UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

ROBERTO NAVARRO KOEPSEL

MÓDULO EXPERIMENTAL PARA SINTONIA DE SISTEMA DE CONTROLE ANTECIPATÓRIO

Maringá – PR – Brasil

Agosto de 2018

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

MÓDULO EXPERIMENTAL PARA SINTONIA DE SISTEMA DE CONTROLE ANTECIPATÓRIO

Roberto Navarro Koepsel
Eng. Eletricista, UEM, 2015
Orientador. Prof. Dr. Marcos de Souza
Coorientador. Prof. Dr. Cid Marcos Gonçalves Andrade

Dissertação de Mestrado submetida à Universidade Estadual de Maringá, como parte dos requisitos necessários para a obtenção do Grau de Mestre em Engenharia Química, área de Modelagem, Controle, Automação de Processos.

Maringá – PR – Brasil

Agosto de 2018

Dados Internacionais de Catalogação na Publicação (CIP)
(Biblioteca Central - UEM, Maringá, PR, Brasil)

K78m

Koepsel, Roberto Navarro
Módulo experimental para sintonia de sistema de controle antecipatório / Roberto Navarro Koepsel. -- Maringá, 2018.
70 f. : il. color., figs., tabs.

Orientador: Prof. Dr. Marcos de Souza.
Coorientador: Prof. Dr. Cid Marcos Gonçalves Andrade.
Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Engenharia Química, Programa de Pós-Graduação em Engenharia Química, 2018.

1. *Feedforward*. 2. Sistemas de Controle Antecipatório. 3. Aquecimento. 4. Arduino. 5. Scilab. 6. Temperatura. I. Souza, Marcos de, orient. II. Andrade, Cid Marcos Gonçalves, coorient. III. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Engenharia Química. Programa de Pós-Graduação em Engenharia Química. IV. Título.

CDD 21.ed. 660.2815

Mariza Nogami CRB 9/1569

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

Esta é a versão final da Dissertação de Mestrado apresentada por Roberto Navarro Koepsel perante a Comissão Julgadora do Curso de Mestrado em Engenharia Química em 24 de agosto de 2018.

COMISSÃO JULGADORA

Prof. Dr. Marcos de Souza
Orientador / Presidente

Prof. Dr. Cid Marcos Gonçalves Andrade
Coorientador

Prof. Dr. Carlos Alexandre Ferri
Membro

Prof. Dr. Abel Fidalgo Alves
Membro

DEDICATÓRIA

Dedico este trabalho aos meus pais Eliana Claudia Navarro Koepsel e Roberto Koepsel que sempre me deram apoio e incentivo, permitindo assim a sua realização.

AGRADECIMENTOS

A minha família, especialmente aos meus pais Roberto Koepsel e Eliana Claudia Navarro Koepsel e à Andreia Navarro Koepsel, minha irmã, pela paciência e suporte durante a elaboração deste trabalho.

Ao meu orientador, Marcos de Souza, ao meu coorientador, Cid Marcos Gonçalves Andrade, pela orientação, confiança e comprometimento.

MÓDULO EXPERIMENTAL PARA SINTONIA DE SISTEMA DE CONTROLE ANTECIPATÓRIO

AUTOR: ROBERTO NAVARRO KOEPSEL

ORIENTADOR: MARCOS DE SOUZA

COORIENTADOR: CID MARCOS GONÇALVES ANDRADE

Dissertação de Mestrado; Programa de Pós-Graduação em Engenharia Química; Universidade Estadual de Maringá; Av. Colombo, 5790, BL E46 – 09; CEP: 87020-900 – Maringá – PR, Brasil, defendida em 24 de agosto de 2018.

RESUMO

Este trabalho apresenta e analisa o processo de construção de um sistema de aquecimento com controle do tipo antecipatório, composto por um tanque de acrílico, uma resistência elétrica, sensores de temperatura, a plataforma Arduino para realizar as conversões analógicas e digitais, um computador responsável por executar ação de controle via software Scilab e um sistema de aquecimento de forma a efetuar perturbações na entrada. Um sistema de controle é caracterizado de forma simples por um processo e um controlador, o qual a partir do erro gerado pela diferença entre o *set point* e a variável controlada, atua sobre a variável manipulada no processo. A utilização de um módulo experimental de um sistema de controle permite implementar os conceitos teóricos em práticas no laboratório. A motivação para a escolha dos softwares e hardware objetivou garantir liberdade e redução de custos. Os resultados mostraram viável a construção de um sistema de controle com uma interface amigável que permite acompanhar todo o processo e um sistema que pode ser facilmente modelado, apresentando excelente estabilidade frente às perturbações simuladas.

Palavras Chave: *Feedforward*; Sistema de Controle Antecipatório; Aquecimento; Arduino; Scilab; Temperatura.

EXPERIMENTAL MODULE FOR TUNING OF ANTECIPATORY CONTROL SYSTEM

AUTHOR: ROBERTO NAVARRO KOEPSEL

SUPERVISOR: MARCOS DE SOUZA

COSUPERVISOR: CID MARCOS GONÇALVES ANDRADE

Master Thesis; Chemical Engineering Graduate Program; State University of Maringá; Av Colombo, 5790, BL E-46 – 09; CEP: 87020-900 – Maringá – PR, Brazil, presented on 24th August 2018.

ABSTRACT

This study presents and analyzes the process of building a heating system of anticipatory type, consisting of an acrylic tank, electric resistance, temperature sensors, the Arduino platform to convert the analog and digital and a computer responsible for execute control action through Scilab software. A heating system has been developed at the entrance to cause disturbances at the entrance. A control system is characterized in a simple way by a process and a controller, which from the failure generated by the difference between the set point and the controlled variable, acts over the variable manipulated in the process. The use of an experimental module of a control system makes implements theoretical concepts in practices carried out in the lab. The motivation for choosing the software and hardware was to guarantee freedom and cost reduction. The results showed a control system with a user-friendly interface that allows to follow the whole process and a system that can be easily modeled, presenting excellent stability against the simulated perturbations.

Keywords: Feedforward; Antecipatory control; Heat; Arduino; Scilab; Temperature.

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A utilização de sistemas de controle é fundamental para aprimorar os processos industriais, de forma a tornar a produção cada vez mais competitiva e eficiente, evitando perdas. Cada vez mais os estudos sobre os sistemas de controle têm sido valorizados para obtenção da máxima eficiência através da modelagem, do controle e da simulação do processo.

No aspecto de antecipação, o controle *feedforward* é um dos métodos empregados em processos em que a malha não possui realimentação, dessa forma, o controlador necessita prever todas as interações presentes no processo para que a atuação consiga prever futuras perturbações.

Para entender a importância desse método basta conhecer, sob o ponto de vista da engenharia de controle de processos, quando seu uso se torna preferido, ou muitas vezes necessário. O primeiro caso, quando os distúrbios e cargas na entrada do processo levam um tempo considerável para afetar a variável controlada na saída. Ou ainda, em casos em que as variáveis na entrada que afetam significativamente a variável controlada são possíveis de serem medidas. Outro exemplo, os processos em que as suas equações termodinâmicas de balanço de matéria e energia são facilmente resolvidas teoricamente. Também para processos em que não é possível analisar os dados de saída, apenas os de sua entrada (CANIATO, 2006).

Assim, sabendo da importância do estudo de sistemas de controle, muitos cursos de engenharia incluem em um componente curricular que abordam questões sobre controle. No qual são abordados fenômenos e ideias, por vezes, complexas e de difícil visualização, abordado frequentemente de forma abstrata. Devido a isso, as práticas em laboratório se tornam excelentes ferramentas para assimilação por parte dos estudantes desses conhecimentos (SCHMID, 2000).

Um dos primeiros estudos apontando a importância sobre a realização de práticas para o aprendizado data de 1980, no qual se aponta a eficácia no aprendizado quando o indivíduo está pessoalmente envolvido na experiência (ORD, 2012). Mais recentemente, Stefanovic et al., (2011) reafirma a importância do aprendizado através do “learning by doing”, no qual através da prática o aluno se familiariza com os conceitos vistos em controle permitindo testar os conhecimentos em relação à modelagem e à identificação de processos.

Sabendo da necessidade desses equipamentos para o ensino, da sua importância no controle *feedforward* e considerando a sua escassez, devido à falta de recursos, que torna limitada a aprendizagem do aluno, que o presente estudo propõe apresentar uma alternativa viável, considerando a dificuldade pelo custo elevado da compra desse tipo de módulo, por meio do desenvolvimento de um módulo de baixo custo que permita simular conceitos, muitas vezes complexos para o aluno, como é o caso da modelagem e da identificação de processos.

1.1 Objetivo

O trabalho tem por objetivo desenvolver um módulo experimental de controle de temperatura do tipo *feedforward* em um tanque utilizando o software livre Scilab e a plataforma Arduino. A interface pretendida deve permitir ao usuário modelar, controlar e simular o processo ao analisar os dados de entrada do sistema de forma a obter um controlador que antecipe e corrige futuras perturbações na saída.

Pretende-se com a construção do módulo disponibilizar uma importante ferramenta de baixo custo para o estudo de técnicas de controle em práticas de laboratório, que permitam ao aluno modelar, controlar e otimizar processos.

1.2 Organização do texto

Este texto encontra-se organizado em sete capítulos. O primeiro capítulo apresenta o tema do trabalho, sua relevância para as instituições de ensino, a motivação para a escolha do tema, os objetivos pretendidos e uma breve descrição da estrutura da dissertação.

O segundo capítulo apresenta uma revisão do controle de processos, aborda sua importância para o mercado global, os principais componentes que o compõe, bem como, realiza uma breve revisão do método de controle *feedforward*. Finaliza com a revisão de trabalhos desenvolvidos na área e discussão da escolha dos hardwares e softwares utilizados na criação do módulo.

O terceiro capítulo descreve os passos para a construção do módulo, assim como os materiais utilizados, os gastos referentes a sua construção e a comunicação entre os componentes do módulo.

No quarto capítulo são detalhados os passos da montagem dos módulos, o desenvolvimento do controlador e da interface amigável.

No quinto capítulo são apresentadas as etapas a serem seguidas pelo usuário para modelar, controlar, simular e obter um sistema eficiente que pretende ser utilizado em práticas laboratoriais.

No sexto capítulo apresenta e discute a interface desenvolvida via Scilab, bem como, as respostas do sistema frente as variações diretas na variável manipulada, assim, como, a modelagem do sistema e sua resposta às perturbações na entrada.

O sétimo capítulo finaliza com as conclusões obtidas e apresenta sugestões de aprimoramento para o módulo desenvolvido.

No Apêndice A é apresentado o código de programação utilizado na plataforma Arduino Uno para receber e enviar os sinais via comunicação serial para o software Scilab, através da porta USB do computador.

No Apêndice B é disponibilizado o código de programação do software Scilab, no código são programados: a interface, a comunicação com a plataforma Arduino e o controlador.

2. REVISÃO BIBLIOGRÁFICA

Os sistemas de controle são vastamente utilizados, vários são os trabalhos que abordam a sua importância para os processos industriais. Entende-se que é necessário conhecer os tipos e as características dos processos de controle mais utilizados e os trabalhos já desenvolvidos na área em vista de sanar possíveis lacunas, assim, como, seguir os acertos trilhados.

2.1 Sistemas Controle

A engenharia busca controlar os materiais e as forças da natureza em busca de benefícios para a humanidade. Engenheiros de sistemas de controle buscam entender e controlar os sistemas de seu ambiente. Esses dois conceitos, entender e controlar, são fundamentais para a área de sistemas de controle, pois para serem eficientes precisam ser entendidos e modelados (DORF, RICHARD C., 2011).

A utilização de sistemas de controle de forma geral permite manter as condições desejadas do processos através da manipulação de certas variáveis, pois visa ajustar a variável de interesse para o valor desejado (LEBLANC; COUGHANOWR, 2009)

As vantagens da automação de processos está relacionada a garantia de qualidade do produto, pois, quanto maior a sua qualidade, menor será a sua tolerância à variação de suas propriedades. Nesse aspecto, um sistema de controle pode garantir pequena variabilidade, proporciona a aquisição confiável dos dados de modo contínuo e preciso assegurando a quantidade e qualidade desejada do produto e a economia do processo. Essa qualidade desejada é viabilizada pela utilização dos sensores, no qual o sistema elimina superaquecimento de fornos, fornalhas ou secadores, por exemplo. Outro aspecto a considerar é quanto a segurança, através da automação de processos as máquinas passam a atuar em locais antes considerados de auto risco livrando da necessidade da presença humana. (RIBEIRO, 2005).

O controle de processos demanda a identificação das partes fundamentais de um sistema de controle, são eles: processo, elemento de medida, controlador e elemento final de controle. Esses elementos do sistema são os componentes presentes no processo estudado. (COUGHANOWR; KOPPEL, 1978).

Além dos termos citados vale ressaltar que outras denominações são muito utilizadas quando se trata de controle de processos: variável controlada, variável manipulada e distúrbio.

Variável controlada é a medida mais importante do processo, pois é a que pretende controlar o processo. Seu valor é sempre comparado com a referência dada pelo controlador, sendo que a diferença entre seu valor e a referência (*set point*) é definido como o erro do processo.

A variável manipulada é a variável que está ligada ao controlador e que será utilizada para manter a variável controlada dentro do valor estipulado, seu valor é atribuído pelo controlador baseado no erro encontrado. (OGATA, 2003).

As indústrias que realizam processos contínuos necessitam de uma alimentação ininterrupta na entrada do processo, na qual a condição para manter a produção de forma continuada depende da correção da variação na entrada do processo, pois, caso ocorra, seja identificada e corrigida, essa correção é realizada através da variável manipulada que reduz a influência da perturbação na variável controlada do processo. (OLIVEIRA, 1999).

Para processos grandes ou visando apenas a facilitação de cálculos é conveniente representar o sistema em um diagrama de blocos, como ilustrado na figura 1. Cada bloco permite representar uma relação de entrada e saída e que é expressa sob a forma de funções de transferência, sendo que essa relação entre a entrada e a saída representa a causa e efeito do processo. (COUGHANOWR; KOPPEL, 1978).

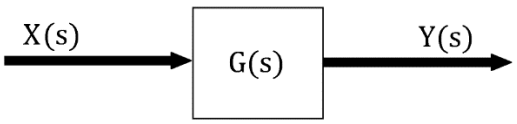

Figura 1. Representação de um bloco.

