário causado por grandes variações de carga. O modo derivativo, com o seu efeito estabilizador (um sistema linear será definido como estável se todos os pólos da função de transferência do sistema tiverem partes reais negativas), permite um aumento do ganho e reduz a tendência para as oscilações, o que conduz a uma velocidade de resposta superior quando comparado com os controladores P e PI. No entanto, estas propriedades assumem um caráter geral, pelo que podem existir exceções em determinados sistemas [25].

Os controladores PID são estudados desde o início do século passado e são utilizados em 90% das aplicações práticas pela facilidade de implementação e de sua flexibilidade. Em particular, quando o modelo matemático da planta não é conhecido e, portanto, métodos de projeto analítico não podem ser utilizados, controles PID se mostram os mais úteis [2].

Exemplos de sistemas que utilizam controle PID são navios, aeronaves, mísseis e satélites. O controle PID é também extensivamente utilizado na indústria, para controle de processos (mecânicos, térmicos, hidráulicos, elétricos e eletrônicos) [24].

Antes de se obter os parâmetros do controlador, é necessário realizar a modelagem e a identificação do sistema, que é uma forma de buscar a representação do comportamento do processo definindo um melhor modelo que o represente. No tópico seguinte é apresentada a teoria sobre modelagem e identificação de sistemas.

2.2.3 Modelagem e Identificação de Sistemas

Um modelo matemático de um sistema real é um análogo matemático que representa algumas das características observadas em tal sistema. Evidentemente, há outros tipos de modelos além dos modelos matemáticos. Por exemplo, a maquete de um edifício também é um modelo, ainda que não seja matemático. É importante perceber que uma maquete possui algumas das características da construção real, mas não todas. A mesma observação é válida para os modelos matemáticos [26].

Esses modelos têm sido utilizados para os mais diversos fins, como, tentar entender e explicar fenômenos que ocorrem, por exemplo, na natureza, em sistemas sociais, biomédicos; sistemas de controle e monitoração; estimação; simulação e treinamento, como, por exemplo, os simuladores de voo [26].

O problema de elaborar modelos matemáticos de sistemas dinâmicos baseado em observações destes sistemas pode ser entendido como um problema da área da engenharia de controle denominada identificação de sistemas [27].

A identificação de sistemas é o método utilizado para buscar a representação do comportamento do processo, pelo menos em parte e de forma aproximada, por meio de um modelo matemático independente do conhecimento prévio a respeito do mesmo [28].

Na Figura 2.3 está ilustrado um processo de modelagem simples. Estritamente falando, um sistema é "algo real" (como um amplificador, um carro ou um corpo humano), enquanto que o modelo é uma "abstração" (um conjunto de equações matemáticas). Pode-se observar que o modelo do processo não é uma cópia real do sistema, e sim uma aproximação do seu comportamento, simulando o sistema como um todo [29].

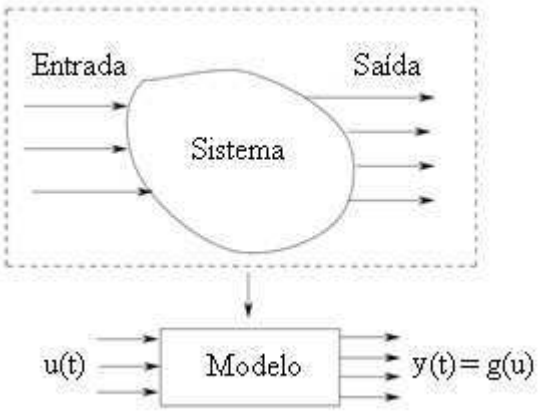


Figura 2.3 – Representação do Processo de Modelagem Simples [29].

A utilização do modelo para simulação do sistema constitui-se em um procedimento de baixo custo e seguro para experimentar o sistema. Entretanto, a validade (adequação) dos resultados de simulação depende completamente da qualidade do modelo matemático do sistema [30].

O MATLAB dispõe de uma ferramenta que faz a identificação computacional de sistemas, chamada *ident*. Com essa ferramenta é possível encontrar modelos matemáticos de sistemas dinâmicos utilizando os dados obtidos da entrada e saída do sistema.

Embora sistemas de controle possam ter ordem elevada, muitos sistemas reais apresentam dominância de primeira ou segunda ordem. Com frequência estes modelos simples são suficientes para realizar um primeiro projeto de controle, ou seja, embora a função de transferência que representa o sistema tenha ordem elevada, pode-se usar um modelo de primeira ou segunda ordem para representá-lo [31].

Sistemas de Primeira Ordem

Considere a função de transferência de primeira ordem com atraso apresentada na Equação (11):

$$\frac{Y(s)}{U(s)} = G(s) = \frac{k}{1 + sT_p} e^{-sT_d} \quad (11)$$

onde *G(s)* é a função de transferência do sistema, *k* é o ganho em regime permanente do sistema, T_d é o atraso de transporte e T_p é a constante de tempo.

O atraso de transporte é definido como o intervalo de tempo decorrido entre a ocorrência de uma perturbação na entrada e o início da resposta do sistema devido a essa ocorrência na entrada [26]. A resposta ao degrau no tempo de sistemas de primeira ordem com atraso de transporte pode ser observada na Figura 2.4.

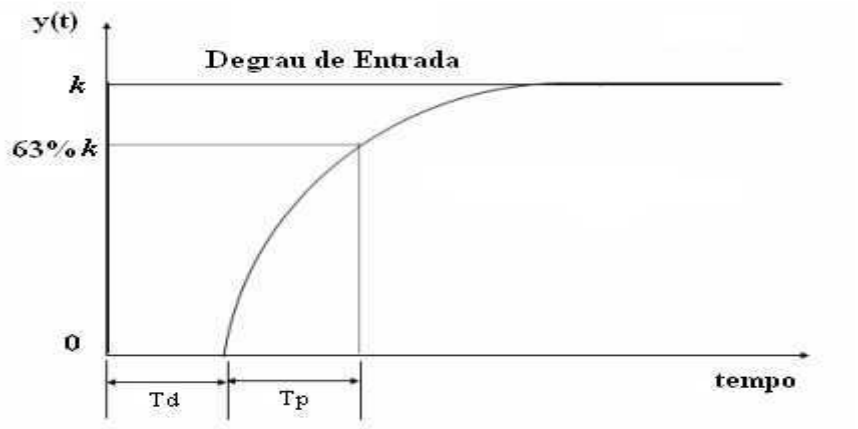


Figura 2.4 – Resposta de um Sistema de Primeira Ordem com Atraso [21].

A resposta atinge 63,2% do seu valor de regime permanente em t = T_d+T_p (Atraso mais constante de tempo).

Sistemas de Segunda Ordem

Considere a função de transferência de segunda ordem padrão com atraso, apresentada na Equação (12):

$$G(s) = \frac{\mathrm{k}e^{-sT_d}}{(\mathrm{T_{p1}s} + 1)(\mathrm{T_{p2}s} + 1)} \quad (12)$$

Onde T_{p1} e T_{p2} são, respectivamente, a primeira e segunda constante de tempo do sistema.

No capítulo seguinte é realizada a descrição da estufa e da plataforma de testes usada no trabalho, onde é apresentada uma representação da estufa utilizada, como também uma fotografia da estufa real. Neste capítulo também será comentado sobre o sensor de temperatura, a placa de aquisição de dados e o *software* da placa de aquisição utilizados no processo de aquisição dos dados.

3 DESCRIÇÃO DA ESTUFA E PLATAFORMA DE TESTES

A plataforma de testes é composta pelo ambiente com sensor, para adquirir os dados de temperatura, atuador, para receber um sinal proveniente do controlador e agir sobre o sistema controlado, placa de aquisição de dados para coletar os dados do sensor e *software* instalado no computador (interface homem-máquina) para fazer o tratamento dos dados e assim monitorar e controlar a temperatura da estufa.

3.1 ESTUFA

Uma representação da estufa utilizada no trabalho pode ser observada na Figura 3.1. Em seu interior foi colocada uma resistência no valor de 11 Ohms na qual é aplicada a tensão para o aquecimento do interior da estufa.

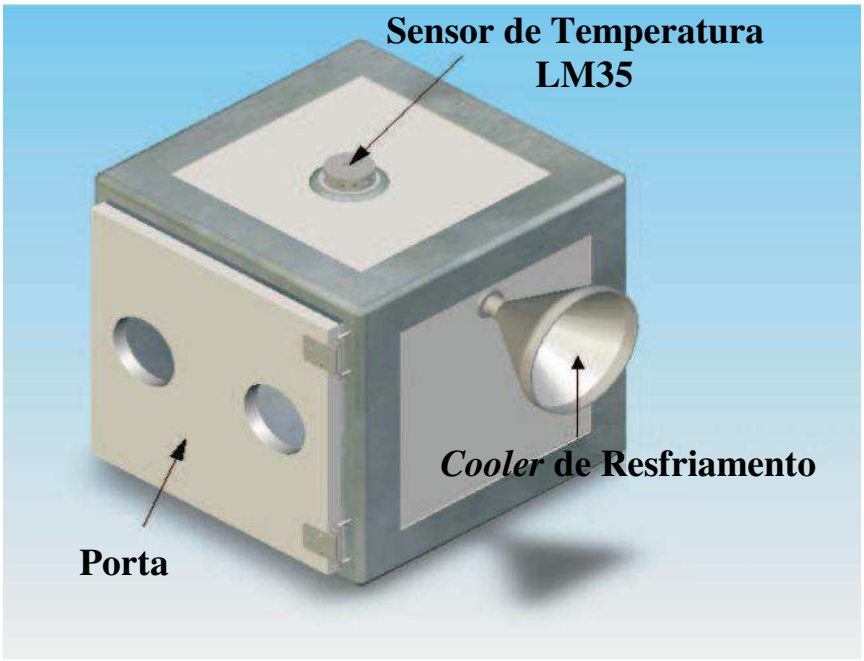


Figura 3.1 - Representação da Estufa usada no Sistema de Controle de Temperatura.

Encontra-se na lateral da estufa um *cooler* que tem como finalidade a refrigeração da mesma, ele atua retirando parte da massa de ar que é aquecida dentro da estufa, sua ação pode ser interpretada como uma nova entrada, por exemplo, uma perturbação no sistema, mas, não foi o caso, neste trabalho, a ação do *cooler* não foi interpretada como outra entrada do sistema, foi utilizada apenas entre um teste e outro para ajudar no resfriamento. Na parte superior, encontra-se

um ponto de medição de temperatura. O sensoriamento realizado neste ponto é feito por meio do sensor de temperatura LM35.

Na Figura 3.2 apresenta-se uma estrutura, que por meio de um *cooler*, proporciona a circulação do ar aquecido pela estufa. Os canos de PVC que são acoplados na porta da estufa permitem uma melhor distribuição da massa de ar quente no interior.

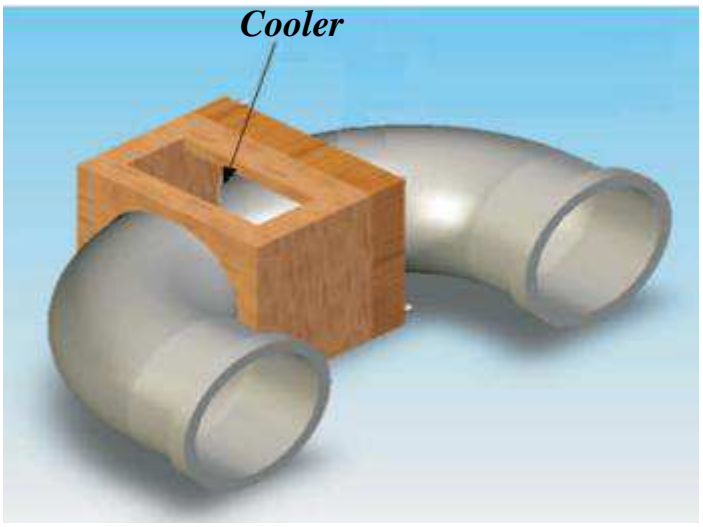


Figura 3.2 – Representação da Estrutura Acoplada à Porta da Estufa.

Na Figura 3.3 apresenta-se uma fotografia da estufa térmica utilizada no trabalho.

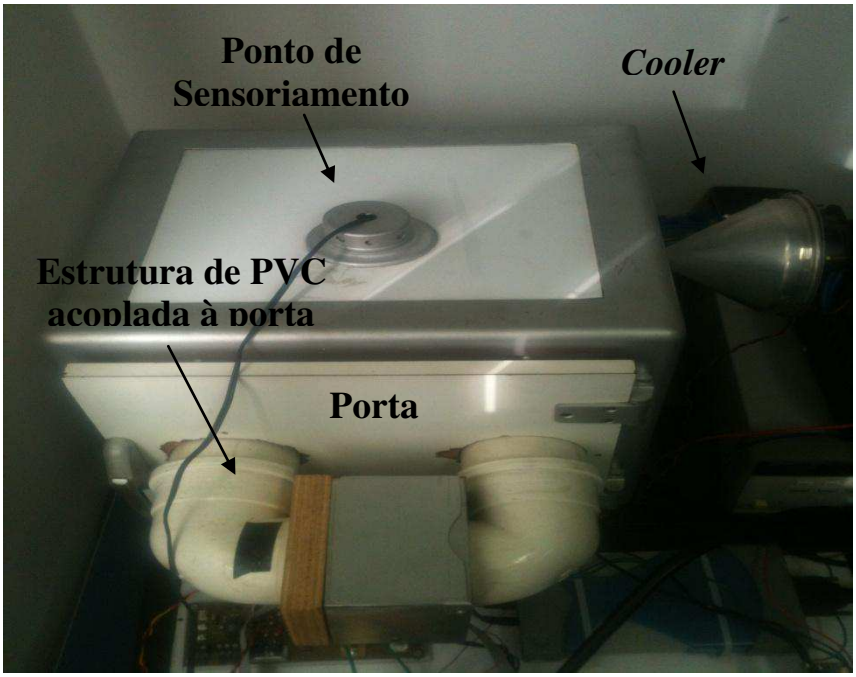


Figura 3.3 – Fotografia da Estufa Térmica.

3.2 SENSOR LM35

O sensor LM35 é um sensor de precisão, fabricado pela *National Semiconductor*, que apresenta uma saída de tensão linear relativa à temperatura em que ele se encontrar no momento em que for alimentado por uma tensão de 4-20V, tendo em sua saída um sinal de 10mV para cada grau Celsius de temperatura.

O LM35 não necessita de qualquer calibração externa ou "*trimming*" para fornecer com exatidão, valores de temperatura com variações de ¼ºC ou até mesmo ¾ºC dentro da faixa de temperatura de –55ºC à 150ºC. Este sensor tem saída com baixa impedância, tensão linear e calibração inerente precisa, fazendo com que o interfaceamento de leitura seja especificamente simples, barateando todo o sistema em função disto. Este sensor poderá ser alimentado com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas independentemente disso, a saída continuará sendo de 10mV/ºC [32].

3.3 CIRCUITO DE MEDIÇÃO UTILIZANDO O LM35

Os projetos de monitoramento de temperatura utilizando o LM35 operam de maneira simples e compacta, são basicamente constituídos de um sensor emissor de sinal, neste caso o LM35, um amplificador operacional que eleva o sinal do sensor, e um circuito decodificador deste sinal (um microcontrolador, sistema de aquisição de dados, *driver*), que recebe o sinal amplificado e atua de alguma forma, simplesmente mostrando o dado num *display*, ou processando a informação e atuando num transdutor. O circuito nada mais é do que um estágio de amplificação para se aproveitar, por exemplo, o máximo da faixa de resolução de um conversor analógico-digital de um sistema de aquisição de dados.

Na Figura 3.4 é apresentado o circuito utilizado na medição utilizando o sensor LM35. Neste circuito, no monitoramento do processo, para identificar o momento em que o sinal do sensor LM35 é perdido, monitora-se a tensão sobre o capacitor C1. Quando a tensão deste capacitor diminui para um valor menor que 0,2V, o sinal do sensor LM35 é perdido e a rotina implementada informa a duração do processo e se a temperatura está ou não estabilizada, caso esteja, infere o valor da temperatura no interior da estufa.

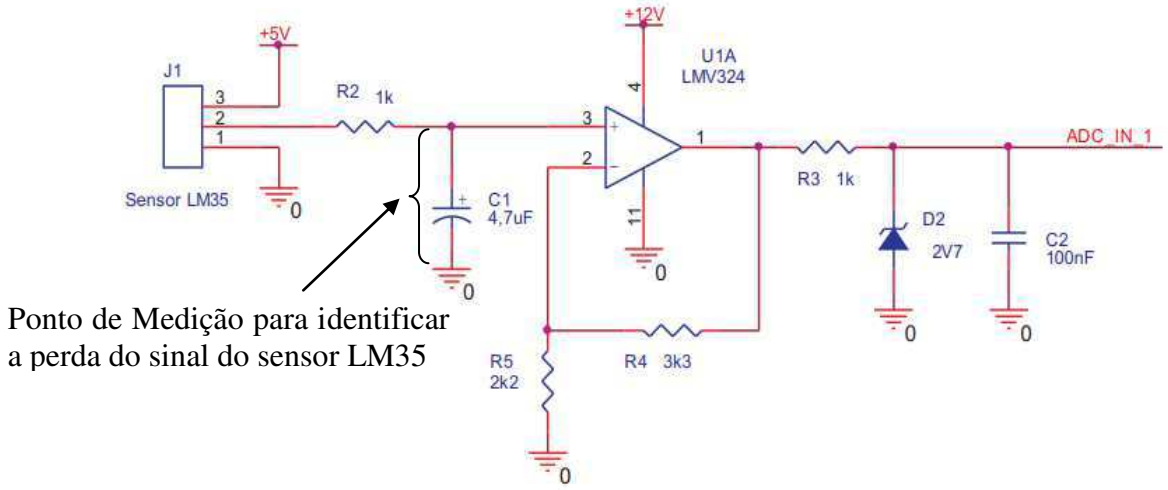


Figura 3.4 – Diagrama Esquemático do Circuito de Medição de Temperatura Usando LM35.

Além do estágio de amplificação o circuito possui também estágios de filtragem antes e após a amplificação. Estes estágios auxiliam atenuando ruídos que venham a diminuir a precisão dos valores de temperatura obtidos no processo de medição.

3.4 SISTEMA DE AQUISIÇÃO DE DADOS

O dispositivo de *hardware* utilizado neste trabalho para efetuar a aquisição dos valores de temperatura é uma placa de aquisição de dados DAQ série U2500A da *Agilent Technologies*. Os dispositivos DAQ série U2500A são compatíveis com inúmeros ambientes de desenvolvimento de aplicativos (ADA), como *Agilent* VEE, LabVIEW, MATLAB e *Microsoft Visual Studio* [33]. Na Figura 3.5 é apresentada uma visão geral do bloco do terminal (vista frontal) do dispositivo.

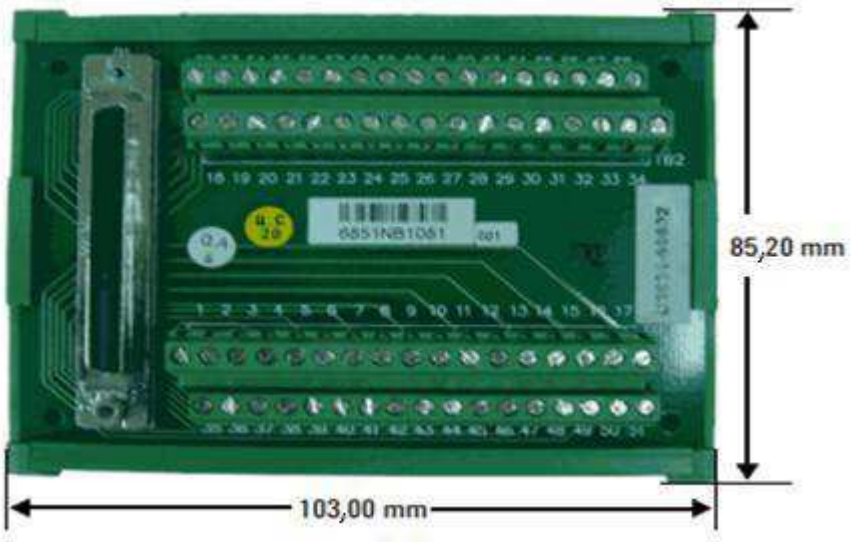


Figura 3.5 – Visão Geral do Bloco de Terminal (Vista Frontal).

O *software* utilizado nos testes para aquisição dos dados foi o *Agilent Measurement Manager*, que é um ambiente fornecido junto com o dispositivo. Na Figura 3.6 é apresentada a interface do *software*.

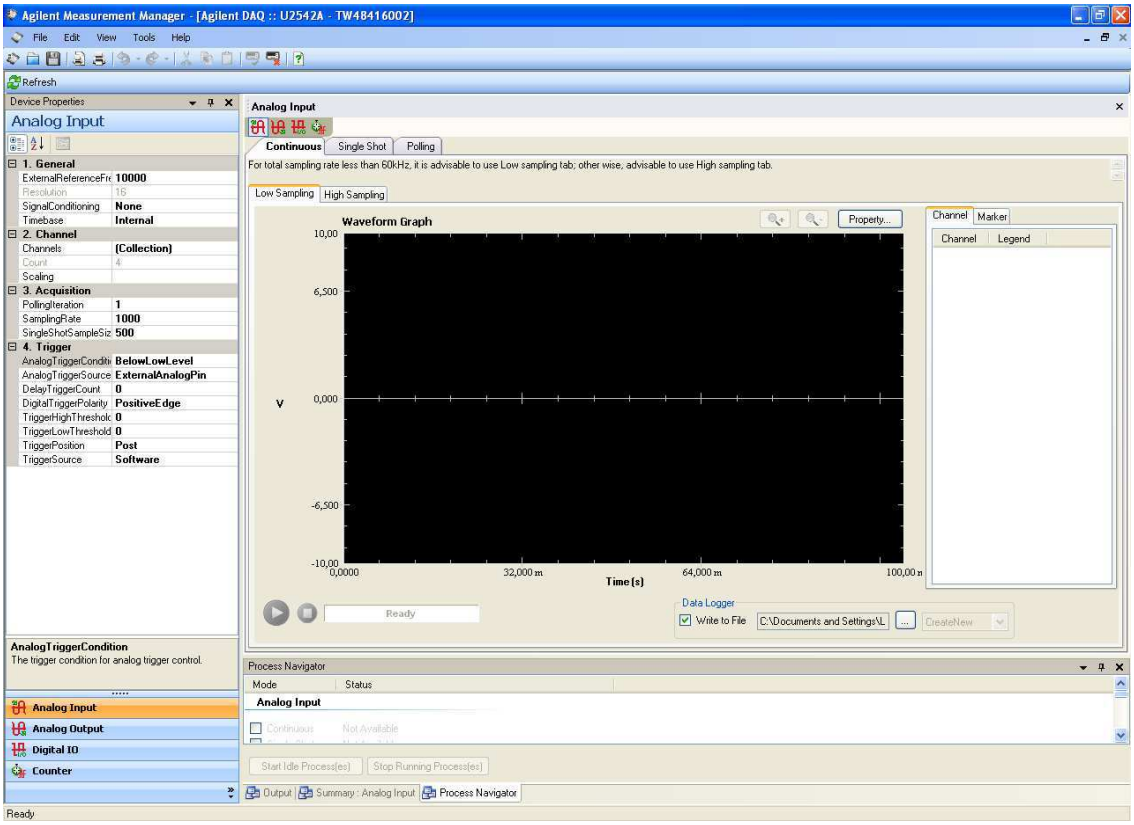


Figura 3.6 – Tela da Interface do *Software* Utilizado na Aquisição dos Dados.

O dispositivo utilizado apresenta as seguintes características [33]:

- amostragem simultânea de até 2 MSa/s por canal;
- 4 canais de entrada analógicos com resolução de 14 a 16 *bits*;
- 2 canais de saída analógicos;
- entrada e saída digitais programáveis de 24 *bits*;
- capacidade para funcionamento autônomo ou modular;
- instalação e configuração do tipo *plug and play*;
- *software* incluso.

No próximo Capítulo serão apresentados os experimentos realizados e os resultados obtidos.

4 EXPERIMENTOS E RESULTADOS

Neste Capítulo são apresentadas as medições realizadas, a identificação dos modelos matemáticos da planta e o projeto dos controladores para atuar na estufa, comentando sobre os métodos SIMC (Skogestad - *Internal Model Control*) e CHR (Chien, Hrones e Reswick), os índices de desempenho utilizados na classificação do melhor controlador, as interfaces gráficas criadas para uma melhor interação com o usuário e, por fim, as simulações e os resultados obtidos no trabalho.

4.1 MEDIÇÕES E TRATAMENTO ESTATÍSTICO

Inicialmente foram feitas as medições, com o intuito de se obter uma relação dos valores de temperatura no interior da estufa e tensão aplicada na resistência para a realização do monitoramento da temperatura da estufa.

Para cada tensão aplicada na resistência, obtém-se uma temperatura correspondente e verifica-se o tempo médio entre as medições (variações na temperatura), que neste caso é em torno de 60 minutos, tendo assim uma referência de que valor de temperatura esperar de acordo com esse tempo médio e com a tensão aplicada. A ideia é que se por algum motivo a medição de temperatura falhar, ser possível inferir um valor de temperatura fundamentado no valor de tensão que esta sendo aplicado para evitar que, por exemplo, um processo industrial seja paralisado.

Variou-se a tensão de 0 V a 5 V, com um passo de 0,5 V, de modo que 0 V está vinculada a temperatura inicial dentro da estufa e 5 V a temperatura máxima alcançada pela estufa. Os testes foram repetidos cinco vezes, número de vezes suficiente para o critério de parada estabelecido para o trabalho, que foi o valor do desvio padrão menor que 1 (um). Os valores obtidos nas medições são apresentados na Tabela 4.1.

Tabela 4.1- Resultados dos Testes Realizados.

Tensão (V)	**Temperatura (°C)**				
	Teste 1	**Teste 2**	**Teste 3**	**Teste 4**	**Teste 5**
0,0	26,5	26,4	26,5	28,5	27,6
0,5	28,8	29,0	28,6	29,2	29,0
1,0	30,8	29,9	30,4	30,5	31,2
1,5	34,4	34,3	34,9	34,9	34,7
2,0	38,9	38,6	38,9	39,0	39,2
2,5	43,5	43,9	44,1	44,3	43,9
3,0	53,6	51,7	53,2	52,3	52,5
3,5	64,7	64,0	64,6	64,2	64,3
4,0	68,8	69,6	71,1	70,2	70,6
4,5	72,4	71,8	71,1	71,3	71,0
5,0	72,4	72,2	72,7	72,4	71,5

Após as medições, foi realizado o tratamento estatístico com os valores obtidos para se ter uma faixa de confiança (variações até 2º C), assim, foi feito o cálculo da média e do desvio padrão dos valores. A Equação (13) utilizada para o cálculo do desvio padrão é apresentada a seguir:

$$Desvio\ padrão = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n-1}} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}} \qquad (13)$$

Onde $\bar{x}$ é a média, x_i é o valor individual e n é o número de valores.

Na Tabela 4.2 estão apresentados os valores médios e o desvio padrão obtidos após o tratamento estatístico.

Tabela 4.2- Valores Médios e Desvio Padrão dos Dados.

Tensão (V)	**Temperatura Média (°C)**	**Desvio Padrão**
0,0	26,50	0,92
0,5	29,00	0,23
1,0	30,50	0,48
1,5	34,70	0,27
2,0	38,90	0,22
2,5	43,90	0,30
3,0	52,50	0,75
3,5	64,30	0,29
4,0	70,20	0,89
4,5	71,30	0,58
5,0	72,40	0,45

Pode ser observado que o maior valor de desvio padrão calculado foi de 0,92, menor que 1 (um), comprovando que a realização de cinco testes são suficientes para o critério de parada estabelecido.

4.2 GERAÇÃO E VALIDAÇÃO DOS MODELOS MATEMÁTICOS

Nesta seção são apresentadas a geração dos modelos matemáticos e a validação dos mesmos, respectivamente.

4.2.1 GERAÇÃO DOS MODELOS MATEMÁTICOS

Com os dados obtidos foi possível realizar a geração dos modelos. O MATLAB dispõe de uma ferramenta que faz a identificação computacional de sistemas, chamada *ident*. Essa ferramenta

constrói modelos matemáticos de sistemas dinâmicos utilizando os dados obtidos da entrada e saída do sistema. Na Figura 4.1 está apresentada a interface da ferramenta *ident*.

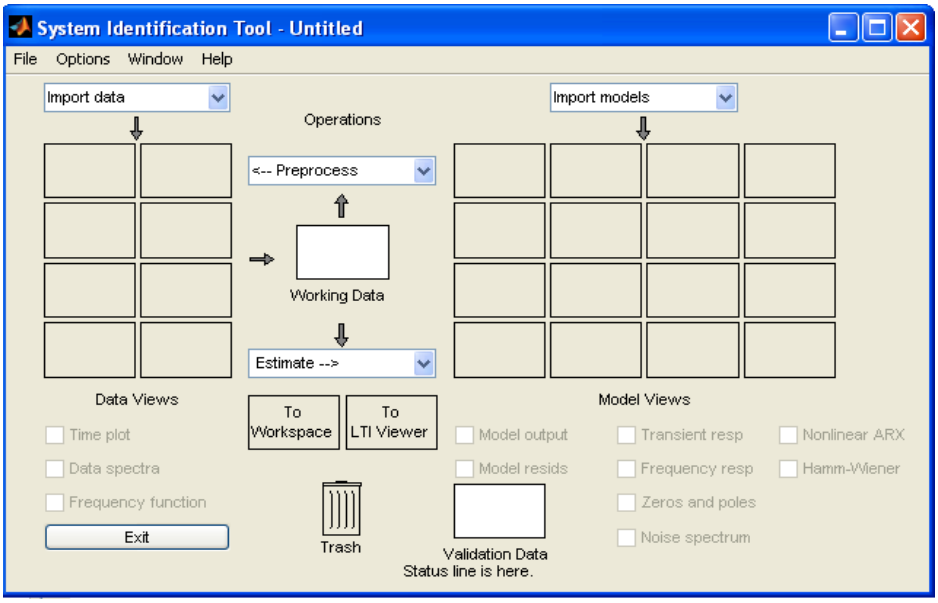


Figura 4.1 – Tela da Interface da Ferramenta *Ident*.

Uma vez carregados os dados no MATLAB, partiu-se para identificação do sistema. Na Figura 4.2 é apresentado o ambiente onde se define a entrada (tensão) e a saída (temperatura) do sistema, que são os dados obtidos com os testes onde se aplicou a tensão e adquiriram-se os valores de temperatura, levando em consideração o tempo de duração para a estabilização da temperatura.

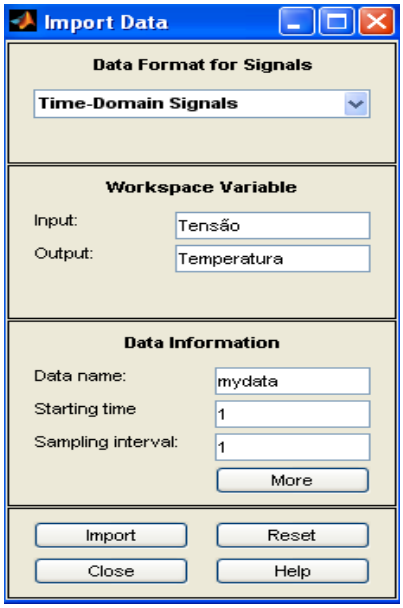


Figura 4.2 – Tela do Ambiente Onde se Define Entrada e Saída para Identificação do Sistema.

Após definir a entrada e a saída do sistema, deram-se início ao reconhecimento da função de transferência. Utilizando a ferramenta *ident* do MATLAB foram gerados modelos de primeira e

segunda ordem com atraso de transporte para o mesmo conjunto de dados. Os modelos de primeira e segunda ordem com atrasos obtidos são apresentados, respectivamente, nas equações (14) e (15):

$$G_1(s) = \frac{ke^{-sT_d}}{T_p s + 1} = \frac{17{,}762e^{-0{,}983s}}{1618{,}1s + 1} \tag{14}$$

$$G(s) = \frac{ke^{-sT_d}}{(T_{p1}s + 1)(T_{p2}s + 1)} = \frac{17{,}75e^{-0{,}95s}}{(1605{,}1s + 1)(44{,}326s + 1)} \tag{15}$$

Onde k é o ganho em regime permanente do sistema, T_d é o atraso de transporte , T_p é a constante de tempo no sistema de primeira ordem e T_{p1} e T_{p2} são as constantes de tempo no sistema de segunda ordem.

4.2.2 Validação dos Modelos Matemáticos

Neste trabalho, para validação dos modelos, também foi utilizada a ferramenta *ident* do MATLAB, que, para comparar a saída do modelo estimado com a saída real, ambas às curvas são plotadas com um índice que qualifica em porcentagem o modelo estimado. Foram utilizados testes diferentes para gerar o modelo e para fazer a comparação na validação. A curva para o modelo de primeira ordem pode ser observada no gráfico da Figura 4.3. O modelo se ajusta a saída aproximadamente 87,62%, mostrando que o modelo pode ser considerado satisfatório.

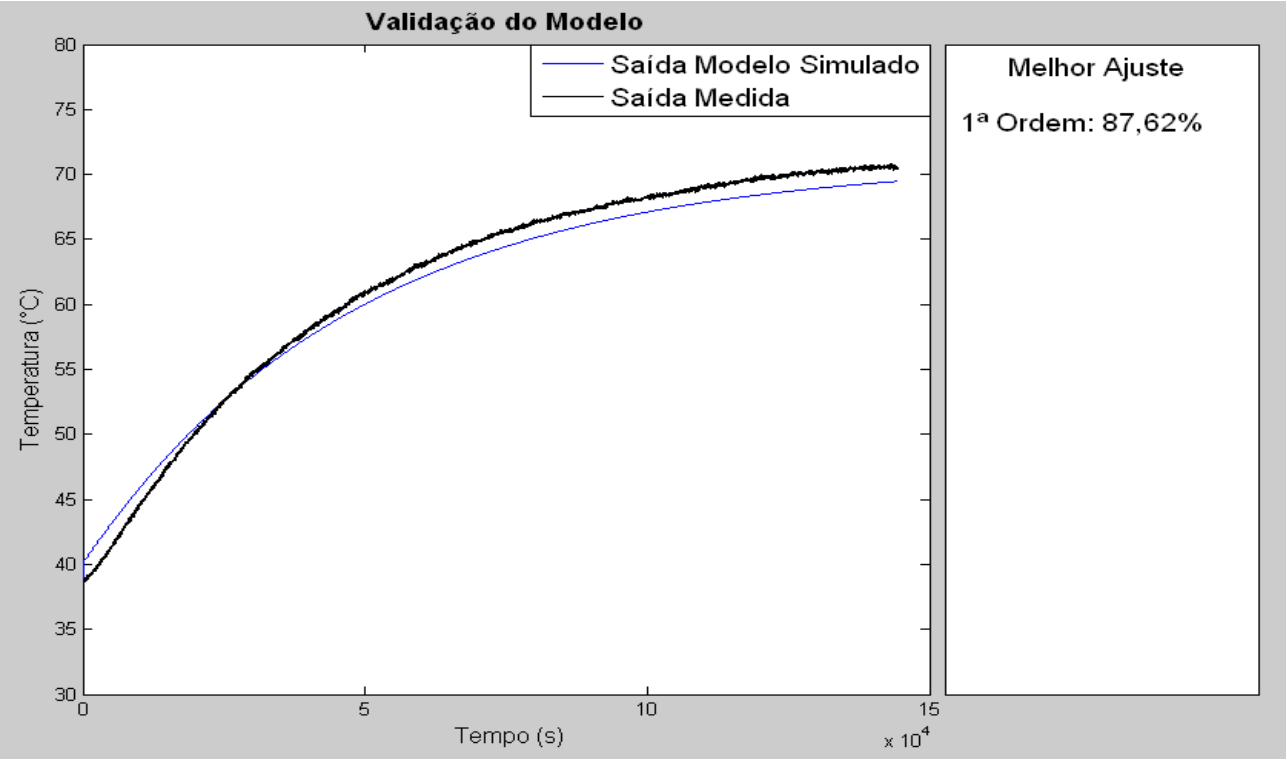


Figura 4.3 – Resultado da Validação do Modelo de Primeira Ordem.

A curva para o modelo de segunda ordem pode ser observada no gráfico da Figura 4.4. O modelo se ajusta a saída aproximadamente 88,07%, mostrando que o modelo de segunda ordem também pode ser considerado satisfatório.