Sabendo da importância do controle de variáveis específicas pelo sistema de controle, as variáveis controladas devem estar relacionadas a variável de controle. Essa relação entre as variáveis é representada pela função de transferência do sistema (DORF, RICHARD C., 2011).

Para o bloco da figura 1 a função de transferência é dada através da relação do sinal que sai pelo que entra no bloco, a qual é representada pela equação 1 abaixo especificada. (LEBLANC; COUGHANOWR, 2009).

$$G(s) = \frac{Y(s)}{X(s)} \quad (1)$$

Basicamente, a função de transferência representa a dinâmica de um processo, ela pode ser obtida de três maneiras:

- pelas equações diferenciais ou algébricas que medem diretamente os parâmetros (massas, atritos, resistências, indutâncias, ganhos de amplificadores, etc.);
- por meio da resposta ao degrau, para os sistemas de dinâmica simples, mede-se a constante de tempo, o valor estacionário, o pico de ressonância, o tempo de subida e o instante de pico ou tempo de acomodação;
- por meio da combinação dos dois primeiros métodos. (CASTRUCCI; BITTAR; SALES, 2011).

Em um sistema de aquecimento a função de transferência pode ser escrita como uma equação de primeira ordem com atraso e ter a seguinte representação (COUGHANOWR; KOPPEL, 1978):

$$G(s) = \frac{k}{\tau s + 1} e^{-Ts} \quad (2)$$

Dessa forma, por meio dos métodos citados a equação do sistema estudado é encontrada e assume a forma: representado pelo ganho (k), pela constante de tempo do sistema τ e pelo atraso e^{-Ts} dado pelo intervalo de tempo (T) em segundos, contados do início até a mudança do valor da variável. (GUERRA, 2006).

2.2 Malhas de controle

Basicamente existem dois tipos de sistemas de controle: os de malha aberta e os de malha fechada. Existe uma diferença básica entre eles: nos de malha aberta, a saída do sistema não é medida e, portanto, não realimenta o controlador para comparação com a entrada, já os de malha fechada, a saída do sistema afeta o modo como o controlador domina a variável manipulada. (OLIVEIRA, 1999).

Para os sistemas de malha aberta a calibração do controlador é fundamental para sua precisão e sua implementação só ocorre quando a entrada e a saída são conhecidas e não ocorre distúrbios internos ou externos. (OGATA, 2003).

Na figura 2 é possível visualizar um sistema de malha aberta com todos os seus componentes.

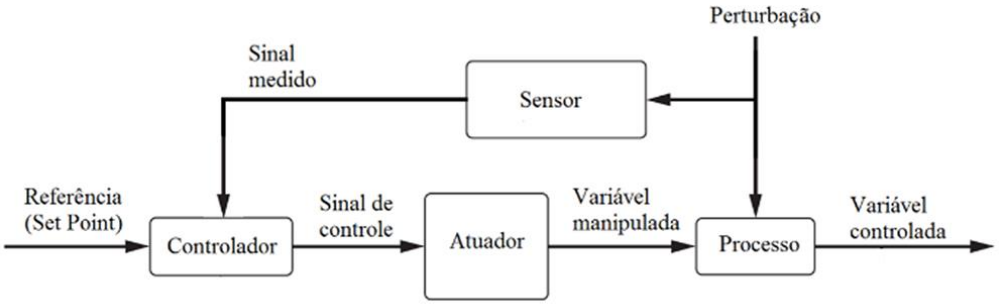

Figura 2. Diagrama de blocos de uma malha aberta com perturbação.

Uma vantagem na utilização da malha aberta é a garantia da estabilidade do sistema, permanecendo inalterada, sendo mais fácil a construção desse tipo de controle, seu uso também reflete na redução de custos, logo, sua aplicação é recomendada sempre que possível, em especial quando se busca evitar distúrbios imprevisíveis nos componentes do sistema. (OGATA, 2003).

Em sistemas nos quais não seja possível estimar ou até mesmo conhecer todos as variações que possam ocorrer dentro ou fora do processo, é necessário utilizar a malha fechada, pois o sinal de saída permite realimentar o sistema que possibilita comparar com o valor de *set point* (valor especificado pelo usuário e almejado pela variável controlada) e gerar um erro a ser usado pelo controlador para definir a resposta do sistema. (OGATA, 2003). Na figura 3 é possível visualizar um sistema de malha fechada com todos os seus componentes.

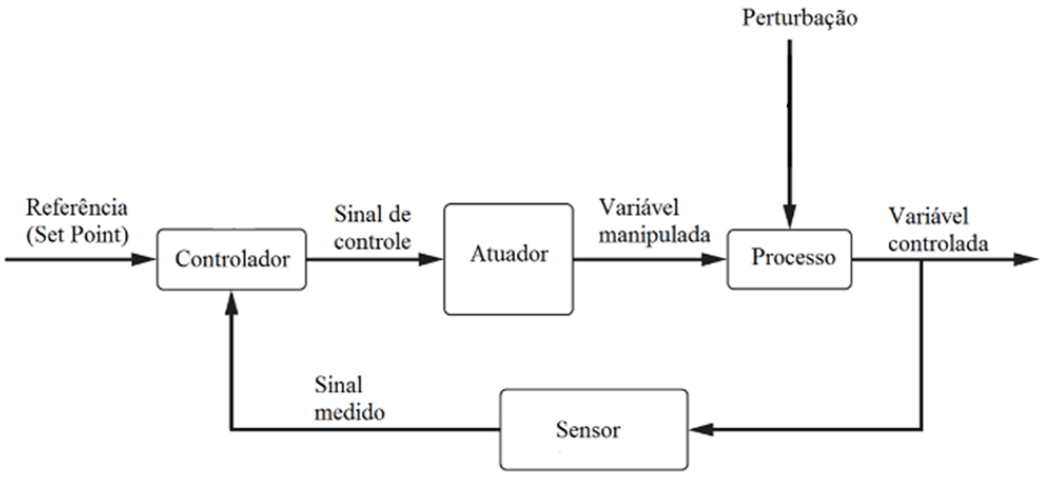

Figura 3. Diagrama de blocos de uma malha fechada com perturbação.

É possível observar que no sistema de malha fechada o sinal analisado vem da saída do processo, o qual difere do sistema de malha aberta que apenas mede a variação na entrada, contudo, ambos enviam a informação do sensor para o controlador.

Portando, um dado que se deve levar em consideração ao se determinar o tipo de malha (aberta ou fechada) que deve ser utilizada depende do conhecimento sobre como são feitas as medidas das variáveis do processo e como é a dinâmica do processo. Para sistemas em que não é possível realizar a medição na saída o uso de uma malha aberta se torna única opção.

2.3 Controle *Feedforward*

Apesar de possuir baixa importância nos textos sobre controle de processos, a alimentação do tipo *feedforward* é antiga e fundamental na engenharia de controle. Seu controle se baseia em quantificar a perturbação na entrada e anular seu efeito na variável de saída do processo. (CASTRUCCI; BITTAR; SALES, 2011).

Diferente dos métodos que medem o erro na saída do processo e retroalimentam o sistema até conseguir o equilíbrio da variável, o controle *feedforward* regula a variável, evita o distúrbio na entrada do processo e impede que o erro ocorra, permitindo a correção antes que possa afetar a saída. (NASCIMENTO; OLIVEIRA, 2006).

É válido, do ponto de vista da engenharia, o uso de controle antecipatório quando (CANIATO, 2006):

- os distúrbios e variações na entrada ou na variável manipulada do processo demoram para afetar a variável controlada. Dessa forma, um controle convencional como realimentação negativa se torna pouco eficiente;
- as variáveis analisadas na entrada do processo são possíveis de serem medidas por equipamentos disponíveis no mercado;
- as equações matemáticas que regem o processo são bem conhecidas e podem facilmente serem resolvidas;
- as equações finais de controle do processo podem ser resolvidas por equipamentos comerciais e a preços factíveis.

Entende-se que é fundamental conhecer a natureza do processo a ser controlado. Para tanto, o engenheiro deve, de acordo com Castrucci et al. (2011), examinar se é possível um controle do tipo antecipatório e caso seja, avaliar a sua viabilidade. Somente

depois, um modelo do tipo realimentação é cogitado, pois existem casos em que o uso de uma malha de realimentação sem um controle por antecipação se torna impossível de obter o desempenho desejado, como é o caso dos reguladores de tensão dos geradores de energia elétrica, no qual a perturbação medida é a corrente solicitada pela carga do gerador.

Para realizar o controle de um sistema *feedforward* é necessário obter o modelo matemático do processo, de forma a analisar como os distúrbios podem influenciar a variável controlada. Assim, ao analisar a entrada, e por meio do modelo encontrado, a variável manipulada é calculada de modo a cancelar o efeito da perturbação e impedir uma variação na saída do processo (RODRÍGUEZ, F., BERENGUEL, M., ARAHAL, 2001).

Para atuar de forma efetiva no processo e atingir o desempenho desejado, uma análise do sistema se faz necessário. Por meio dos métodos abaixo é possível encontrar as equações que regem o sistema a ser controlado, quais sejam segundo Martinelli (2018):

- caixa preta;

O modelo caixa preta garante a melhor relação entrada/saída para o sistema. Os parâmetros desse método são ajustados através de regressão linear pela teoria de identificação por subespaços. Nesse método a relação entre entrada e saída não possui significado físico e a construção desse modelo é mais rápido e direto se comparado ao modelo caixa branca. Esse modelo é ainda o mais recomendável para trabalhar com sistemas MIMO (*Multiple Inputs Multiple Outputs*).

- caixa branca;

Pouco utilizada para processo de gestão energética, nesse modelo presume-se conhecimento total das propriedades físicas atuantes no sistema, como por exemplo condutividades e capacidades térmicas dos materiais envolvidos. A modelagem através da caixa branca resulta num modelo com os parâmetros com significados físicos precisos, contudo, apresenta erros associados a variáveis não levadas em conta. Esse modelo, diferente da caixa preta, não precisa de qualquer medida do comportamento do sistema para identificar os seus parâmetros e é validado após condizer com as medidas reais constatadas.

- caixa cinza.

Os modelos caixa cinza são uma combinação entre os dois métodos anteriores, pois utilizam os dados coletados para definir um modelo com interpretações físicas. São utilizados principalmente na gestão de consumo energético ao encontrar parâmetros desconhecidos através de medições no sistema.

2.4 Tipos de sistemas de controle

Para a construção do módulo com controle *feedforward* tendo a temperatura como variável controlada, buscou-se conhecer os principais trabalhos publicados no meio acadêmico visando obter os melhores componentes utilizados e compatibilizá-lo a um sistema de baixo custo.

Para definir os parâmetros do sistema a ser desenvolvido, foram analisados inúmeros trabalhos de diversas naturezas, tanto na área que envolve aplicações para a área acadêmica, quanto em sistemas já empregados na indústria. Foram analisados desde o tipo de controle utilizado até os tipos de sensores, visto que cada forma de atuação e variável controlada determinam a escolha de cada componente. Entretanto, alguns pontos podem ser comuns independente dessas peculiaridades do sistema. Essa pesquisa pode ser visto na Tabela 1.

Diante das inúmeras opções e dos objetivos iniciais do trabalho, foram selecionados alguns parâmetros que visam atender o objetivo geral do trabalho, um sistema de aquecimento baseado em softwares livres.

Para selecionar o sensor de temperatura foi levado em consideração para a escolha: a precisão, o preço e o tempo para medição. Com os tipos encontrados na tabela 1 e mediante análise feita por Martinazzo e Orlando (2016) o sensor que mais se adequou ao projeto foi o DS18B20.

Dentre as opções de componentes para converter os sinais analógicos e digitais, a plataforma Arduino Uno se mostrou a melhor escolha devido à sua fácil programação e por já ser utilizada em outras disciplinas na Universidade.

O aquecimento dos sistemas analisados basicamente se dividem entre aquecedores elétricos e queimadores, optou-se pela resistência elétrica devido a facilidade de regulagem, menor custo e fácil instalação.

Tabela 1. Trabalhos analisados.

Controle	V.Contr.	Atuador	Software	Aquisição dados	Sensor	Autores
PID	Temperatura	Resistencia elétrica	Simulink	PCI 1711	Variac	(GUERRA, 2006)
PID	Temperatura	Lâmpada Cooler	Amarino Xcos Scilab	Arduino Nano	LM35/DHT11	(BOTELHO et al., 2016)
PID	Temperatura	Ebulidor elétrico	TwidoSuite	CLP	LM35	(HOTZ, 2014)
Feedforward PID	Temperatura	Resistor espiral	CD600	CD600	PT-100/	(SILVA; LOPES; AMARAL, 2012)
PI	Temperatura	Resistencia	Matlab simulink	NI - USB 6009	Pt100	(BAMPI, 2016)
PI	Temperatura	Resistencia elétrica	Emerson DeltaV Distributed Control System	Foundation Fieldbus modules Watlow DIN-A-MITE	EchoPod DL14 Termopar	(TZOUANAS; STEVENSON, 2013)
PI PID Feedforward	Temperatura	Queimadores	FCC sistema de otimização dos fornos	TIC	Termopar	(TEIXEIRA; JOTA; TEIXEIRA, 2007)
Redes neurais	Nível Temperatura	Resistência	OPCtool Matlab Profibus	SMAR PD-3	-	(BERTACHI et al., 2013)
------	Nível	Válvula Servomotor	-	CLP Arduino	Tipo turbina	(GNOATTO et al., 2016)
PID/espaços de estados	Nível	Moto bomba	Programação Linguagem C	Placa aquisição 12 bits	Boia	(GOSMANN, 2002)
PID	Nível Temperatura	Resistência Bomba para-brisa	NetBeans IDE/JAVA/Blender	Arduino Severino	Infravermelho GP2Y0A21YK LM35	(GOMES et al., 2011)
PID Escalonamento	Velocidade hélice	Motor elétrico	Sisotool Matlab	Placa de controle com AmpOp e Transistores	Ângulo tensão	(CAVAZZANA ; FILHO; SOUZA, 2011)
PID/PI	Velocidade hélice	ServoMotor DC	Linguagem de programação	PIC 16F628	Foto receptores	(ISHIGAKI et al., 2004)
Feedforward MPC	Temperatura Pressão	Válvulas gás/retorno/chaminé	SDCD	SDCD	-	(ROCHA et al., 2010)
Feedforward Adaptativo	Compressão	Motor elétrico CC	LabView	CompactRIO		(LIMA et al., 2015)

Uma das peculiaridades desse trabalho que se diferencia dos demais refere-se ao software utilizado para desenvolver o controlador, pois todos os trabalhos analisados utilizaram softwares pagos e ou fechados, sem possibilidade de alteração em seu código.