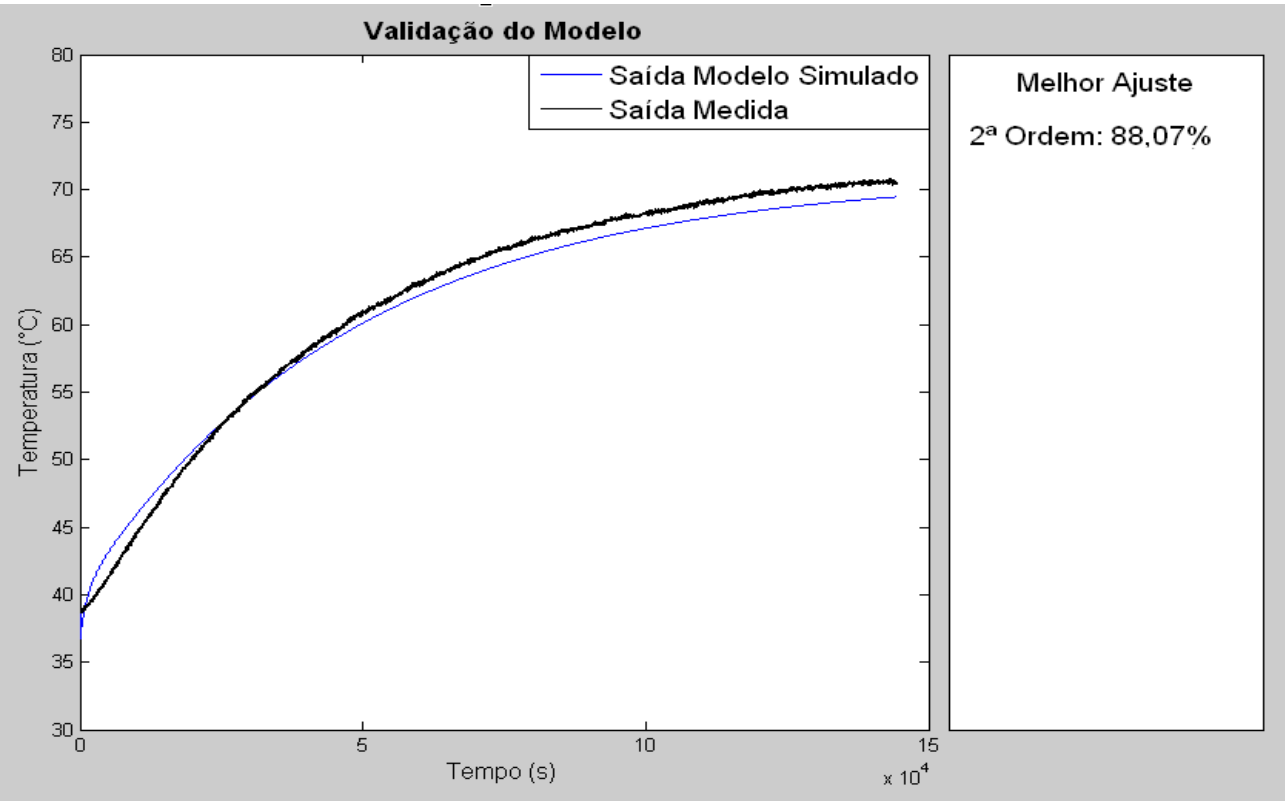


Figura 4.4 – Resultado da Validação do Modelo de Segunda Ordem.

4.3 PROJETO DOS CONTROLADORES

Uma vez realizada a identificação do sistema, parte-se para a próxima etapa, a realização do projeto dos controladores para atuar na estufa. Foram utilizados os métodos de sintonia propostos por *Skogestad* (SIMC) e por *Chien, Hrones e Reswick* (CHR) para o cálculo dos parâmetros: k_p (Ganho Proporcional), T_i (Constante de Tempo Integral) e T_{td} (Constante de Tempo Derivativa) dos controladores.

4.3.1 MÉTODO *SKOGESTAD* (SIMC)

O método SIMC, é sugerido em vez do método IMC (Controle por Modelo Interno) clássico, pois, utilizando um parâmetro de ajuste, T_c, e fazendo esse parâmetro igual ao atraso da planta (T_d), a resposta é mais rápida e possui uma boa rejeição a perturbação [34].

Assim, para ajustar o controlador de cada malha de controle em estudo, foram aplicadas as regras de ajuste associados com a ordem da planta, conforme apresentado na Tabela 4.3:

Tabela 4.3- Regra de Sintonia SIMC.

Planta	k_p	T_i	T_{td}
1ª Ordem	$\frac{1}{k}\frac{T_p}{T_c+T_d}$	Min$\{T_p, 4(T_c+T_d)\}$	-
2 ª Ordem	$\frac{1}{k}\frac{T_{p1}}{T_c+T_d}$	Min$\{T_{p1}, 4(T_c+T_d)\}$	T_{p2}

Os controladores PI e PID projetados pelo método SIMC são apresentados, respectivamente, em (16) e (17):

$$G_{c1_{SIMC}}(s) = 46{,}34\left(1+\frac{1}{7{,}86s}\right) \tag{16}$$

$$G_{c2_{SIMC}}(s) = 47{,}59\left(1+\frac{1}{7{,}6s}+44{,}32s\right) \tag{17}$$

4.3.2 MÉTODO *CHIEN, HRONES E RESWICK* (CHR)

Pelo método de sintonia proposto por Chien, Hrones e Reswick (CHR), por meio do modelo de primeira ordem com atraso, podem-se obter os parâmetros do controlador em uma rotina do MATLAB utilizando o método com 0% de sobresinal para cálculo desses parâmetros [35]. Neste método, para realizar o ajuste do controlador, foram aplicadas as regras de sintonia associada a cada tipo de controlador, conforme apresentadas na Tabela 4.4.

Tabela 4.4- Regra de Sintonia CHR.

Planta	k_p	T_i	T_{td}
1ª Ordem	$\frac{0{,}35T_p}{k\,T_d}$	$1{,}2T_p$	0
2 ª Ordem	$\frac{0{,}6T_{p1}}{k\,T_d}$	T_{p1}	$0{,}5T_d$

Os controladores PI e PID projetados pelo método CHR são apresentados, respectivamente, em (18) e (19):

$$\mathrm{G_{c1_{CHR}}}(\mathrm{s}) = 32{,}44\left(1+\frac{1}{1941{,}7\mathrm{s}}\right) \tag{18}$$

$$G_{c2_{CHR}}(s) = 57{,}11\left(1 + \frac{1}{1605{,}1s} + 0{,}475s\right) \quad (19)$$

4.4 ÍNDICES DE DESEMPENHO

A teoria de controle moderno admite que o engenheiro de sistemas pode especificar quantitativamente o desempenho requerido para o mesmo. Então, um índice de desempenho é uma medida quantitativa de desempenho do sistema e é escolhido de modo que a ênfase seja dada para as especificações do sistema [36].

Neste trabalho são apresentados dois índices para avaliação do desempenho e robustez dos sistemas de controle em malha fechada, o valor de pico da função de sensibilidade (M_s) e a integral do erro absoluto (*IAE*).

4.4.1 VALOR DE PICO DA FUNÇÃO DE SENSIBILIDADE (M_S)

Os processos estão sujeitos à mudanças no ambiente, envelhecimento de seus componentes, imprecisão dos valores dos parâmetros, etc. Assim, em um sistema em malha fechada verifica-se a variação na saída devido às mudanças no processo e procura-se corrigir as variações na saída. Por isso, a sensibilidade de um sistema é de importância fundamental [36].

A partir do ganho de malha *L(s)* pode-se determinar a função de sensibilidade *S(s)* de um sistema com realimentação de acordo com a Equação (20):

$$S(s) = \frac{E(s)}{R(s)} = \frac{1}{1 + L(s)} \quad (20)$$

onde *E(s)* é o sinal de erro e *R(s)* é o sinal de referência.

Com isso, um sistema em malha fechada permite que *G(s)* seja especificado com menor exatidão porque a sensibilidade a variações ou erros na planta é reduzida aumentando-se *L(s)* sobre a faixa de frequência de interesse [36].

O valor de pico M_s é definido como o máximo valor absoluto da função de sensibilidade em função da frequência, de acordo com a Equação (21):

$$M_s = \max|S(jw)| = \left|\frac{1}{1 + G_c(jw)G(jw)}\right| \quad (21)$$

onde valores pequenos são desejados para uma melhor margem de ganho (*GM*) e margem de fase (*PM*), que são calculados conforme as Equações (22) e (23) [37].

$$GM = \frac{M_s}{M_s - 1} \tag{22}$$

$$PM = 2arcsen\frac{1}{2M_s} \tag{23}$$

4.4.2 Integral do Erro Absoluto (IAE)

Para avaliar o desempenho em malha fechada, pode-se considerar um degrau unitário para o sinal de referência e perturbação (r(t) = d(t) = 1).

Dessa forma, tem-se que o desempenho da saída pode ser dado pela integral do erro absoluto (*IAE*), erro este provindo do controlador [34].

Matematicamente, na Equação (24), tem-se que

$$IAE = \int_0^\infty |e(t)|dt \tag{24}$$

onde o valor de *IAE* deve ser o menor possível.

Quanto à aplicação, esse índice de desempenho é particularmente útil para estudos de simulações computacionais.

4.5 Interfaces Gráficas e Rotinas implementadas

Foram desenvolvidas interfaces de monitoramento e controle para facilitar a comunicação entre o usuário e as rotinas de monitoramento e controle. Para a criação dessas interfaces gráficas foi feito o uso do GUIDE (*Graphical User Interface Development Environment*), uma ferramenta fornecida pelo MATLAB projetada para que o usuário construa interfaces gráficas com maior facilidade e rapidez.

Monitoramento

Na Figura 4.5 pode ser observada a interface desenvolvida para o monitoramento do processo.

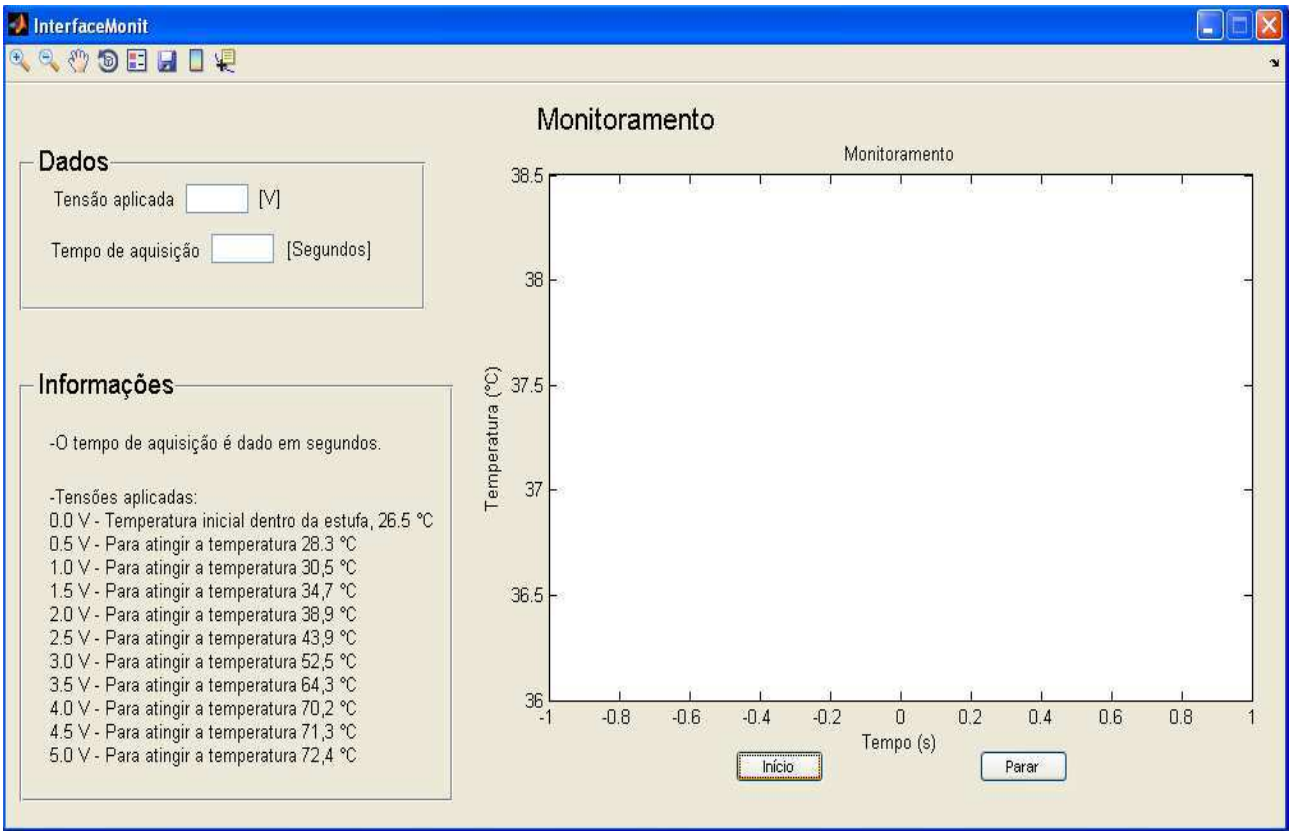


Figura 4.5 – Tela da Interface Gráfica Para Monitoramento do Processo.

Na interface para o monitoramento, no menu 'Dados', é possível ao usuário definir a tensão, em Volts, que deseja aplicar na resistência localizada no interior da estufa, como também o tempo de execução do processo, em segundos. Na própria interface existe uma seção com a informação da relação da tensão aplicada e da temperatura vinculada a esta tensão, referentes aos valores médios calculados após o conjunto de medições realizado.

Ao fim de cada experimento de monitoramento, o programa retorna uma caixa de texto informando o tempo de duração do experimento, como também o valor da temperatura, caso esteja estabilizada, lembrando que a temperatura é considerada estabilizada se, após a aplicação da tensão desejada, a duração do processo for maior que 60 minutos, caso ainda não tenha atingido o tempo necessário para a estabilização, é informada na caixa de texto que a temperatura ainda não esta estabilizada. O processo não necessariamente precisa ser paralisado, por exemplo, se a duração do processo no momento em que foi perdido o sinal está em torno de 40 minutos, é possível aguardar o tempo restante necessário para a estabilização da temperatura (neste caso, em torno de 20 minutos), podendo assim, inferir o valor de temperatura. Nas Figuras 4.6 e 4.7, pode-se observar o exemplo dessas caixas de texto para o caso da temperatura estar estabilizada e não estar estabilizada, respectivamente.

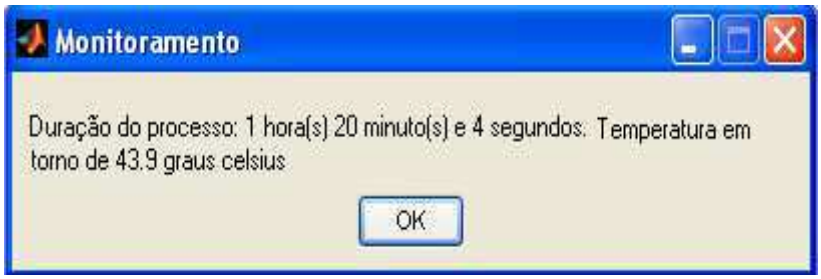


Figura 4.6 – Tela da Caixa de Texto com Informações do Processo (Temperatura Estabilizada).

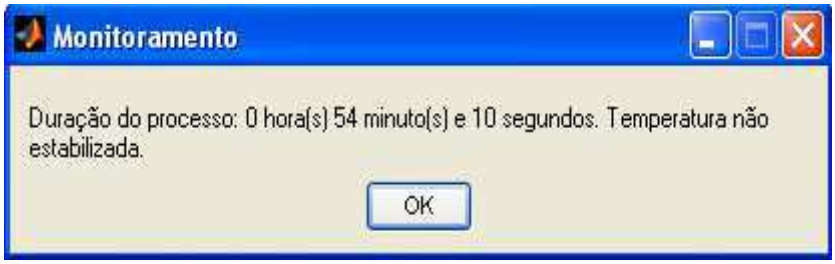


Figura 4.7 – Tela da Caixa de Texto com Informações do Processo (Temperatura não Estabilizada).

Controle

Na Figura 4.8 pode ser observada a interface desenvolvida para o controle do processo. Caso o usuário queira fazer o experimento com apenas uma temperatura de referência, basta digitar o número 0 (zero) no espaço indicado para a segunda temperatura de referência, simbolizando que não será utilizada essa temperatura e o experimento será realizado com apenas uma temperatura de referência.

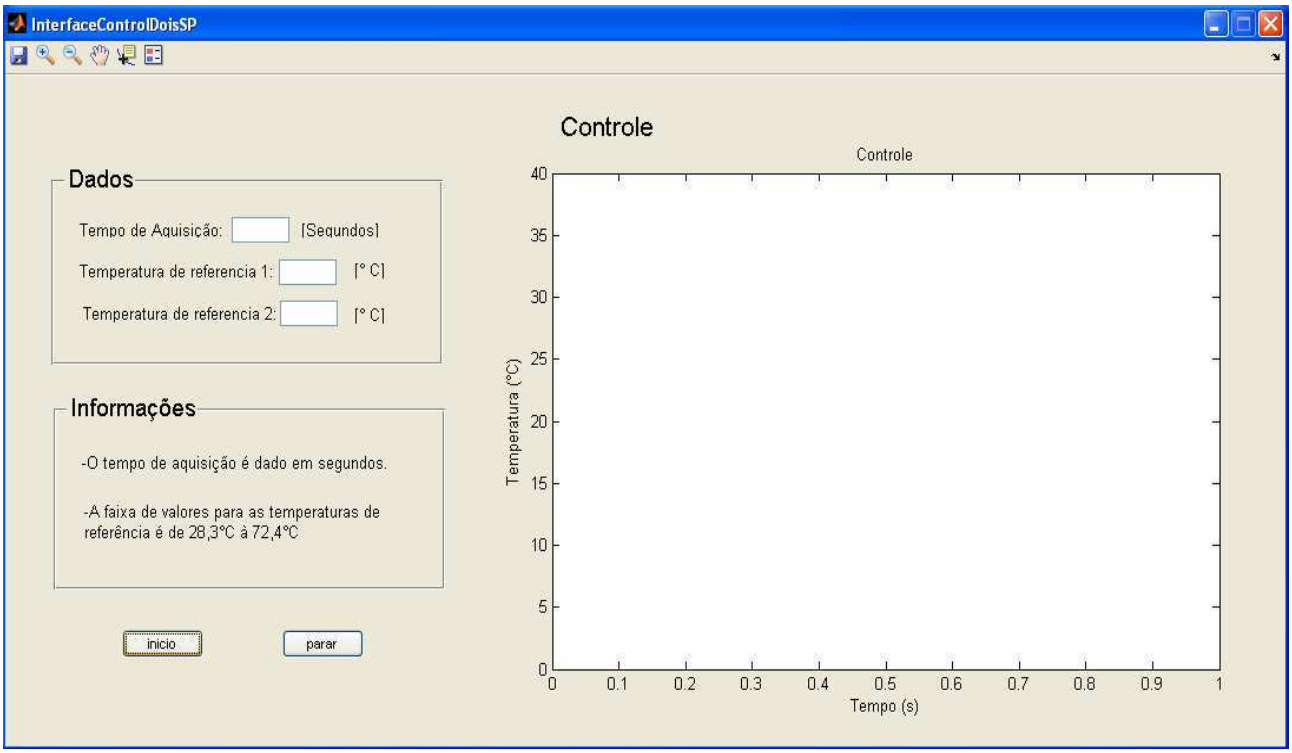


Figura 4.8 – Tela da Interface Para Controle do Processo.

No menu ‘Dados’, o usuário define o tempo de execução do processo, em segundos, e a(s) temperatura(s) de referência (temperatura que se deseja alcançar), em Graus Celcius.

Nos gráficos obtidos com essas interfaces, o eixo ‘x’ representa o tempo de duração do processo e o eixo ‘y’ a temperatura em Graus Celcius.

4.6 Rotina Implementada Para Controle do Processo

O algoritmo que possibilita o controle pelo MATLAB em tempo real baseia-se nas funções de cronômetro do MATLAB, (TIC e TOC), onde TIC inicia a contagem do cronômetro no instante que é executada, e TOC armazena o tempo decorrido entre execução da função TIC e a sua execução, em conjunto com a função *while* que executará um loop durante o tempo estabelecido pelo usuário.

Na Figura 4.9 é apresentado o diagrama de fluxo do algoritmo de controle em tempo real.

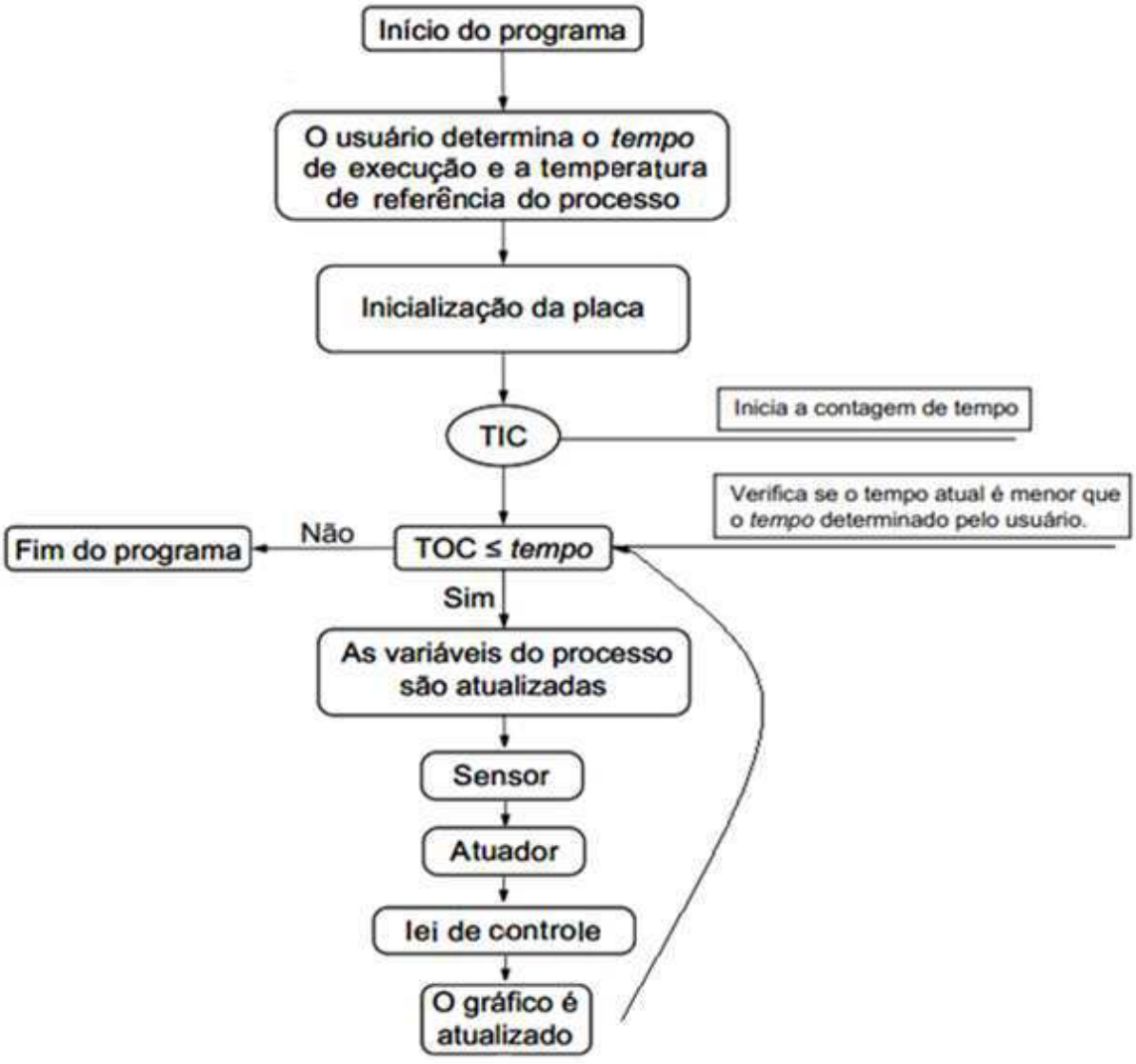


Figura 4.9 – Diagrama de Fluxo do Algoritmo em Tempo Real.

O programa é iniciado e após ser definido o tempo de execução e a temperatura de referência do processo, dados pelo usuário, o primeiro passo é executar a inicialização da placa de aquisição, depois são realizadas as definições das variáveis utilizadas no algoritmo, os parâmetros do controlador, variáveis de tempo e taxa de amostragem.

A função TIC é executada iniciando a contagem de tempo, em seguida inicia-se o *loop* que será executado enquanto o tempo do processo for menor que o tempo definido pelo usuário.

É realizado o cálculo do sinal de controle para atuar na estufa e esse sinal é enviado para a planta através da saída analógica. Devido à saída analógica do DAQ aceitar valores de tensões entre -5V e 5V, para adequar a saída é necessário uma conversão de valor antes de enviar o sinal de controle para a planta.

Com os níveis de tensões aceitáveis, o sinal de controle é enviado para atuar na planta, em seguida o gráfico da interface é atualizado e inicia-se o loop novamente repetindo todo o processo até que o tempo do processo seja igual ao tempo definido pelo usuário.

4.7 SIMULAÇÕES

Para a avaliação e robustez dos sistemas de controle em malha fechada com os controladores projetados foram simuladas as respostas ao aplicar uma variação no sinal de referência e uma perturbação de carga, ambas do tipo degrau unitário e calculados os índices de desempenho M_s (Valor de pico da função de sensibilidade) e IAE (Integral do Erro Absoluto) [38].

Na Figura 4.10 está apresentado o diagrama de blocos do sistema de controle de malha fechada para a simulação.

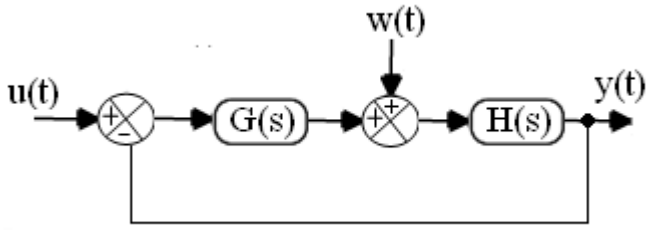


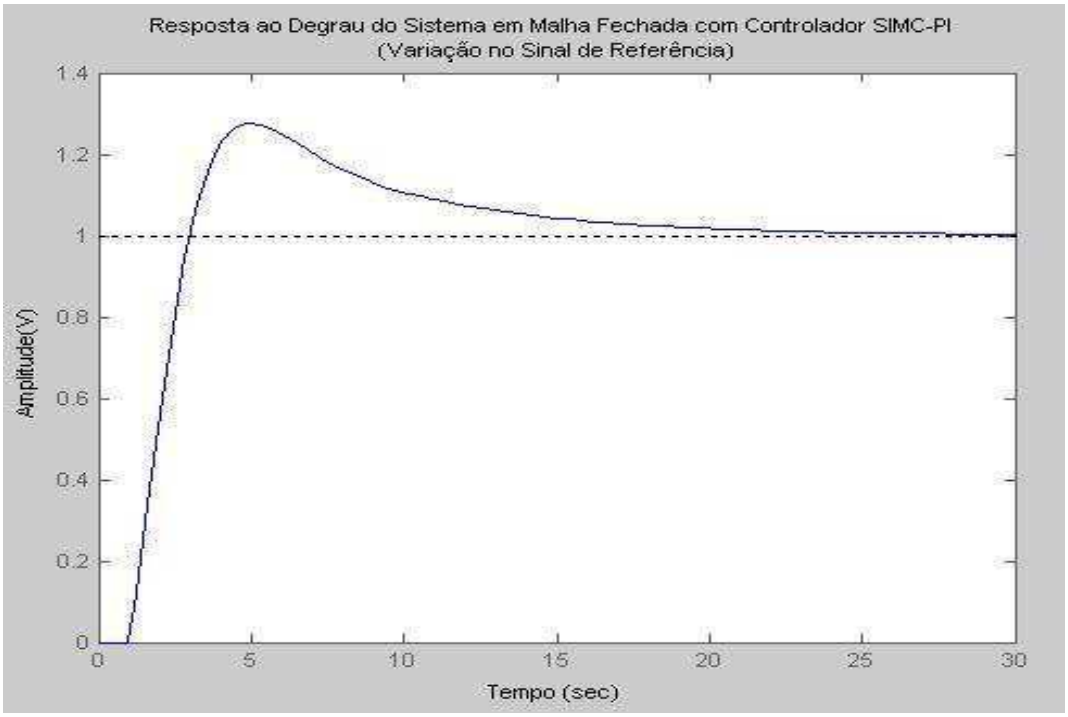
Figura 4.10. Diagrama de Blocos do Sistema de Controle de Malha Fechada Para a Simulação.

Em que u(t) é a entrada (degrau unitário), w(t) a perturbação de carga e y(t) é a saída.

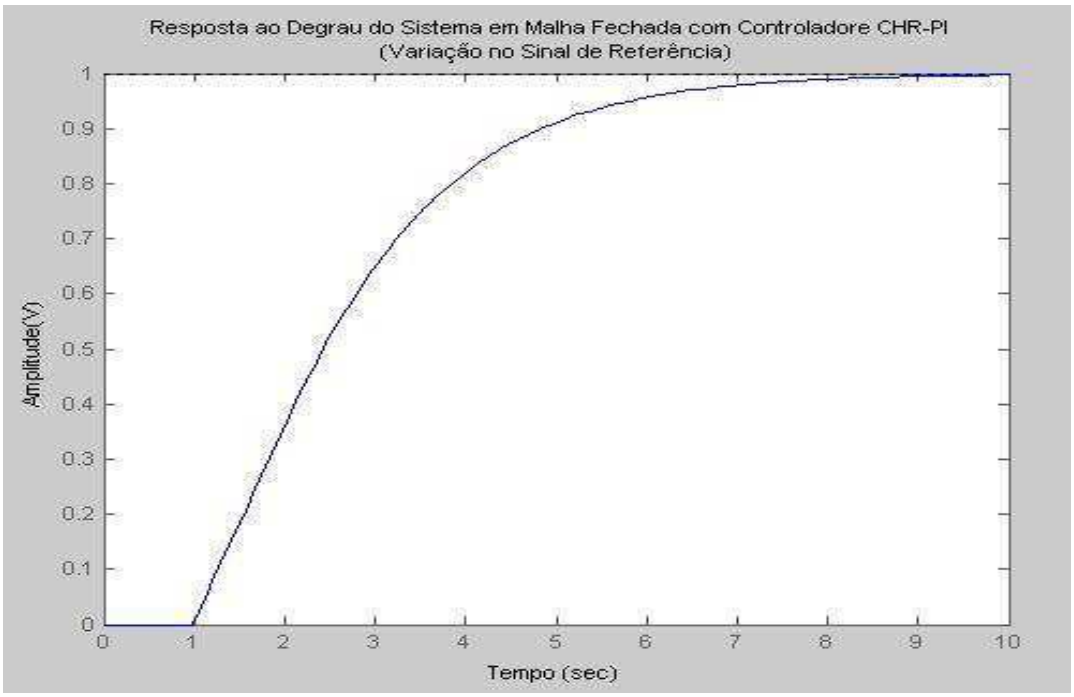
Para ambos os métodos (SIMC e CHR), os resultados obtidos nas simulações para o controlador PI podem ser observados nos gráficos das Figuras 4.11 e 4.12.

Em relação ao comportamento do sistema em malha fechada, dada uma variação no sinal de referência, observa-se que, para o controlador PI, a sintonia obtida pelo método CHR (Figura 4.11-

b) alcança a estabilidade em menos tempo, apresenta uma curva sem *overshoot* (medida de quanto a resposta excede o valor de referência), além de obter um valor baixo de M_s, igual a 1,48 e minimizar a integral do erro absoluto, alcançando-se um IAE = 2,80. Diferentemente da estratégia SIMC (Figura 4.11-a), que apresentou uma curva com *overshoot* igual a 1,34, M_s = 2,27 e IAE = 3,82.



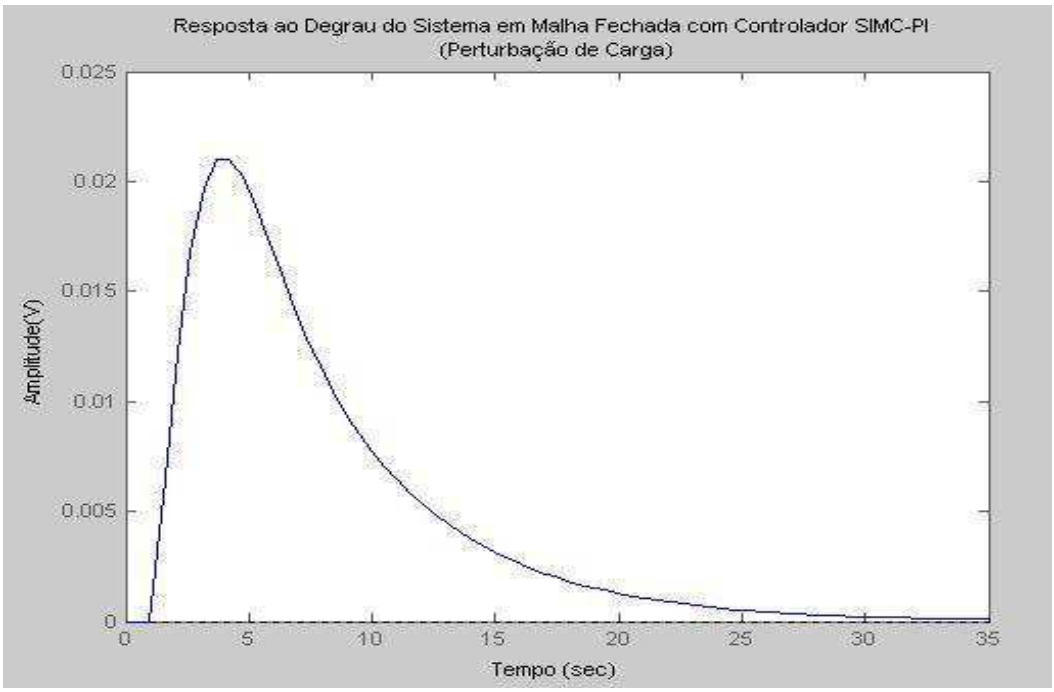
(a)



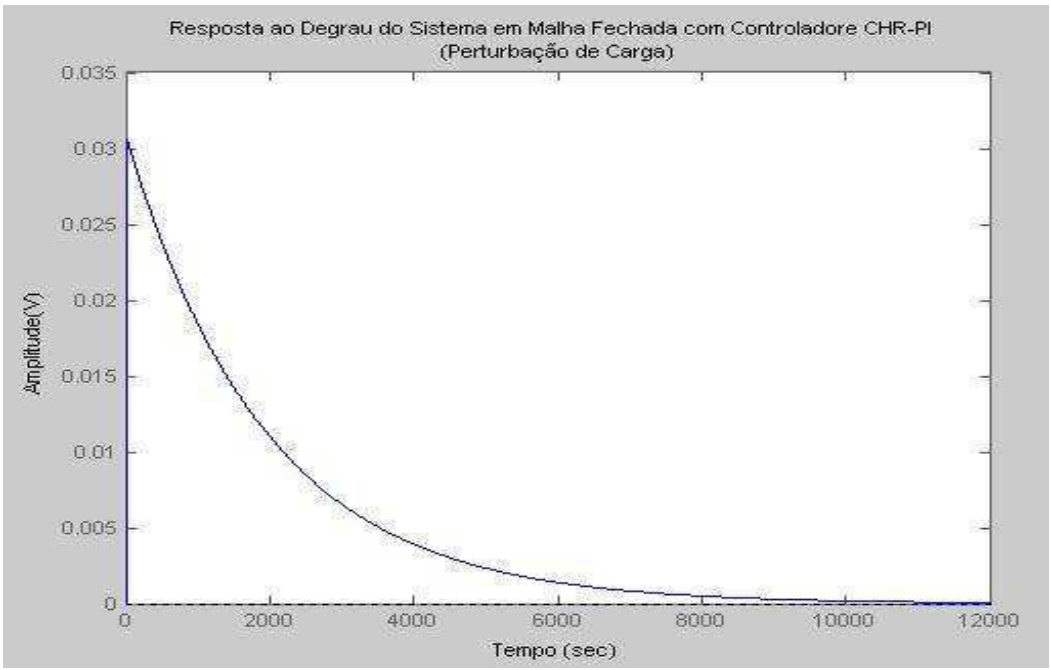
(b)

Figura 4.11. Resposta ao Degrau do Sistema em Malha Fechada dada uma Variação no Sinal de Referência: (a) Controlador SIMC-PI; (b) Controlador CHR-PI.

Já quando aplicada a perturbação de carga (Figura 4.12), observou-se que o método SIMC teve melhor resposta, com um *overshoot* em torno de 0,02, por este método a curva volta ao valor de origem (zero) em menos tempo.