3. METODOLOGIA

Neste capítulo são descritos a comunicação entre os componentes do sistema de controle e o controlador, pois foi necessário determinar o tipo de sinal a ser utilizado, bem como, os componentes compatíveis. Na sequência, apresentam-se os materiais e o método escolhido para a comunicação entre os componentes presentes no sistema. Logo em seguida, detalham-se a montagem dos módulos, do desenvolvimento do controlador e da interface amigável.

Para a criação do módulo primeiro foi feito o estudo e planejamentos dos componentes e posteriormente a construção, seguindo as seguintes etapas:

- estudo da comunicação do software Scilab com a plataforma Arduino;
- planejamento e dimensionamento do tanque;
- estudo dos tipos de sensores de temperatura;
- planejamento e dimensionamento do tipo de aquecimento;
- estudo do tipo de controle da vazão;
- estudo do local para instalação do módulo;
- construção do tanque em acrílico;
- construção da placa dos sensores de temperatura;
- construção da placa responsável por controlar a potência da resistência;
- construção da placa responsável por controlar a vazão da bomba d'água;
- instalação dos componentes desenvolvido no local escolhido;
- criação da interface de controle no software Scilab;
- testes de funcionamento;
- início do experimento.

Para determinar a escolha dos componentes do módulo, além de partir da escolha do uso do Scilab e da plataforma Arduino foram analisados outras módulos responsáveis por controlar processos, essa pesquisa pode ser vista na tabela 1. Através desse estudo foi montado o diagrama da figura 4 com todos os componentes selecionados e utilizados para a construção do módulo e como se conectam ao controlador desenvolvido no Scilab.

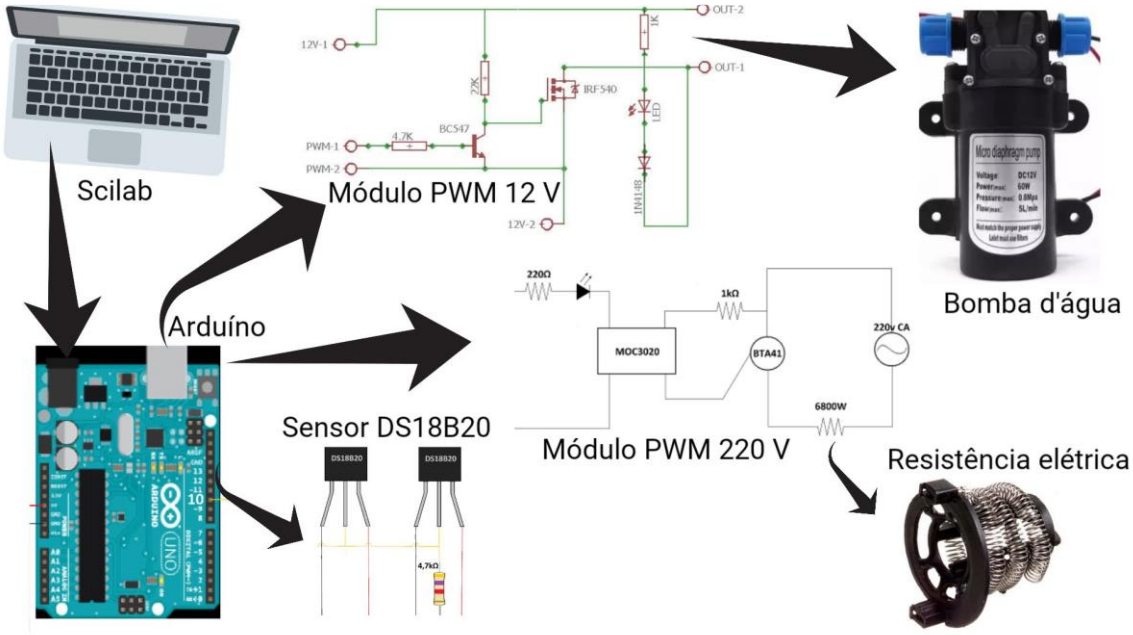

Figura 4. Diagrama da montagem dos equipamentos.

Nesse diagrama é possível identificar as placas desenvolvidas para compatibilizar o sinal enviado do Arduino, pelo controlador, para a resistência elétrica e bomba d'água, assim, como, a placa responsável por conectar os sensores de temperatura. A partir dessas informações a lista de materiais e equipamentos necessários para o projeto foi elaborada.

3.1 Materiais e equipamentos

Tendo como base o módulo sendo controlado por meio do Software Scilab com interação com a plataforma Arduino UNO os equipamentos foram selecionados de forma a compatibilizar esses dois elementos. Partindo da necessidade de um tanque transparente o material que se mostrou mais adequado para utilização foi o acrílico. Os sensores de temperatura foram escolhidos buscando o melhor custo benefício, que a partir de testes se obteve melhores resultado com o DS18B20. Para o aquecimento do tanque foi decidido pela resistência elétrica devido ao seu fácil controle de potência e para manter o nível constante se optou por uma bomba d'água. Logo se obteve a seguinte lista de materiais:

- tanque de acrílico transparente de 80 L;
    - chapa Acrílico Cast 1 m^2;
    - cola acrílico sinteglas S-330;
    - pistola Ar quente;
- Arduino UNO;
- sensor de temperatura DS18B20;

- resistor;
- bomba d’água 60 W;
- modulo PWM 5V CC/12V CC;
    - IRF540;
    - resistores;
    - *born*;
    - placa PCI;
    - led;
    - BC547;
    - diodo;
    - dissipador;
- módulo PWM 5V CC/220V AC;
    - resistores;
    - placa PCI;
    - led;
    - *born*;
    - diodo;
    - MOC3020;
    - BTA41;
    - dissipador;
    - cooler 12V;
- resistência elétrica 6800 W;
- fonte 12V;
- disjuntor bifásico 32ª tipo B;
- placa de madeira;
- caixa de madeira;
- chuveiro elétrico ;
- cabo 6 mm^2;
- computador com software Scilab;
- cabo impressora 2 m;
- tubulação PVC ¾;
- cola quente.

Por meio da lista foi feito o orçamento (tabela 2) e compra dos materiais para iniciar a construção do módulo de controle de temperatura.

Tabela 2. Gastos com materiais.

Material	Valor
Chapa acrílico	50,00
Cola S-330	65,80
Arduino UNO	45,00
2 x DS18B20	23,00
Bomba d'água 60 W	120,00
Modulo PWM 5V CC/12V CC	35,00
Módulo PWM 5V CC/220V AC	29,50
Resistência elétrica 6800 W	16,53
Fonte 12V	40,00
2 x Disjuntor bifásico 32 A tipo B	57.35
Caixa de madeira	15,00
Chuveiro elétrico	50,00
Cabos e canos	45,00
Total	**592,18**

Com o módulo pronto foi possível alimentar o controlador presente no software Scilab utilizando as equações matemáticas e testes empíricos que permitem analisar o desempenho do método *feedforward*.

3.2 Sinal de controle

Para controlar o acionamento dos componentes o sinal do tipo PWM foi definido como ideal, sendo muito utilizado para o controle de cargas de potência como motores e elementos de aquecimento, os controles do tipo pulsantes ou PWM (Modulação de Largura de Pulso) permitem inúmeras aplicações industriais ao garantir partidas suaves para os motores e menor consumo de potência, se comparado aos controles do tipo linear. (BRAGA, [s.d.]).

Para o controle de cargas muito grandes a utilização do método linear se torna inviável, devido à dificuldade de encontrar componentes capazes de controlar a corrente

dessas cargas, dessa forma o sinal PWM se mostrou a melhor opção para o controle da potência enviada a carga (BRAGA, [s.d.]).

A modulação por largura de pulso senoidal é uma técnica simples e muito utilizada. A ideia desta modulação é controlar a razão de tempo de fechamento das chaves semicondutoras de forma a obter na saída um valor médio igual ao valor desejado. A potência média nesse tipo de controle se dá no decurso entre o tempo em que o circuito permanece ativo e o que permanece desligado. Essa razão é conhecida como o ciclo de operação ou *duty cycle*.

Um sinal PWM do tipo retangular, visualizado na figura 5, exemplifica a tensão média (V_m) enviada, sendo que a carga será dada pela proporção entre o tempo ativo (t_1) e o tempo total do ciclo, soma do tempo ligado (t_1) e desligado (t_2).

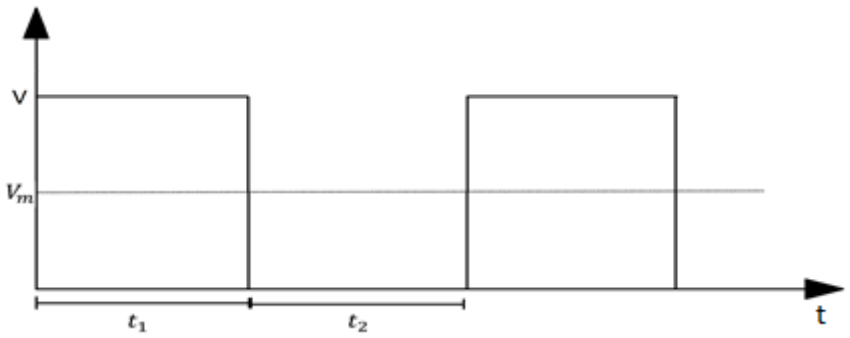

Figura 5. Forma de um sinal PWM do tipo retangular.

Na prática os dispositivos de estado sólido são os componentes responsáveis por controlar as mudanças de estado de modo rápido e eficiente. Eles, por intermediação de um sinal PWM, controlam a passagem da corrente pelo circuito. (GHIRARDELLO, [s.d.])

Diferentemente das cargas CC, o controle por meio do sinal PWM em cargas CA demandam alguns passos a mais, como a escolha do método de controle, uma vez que é necessário além do dispositivo de estado sólido a utilização de chaves eletrônicas como tiristores em ligação antiparalelo ou TRIACs.

O uso dessas chaves permite controlar a potência média enviada à carga, usualmente esse controle é feito por meio de dois métodos: controle por ângulo de fase

ou por intermediação do controle por ciclos inteiros, sendo esse último o escolhido para ser utilizado pelo sistema desenvolvido (KOMATSU; JUNIOR; KAISER, 2017).

No método escolhido é controlado a quantidade de ciclos inteiros enviados à carga (K) a cada determinado número de ciclos inteiros da rede (N). Através dessa relação se obtém a tensão enviada à carga por meio da equação 3 (KOMATSU; JUNIOR; KAISER, 2017).

$$V_{carga} = V_{rede-eficaz}\sqrt{\frac{K}{N}} \tag{3}$$

3.3 Arduino

O Arduino é uma plataforma de fácil utilização, permite que usuários menos experientes consigam utilizar suas funções de modo simples sendo amplamente utilizado para fins educacionais.

O Arduino possui uma grande quantidade de *softwares*, *hardwares* e arquivos bases, fato esse que permite uma vasta aplicação. (ARDUINO CORPORATION, 2018). Além disso, é uma plataforma *open source* e utiliza um microcontrolador ATmega328P com 14 pinos digitais de entrada e ou saída, desses pinos, seis podem emitir sinais PWM e outros seis são exclusivos para entradas analógicas. (ARDUINO CORPORATION, 2018)

A comunicação na referida plataforma permite que os sinais recebidos possam ser repassados via comunicação serial, assim, como, propicia receber por ela os dados e transmiti-los através de suas saídas. Dessa forma, ela pode funcionar como uma ponte entre os sensores, o controlador e o atuador do sistema.

A comunicação serial entre a placa Arduino e um computador ou dispositivo pode ser feita através da sua porta serial por meio dos pinos digitais 0 (RX) e 1 (TX) ou via USB quando comunicado com um computador. Para que a plataforma receba os dados basta programar algumas linhas de comando (ARDUINO CORPORATION, 2018). O código desenvolvido e utilizado para receber e enviar os dados se encontra no apêndice A.

3.4 Scilab

O Scilab é um software *open source* para computação numérica, constitui uma poderosa ferramenta para usos na área de engenharia e em inúmeras aplicações científicas (ESI GROUP, 2017)

Este software está disponível para diversas plataformas, tais como: Linux, Mac OS e Windows. Possui centenas de funções matemáticas além de ser avançado, a ponto de permitir visualização de gráficos em 2D e 3D. Suas principais aplicações são:

- simulações matemáticas;
- visualização gráfica em 2D e 3D;
- otimização de problemas;
- estatística;
- análise e controle;
- processamento de sinais.

Para aplicações mais específicas, tal como estabelecer comunicação com outros dispositivos via comunicação serial, é possível por intermédio de uma plataforma chamada ATOMS (gerenciador automático de módulos para o Scilab) fazer o download de módulos que liberam ao usuário novas funções.

Com o módulo responsável pela comunicação serial instalado o software passa a reconhecer novas funções vindas desse pacote (*openserial, closeserial, serialwrite e serialread*), por meio dessas funções é possível controlar o envio e recebimento de sinais quando um dispositivo apto, como o Arduino, estiver conectado à porta USB do computador. O código criado para essa comunicação se encontra no apêndice B.

4. MONTAGEM E IMPLEMENTAÇÃO

4.1 Sistema de medição

4.1.1 DS18B20

Como o tipo de controle desenvolvido permite controlar a temperatura de saída de um tanque, foi necessário escolher um sensor de temperatura que mais se adequasse ao projeto (baixo custo, preciso, fácil integração com a plataforma de aquisição de dados e resistente a água) dessa forma, o escolhido foi o sensor DS18B20, devido as suas características.

Esse sensor pode ser facilmente encontrado no mercado nacional e possui uma excelente precisão, podendo ser utilizado com resolução que vai de 9 a 12 bits, como mostrado na tabela 3, e possui uma acurácia de 0,5°C operando na faixa que vai de -10 a 85°C. (MAXIM INTEGRATED, 2015)

Tabela 3. Resolução do sensor DS18B20

Resolução	Tempo de conversão (ms)	Precisão (°C)
9 bits	93,75	0,5
10 bits	187,5	0,25
11 bits	375	0,125
12 bits	750	0,0625

Fonte: MAXIM INTEGRATED, 2015.

Na tabela 3 é possível identificar o tempo e a precisão para cada resolução definida para o sensor, para o trabalho a resolução escolhida foi a de 11 bits, pois apresenta uma precisão satisfatória com um tempo de conversão que não interfere no recebimento dos dados do sistema, uma vez que ao utilizar os 12 bits o tempo de conversão acabava criando intervalos muito grandes entre as medições atrapalhando a atuação do controlador.