(a)

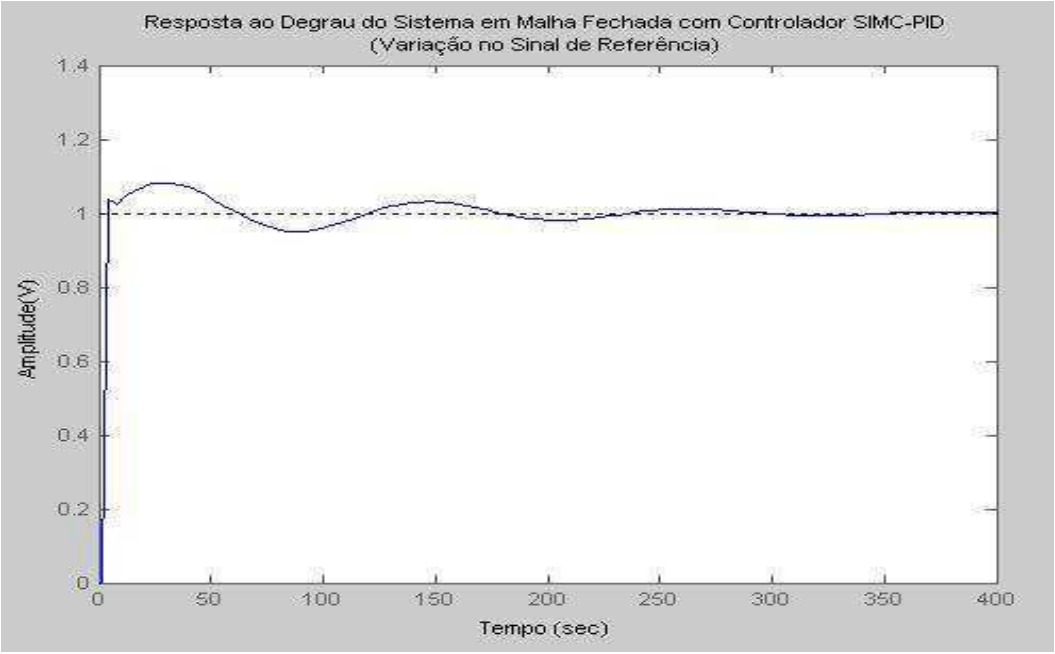


(b)

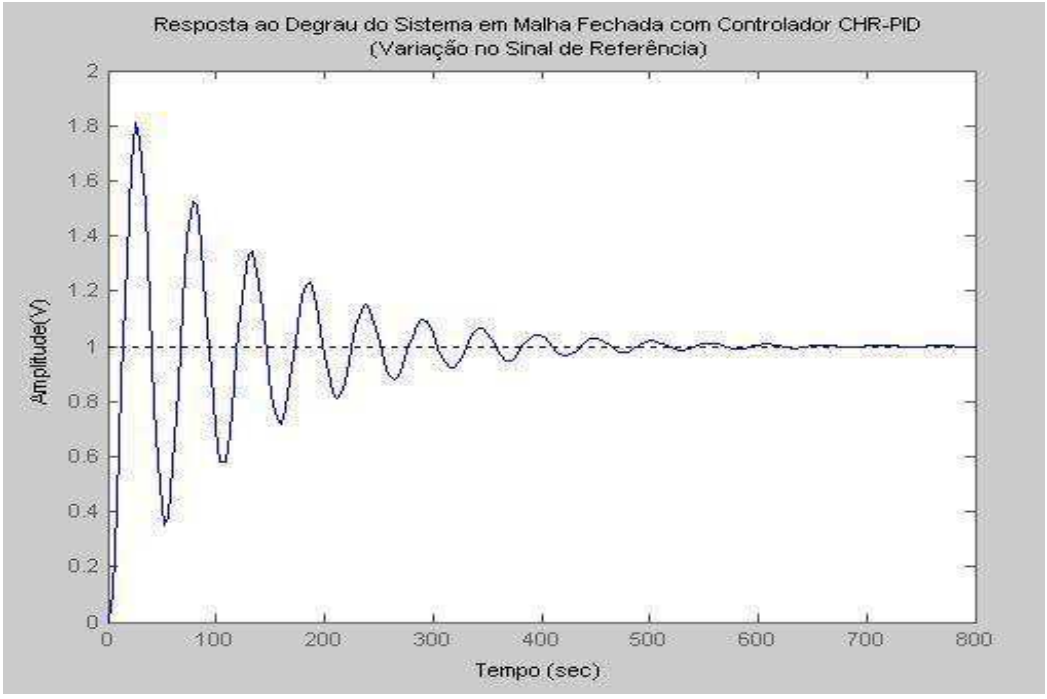
Figura 4.12- Resposta ao Degrau do Sistema em Malha Fechada dada uma Perturbação de Carga: (a) Controlador SIMC-PI; (b) Controlador CHR-PI.

Nas Figuras 4.13 e 4.14 observam-se os gráficos dos resultados obtidos para os controladores PID para ambos os métodos (SIMC e CHR).

Já para os controladores PID, a sintonia obtida pelo método SIMC aplicado um degrau no sinal de referência (Figura 4.13-a) tem um resultado melhor comparado com o obtido pelo método CHR, pois possui menor valor de *overshoot*, igual a 0,08, valor de M_s = 2,09 e IAE = 10,09. Diferente do método CHR, que apresentou uma curva com *overshoot* em torno de 0,8, M_s = 8,47 e IAE = 81,46.



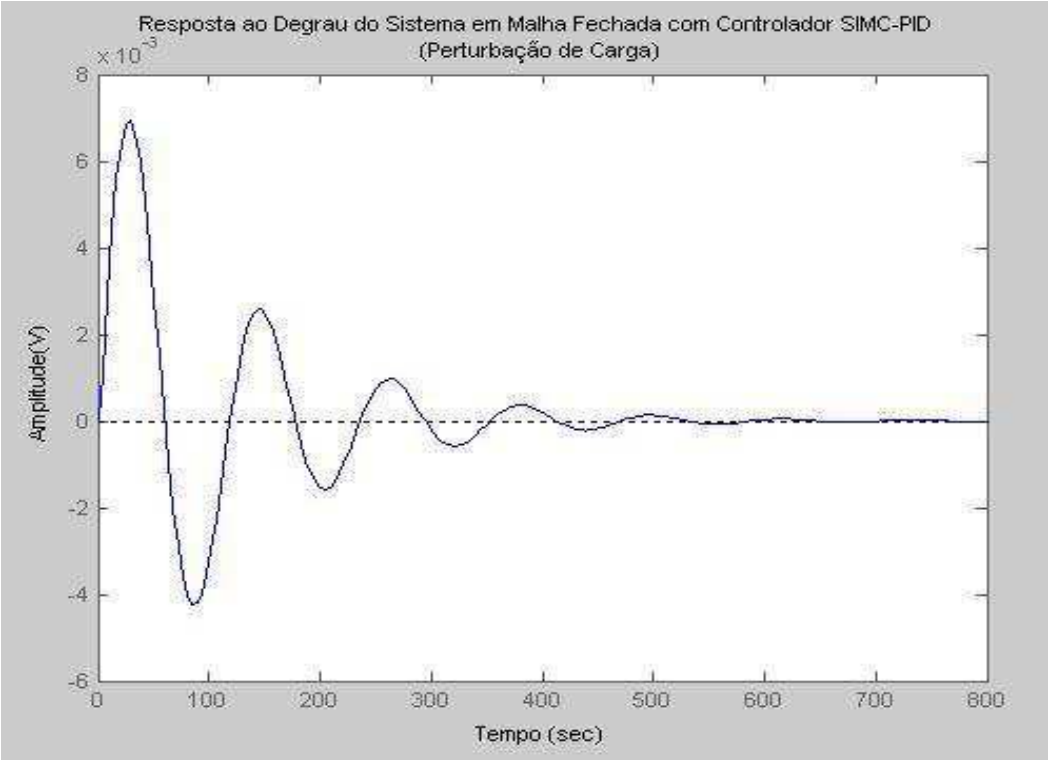
(a)



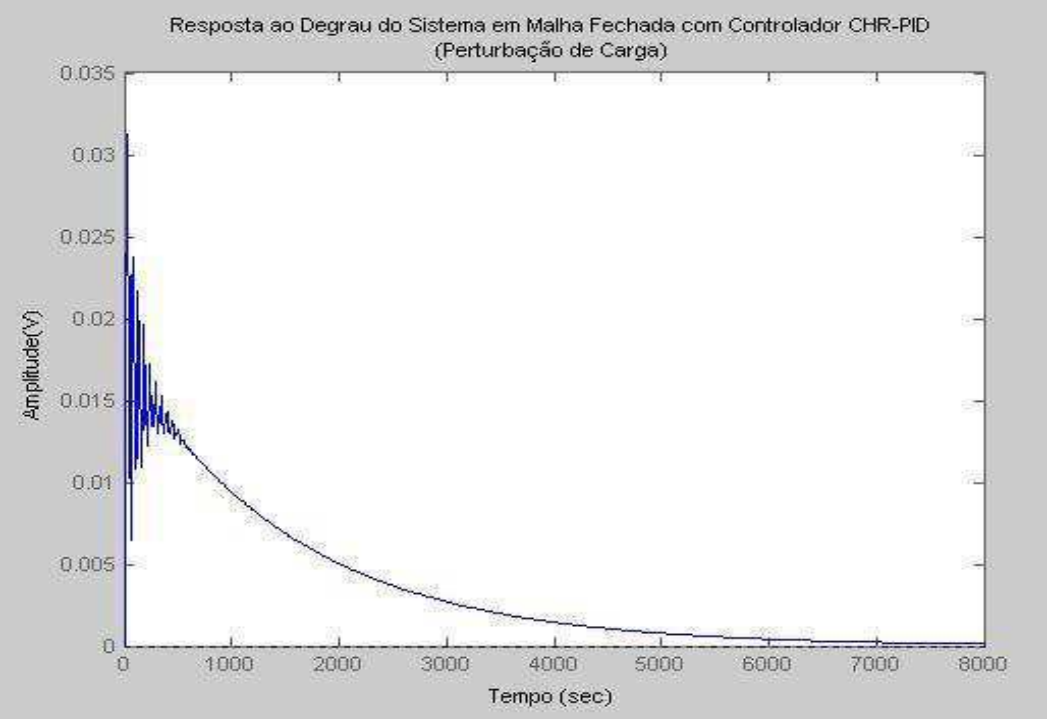
(b)

Figura 4.13. Resposta ao Degrau do Sistema em Malha Fechada dada uma Variação no Sinal de Referência: (a) Controlador SIMC-PID; (b) Controlador CHR-PID.

Para a perturbação de carga (Figura 4.14), também se observou que o método SIMC obteve melhor resposta, com menor *overshoot* (em torno de 0,007) e com sua curva voltando ao valor de origem (zero) em menos tempo.



(a)



(b)

Figura 4.14- Resposta ao Degrau do Sistema em Malha Fechada dada uma Perturbação de Carga: (a) Controlador SIMC-PID; (b) Controlador CHR-PID.

4.8 RESULTADOS DOS EXPERIMENTOS

4.8.1 RESULTADOS OBTIDOS COM O MONITORAMENTO

Com o objetivo de atingir uma temperatura em torno de 43,9°C foi aplicada a tensão de 2,5V na resistência localizada no interior da estufa. No primeiro teste, retirou-se o sensor LM35 (perdendo o sinal) antes da duração do processo atingir 60 minutos. Na Figura 4.15, pode-se observar o gráfico obtido com o teste.

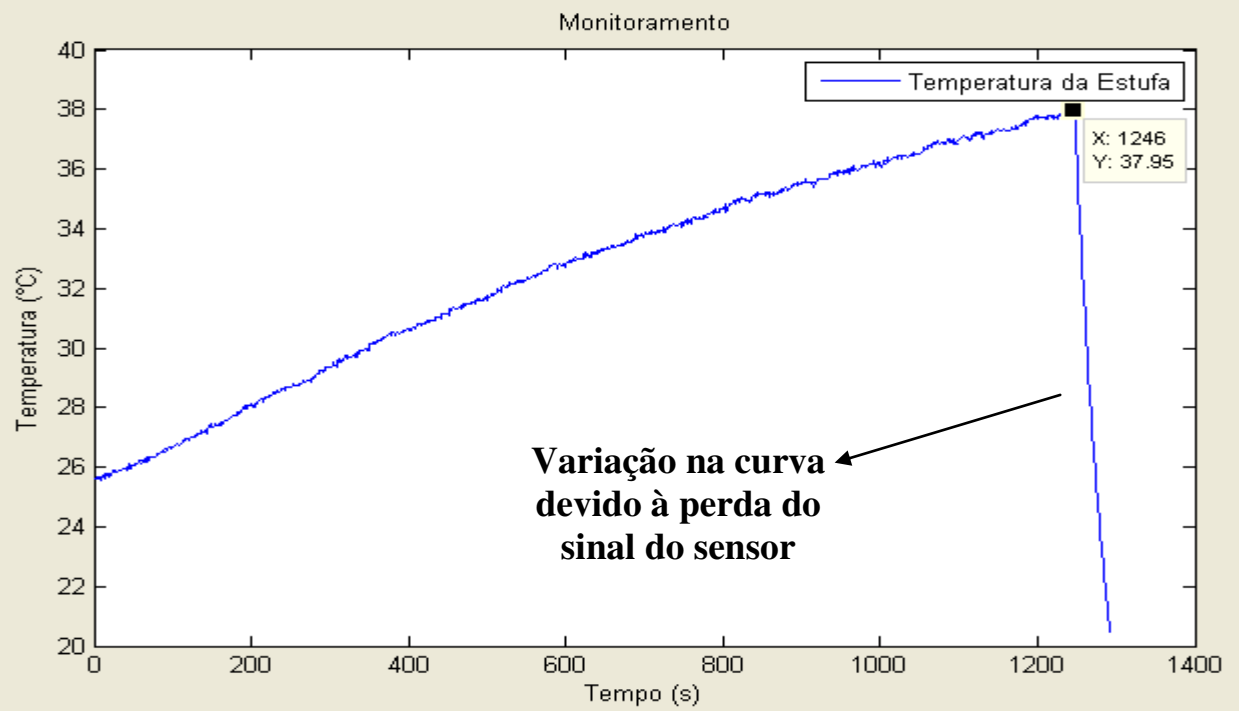


Figura 4.15- Resultado do Monitoramento com Perda do Sinal do Sensor (Temperatura Não Estabilizada).

É possível observar no gráfico a última temperatura medida pelo sensor (37,95°C), a partir desse ponto observa-se uma queda na curva da temperatura, essa queda é devido a perda do sinal, como apontado no gráfico. Como a temperatura ainda não estava estabilizada, pois a duração do processo era inferior a 60 minutos, uma caixa de texto gerada pelo programa mostra a duração do processo informando que a temperatura ainda não estava estabilizada. Na Figura 4.16 pode-se observar esta caixa de texto.

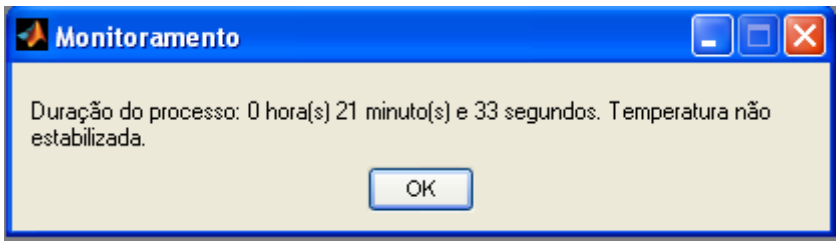


Figura 4.16- Tela da Caixa de Texto com Informações sobre o Monitoramento com Perda do Sinal do Sensor (Temperatura Não Estabilizada).

Realizou-se um novo teste de monitoramento, na Figura 4.17 pode-se obsevar o gráfico obtido com o teste realizado.

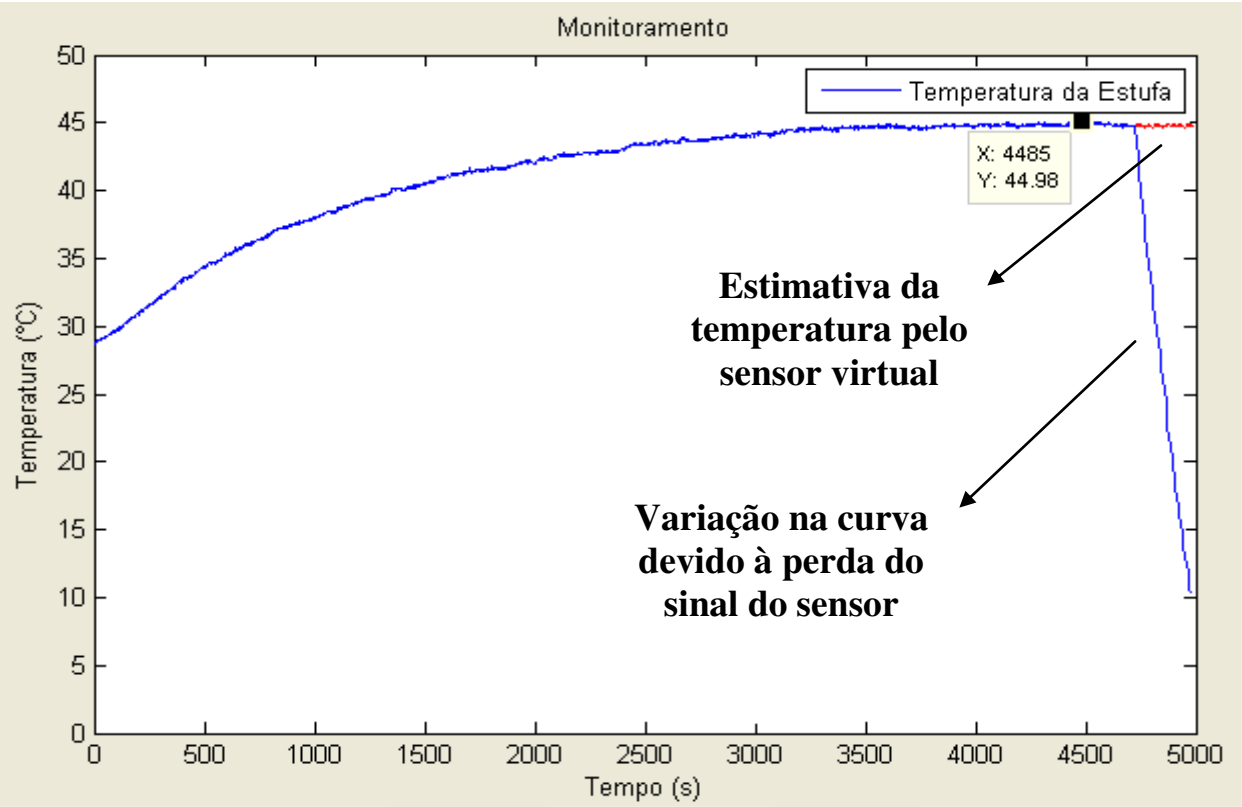


Figura 4.17- Resultado do Monitoramento com Perda do Sinal do Sensor (Temperatura Estabilizada).

Analisando o gráfico, é possível observar que, no momento que se perdeu o sinal do sensor LM35, a duração do processo era maior que o tempo necessário para a estabilização da temperatura (60 minutos), podendo-se assim, inferir o valor da temperatura, de acordo com a relação de tensão e temperatura obtidas nos testes realizados anteriormente. Assim, obteve-se êxito no propósito do programa, caso ocorra a perda de sinal do sensor, sabendo-se a tensão que está sendo aplicada e o tempo de duração do processo, é possível inferir o valor de temperatura dentro da estufa, que neste caso, foi de 43,9°C.

A caixa de texto com a duração do processo informando a temperatura no interior da estufa pode ser observada na Figura 4.18.

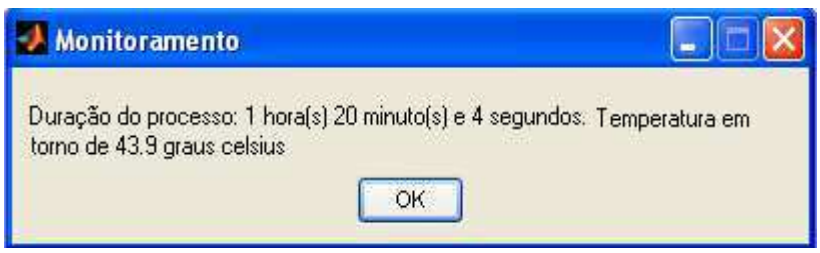


Figura 4.18- Tela da Caixa de Texto com Informações sobre o Monitoramento com Perda do Sinal do Sensor (Temperatura Estabilizada).

4.8.2 Resultados Obtidos com o Controle

Com base no cálculo dos controladores implementados no MATLAB e nas simulações realizadas, foi definido na rotina de controle o melhor controlador para cada caso.

Nos testes com uma temperatura de referência e partindo-se da temperatura inicial dentro da estufa, caso a temperatura de referência seja menor que 38,9°C (Critério para escolha do controlador), foi considerado um sistema de primeira ordem, por ter variação de temperatura pequena, assim, a rotina considera o controlador PI pelo método de sintonia CHR a melhor opção de controle. Se a temperatura de referência for maior que 38,9°C, considera-se um sistema de segunda ordem, variação de temperatura maior em comparação com o sistema considerado de primeira ordem, assim, na rotina implementada determina-se o controlador PID pelo método de sintonia SIMC a melhor opção de controle.

Com uma Temperatura de Referência

Na Figura 4.19 pode-se observar o gráfico obtido para o experimento com a temperatura de referência igual a 34,7°C.

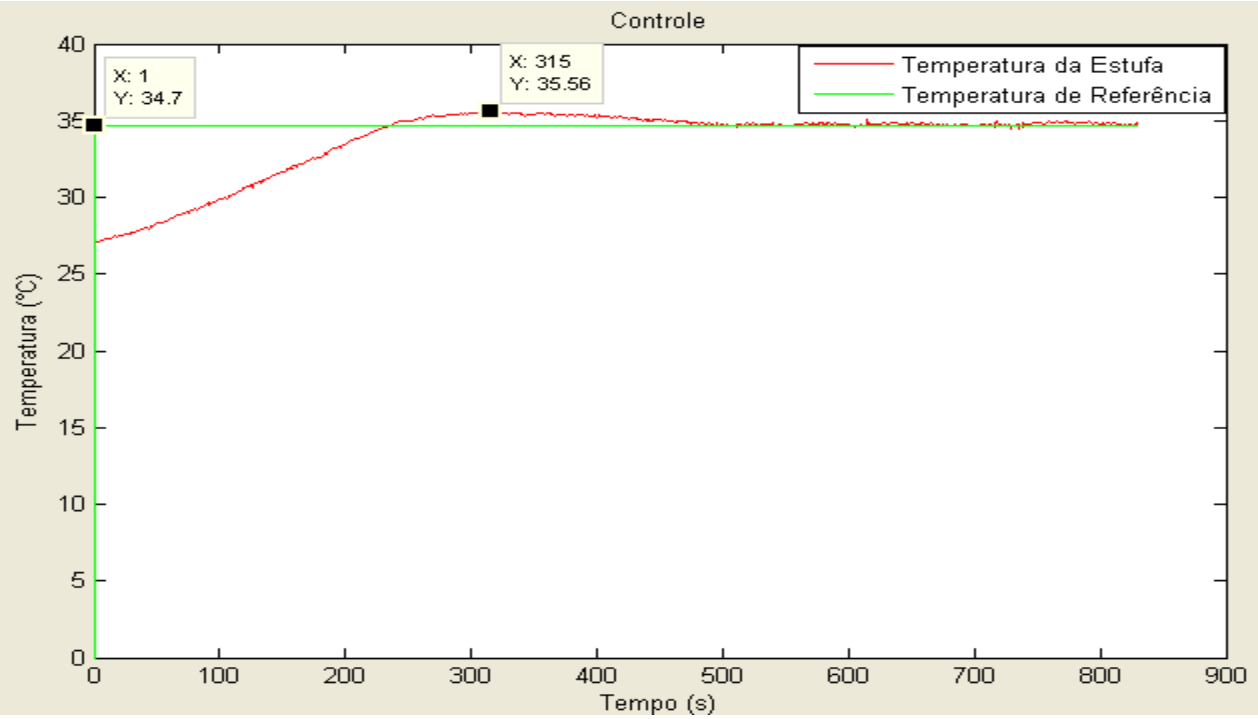


Figura 4.19- Resultado do Experimento com Temperatura de Referência Igual a 34,7°C.

Como o valor da temperatura de referência é menor que 38,9°C, neste caso, na rotina é definido o controlador PI pelo método de sintonia CHR para atuar na estufa.

A curva apresentada na figura é considerada um resultado satisfatório, o controlador agiu na planta, fazendo com que a temperatura atingisse a temperatura de referência definida pelo usuário

(34,7°C). Apesar de ter um pequeno *overshoot,* no valor de 0,86 (Diferença entre o valor máximo de temperatura medido, pelo gráfico, 35,56°C e a temperatura de referência, no caso, 34,7°C), a curva tem um bom tempo de acomodação, em torno de 500 segundos (aproximadamente 8 minutos), como se pode observar no gráfico da Figura 5.17.

Na Figura 4.20 pode-se observar o gráfico obtido para o teste com a temperatura de referência igual a 52,5°C.

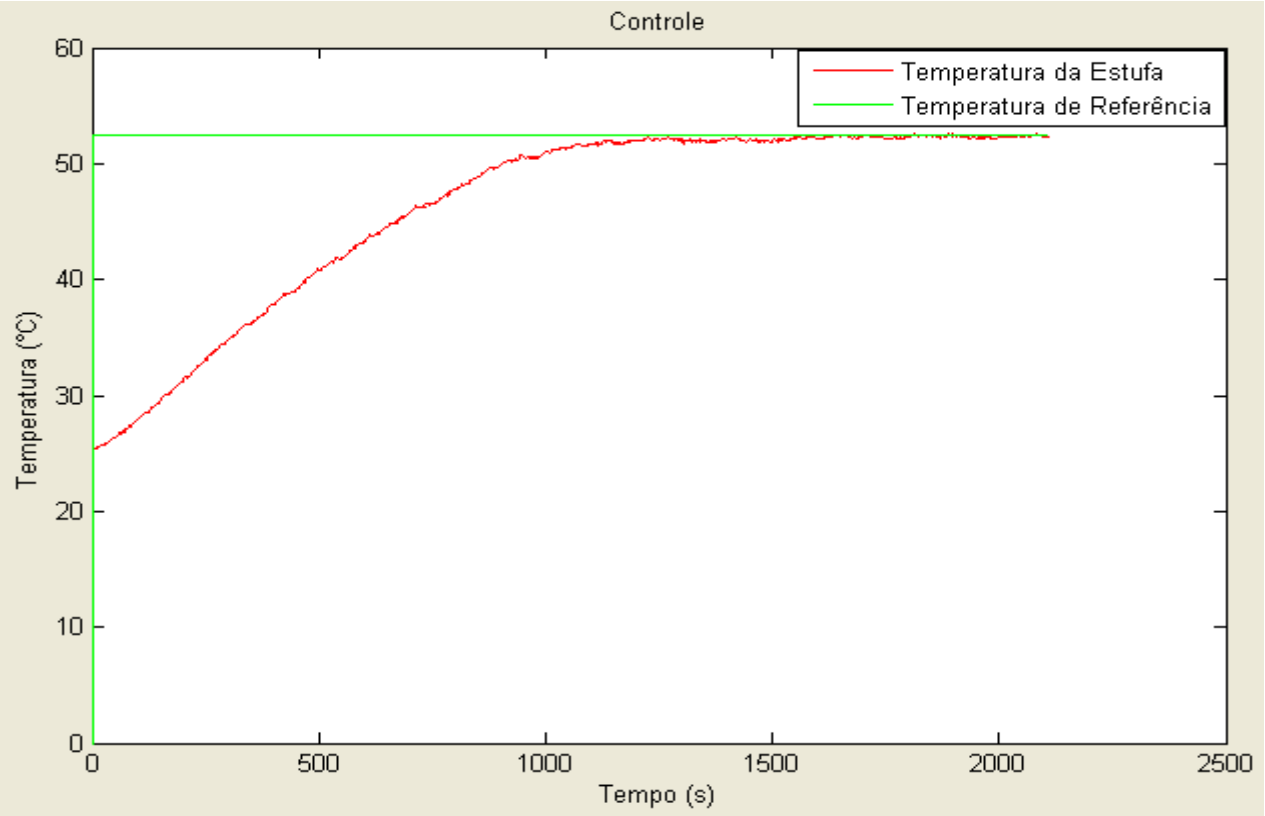


Figura 4.20- Resultado do Experimento com Temperatura de Referência Igual a 52,5°C.

Como o valor da temperatura de referência é maior que 38,9°C, neste caso, na rotina é definido o controlador PID pelo método de sintonia SIMC para atuar na estufa, que agiu na planta fazendo com que a temperatura atingisse a temperatura de referência definida pelo usuário. Não apresentou *overshoot* e seu tempo de acomodação foi em torno de 1250 segundos (aproximadamente 20 minutos), como se pode observar no gráfico da Figura 5.18.

Com duas Temperaturas de Referência

Na Figura 4.21 pode-se observar o gráfico obtido para o experimento com duas temperaturas de referência, 43,9°C aplicado no instante inicial e 64,3°C aplicado após 600 segundos.

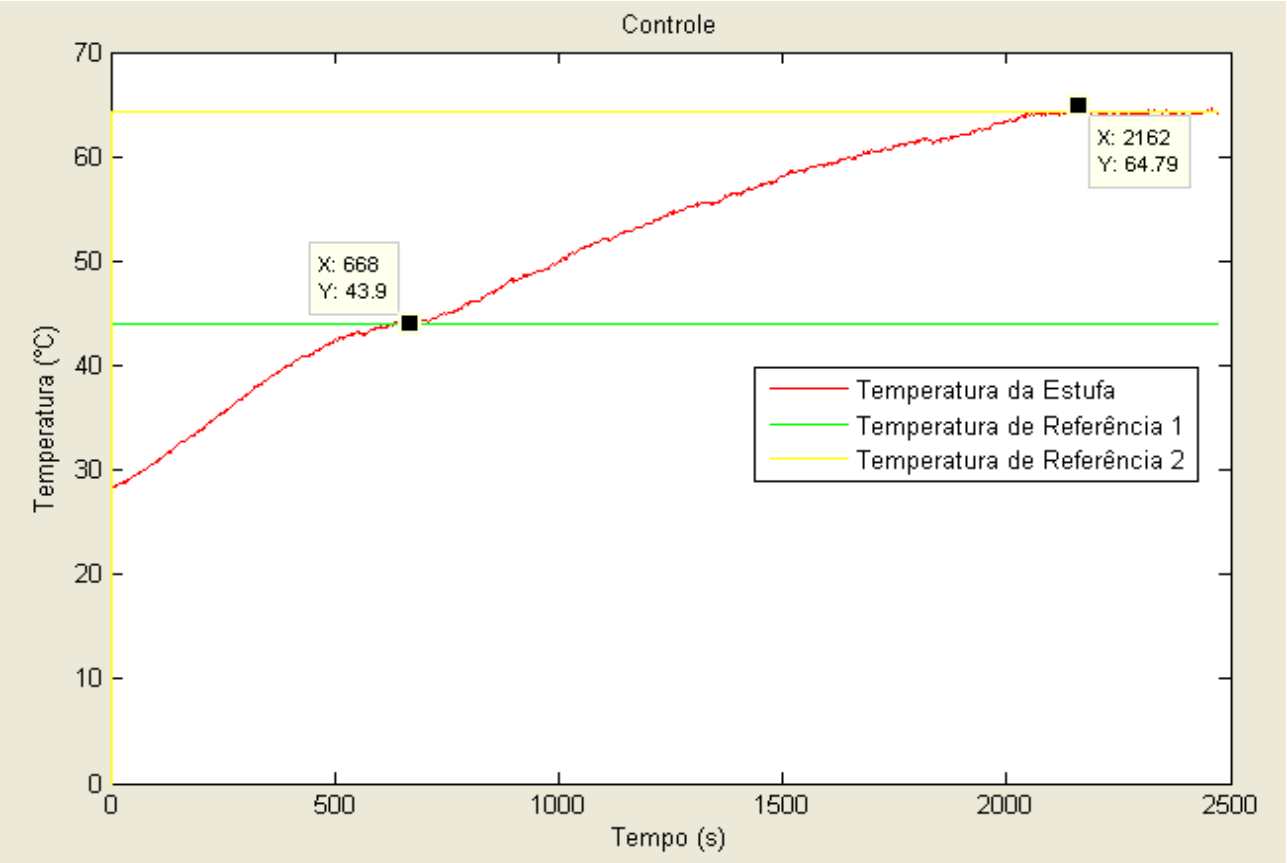


Figura 4.21- Resultado do Experimento com Duas Temperaturas de Referência, 43,9°C e 64,3°C.

Como o primeiro valor da temperatura de referência é maior que 38,9°C, na rotina é definido o controlador PID pelo método de sintonia SIMC para atuar na estufa. Já na segunda parte da curva, a diferença das temperaturas de referência é menor que 38,9°C, sendo assim, neste caso, na rotina foi definido o controlador PI pelo método de sintonia CHR para atuar na estufa.

Pela curva apresentada no gráfico é possível observar a eficiência dos controladores, que agiram na planta fazendo com que a temperatura da estufa atingisse as temperaturas de referência definidas pelo usuário. Para alcançar a segunda temperatura de referência, apesar de ter um pequeno *overshoot,* no valor de 0,49 (Diferença entre o valor máximo de temperatura medido, pelo gráfico, 64,79°C e a temperatura de referência, no caso, 64,3°C), a curva estabilizou na temperatura de referência definida pelo usuário.

Foi realizado um segundo experimento com duas temperaturas de referência. Na Figura 4.22 pode-se observar o gráfico obtido para este experimento, com a primeira temperatura de referência aplicada no instante inicial no valor de 35°C e a segunda de 55°C aplicado após 1600 segundos.

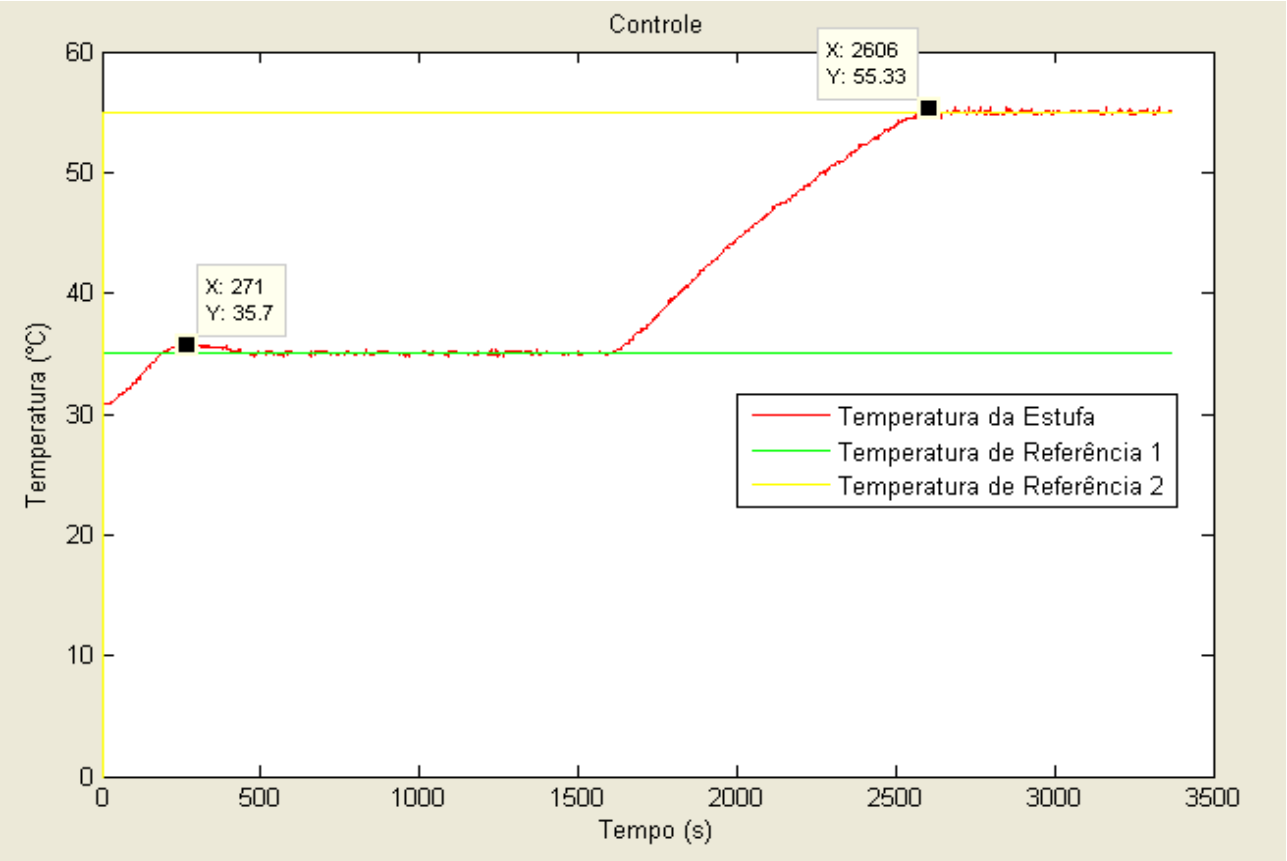


Figura 4.22- Resultado do Experimento com Duas Temperaturas de Referência, 35°C e 55°C.

Como o primeiro valor da temperatura de referência é menor que 38,9°C, na rotina é definido o controlador PI pelo método de sintonia CHR para atuar na estufa. Assim como na segunda parte da curva, a diferença das temperaturas de referência também é menor que 38,9°C, deste modo, novamente foi definido na rotina o controlador PI pelo método de sintonia CHR.

Os controladores atuaram na planta fazendo com que a temperatura da estufa atingisse as temperaturas de referência definidas pelo usuário, mostrando a eficiência dos mesmos. Na primeira parte da curva, apesar de ter um pequeno *overshoot,* no valor de 0,7 (Diferença entre o valor máximo de temperatura medido, pelo gráfico, 35,7°C e a temperatura de referência, no caso, 35°C), a curva teve um tempo de acomodação em torno de 500 segundos (aproximadamente 8 minutos) e na segunda parte da curva, o *overshoot* foi ainda menor, no valor de 0,33, neste caso, também se obteve êxito, a temperatura atingiu a temperatura de referência fornecida pelo usuário e permaneceu estabilizada.

4.8.3 Resultados Obtidos com o Controle com Perturbação

Uma perturbação externa foi causada para testar o controle da planta. Durante o experimento, colocou-se o sensor de temperatura LM35 em contato com água em temperatura ambiente, para que sua temperatura diminuísse.

Na Figura 4.23 pode ser observado o gráfico desse experimento realizado com a perturbação, em que a temperatura de referência foi de 38,9°C.