O sensor utilizado possui encapsulamento TO-92 com três terminais identificados como GND (terra), DQ (entrada/saída de dados) e V_{DD}. O terminal de dados se comunica através do protocolo *one-wire* e permite que mais de um sensor possa ser conectado na mesma entrada, pois cada sensor possui um número de identificação que permite sua localização entre vários sensores (Figura 6).

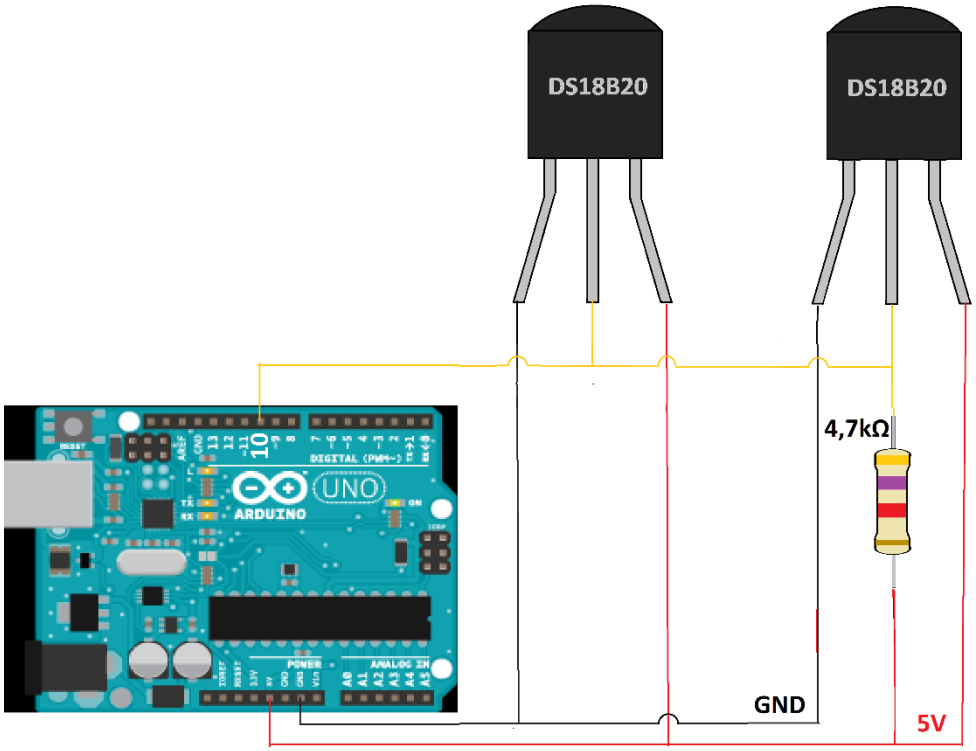

Figura 6. Circuito dos sensores de temperatura.

A comunicação com os sensores para recebimento do sinal é feita através da plataforma Arduino, no qual o mesmo fica responsável por alimentar o terminal V_{DD} do DS18B20 com tensão de 5V e através da protocolo *one-wire* e da biblioteca *DallasTemperature* é feita a comunicação com os sensores.

Segundo Maxim Integrated (2015) os sensores já são distribuídos calibrados e não necessitam de qualquer ajuste por parte do usuário, permitindo sua imediata aplicação como termômetro.

4.2 Construção do sistema

4.2.1 Tanque

O reservatório utilizado foi um tanque produzido em acrílico *Cast* (confeccionado entre lâminas de vidro temperado). Tal escolha se deu devido as suas características de grande durabilidade e transparência. Suas propriedades térmicas também atendem as necessidades do projeto e podem ser vistas na tabela 4.

Tabela 4. Propriedades térmicas e moldagem para chapas acrílicas

Propriedades térmicas e moldagem	**Valores**
Temperaura de moldagem de chapas “cast”	165 a 190 °C
Tempo de aquecimento em estufas de chapas “cast”	3 a 4 min/mm
Temperaura de uso contínuo das chapas	-40 a 80 °C
Condutividade térmica	0,18 W/m°C
Auto ignição	Acima de 490°C
Flamabilidade	25

Fonte: (INSTITUTO NACONAL PARA DESENVOLVIMENTO DE ACRÍLICO, [s.d.])

A fabricação do acrílico tipo Cast ocorre em autoclaves e tem como matéria prima o Metil Metacrilato, ele em estado líquido ao ser combinado com aditivos, pigmentos e catalisadores inicia sua polimerização, passando de liquido para sólido. (EMPÓRIO DO ACRÍLICO, [s.d.]). Para determinar a qualidade das placas elas passam por um controle de qualidade para que atendam à norma ISO 7823-1.

A partir desse pressuposto, para o tanque o formato escolhido foi o cilíndrico, pois têm menos pontos de cola se comparado a tanques de outros formatos. A colagem das bordas e da base foi feita utilizando uma cola específica que garante até 80% da resistência do acrílico.(SINTEGLAS, [s.d.]).

A colagem seguiu as recomendações do fabricante, que indica preencher um vão de no mínimo 0,8mm entre as placas com cola, para o processo uma fita é posicionada na parte inferior do espaço formado entre as placas até que a cola seque por completo e atinja sua cura total. A figura 7 ilustra o processo de construção dos tubos de acrílico durante o processo de moldagem (SINTEGLAS, 2012).

Figura 7. Tubos de acrílico durante o processo de moldagem.

Após o tempo de secagem completa dos pontos de emenda foi possível realizar os testes para verificar se havia vazamento e avaliar a sua resistência a pressão, sendo corrigidos pontos com vazamentos.

Para vedar a base do tanque (figura 8) foi realizado um teste com silicone, por ser mais fácil de aplicar e possuir excelente resistência e não interagir com os materiais. Entretanto, após alguns testes ele se mostrou ineficiente à alta pressão, apresentando vazamentos.

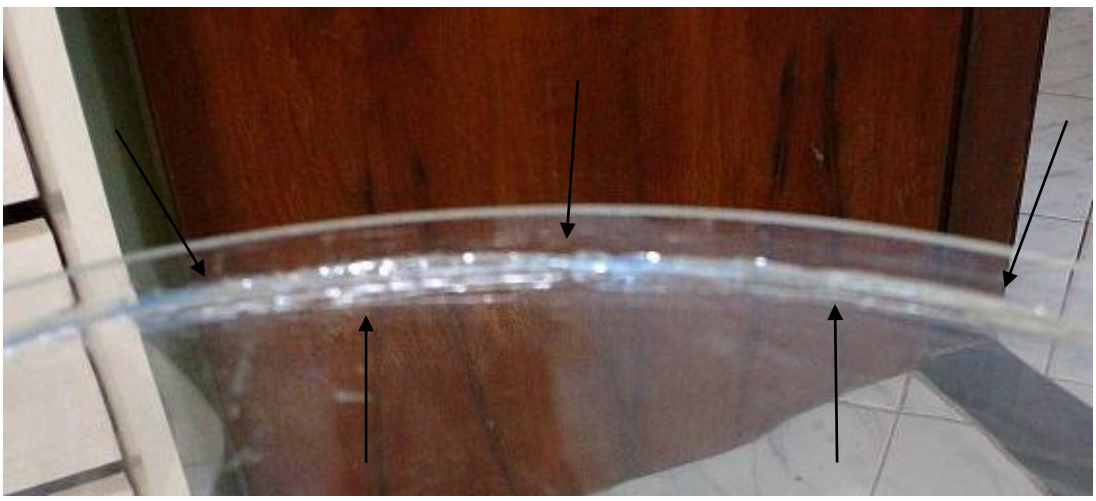

Figura 8. Emenda da base do tanque de acrílico.

Para solucionar o vazamento foi necessário utilizar mais uma camada da cola específica para acrílico para preencher completamente todas as emendas e dispensar o uso do silicone para suprimir as lacunas deixadas pela emenda da base com a coluna.

4.2.2 Atuador

Para que o sinal do controlador, presente no Scilab, possa representar efetivamente uma atuação no sistema é necessário um circuito atuador que receba o sinal do controlador, converta para uma saída PWM de 5V presente na plataforma Arduino e ative um TRIAC BTA41 conectado à resistência elétrica (figura 9).

O sinal de ativação enviado ao TRIAC pelo Arduino é limitado por sua porta de saída analógica, o sinal PWM gerado por ela é fixada numa escala de 0 a 255, esse intervalo representa o *duty cycle* do sinal emitido pela saída analógica, sendo 255 equivalente ao *duty cycle* de 100%. Dessa forma, um sinal de 255 enviado por meio do comando *analogWrite* representa uma potência enviada a carga de 100%.

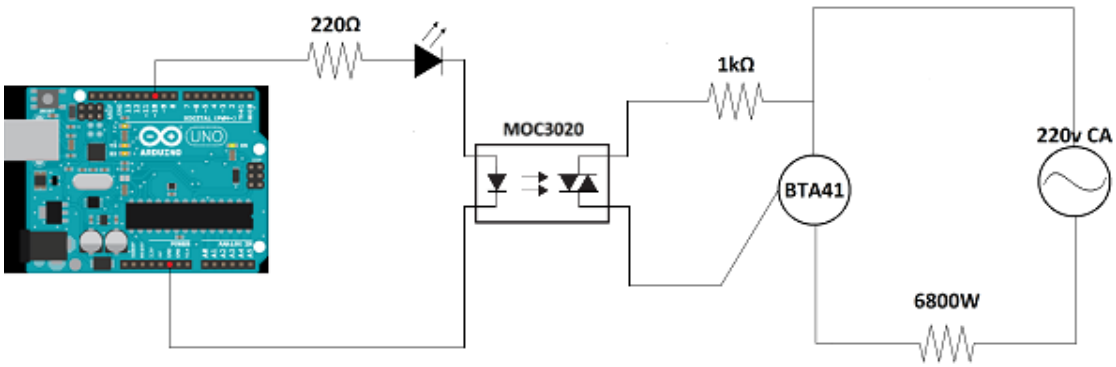

Figura 9. Circuito elétrico do atuador.

O circuito elétrico do atuador além de regular a carga enviada à resistência tem a função de isolar o Arduino através do componente MOC3020, que faz parte da família dos isoladores-disparadores bloqueando o circuito contra tensão de pico de 400 V, esse componente também tem a função de disparar o TRIAC,

Tendo como isolamento óptico a ativação do MOC3020 é feita através do acionamento do LED em seu interior, sendo necessário uma corrente de disparo de apenas 30 mA (figura 10).

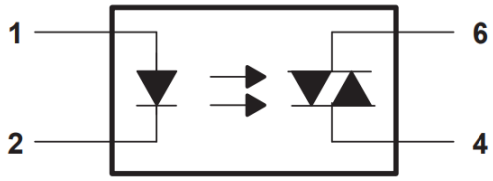

Figura 10. MOC3020

Fonte: TEXAS, 1998

4.2.3 Controle de fluxo

O uso de uma Bomba d'água se deu a partir da necessidade de variar e manter um fluxo de água na saída do tanque, uma vez que a vazão deverá permanecer constante para que a única variável controlada pelo sistema seja a temperatura. Em relação à escolha da bomba, a decisão se baseou em suas características: potência, tensão e tamanho.

O objetivo é fazer com que o fluxo de água transportado seja próximo ao de uma torneira comum (6L/min), pois será a vazão utilizada na entrada do tanque, sendo então necessário uma bomba com valores próximos para retirar a água e manter o mesmo nível do tanque, por isso, uma bomba com menos potência foi necessária. Considera-se relevante a definição de tamanho e de resistência em elevadas temperaturas, conforme características:

- entrada/saída: 1/2 " macho;
- potência: 60 W;
- tensão: 12 Volts corrente contínua;
- corrente nominal máxima: 5A;
- fluxo máximo: 5 L/min;
- pressão de trabalho: 0,8 Mpa;
- temperatura máxima do fluido: 100 ºC.

Visando permitir maior variação da vazão e se adequar à escala do sistema foi criado um circuito, representado pela figura 11, que define a vazão através de um sinal do tipo PWM. Esse sinal é controlado através da interface desenvolvida e inserida pelo usuário.

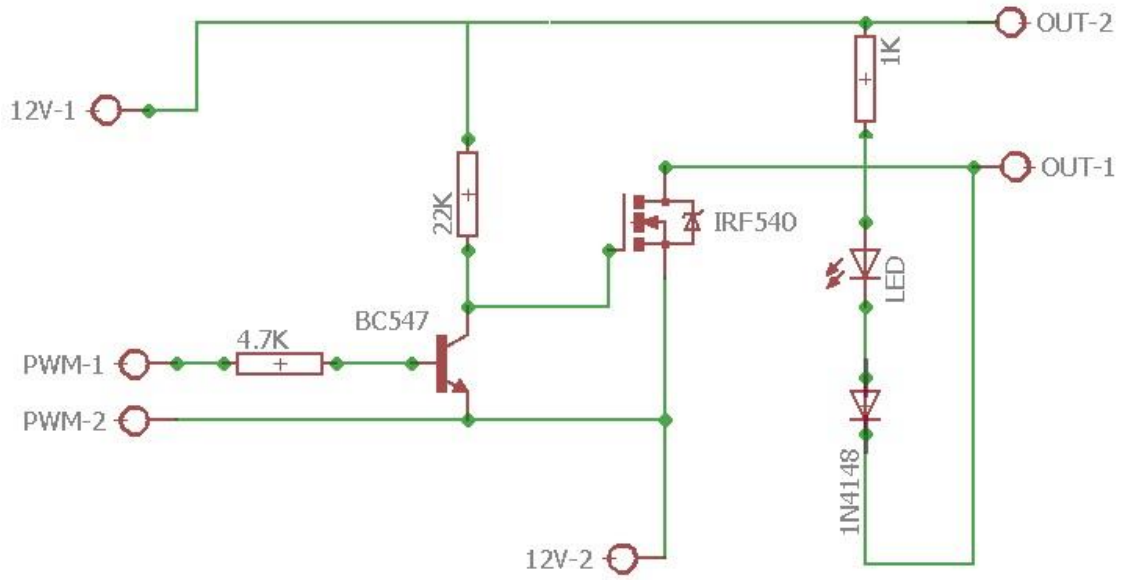

Figura 11. Circuito elétrico para controle PWM da bomba d'agua.

4.3 Montagem dos módulos

Para a montagem dos módulos buscou-se interligar todos os componentes desenvolvidos e testados separadamente de modo a funcionarem em conjunto. A montagem seguiu o planejado e teve apenas pequenas adaptações quanto ao posicionamento de alguns componentes. O seu diagrama, assim como o resultado final da montagem podem ser considerados nas figuras 12 e 13 respectivamente.