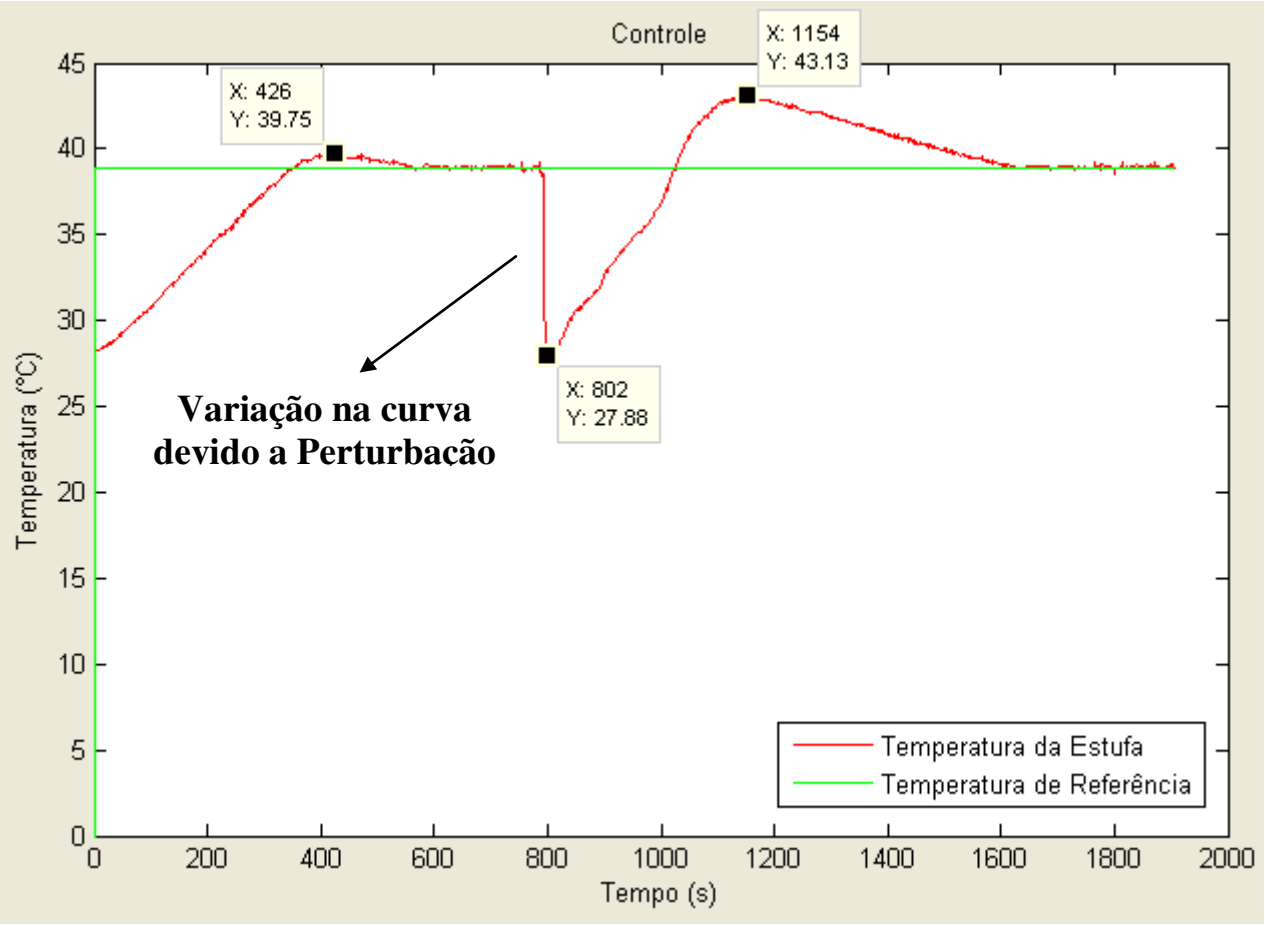


Figura 4.23- Resultado do Experimento com Perturbação na Planta.

No gráfico é possível observar na primeira parte da curva que, apesar de um pequeno *overshoot* no valor de 0,85, a temperatura atinge o valor de regime permanente no tempo em torno de 600 segundos (10 minutos). Após a temperatura atingir o valor de regime permanente, gerou-se uma perturbação, em que o sensor LM35 foi colocado em contato com água em temperatura ambiente, é possível observar a variação na curva, em que a temperatura chega a atingir 27,88°C devido a perturbação causada, após a perturbação, a curva tem um *overshoot* em torno de 4,23, pois apesar de perturbar o sistema como um todo, só o sensor entrou em contato com a água em temperatura ambiente, mas a massa de ar quente que já estava no interior da estufa permaneceu, e o controlador agiu como se toda a estufa estivesse a 27,88º C, aplicando um sinal maior que o necessário, o que justifica um *overshoot* maior que na primeira parte da curva. Mas apesar disso, a temperatura voltou ao valor de temperatura desejado, definido pelo usuário no início do experimento.

5 CONCLUSÕES

Os objetivos do trabalho é realizar o monitoramento e controle da temperatura de uma estufa utilizando o conceito de sensor virtual, baseado em duas vertentes: medições, para o monitoramento do processo, e modelagem e controle, para o controle da temperatura no interior da estufa.

Com o monitoramento do processo foi possível garantir que, uma vez perdido o sinal do sensor LM35, sabendo-se a tensão que está sendo aplicada na resistência e o tempo de duração do processo, o valor de temperatura do interior da estufa pode ser inferido de forma automática. Ao fim de cada experimento de monitoramento, o programa retorna uma caixa de texto informando o tempo de duração do experimento, como também o valor da temperatura, caso tenha atingido o valor de regime permanente, lembrando que esse valor de temperatura é considerado estar em regime permanente se, após a aplicação da tensão desejada, a duração do processo for maior que 60 minutos, e caso ainda não tenha atingido esse tempo, é informada na caixa de texto que a temperatura ainda não esta em regime permanente. O processo não necessariamente precisa ser paralisado, por exemplo, se a duração do processo no momento em que foi perdido o sinal está em torno de 40 minutos, é possível saber o último valor de temperatura medido e aguardar o tempo restante necessário para que a temperatura atinja o valor de regime permanente (neste caso, em torno de 20 minutos), podendo assim, inferir o valor de temperatura.

Com os dados obtidos nos testes iniciais foram determinados os parâmetros para modelos de primeira e segunda ordem com atraso e a partir desses modelos foi realizada a validação, e em seguida a determinação dos controladores para cada modelo.

No controle do processo, foram apresentadas a avaliação dos métodos de sintonia propostos por *Skoghestad* (SIMC) e por *Chien, Hrones e Reswick* (CHR), como estratégias de controle para atuar no controle da temperatura da estufa. Nas simulações, em relação ao comportamento do sistema em malha fechada, dada uma variação no sinal de referência, observa-se que, para o controlador PI sintonizado pelo método CHR, o tempo de acomodação é menor, apresenta uma curva sem *overshoot* (medida de quanto a resposta excede o valor de referência), além de obter um valor baixo de M_s, igual a 1,48 e minimizar a integral do erro absoluto, alcançando-se um IAE = 2,80. Diferentemente da estratégia SIMC, que apresentou uma curva com *overshoot* igual a 1,34, M_s = 2,27 e IAE = 3,82. Já quando aplicada a perturbação de carga, observou-se que o método SIMC teve melhor resposta, com um *overshoot* em torno de 0,02, e por este método a curva volta ao valor de origem (zero) em menos tempo.

Para os controladores PID, nas simulações, a sintonia obtida pelo método SIMC aplicado um degrau no sinal de referência tem um resultado melhor comparado com o obtido pelo método CHR, pois possui menor valor de *overshoot*, igual a 1,08, valor de M_s = 2,09 e IAE = 10,09. Diferente do método CHR, que apresentou uma curva com *overshoot* em torno de 1,8, M_s = 8,47 e IAE = 81,46. Para a perturbação de carga, também se observou que o método SIMC obteve melhor resposta, com menor *overshoot* (em torno de 0,007) e com sua curva voltando ao valor de origem (zero) em menos tempo.

Com o cálculo dos controladores e as simulações realizadas, definiu-se o critério de escolha para os controladores e aplicou-se esse critério na rotina implementada no MATLAB. Com os experimentos realizados, foi possível confirmar os desempenhos de regime permanente obtidos com as simulações. Os controladores obtiveram desempenho satisfatório, com pequenos valores de *overshoot* e tempos de acomodação dentro dos limites esperados, pois atuaram na planta fazendo com que a temperatura atingisse a temperatura de referência definida pelo usuário e, em caso de perturbação, também obtiveram êxito, retornando à temperatura de referência.

5.1 TRABALHOS FUTUROS

Quanto aos trabalhos futuros, as seguintes etapas podem ser realizadas:

- a utilização de dois ou mais pontos de sensoriamento, para comparação dos valores de temperatura medidos em diferentes pontos dentro da estufa.
- a realização de testes com diferentes tipos de perturbações;
- a análise e implementação de diferentes métodos de sintonia para o cálculo dos controladores.
- a análise de novos índices de desempenhos para verificar a performance do sistema.
- a análise de instabilidade do sistema em estudo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Anacleto, A. M. C. *Temperatura e sua Medição*. Dissertação de Mestrado em Física para o Ensino, Faculdade de Ciências da Universidade do Porto, 2007.

[2] Ogata, K. *Engenharia de Controle Moderno*. 4ed. São Paulo: Prentice Hall, 2003.

[3] Kadlec, P., Gabrys, B., Strandt, S. *Data-driven Soft Sensors in the Process Industry*. Computers and Chemical Engineering, pp. 795-814, 2008.

[4] Zanata, D. R. P. *Desenvolvimento de Sensor Virtual Empregando Redes Neurais para Medição da Composição em uma Coluna de Destilação*. Dissertação de Mestrado em Engenharia (Engenharia de Sistemas), Escola Politécnica, Universidade de São Paulo, São Paulo, 229p, 2005.

[5] Soares, F. M. *Aplicação de Sensores Virtuais na Inferência da Temperatura de Banho no Processo de Fabricação de Alumínio Primário*. Dissertação de mestrado em Engenharia Elétrica, Universidade Federal do Pará, Belém, 2009.

[6] Choi, D. J., Park, H. *A hybrid Artificial Neural Network as a Software Sensor for Optimal Control of a Wastewater Treatment Process*. Water Research, v.35, n.16, p.3959-3967, 2001.

[7] Atkinson, C. M., Long, T. W., Hanzevack, E. L. *Virtual Sensing : A neural Network-Based Intelligent Performance and Emissions Prediction System for On-Board Diagnostics and Engine Control.* In: International Congress and Exposition, Detroit, Michigan, USA, p.1-12, 1998.

[8] Fortuna, L., Graziani, S., Rizzo, A., Xibilia, M. G. *Soft Sensor for Monitoring and Control of Industrial Processes.* Londres, Editora Springer, 270p, 2007.

[9] Abusnina, A. *Adaptive Soft Sensors in Industry. EngD Qualifying Dissertation*. Department of Computer Science Large Scale Complex IT Systems, Janeiro, 2013.

[10] Amazouz, M. and Pantea, R. *Use of Multivariate Data Analysis for Lumber Drying Process Monitoring and Fault Detection. International Conference on Data Mining*. DMIN, Las Vegas, Nevada, USA, Junho, 2006.

[11] Reda, S., Cochran, R. J., Nowroz, A. N. *Improved Thermal Tracking for Processors Using Hard and Soft Sensor Allocation Techniques*. IEEE Transactions on Computers, Vol. 60, NO. 6. Junho, 2011.

[12] Thanayankizil, L. V., Ghai, S. K., Chakraborty, D., Seetharam, D. P. *Softgreen-Towards Energy Management of Green Office Buildings with Soft Sensors*. IEEE Fourth International Conference on Communication Systems and Networks (COMSNETS), Janeiro, 2012.

[13] Júnior, C. A. A. L. *Desenvolvimento de um Sensor Virtual Baseado em Redes Neurais Artificiais para Previsão do Oxigênio Consumido no Processo de Tratamento de Água*. Monografia de Graduação em Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, Dezembro, 2010.

[14] Santos, R. L. *Modelagem de um Sensor Virtual para Controle do Conforto Térmico*. Dissertação de Mestrado em Engenharia, Faculdade de Tecnologia, Universidade de Brasília, Brasília, Março, 2011.

[15] Gonçalves, L. C. *Configuração Interferómetrica Diferencial para Medição de Deformação e Temperatura.* Dissertação de Mestrado em Engenharia de Telecomunicações e Redes, Universidade da Madeira, Outubro, 2010.

[16] James, S. C., Legge, R. L., Budman, H. *On-line Estimation in Bioreactors: A review. Reviews in Chemical Engineering*, v.16, n.4, p.311-340, 2000.

[17] Bernini, C. C. *Implementação em Hardware/Filmware de um Sensor Virtual Utilizando Algoritmo de Identificação Nebulosa*. Dissertação de Mestrado em Engenharia, Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

[18] Lotufo, F. A. *Desenvolvimento de um Sensor Virtual para Processos Não-Lineares e Variantes no Tempo, com Aplicação em Planta de Neutralização de pH*. Tese de Doutorado em Engenharia Mecânica, Guaratinguetá, 2010.

[19] Júnior, A. A. M. *Elaboração de um Analisador Virtual Utilizando Sistema Híbrido Neuro-Fuzzy para Inferir a Composição num Processo de Destilação*. Dissertação de Mestrado em Engenharia Química, Universidade Federal de Alagoas, Maceió, 2011.

[20] Souza, S. S. *Controle de Nível e Temperatura com Aplicação do Ambiente LABVIEW.* Dissertação de Mestrado em Engenharia Mecânica, Universidade Federal do Rio de Janeiro, Maio, 2002.

[21] Ribeiro, M. A. *Controle de Processos.* Tek Treinamento & Consultoria, 8ª Edição, Salvador, Bahia, 2005.

[22] Melo, M. M. *Modelagem de uma estufa térmica e sintonia do controlador PID.* Monografia e seminário do curso de Engenharia Elétrica, Universidade Federal de Viçosa, Minas Gerais, Dezembro, 2007.

[23] Campestrini, L. *Sintonia de Controladores PID Descentralizados Baseada no Método do Ponto Crítico.* Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006.

[24] Carrara, V. *Análise e Controle de Sistemas Lineares*. Instituto Nacional de Pesquisas Espaciais- INPE. São José dos Campos, 2012.

[25] Lourenço, J. *Sintonia de Controladores PID*. Escola Superior de Tecnologia. Realizado em Janeiro de 96 e revisto em Janeiro de 97.

[26] Aguirre, L. A. *Introdução à Identificação de Sistemas: Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais.* Editora da UFMG, 1ed, Belo Horizonte, 2000.

[27] Garcia, C. *Modelagem e Simulação de Processo Industriais e de Sistemas Eletromecânicos*. Editora da Universidade de São Paulo, 2ed, São Paulo, 2005.

[28] Santos, J. E. S. *Controle Preditivo Não-Linear para Sistemas de Hammerstein*. Tese de Doutorado em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Abril de 2007.

[29] Cassandras, C. G., Lafortune, S. *Introduction to Discrete Event Systems*. Springer Science-Business Media, LLC, 2008.

[30] Coelho, A. A. R., Coelho, L. S. *Identificação de Sistemas Dinâmicos Lineares*. Universidade Federal de Santa Catarina (UFSC), Editora da UFSC, Florianópolis, 2004.

[31] Silva, A. S. *Fundamentos do controle clássico*. Universidade Federal de Santa Catarina, 2008.

[32] National Semicondutor. *LM35 Precision Centigrade Temperature Sensors*, 2000.

[33] Agilent Technologies. *Agilent U2500A User's Guide*. Agilent Technologies, 2ed, 2008.

[34] Skogestad, S. *Simple Analytic Rules for Model Reduction and PID Controller Tuning*. In: Journal of Process Control 13, pp. 291 – 309, 2003.

[35] Abdallah, Y. M. *Sintonia de Controlador PID via Procedimento Adaptativo para Controle de Atitude de Veículos Lançadores*. Dissertação de Mestrado em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle. INPE, São José dos Campos, São Paulo, 2004.

[36] Dorf, R. C. e Bishop, R. H. *Sistemas de Controle Modernos*. 11ed. Rio de Janeiro: LTC, 2009.

[37] Zhao, Z., Liu, Z. Zhang, J. *IMC-PID Tuning Method Based on Sensitivity Specification for Process with Time-Delay.* In: J. Cent. South Univ. Technol. 18, pp. 1153−1160, 2011.

[38] Monteiro, N. A. B., Silva, J. J., Rocha Neto, J. S. Monitoramento e Controle de Temperatura de uma Estufa Utilizando Sensor Virtual. In: XLII Congresso Brasileiro de Educação em Engenharia-COBENGE. Juiz de Fora, Minas Gerais, 2014.