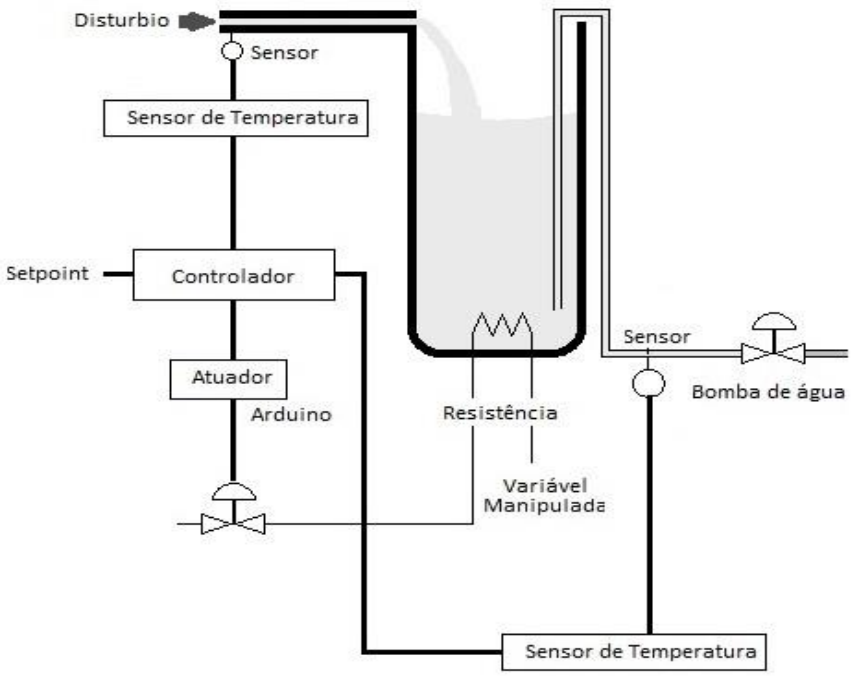

Figura 12. Diagrama do sistema a ser montado.

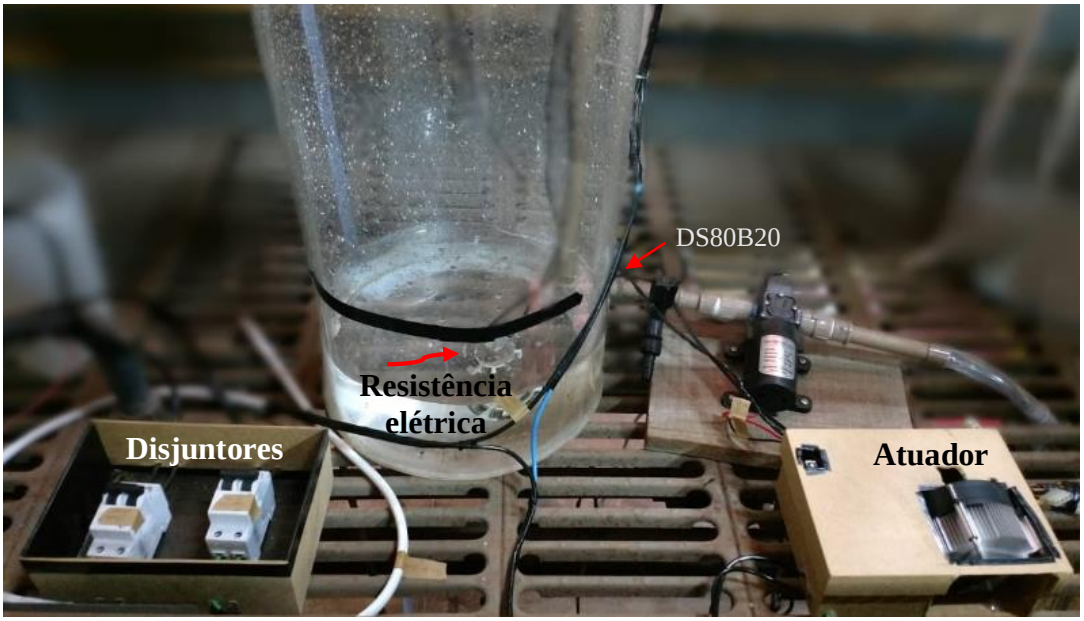

Figura 13. Sistema completo montado.

Para simular distúrbios durante o processo, outra resistência foi instalada antes da entrada do sistema (figura 14), ela assim como a resistência principal foram protegidas por disjuntores bipolar de 32A tipo B por conta da alta potência das resistências (6800 watts). No caso da resistência utilizada como distúrbio, o disjuntor funciona como interruptor para controlar manualmente de forma aleatória a perturbação durante as simulações.

Figura 14. Resistência elétrica responsável por simular distúrbio na entrada.

Uma das adaptações feitas foi na resistência do interior do tanque que a princípio seria instalada por baixo do mesmo, mediante pequenos furos em sua base, entretanto,

durante a confecção dos tanques a pressão e o formato irregular da superfície se mostrou um problema, assim, devido a dificuldade de encontrar uma vedação eficiente optou-se por passar os fios pela parte superior.

Assim como para a entrada dos fios da resistência, havia a necessidade de uma nova abertura na base do tanque para a passagem da saída do fluido. O qual também se mostrou um problema, a opção, portanto, foi passar uma tubulação por cima do tanque e conectá-la à bomba de água. Os demais equipamentos foram instalados sem mais percalços e todos conectados ao computador possibilitando analisar a resposta e a perda de energia térmica do sistema, assim como a equação de envio de potência e a sensibilidade dos sensores.

4.4 Controlador e interface

O controlador foi desenvolvido através do software Scilab e em ligação serial com o Arduino, sendo responsável por controlar todo o sistema. Nele foi desenvolvido uma interface amigável para o operador definir os parâmetros que deveriam ser seguidos pelo sistema, como pode ser observado o seu princípio de funcionamento na figura 15.

Para efetuar a ação de controle propriamente dita, foi programado um cenário em que o usuário pode definir qual o tipo de atuação de controle, acionando a linha de comando correspondente. Para o código do controle do tipo *feedforward* é necessário ter equacionado todas as perdas do sistema, assim como uma equação que relaciona o sinal de saída com a resposta sentida pelo sistema, essas equações são apresentadas no próximo capítulo.

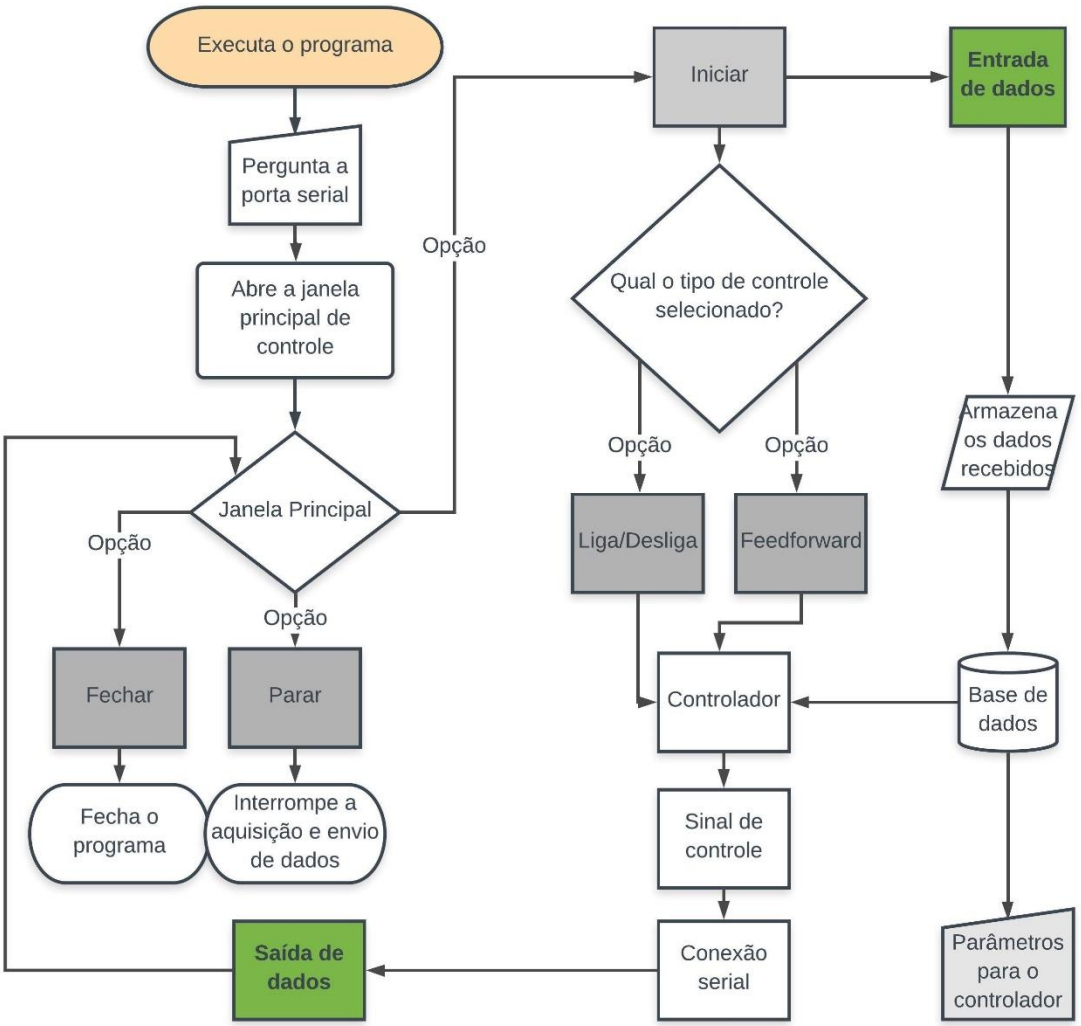

Figura 15. Fluxograma do programa desenvolvido no Scilab.

O código do programa através do balanço de energia equacionado, identifica o valor dentro da faixa de saída do Arduino que corrige a temperatura para corresponder ao *set point*. O código desenvolvido no Scilab se encontra no apêndice B.

5. APLICABILIDADE

Neste capítulo são desenvolvidas as etapas a serem seguidas pelo usuário para modelar, controlar, simular e obter um sistema eficiente para ser utilizado como estudo em práticas laboratoriais.

5.1 Modelagem

Para que o controlador possa atuar de forma eficiente em um sistema do tipo *feedforward*, em malha aberta. É preciso conhecer de maneira precisa as informações do sistema, de forma a saber como um sinal irá repercutir no sistema, assim como avaliar a perda de energia térmica, tornando possível conhecer a influência de uma perturbação.

Devido a natureza do sistema é necessário combinar conceitos teóricos com os dados encontrados na prática, uma vez que a constante referente à perda de energia térmica para o ambiente depende de dados encontrados experimentalmente. As equações e constantes que devem ser inseridas no controlador foram encontrados a partir dos seguintes passos.

Encontra a quantidade de calor recebido pelo sistema ao enviar utilizar 100% da potência da resistência, através da equação do calor (4).

$$Q = m \cdot c \cdot \Delta T \tag{4}$$

Sendo ΔT a diferença entre a temperatura alcançada e a temperatura inicial e m a massa de fluido por um determinado intervalo.

Para encontrar o valor perdido para o ambiente basta tirar a potência recebida, ao enviar 100% do *duty cycle* a carga, da potência enviada potência máxima da resistência dada pelo fabricante. A potência enviada é dada por:

$$Q_{perdido} = Q_{enviado} - Q_{recebido} \tag{5}$$

Sabendo a quantidade de energia térmica perdida é possível encontrar a constante $h \cdot A$ de perda de calor por convecção do sistema, por meio da lei do resfriamento de Newton (equação 6), sendo h o coeficiente de transferência térmica e A a área.

$$Q_{perdido} = h \cdot A \cdot (T_{externa} - T_{fluido}) \tag{6}$$

Com isso encontra a energia térmica perdida para cada diferença de temperatura, de forma a obter uma equação que relaciona o valor do sinal PWM com quantidade de energia térmica transferida ao fluido.

Com essa equação é possível deixar a saída do controlador em função de todas as informações obtidas pelo sistema como: temperatura ambiente, temperatura da entrada e a equação de envio do sistema. Assim para cada variação nesses parâmetros um novo valor na saída é gerado. A equação encontrada pode então ser inserida no código fonte do controlador.

5.2 Práticas em laboratório

Visando a utilização em práticas de laboratório o uso do módulo permite vivenciar na prática a aplicação dos conhecimentos teóricos de transferência de calor. Para facilitar e permitir uma aproximação maior do usuário com a prática, é possível acompanhar todo o processo via uma interface amigável que permite mudar os parâmetros de controle (tipo de controle, *set point*, temperatura ambiente e vazão) de modo fácil e rápido. Essa interface pode ser observada na figura 16.

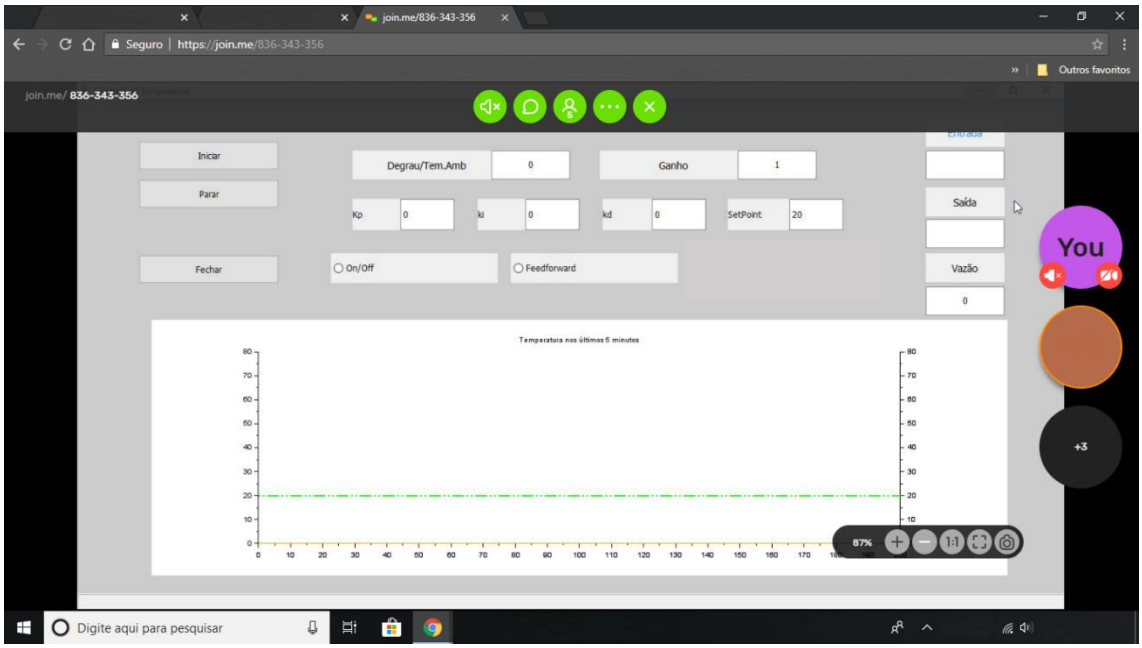

Figura 16. Captura de tela através da perspectiva de um usuário.