ANEXO A-ROTINA PARA CÁLCULO DO CONTROLADOR PELO MÉTODO SIMC

```
%% Projeto dos Controladores PI e PID pelo Método SIMC
clear all
close all

%Parâmetros do Processo: Ganho, Cte. de Tempo e Atraso
kp = 17.762;
t1 = 1618.1;
l = 0.983;

%Aproximação de Primeira Ordem com Atraso do Processo
num_g_aprox_1 = [kp];
den_g_aprox_1 = [t1 1];
Gp_s  = tf (num_g_aprox_1, den_g_aprox_1, 'InputDelay', l)
Gd_s = Gp_s; %Modelando o modelo da perturbação

%PID
kppid =  17.75;
t1pid = 1605.1;
t2 = 44.326;
lpid = 0.95;

%Aproximação de Segunda Ordem com Atraso do Processo
num_g_aprox_2 = [kppid];
den_g_aprox_2 = [(t1pid * t2)  (t1pid + t2)  1];
G_s_modelo_2 = tf (num_g_aprox_2, den_g_aprox_2, 'InputDelay', lpid)
Gd_spid = G_s_modelo_2;

%Parâmetros do Controlador
%PI
k_simc =  (0.5/ kp) * (t1/l);
ti_simc = min (t1, (8*l));

%Controlador
num_control_simc = [k_simc   k_simc /ti_simc];
den_control_simc  = [1  0];
C_s_simc = tf (num_control_simc, den_control_simc);

Gc_s_simc = tf (num_control_simc, den_control_simc);
L_s_simc = Gp_s * Gc_s_simc;

%PID
k_pid =  (0.5/ kppid) * (t1pid/lpid);
ti_pid = min (t1pid, (8*lpid));
td_pid = t2;

num_control_pid = [(k_pid * td_pid)  k_pid  (k_pid/ti_pid)];
den_control_pid  = [1  0];

C_s_pid = tf (num_control_pid, den_control_pid);
L_s_pid = G_s_modelo_2 * C_s_pid;

% Rastreamento do Sinal de Referência
T_s_simc = feedback (ss(L_s_simc),1);
T_s_simc_pid = feedback (ss(L_s_pid),1);
```

```
%Perturbação de Carga
Td_s_simc =  Gd_s*feedback(1,ss(L_s_simc));
Td_s_pid =  Gd_s*feedback(1,ss(L_s_pid));

%FIGURAS
%Resposta ao Degrau (Variação no Sinal de Referência)
figure (1), step(T_s_simc), title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método SIMC c/ Controlador PI(Variação no Sinal de
Referência)'))
figure (2), step(T_s_simc_pid), title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método SIMC c/ Controlador PID(Variação no Sinal de
Referência)'))

%Resposta ao Degrau (Perturbação de Carga)
figure (3),   step(Td_s_simc),title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método SIMC c/ Controlador PI(Perturbação de Carga)'))
figure (4),   step(Td_s_pid),title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método SIMC c/ Controlador PID(Perturbação de
Carga)'))

%%% ÍNDICES DE DESEMPENHO
% PI
%Ms - Valor de pico da função de sensibilidade
Ms_pi_ref = max(abs(bode(1/(1 + ss(T_s_simc)))));
Ms_pi_pert = max(abs(bode(1/(1 + ss(Td_s_simc)))));

%IAE - Integral do Erro Absoluto
    [y1,t_1]= step(T_s_simc);
    IAE_DS_PI_ref = trapz(t_1,abs(y1-1));

    [y2,t_2]= step(Td_s_simc);
    IAE_DS_PI_pert = trapz(t_2,abs(y2-1));

% PID
%Ms
Ms_pid_ref = max(abs(bode(1/(1 + ss(T_s_simc_pid)))));
Ms_pid_pert = max(abs(bode(1/(1 + ss(Td_s_pid)))));

%IAE
    [y5,t_5]= step(T_s_simc_pid);
    IAE_DS_PID_ref = trapz(t_5,abs(y5-1));

    [y6,t_6]= step(Td_s_pid);
    IAE_DS_PID_pert = trapz(t_6,abs(y6-1));
```

ANEXO B-ROTINA PARA CÁLCULO DO CONTROLADOR PELO MÉTODO CHR

```
%%%% Projeto dos Controladores PI e PID pelo Método CHR
clear all
close all

%Parâmetros do Processo: Ganho, Cte. de Tempo e Atraso
%PI
kp =  17.762;
t1 =  1618.1;
l =  0.983;

%Aproximação de Primeira Ordem com Atraso do Processo
num_g_aprox_1 = [kp];
den_g_aprox_1 = [t1 1];
Gp_s  = tf (num_g_aprox_1, den_g_aprox_1, 'InputDelay', l)
Gd_s = Gp_s;

%PID
kppid =  17.75;
t1pid = 1605.1;
t2 = 44.326;
lpid = 0.95;

%Aproximação de Segunda Ordem com Atraso do Processo
num_g_aprox_2 = [kppid];
den_g_aprox_2 = [(t1pid * t2)  (t1pid + t2)  1];
G_s_modelo_2 = tf (num_g_aprox_2, den_g_aprox_2, 'InputDelay', lpid)
Gd_spid = G_s_modelo_2;

%Parâmetros do Controlador
%PI
k_chr_pi =  (0.35 * t1)/(kp*l);
ti_chr_pi = (1.2*t1);

num_control_chr_pi = [k_chr_pi   k_chr_pi /ti_chr_pi];
den_control_chr_pi  = [1  0];

C_s_chr_pi = tf (num_control_chr_pi, den_control_chr_pi);
L_s_chr_pi = Gp_s * C_s_chr_pi;

%PID
k_chr_pid =  (0.6*t1pid)/(kppid*lpid);
ti_chr_pid = t1pid;
td_chr_pid = 0.5*lpid;

num_control_chr_pid = [(k_chr_pid * td_chr_pid)  k_chr_pid
(k_chr_pid/ti_chr_pid)];
den_control_chr_pid  = [1  0];

C_s_chr_pid = tf (num_control_chr_pid, den_control_chr_pid);
L_s_chr_pid = G_s_modelo_2 * C_s_chr_pid;

% Rastreamento do Sinal de Referência
T_s_chr_pi = feedback (ss(L_s_chr_pi),1);
T_s_chr_pid = feedback (ss(L_s_chr_pid),1);
```

```
%Perturbação de Carga
Td_s_chr_pi =  Gd_s*feedback(1,ss(L_s_chr_pi));
Td_s_chr_pid =  Gd_s*feedback(1,ss(L_s_chr_pid));

%FIGURAS
%Resposta ao Degrau (Variação no Sinal de Referência)
figure (1), step(T_s_chr_pi), title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método CHR c/ Controlador PI(Variação no Sinal de
Referência)'))
figure (2), step(T_s_chr_pid), title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método CHR c/ Controlador PID(Variação no Sinal de
Referência)'))

%Resposta ao Degrau (Perturbação de Carga)
figure (3),  step(Td_s_chr_pi),title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método CHR c/ Controlador PI(Perturbação de Carga)'))
figure (4),  step(Td_s_chr_pid),title(sprintf('Resposta ao Degrau do Sistema em
Malha Fechada \n usando o Método CHR c/ Controlador PID(Perturbação de Carga)'))

% ÍNDICES DE DESEMPENHO
% PI
% Ms - Valor de pico da função de sensibilidade
Ms_pi_ref = max(abs(bode(1/(1 + ss(T_s_chr_pi)))));
Ms_pi_pert = max(abs(bode(1/(1 + ss(Td_s_chr_pi)))));

%IAE - Integral do Erro Absoluto
    [y1,t_1]= step(T_s_chr_pi);
    IAE_DS_PI_ref = trapz(t_1,abs(y1-1));

    [y2,t_2]= step(Td_s_chr_pi);
    IAE_DS_PI_pert = trapz(t_2,abs(y2-1));

% PID
Ms_pid_ref = max(abs(bode(1/(1 + ss(T_s_chr_pid)))));
Ms_pid_pert = max(abs(bode(1/(1 + ss(Td_s_chr_pid)))));

%IAE
    [y5,t_5]= step(T_s_chr_pid);
    IAE_DS_PID_ref = trapz(t_5,abs(y5-1));

    [y6,t_6]= step(Td_s_chr_pid);
    IAE_DS_PID_pert = trapz(t_6,abs(y6-1));
```

ANEXO C-ROTINA DE MONITORAMENTO COM INTERFACE GRÁFICA

```
function varargout = InterfaceMonit(varargin)
% Inicialização do código
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfaceMonit_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfaceMonit_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fim da inicialização do código
% Executa antes da interface tornar-se visível.
function InterfaceMonit_OpeningFcn(hObject, eventdata, handles, varargin)

axes(handles.axes1);
grid on

handles.output = hObject;

guidata(hObject, handles);

function varargout = InterfaceMonit_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function tensao_Callback(hObject, eventdata, handles)

% Executa durante a criação do objeto, depois de definir as propriedades
function tensao_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tempo_Callback(hObject, eventdata, handles)

function tempo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Executa quando o botão 'parar' é pressionado.
function parar_Callback(hObject, eventdata, handles)
set(handles.parar,'Enable','off');
```

```
% Inicia o programa quando o botão 'início' é pressionado.
function Start_Callback(hObject, eventdata, handles)

axes(handles.axes1);
cla; %limpa figura para início do programa

%inicia a placa/Cria entrada e saída analógica e adiciona canal a elas
out=daqhwinfo('agilentu2500');
out.ObjectConstructorName(:)
ai=analoginput('agilentu2500');
ai2=analoginput('agilentu2500');
ao=analogoutput('agilentu2500');
addchannel(ai, 0)
addchannel(ai2, 1)
addchannel(ao, 0)
%Definição de variáveis
SampleRate = 3;
tensaoaplicada = str2num(get(handles.tensao,'String'));
AcquisitionTime = str2num(get(handles.tempo,'String'));
%Configura a entrada analógica
set (ai, 'SampleRate', SampleRate)
set (ai, 'samplespertrigger', SampleRate*AcquisitionTime)  % inf para adquirir
dados continuamente
set (ai2, 'SampleRate', SampleRate)
set (ai2, 'samplespertrigger', SampleRate*AcquisitionTime)  % inf para adquirir
dados continuamente

%início do laço para aquisição
tic;
while toc < AcquisitionTime
    putsample(ao,tensaoaplicada)   %Aplica a tensão. Faixa de 0~5V e passo 0.5V
    lm35 = (getsample (ai))/0.05;
    i = ceil(toc)
    VetorTemperatura(i)= lm35      %Vetor com os valores de temperatura
    TensaoCapacitor = getsample (ai2)
    % Plota o gráfico tempo por temperatura
    tempo=0:1:length(VetorTemperatura)-1;
    plot(tempo,VetorTemperatura);
    title ('Monitoramento')
    xlabel('Tempo (s)');
    ylabel('Temperatura (°C)');
    drawnow
    %parar processo caso o botão 'parar' seja pressionado
    if strcmp('off',get(handles.parar,'Enable')) == 1
        break;
    end
    %teste para saber quando o sinal do lm35 foi perdido
    if TensaoCapacitor < 0.2
        break
    end
end
%Cálculo do tempo do processo
TempodeExecucao = toc;
horas = floor(TempodeExecucao/3600);
minutos = mod(floor(TempodeExecucao/60),60);
segundos = round(mod(TempodeExecucao,60));

%Condições para se monitorar a temperatura. Se já estiver passado de uma hora
%(3600 segundos) da tensão sendo aplicada, caso o processo pare, é possível
%inferir um valor de temperatura.
if toc > 3600
    switch SampleRate == 3
        case tensaoaplicada == 0
```

```
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 26.5 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 0.5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 28.9 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 1
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 30.6 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 1.5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 34.5 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 2
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 38.9 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 2.5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 43.9 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 3
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 52.5 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 3.5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 64.2 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 4
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 69.9 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 4.5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 71.2 graus celsius'], 'Monitoramento');
        case tensaoaplicada == 5
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
em torno de 72.5 graus celsius'], 'Monitoramento');
        otherwise
            h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
não estabilizada.'], 'Monitoramento')
    end
else
    h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) '
num2str(minutos)  ' minuto(s) e ' num2str(segundos) ' segundos. ' 'Temperatura
não estabilizada.'], 'Monitoramento')
end
stop(ai2);
delete(ai2);
stop(ao);
delete(ao);
daqfind;
delete(daqfind);
```

ANEXO D-ROTINA DE CONTROLE COM INTERFACE GRÁFICA

```
function varargout = InterfaceControlDoisSP(varargin)

% Inicialização do código
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfaceControlDoisSP_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfaceControlDoisSP_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Fim da inicialização

% Executa antes da interface tornar-se visível.
function InterfaceControlDoisSP_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;

guidata(hObject, handles);

function varargout = InterfaceControlDoisSP_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;

function tempo_Callback(hObject, eventdata, handles)

function tempo_CreateFcn(hObject, eventdata, handles)

      if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
      end

% Executa quando botão 'parar' é pressionado.
function parar_Callback(hObject, eventdata, handles)
set(handles.parar,'Enable','off');

function setpoint1_Callback(hObject, eventdata, handles)

function setpoint1_CreateFcn(hObject, eventdata, handles)

      if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
      end

function setpoint2_Callback(hObject, eventdata, handles)

function setpoint2_CreateFcn(hObject, eventdata, handles)
```

```
      if ispc && isequal(get(hObject,'BackgroundColor'),
      get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
      end

% Executa quando botão 'início' é pressionado.
function inicio_Callback(hObject, eventdata, handles)

axes(handles.axes1);
cla; %limpa figura para inicio do programa

%inicia a placa/Cria entrada e saída analógica e adiciona canal a elas
out=daqhwinfo('agilentu2500');
out.ObjectConstructorName(:)
ai=analoginput('agilentu2500');
ao=analogoutput('agilentu2500');
addchannel(ai, 0)
addchannel(ao, 0)

%Definição de variáveis
%pi
kpchr = 32.44;
tichr = 1941.72;

%pid
kpsimc = 47.59;
tisimc = 7.6;
tdsimc = 44.32;

SampleRate = 3;
lm35 = (getsample (ai))/0.05;
controlador = 5;
sp1 = str2num(get(handles.setpoint1,'String'));
sp2 = str2num(get(handles.setpoint2,'String'));
difsp = sp2- sp1;
AcquisitionTime = str2num(get(handles.tempo,'String'));
ta = 0.33;

%Configura a entrada analógica
set (ai, 'SampleRate', SampleRate)
set (ai, 'samplespertrigger', SampleRate*AcquisitionTime)  % inf para adquirir
dados continuamente

if sp2 == 0
    %início do laço para aquisição
    tic;
    while toc < AcquisitionTime

        %As variáveis são atualizadas
        lm35 = (getsample(ai))/0.05;
        i = (ceil(toc))+1;
        VetorTemperatura(i)= lm35;
        vetorsp1(i) = sp1;

        if sp1 < 38.9

            %início da rotina de controle
            erro(i) = vetorsp1(i) - VetorTemperatura(i);
            controlador (i) = controlador(i-1) + kpchr*[(1+ta/tichr)*(erro(i))];
```

```
            %Níveis de tensões aceitáveis no DAQ
            if controlador(i) >= 5
                controlador(i) = 5;
            elseif controlador(i) <=0;
                controlador(i) = 0;
            end

            sinaldecontrole = controlador(i)
            putsample(ao, sinaldecontrole);
            %fim da rotina de controle

            % Plota o gráfico tempo(s) por temperatura(°C)
            tempo=0:1:length(VetorTemperatura)-1;
            plot(tempo,VetorTemperatura,'r',tempo, vetorsp1,'g');
            title ('Controle')
            xlabel('Tempo (s)');
            ylabel('Temperatura (°C)')
            drawnow

        else
            if sp1 >= 38.9

                %PID
                %início da rotina de controle
                erro(i) = vetorsp1(i) - VetorTemperatura(i);
                controlador(i) = controlador(i-1)+kpsimc*erro(i)*(1+(ta/tisimc))
+kpsimc*tdsimc*(erro(i)-erro(i-1));

                %Níveis de tensões aceitáveis no DAQ
                if controlador(i) >= 5
                    controlador(i) = 5;
                elseif controlador(i) <=0;
                    controlador(i) = 0;
                end

                sinaldecontrole = controlador(i)
                putsample(ao, sinaldecontrole);
                %fim da rotina de controle

                % Plota o gráfico tempo(s) por temperatura(°C)
                tempo=0:1:length(VetorTemperatura)-1;
                plot(tempo,VetorTemperatura,'r',tempo, vetorsp1,'g');
                title ('Controle')
                xlabel('Tempo (s)');
                ylabel('Temperatura (°C)')
                drawnow
            end
        end

        % parar processo caso o botão 'parar' seja pressionado
        if strcmp('off',get(handles.parar,'Enable')) == 1
            break;
        end
    end
else

    tic;
    while toc < AcquisitionTime

        %As variáveis são atualizadas
        lm35 = (getsample(ai))/0.05;
        i = (ceil(toc))+1;
```

```
        VetorTemperatura(i)= lm35;
        vetorsp1(i) = sp1;
        vetorsp2(i) = sp2;

        if sp1 < 38.9 && toc < 1600

            %início da rotina de controle
            erro(i) = vetorsp1(i) - VetorTemperatura(i);
            controlador (i) = controlador(i-1) + kpchr*[(1+ta/tichr)*(erro(i))];

            %Níveis aceitáveis
            if controlador(i) >= 5
                controlador(i) = 5;
            elseif controlador(i) <=0;
                controlador(i) = 0;
            end

            sinaldecontrole = controlador(i)
            putsample(ao, sinaldecontrole);
            %fim da rotina de controle

            % Plota o gráfico tempo por temperatura
            tempo=0:1:length(VetorTemperatura)-1;
            plot(tempo,VetorTemperatura,'r',tempo,vetorsp1,'g',tempo,
vetorsp2,'y');
            title ('Controle')
            xlabel('Tempo (s)');
            ylabel('Temperatura (°C)')
            drawnow

        else
            if sp1 >= 38.9 && toc < 1600

                %PID
                %início da rotina de controle
                erro(i) = vetorsp1(i) - VetorTemperatura(i);
                controlador(i)=controlador(i-1) + kpsimc*erro(i)*(1+(ta/tisimc))
+ kpsimc*tdsimc*(erro(i)-erro(i-1));

                %Níveis aceitáveis
                if controlador(i) >= 5
                    controlador(i) = 5;
                elseif controlador(i) <=0;
                    controlador(i) = 0;
                end

                sinaldecontrole = controlador(i)
                putsample(ao, sinaldecontrole);
                %fim da rotina de controle

                % Plota o gráfico tempo por temperatura
                tempo=0:1:length(VetorTemperatura)-1;
                plot(tempo,VetorTemperatura,'r',tempo, vetorsp1,'g',tempo,
vetorsp2,'y');
                title ('Controle')
                xlabel('Tempo (s)');
                ylabel('Temperatura (°C)')
                drawnow

            else
                if difsp <= 31.3 && toc >= 1600
```

```
                    %início da rotina de controle
                    erro(i) = vetorsp2(i) - VetorTemperatura(i);
                    controlador(i)=controlador(i-
1)+kpchr*[(1+ta/tichr)*(erro(i))];

                    %Níveis aceitáveis
                    if controlador(i) >= 5
                        controlador(i) = 5;
                    elseif controlador(i) <=0;
                        controlador(i) = 0;
                    end

                    sinaldecontrole = controlador(i)
                    putsample(ao, sinaldecontrole);
                    %fim da rotina de controle

                    % Plota o gráfico tempo por temperatura
                    tempo=0:1:length(VetorTemperatura)-1;
                    plot(tempo,VetorTemperatura,'r',tempo, vetorsp1,'g',tempo,
vetorsp2,'y');
                    title ('Controle')
                    xlabel('Tempo (s)');
                    ylabel('Temperatura (°C)')
                    drawnow

                else
                    if difsp > 31.3 && toc >= 1600

                        %início da rotina de controle
                        erro(i) = vetorsp2(i) - VetorTemperatura(i);
                        controlador (i) = controlador(i-1) +
kpsimc*erro(i)*(1+(ta/tisimc)) + kpsimc*tdsimc*(erro(i)-erro(i-1));

                        %Níveis aceitáveis
                        if controlador(i) >= 5
                            controlador(i) = 5;
                        elseif controlador(i) <=0;
                            controlador(i) = 0;
                        end

                        sinaldecontrole = controlador(i)
                        putsample(ao, sinaldecontrole);
                        %fim da rotina de controle

                        % Plota o gráfico tempo por temperatura
                        tempo=0:1:length(VetorTemperatura)-1;
                        plot(tempo,VetorTemperatura,'r',tempo,
vetorsp1,'g',tempo, vetorsp2,'y');
                        title ('Controle')
                        xlabel('Tempo (s)');
                        ylabel('Temperatura (°C)')
                        drawnow
                    end

                end

            end
        end
```

```
        %parar processo caso o botão 'parar' seja pressionado
        if strcmp('off',get(handles.parar,'Enable')) == 1
            break;
        end
    end
    end

%Cálculo do tempo do processo
TempodeExecucao = toc;
horas = floor(TempodeExecucao/3600);
minutos = mod(floor(TempodeExecucao/60),60);
segundos = round(mod(TempodeExecucao,60));

h = msgbox(['Duração do processo: ' num2str(horas) ' hora(s) ' num2str(minutos)
' minuto(s) e ' num2str(segundos) ' segundos.'])

stop(ao);
delete(ao);
daqfind;
delete(daqfind);
```