A interface disponibilizada pode ser compartilhada via rede através de um link gerado pelo programa e que pode ser acessado de qualquer navegador para acompanhar os dados da interface em cada prática, através dele é possível que até 5 usuários acompanhem sem programas adicionais ao processo.

Vale observar que apenas um usuário será o responsável por realizar a definição dos parâmetros de controle, já os demais irão acompanhar o processo, isso se torna lógico

uma vez que um sistema de controle não depende da intervenção do usuário, funcionando de modo independente e autônomo.

Para ser utilizado em futuras práticas de laboratório é possível seguir alguns passos. Sabendo que o modulo será do tipo *feedforward* é necessário desvendar todas os fenômenos que regem esse sistema. Na prática é necessário encontrar:

- a energia térmica fornecida pelo sistema;
- a energia térmica perdida;
- a fórmula do calor necessário para cada variação do sistema.

Depois de encontrar as equações é possível simular diferentes perturbações tanto na entrada quanto na vazão do sistema, inclusive simular alterações no nível do tanque, uma vez que o sistema também identifica e analisa a mudança na vazão.

Para o início do experimento deve se verificar se a vazão de saída da bomba está de acordo com o valor lido pelo LDR, sendo necessário calibrar o sistema. Para isso basta um cronômetro e uma balança. Com o resultado atualizar a proporção correta no campo destinado ao ganho.

Por intermédio do software que disponibiliza a interface de controle online é enviado os dados armazenados pelo controlador desenvolvido, esses dados são fundamentais e armazenam todas as informações do sistema durante o decorrer do processo.

5.3 Resposta ao degrau

O sistema funciona totalmente através da interface apresentada, nela é possível realizar degraus com diferentes valores e analisar a resposta do sistema, podendo obter a função de transferência do modelo testado. Como pode ser observado na figura 17.

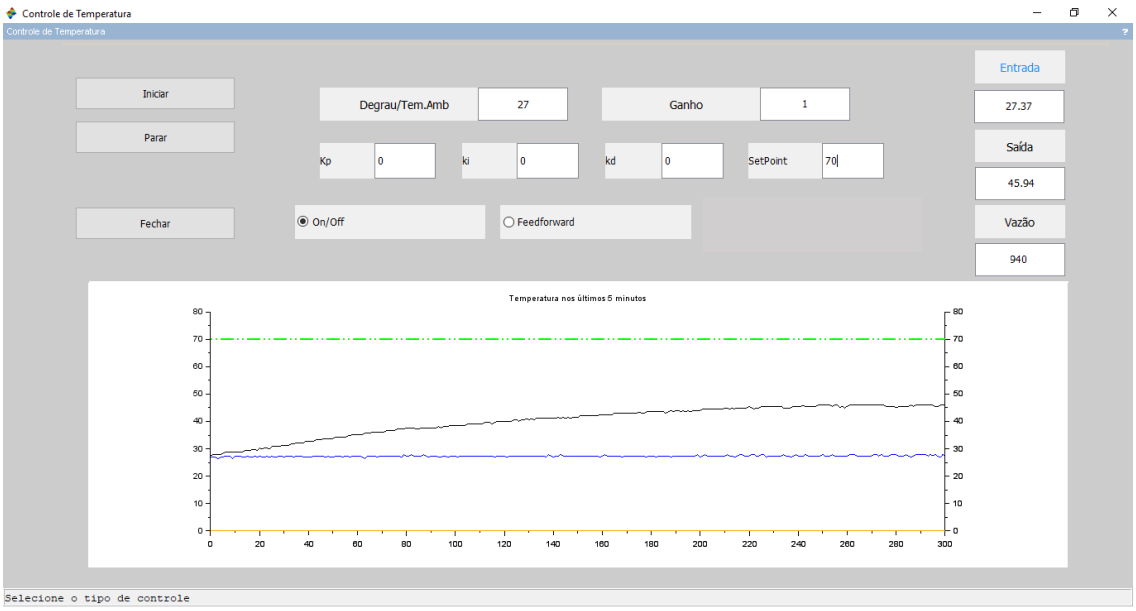

Figura 17. Interface apresentado uma resposta ao degrau.

Através da interface é possível analisar a forma geral do ganho assim como acompanhar a variação dada pelos sensores. Como pode ser vista na interface, o degrau é gerado ao se colocar o sistema em modo *on/off*, estabelecer um valor para o degrau e extrapolar o *set point* para não limitar o ganho.

Com a resposta gerada é possível encontrar a constante de tempo, o ganho e o atraso do sistema, esses dados são fundamentais para se encontrar a função de transferência do sistema. Para gerar o gráfico com os pontos e seus tempos de medida é possível exportar os dados gerados pelo programa, armazenado em arquivo txt, para qualquer outro software, o qual pode visualizar o arquivo de forma mais detalhada com as informações medidas.

5.4 Sistema com controle *feedforward*

Com a interface desenvolvida, se obtém todos as variáveis do sistema na mesma tela em que os parâmetros são alterados. Ao selecionar o modo *feedforward* e aplicar uma variação no *set point* se analisa o tempo que o sistema demora até atingir seu estado de equilíbrio, assim como, o atraso do sistema e o ganho de temperatura após o envio do sinal.

No modo de controle *feedforward*, ao realizar uma mudança no *set point*, o controlador muda a intensidade de atuação, essa alteração do sistema é baseada na equação encontrada por meio da modelagem utilizando o método de caixa cinza.

O resultado da equação encontrada e inserida no controlador muda caso os parâmetros utilizados se alterem, portanto, é possível encontrar diferentes respostas para cada modelagem feita. Para analisar a eficiência do controlador se observa se o *set point* e a entrada permanecem constantes para que seja possível determinar se a equação de transferência de calor e a equação de perda do sistema encontrada pelo usuário representam de forma satisfatória o sistema.

Um erro na equação de transferência de calor do controlador, assim como um coeficiente de perda de calor equivocado resulta em uma temperatura abaixo ou a cima do desejado, uma vez que o sistema analisa apenas a temperatura na entrada. Dessa forma é recomendado realizar mais de uma medida para obter a média desses parâmetros.

6. RESULTADOS E DISCUSSÃO

A seguir apresenta-se e discute a interface desenvolvida via Scilab, bem como, as respostas do sistema frente as variações diretas em sua variável manipulada, assim, como, a modelagem do sistema e sua resposta às perturbações na entrada.

6.1 Interface de controle

O primeiro resultado obtido foi a interface do sistema supervisório, responsável por definir os parâmetros a serem seguidos e o tipo de controle. Ao executar o código, uma janela é apresentada ao usuário requisitando a porta referente a comunicação serial entre o Scilab e o Arduino. (Figura 18).

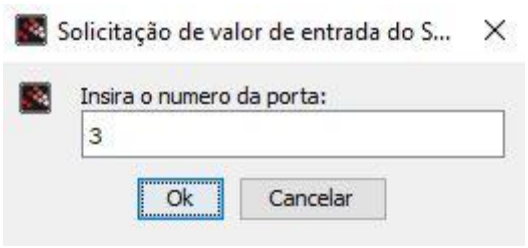

Figura 18. Primeira tela apresentada ao usuário.

Com a porta escolhida e liberada, a janela principal do sistema contendo todos os campos de dados e parâmetros de controle é exibida (figura 19).

Figura 19. Interface de controle do sistema.

Na janela principal do programa as seguintes áreas podem ser identificadas: uma que determina a aquisição e envio dos dados, outra que recebe os parâmetros de controle e, por último, os campos nos quais são apresentadas as informações gerais do sistema.

Na figura 20 é possível visualizar a área responsável pela aquisição dos dados, nela o usuário determina quando o programa deve iniciar, parar ou simplesmente fechar a janela principal de controle, essa é muito importante pois encerra corretamente a comunicação serial evitando que a comunicação continue.

Figura 20. Botões de controle do recebimento e envio de dados.

Um dado importante a se registrar diz respeito ao fato de que, assim que é iniciado o programa, uma nova janela surge para que seja definido o nome do arquivo .txt a ser gerado, contendo todos os dados do sistema. (Figura 21).

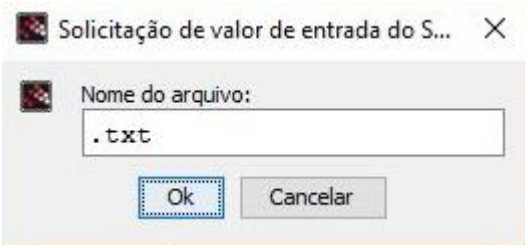

Figura 21. Janela para salvamento dos dados adquiridos pelo programa.

A área apresentada na figura 22 é responsável por receber os indicadores sendo possível selecionar o tipo de controle, assim como os parâmetros que o controlador vai utilizar. Para que para as funções feedforward e Feedback funcionem é preciso deixar marcado o campo On/Off que quando sozinho funciona apenas como liga e desliga.

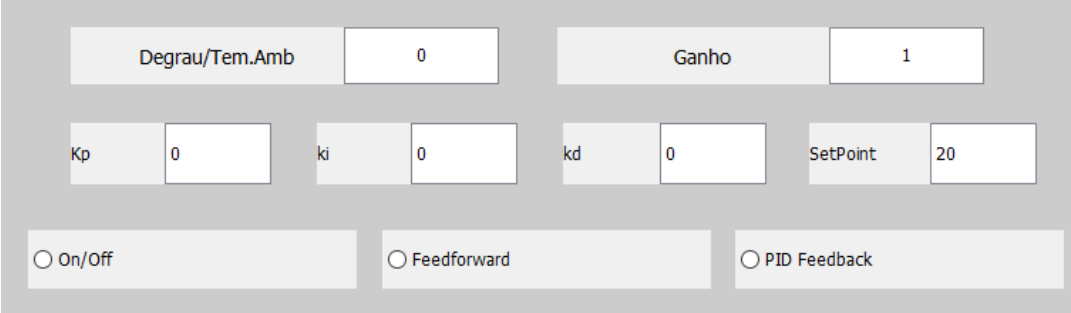

Figura 22. Área destinada à inserção de dados pelo usuário.

Os dados adquiridos do sistema como temperatura e vazão podem ser visualizados na área indicada pela figura 23, os valores apresentados nesse espaço são também mostrados num gráfico, nele são mostrados a variação da temperatura tanto de entrada, quanto de saída no intervalo de 5 minutos.

Figura 23. Campos de visualização dos dados dos sensores.

Além do gráfico gerado pelo próprio programa é possível com os dados armazenados no arquivo de texto gerar gráficos mais completos, contendo informações adicionais como o tempo transcorrido assim como todos os parâmetros utilizados durante todo o experimento.

6.2 Resposta do sistema a um degrau na resistência

Para encontrar função de transferência do sistema, basta obter a resposta frente um degrau na resistência, para isso o sistema é colocado em malha aberta, com um volume e vazão fixos. Para uma vazão constante de 50g/s segue, na figura 24, resposta ao degrau obtida pelo sistema.

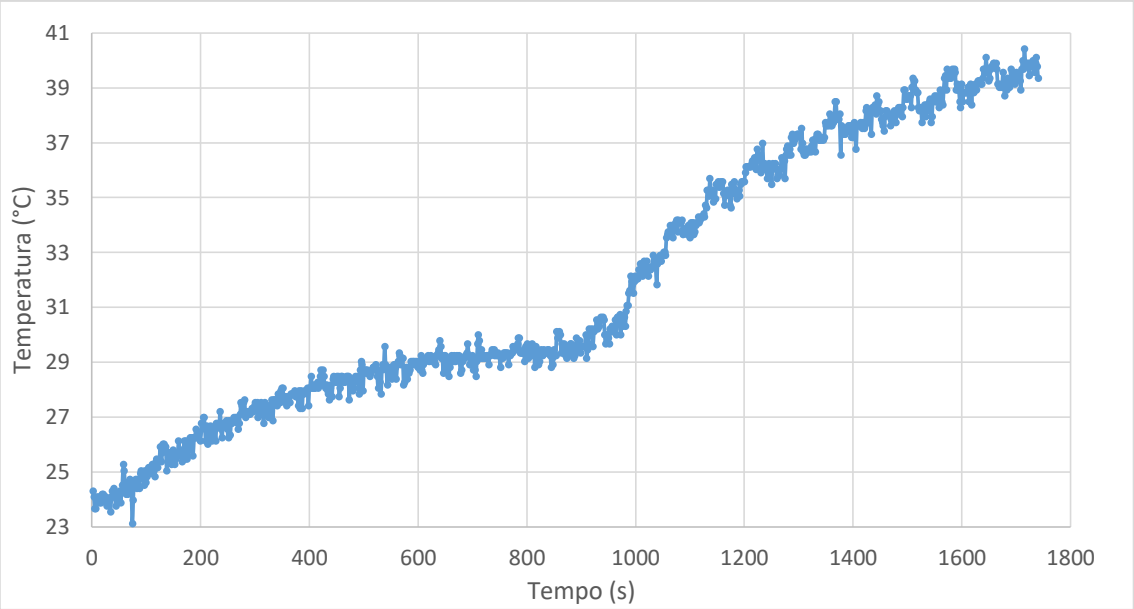

Figura 24. Resposta do sistema aos degraus na resistência.

O degrau dado na resistência será o considerado utilizando o intervalo entre os 10% e os 40% do *duty cycle*. Através da resposta e com os valores definidos inicialmente se obtém a função de transferência.

Pela resposta na saída do sistema se obtém a variação da temperatura, que passa de 29,6°C para 40°C.

Para uma variação de 63% na temperatura se obtém a constante de tempo como sendo τ de 315 segundos e coeficiente de atraso de L de 21 segundos.

Logo a função de transferência de primeira ordem do sistema para esse nível e essa vazão pode ser dada por:

$$G(s) = \frac{0{,}1312}{315s + 1} e^{-21s} \quad (07)$$

6.3 Modelagem para controle feedforward

Encontra o valor recebido de Q para 6800W (equação do calor que se relaciona com o PWM) considerando um *duty cycle* de 100%. Como a temperatura atingida nesse experimento foi de 53,87 °C para uma entrada de 22,9°C. Sendo c=1 e massa igual a 936,96g em 20 segundos.

$$Q = 5223177{,}022\ cal \quad (08)$$

$$Q = 6073{,}81\ W \tag{09}$$

Para encontrar o valor perdido para o ambiente basta tirar a potência recebida da potência enviada. A potência enviada é dada por:

$$Q_{perdido} = Q_{enviado} - Q_{recebido} \tag{10}$$

$$Q_{perdido} = 726{,}19\ W \tag{11}$$

Com o Q perdido encontra h.A

Sabendo a quantidade de energia térmica perdida é possível encontrar a constante h.A de perda de calor por convecção do sistema.

$$Q_{perdido} = h \cdot A \cdot (T_{externa} - T_{fluido}) \tag{12}$$

$$h \cdot A = -22{,}973423 \tag{13}$$

Com isso encontra o Q perdido para cada diferença de temperatura.

Através da tabela 5 é possível ver a energia térmica recebida para cada valor do *duty cycle.*

Tabela 5. Ganho de temperatura para cada valor de saída

PWM	% *duty cycle*	ΔT	Vazão(g/20s)	$Q_{recebido}$
0	0	0	1000	**0**
32	12,5	7,75	1115,78	**1810,00**
64	25	11,07	1113,23	**2579,48**
96	37,6	15,06	1048,37	**3304,75**
128	50,2	19,46	1011,22	**4118,96**
160	62,7	23,22	999,8	**4859,31**
192	75,3	26,35	969,95	**5349,70**
224	87,8	29,68	956,75	**5943,77**
255	100	30,97	936,96	**6073,82**

Essa tabela foi obtida ao se colocar o sistema em modo liga e desliga, ao inserir valores de saída do controlador e extrapolando o *set point* para atingir o maior valor possível para cada valor de saída. Essa variação pode ser vista na figura 25.

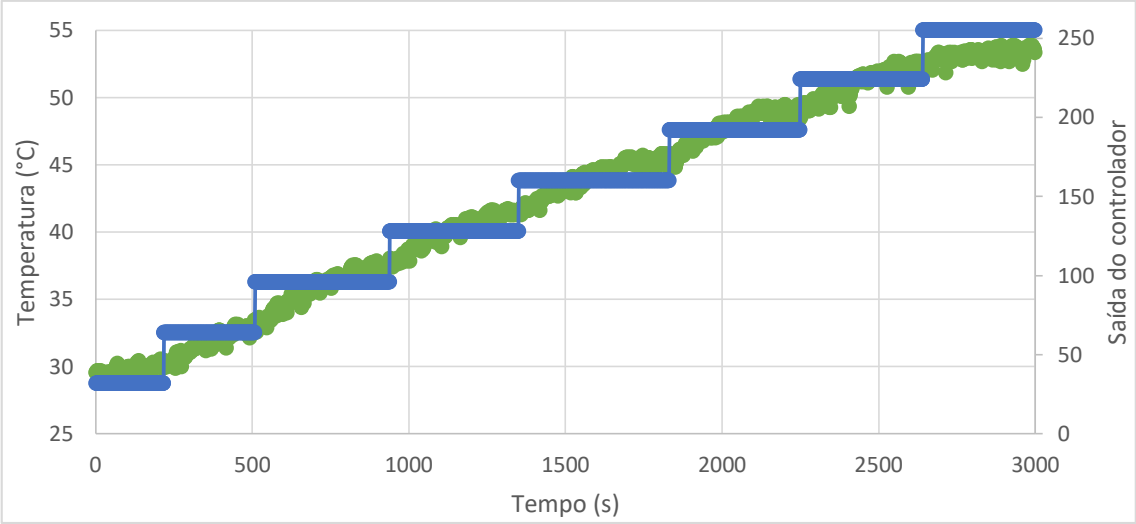

Figura 25. Variação da temperatura em função do degrau

Com isso se encontra uma equação, a qual relaciona o valor enviado pelo PWM com o valor recebido pelo sistema e somado ao valor perdido obtém-se o valor real enviado. Esses valores podem ser vistos na tabela 6.

Tabela 6. Quantidade de energia térmica fornecida, recebida e enviada

$Q_{fornecido} = Q_{perdido} + Q_{recebido}$	$Q_{perdido} = h \cdot A \cdot \Delta T$	$Q_{recebido} = m \cdot c \cdot \Delta T$	ΔT
0	0	0	0
1997,70	187,69	1810,00	7,75
2826,44	246,96	2579,48	11,07
3655,55	350,80	3304,75	15,06
4560,97	442,01	4118,96	19,46
5397,80	538,49	4859,31	23,22
5972,04	622,36	5349,70	26,35
6642,84	699,08	5943,77	29,68
6800	726,19	6073,82	30,97

Através dos valores encontrados é possível obter o gráfico da figura 26. No qual representa a quantidade de energia térmica real que chega ao sistema de acordo com o valor do sinal PWM enviado pelo Arduino à placa de potência desenvolvida.

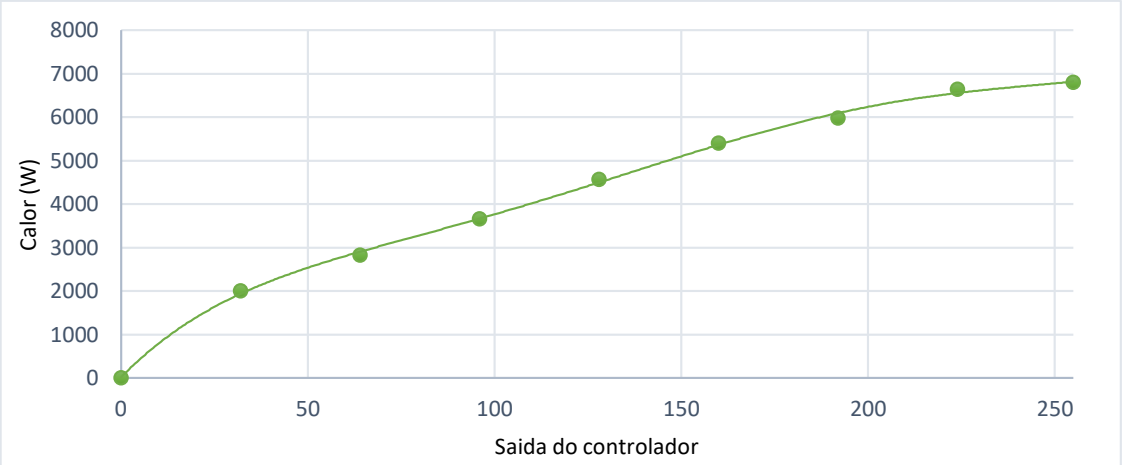

Figura 26. Energia térmica absorvida pelo sistema em função da saída.

Através da figura 26 é possível encontrar a equação abaixo.

$$Q_{fornecido} = 4 \cdot 10^{-8}x^5 - 3 \cdot 10^{-4}x^4 + 0{,}009x^3 - 1{,}1124x^2 + 87{,}439x + 15{,}18 \quad (14)$$

Com

$$R^2 = 0{,}9991 \quad (15)$$

Com essa equação é possível deixar a saída do controlador em função de todas as informações obtidas pelo sistema como: temperatura ambiente, temperatura da entrada e a equação de envio do sistema.

Assim para cada variação nesses parâmetros um novo valor na saída é gerado.

$$Output = a \cdot x^5 + b \cdot x^4 + c \cdot x^3 + d \cdot x^2 + e \cdot x^1 + f \quad (16)$$

Sendo:

$$a = 4 \cdot 10^{-8} \quad (17)$$

$$b = -3 \cdot 10^{-4} \quad (18)$$

$$c = 0{,}009 \quad (19)$$

$$d = -1{,}1124 \quad (20)$$

$$e = 87{,}439 \quad (21)$$

$$f = 15{,}18 + 22{,}973269 \cdot (TempAmb - Setpoint) - \ldots$$
$$\ldots - 180 \cdot Vazão \cdot (Setpoint - Entrada) \cdot 0{,}00116299354 \quad (22)$$

A equação é inserida no código fonte do controlador presente no software Scilab como a seguir:

Qperd=-22.973269(TempAmb-%Setpoint);*

*Qprecisa=180*Vazao1*(%Setpoint-data1)*0.00116299354;*

xx=15.18-Qperd-Qprecisa;

p=poly([xx,87.439,- 1.1124,0.009,- 0.00003,0.00000004],'x','coeff');

6.4 Respostas do sistema ao controle *feedforward*

Através das equações encontradas e inseridas no controlador pelo usuário é possível se obter a eficiência do controlador. Com a equação encontrada e inserida a resposta do sistema a uma variação no *set point* é vista na figura 27.

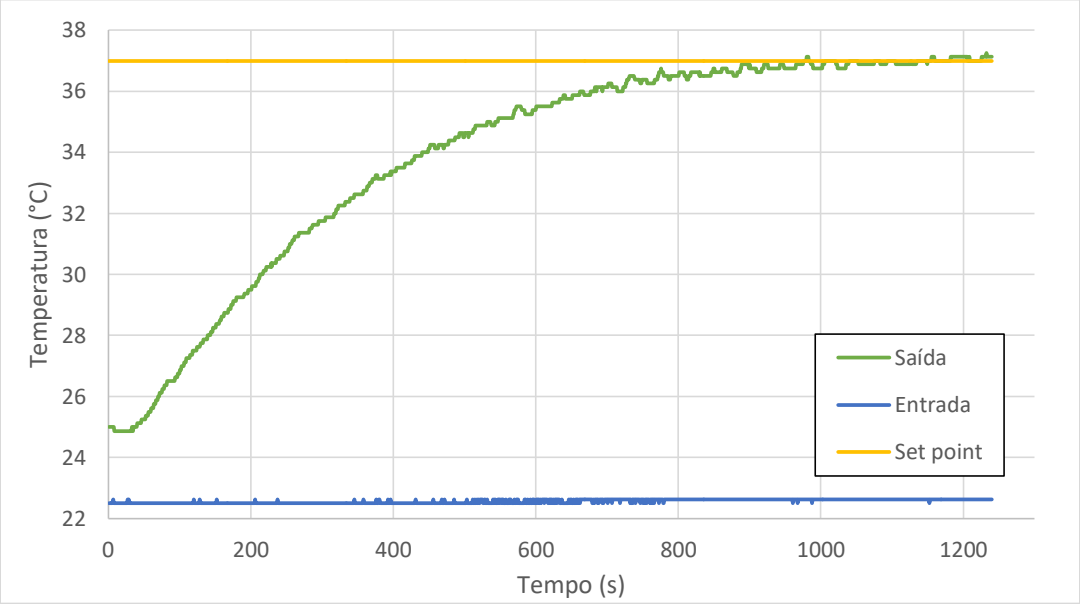

Figura 27. Resposta do sistema a uma variação no *set point*

É possível analisar o tempo de subida e o quanto de energia térmica o sistema fornece para uma entrada constante. Devido as características do controlador desse tipo após uma variação no *set point* o sistema responde de forma lenta e estabiliza próximo do valor estabelecido.

Após uma variação no *set point* analisa-se a estabilidade do sistema para uma entrada constante, sua resposta também reflete o quão próximo o sistema identifica a perda de energia térmica para o ambiente e o compensa (figura 28).

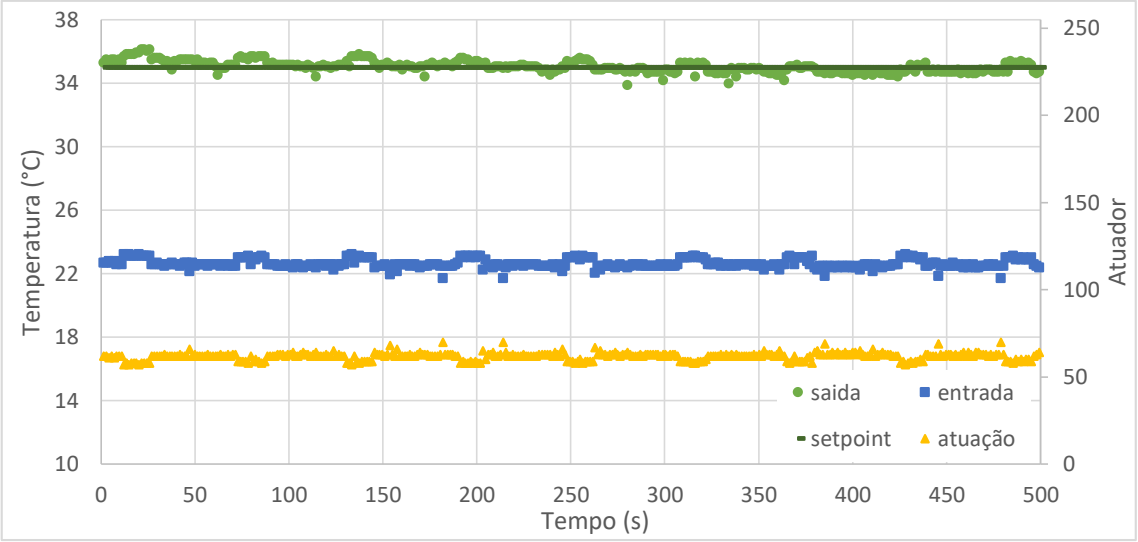

Figura 28. Resposta do sistema para um controle *feedforward* sem perturbação.

O objetivo de um controle desse tipo é o de atuar de forma a anular a perturbação em sua entrada, sendo assim foram simulados distúrbios aleatórias na entrada do sistema. Sua atuação frente às variações em sua entrada podem ser observada na figura 29.

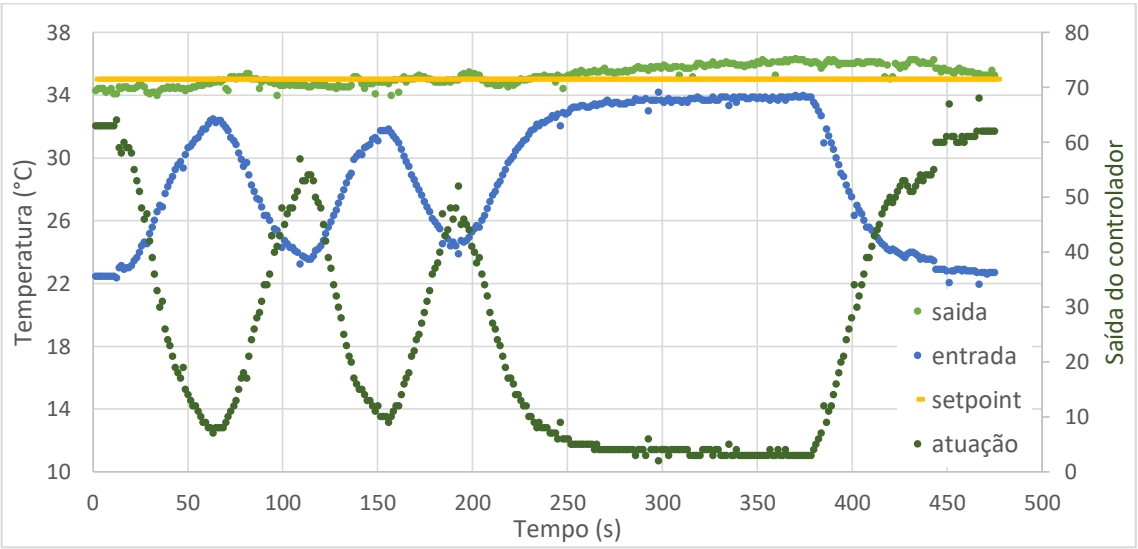

Figura 29. Resposta do sistema para um controle *feedforward* com perturbação.

Essa análise mostra que o sistema consegue responder de forma rápida às variações grandes em sua entrada, no qual uma variação de 10°C representa uma alteração de apenas 1°C grau na saída.

7. CONCLUSÃO E TRABALHOS FUTUROS

Conclui-se que os resultados obtidos alcançaram o objetivo principal do trabalho, a viabilidade da construção de um módulo experimental de baixo custo e baseado em softwares livres. O equipamento construído mostra-se de fácil utilização para fins educacionais, permite ao aluno aplicar os conhecimentos de transferência de calor, modelagem e controle de processos. Portanto, é um sistema adequado para a aplicabilidade de correções de perturbações na entrada do processo.

Visando a aplicação em trabalhos futuros o módulo desenvolvido permite incorporar novos sensores e ainda possibilita a inclusão de novos métodos de controle. Outra vantagem refere-se a liberdade de configuração do módulo, tendo em vista que em conjunto alunos e docentes podem atribuir novas funcionalidades ao módulo.

REFERÊNCIAS

ARDUINO CORPORATION. **Arduino Uno Rev3**. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 2 jan. 2018.

BAMPI, C. L. **Recondicionamento e projeto de controle para uma bancada didática de processo térmico**. [s.l: s.n.].

BERTACHI, A. H. et al. **Controle de um processo multivariável em uma planta didática industrial utilizando redes neurais**. XI Simpósio Brasileiro de Automação Inteligente (SBAI 2013), 2013, Fortaleza - CE. **Anais**...2013

BOTELHO, V. R. et al. **Sistema didático de controle de temperatura**. COBEQ. **Anais**...2016

BRAGA, N. C. **Controles PWM de potência (art006)**. Disponível em: <http://www.newtoncbraga.com.br/index.php/projetos/375-controles-pwm-de-potencia-art006>. Acesso em: 10 jan. 2017.

CANIATO, L. C. **Modelagem e controle de nível e temperatura em sistema de armazenamento de água purificada para uso em empresa farmacêutica**. [s.l.] Centro Universitário do Instituto Mauá de Tecnologia, 2006.

CASTRUCCI, P. DE L.; BITTAR, A.; SALES, R. M. **Controle automático**. [s.l.] LTC, 2011.

CAVAZZANA, E.; FILHO, J. D.; SOUZA, E. M. R. DE. **Construção de uma plataforma didática para estudo da técnica de controle gain schedulling utilizando um escalonador mecânico de ganhos**. COBENGE. **Anais**...ABENGE, 2011

COUGHANOWR, D.; KOPPEL, L. **Análise e controle de processos**. [s.l: s.n.].

DORF, RICHARD C., AND R. H. B. **Modern Control Systems**. [s.l: s.n.].

EMPÓRIO DO ACRÍLICO. **Chapas de acrílico**. Disponível em: <http://www.emporiodoacrilico.com.br/produto/10/chapasdeacrlico.html>. Acesso em: 2 fev. 2017.

ESI GROUP. **About Scilab**. Disponível em: <https://www.scilab.org/en/scilab/about>. Acesso em: 25 jan. 2017.

GHIRARDELLO, A. **Curso técnico em eletrônica eletrônica industrial -**

apostila sobre modulação PWMColégio Politec, [s.d.]. Disponível em: <www.mecatronicadegaragem.blogspot.com>

GNOATTO, F. et al. Projeto e desenvolvimento de módulo de controle de nível em escala piloto. **Engevista**, p. 280–293, 2016.

GOMES, F. J. et al. **Módulo laboratorial de baixo custo, baseado em FOSS, para educação em engenharia de controle de processos industriais**. CLAGTEE. **Anais**...2011

GOSMANN, H. L. **Um sistema multivariável de tanques acoplados para avaliação de técnicas de controle**. [s.l.] UNIVERSIDADE DE BRASÍLIA FACULDADE DE TECNOLOGIA, 2002.

GUERRA, L. N. DE A. **Uso de compensador PID no controle da taxa de variação de temperatura em um forno elétrico a resistência**. [s.l: s.n.].

HOTZ, J. D. S. **Bancada didática para controle de nível e temperatura**. [s.l.] UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, 2014.

INSTITUTO NACONAL PARA DESENVOLVIMENTO DE ACRÍLICO. **Acrílico, do começo ao fim**, [s.d.]. Disponível em: <http://www.indac.org.br/arquivos/acrilico_indac.pdf>. Acesso em: 20 fev. 2017

ISHIGAKI, F. M. et al. **Inclusão do protótipo didático 'fan plate' em laboratório de experimentação remota**. COBENGE. **Anais**...Brasília: 2004Disponível em: <http://www.abenge.org.br/cobenge/arquivos/15/artigos/03_507.pdf>

KOMATSU, P. W.; JUNIOR, L. M.; KAISER, W. **Eletrônica de potência I**, 2017.

LEBLANC, S. E.; COUGHANOWR, D. R. **Process Systems Analysis and Control**. 3ª ed. New York: McGraw-Hill Education, 2009.

LIMA, A. et al. **Controle feedforward adaptativo através de estimação de amplitude para uma máquina eletromecânica de ensaio de fadiga**. XII Simpósio Brasileiro de Automação Inteligente (SBAI). **Anais**...2015

MARTINAZZO, C. A.; ORLANDO, T. Comparação entre três tipos de sensores de temperatura em associação com arduíno. **Perspectiva**, p. 93–104, 2016.

MARTINELLI, B. Z. **Aplicações de técnicas de controle preditivo baseado em modelo**. [s.l.] Universidade Federal do Rio de Janeiro, 2018.

MAXIM INTEGRATED. Datasheet DS18B20. **Maxim Integrated**, v. 92, p. 20, 2015.

NASCIMENTO, A. G. DO; OLIVEIRA, L. S. DE. **Controle automático de processos**Rio de JaneiroSENAI-RJ, , 2006.

OGATA, K. **Engenharia de controle moderno**. 4. ed. [s.l: s.n.].

OLIVEIRA, A. L. DE L. **Fundamentos de controle de processo**, 1999.

ORD, J. John Dewey and Experiential Learning : Developing the theory of youth work. **Youth & Policy**, n. 108, p. 55–72, 2012.

RIBEIRO, M. A. **Controle de Processo**. Salvador: Tek Treinamento & Consultoria, 2005.

ROCHA, L. F. et al. **Controle preditivo na otimização de moinho secador de carvão na indústria de mineração**. Brasil Automation. **Anais**...Brasil Automation, 2010

RODRÍGUEZ, F., BERENGUEL, M., ARAHAL, M. R. Feedforward controllers for greenhouse climate control based on physical models. **Procedings of the European Control Conference 2001**, v. 5, p. 2158–2163, 2001.

SCHMID, C. Remote experimentation techniques for teaching control engineering. n. June, 2000.

SILVA, R. B.; LOPES, M. P.; AMARAL, L. S. **Projeto e construção de uma planta didática para ensino de estratégias de controle de nível, vazão e temperatura em cursos de engenharia**. COBENGE. **Anais**...2012

SINTEGLAS. **S-330**. Disponível em: <https://sinteglas.com.br/images/stories/pdfs/s330.pdf>. Acesso em: 10 fev. 2017.

SINTEGLAS. **Informações técnicas sobre colagem de chapas acrílicas**. Disponível em: <http://www.indac.org.br/arquivos/catalogo-colagem.pdf>. Acesso em: 10 fev. 2017.

STEFANOVIC, M. et al. A LabVIEW-based remote laboratory experiments for control engineering education. **Computer Applications in Engineering Education**, v.

19, n. 3, p. 539–549, 2011.

TEIXEIRA, B. O. S.; JOTA, F. G.; TEIXEIRA, M. H. **Modelagem, controle e otimização do processo dos fornos de reaquecimento de placas**. Controle & Automação Sociedade Brasileira de Automatica. **Anais**...2007

TEXAS, I. Moc3020. n. October 1986, p. 8, 1998.

TZOUANAS, V.; STEVENSON, M. **Temperature and level control of a multivariable water tank process**. American Society for Engineering Education, 2013

APÊNDICE A

Código de programação do atuador no Arduino UNO

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 10
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress sensor1, sensor2;
const int Actuatorresistencia = 11;
const int Actuatorbomba = 3;
int estado1;
int bomba;
int resistencia;
int indice;
String new_str[2]; //array de 2 posições
String tempString; //temporario
char a; //para armazenar o caracter da serial
void setup(){
  pinMode(Actuatorresistencia, OUTPUT);
  pinMode(Actuatorbomba, OUTPUT);
  Serial.begin(9600);
  sensors.begin();
  sensors.getAddress(sensor1, 0);
  sensors.getAddress(sensor2, 1);
  TCCR2B = TCCR2B & 0b11111000 | 0x07;
  //precisão do sensor (9, 10, 11, ou 12 bits)
  sensors.setResolution(sensor1, 11);
  sensors.setResolution(sensor2, 11);
}
void loop(){
  float inByte;
  estado1=0;
  sensors.requestTemperatures();
  float tempC1 = sensors.getTempC(sensor1);
  float tempC2 = sensors.getTempC(sensor2);
  if(!Serial.available()){
    Serial.print(tempC1);
    Serial.print("   ");
    Serial.print(tempC2);
    Serial.print("   ");
    Serial.print(bomba);
    Serial.println();
  }
  indice=0;
    while (Serial.available() > 0) {
      tempString = " "; //Um espaço para esvaziar a string.
      while(true){
        a = Serial.read(); //leia um caractere
        if(a == ';') break; //ser for ponto-e-virgula, sai do while
        else tempString += a; //adiciona na string temporária
      }
      new_str[indice] = tempString; //armazeno no array de strings
      indice++; //incremento o contador
      resistencia=new_str[0].toInt();
      bomba=new_str[1].toInt();
```

```
    analogWrite(Actuatorresistencia, resistencia);
    analogWrite(Actuatorbomba, bomba);
  }
}
```

APÊNDICE B

Código de programação do controlador no Scilab

```
//Limpa
clc
//Pergunta porta de entrada do Arduino
port_name=evstr(x_choose(['  1';'  2';'  3';'  4';'  5'],['Selecione o número da porta: ']))
if port_name == 0 then
   delete(f);
   mclose;
end
////////// Janela
f=figure('figure_position',[400,50],'figure_size',[878,631],'auto_resize','on','background',[33],'figure_nam
e','Temperature Control','tag','mainWindow','menubar_visible','off','toolbar_visible','off');
////////// Botões
handles.dummy = 0;
handles.obj1= newaxes();handles.obj1.margins = [ 0 0 0 0];handles.obj1.axes_bounds =
[0.0754060,0.4299242,0.8665893,0.5321970];
handles.Iniciar=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.0623666,0.8825757,0.142
529,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Start','Style','pushbutton','Value',[0],'Vertic
alAlignment','middle','Visible','on','Tag','Iniciar','Callback','iniciar')
handles.Parar=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.0623666,0.8036616,0.142
529,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Stop','Style','pushbutton','Value',[0],'Vertic
alAlignment','middle','Visible','on','Tag','Parar','Callback','Parar')
handles.Fechar=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.0623666,0.6458333,0.142
529,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Close','Style','pushbutton','Value',[0],'Verti
calAlignment','middle','Visible','on','Tag','Fechar','Callback','Fechar')
handles.obj6=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.2790255,0.7578788,0.103248
3,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','Kp','Style','text','Value',[0],'VerticalAlignment'
,'middle','Visible','on','Tag','obj6','Callback','')
handles.obj7=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.4058623,0.7578788,0.103248
3,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','ki','Style','text','Value',[0],'VerticalAlignment','
middle','Visible','on','Tag','obj7','Callback','')
handles.obj8=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.5326992,0.7578788,0.103248
3,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','kd','Style','text','Value',[0],'VerticalAlignment',
'middle','Visible','on','Tag','obj8','Callback','')
handles.obj9=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
```

```
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.659536,0.7578788,0.1032483
,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','SetPoint','Style','text','Value',[0],'VerticalAlign
ment','middle','Visible','on','Tag','obj9','Callback','')
handles.OnOff=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.2574246,0.6480303,0.169373
5,0.0625],'Relief','default','SliderStep',[0.01,0.1],'String','On/Off','Style','radiobutton','Value',[0],'VerticalAli
gnment','middle','Visible','on','Tag','OnOff','Callback','OnOff')
handles.feedforward=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.4392575,0.6480303,0.169373
5,0.0625],'Relief','default','SliderStep',[0.01,0.1],'String','Feedforward','Style','radiobutton','Value',[0],'Vert
icalAlignment','middle','Visible','on','Tag',' feedforward','Callback',' feedforward')
handles.feedback=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.6210905,0.6480303,0.169373
5,0.0625],'Relief','default','SliderStep',[0.01,0.1],'String','PID
Feedback','Style','radiobutton','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','feedback','Callbac
k','feedback')
handles.kp=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.3272668,0.7578788,0.055127
6,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','0','Style','edit','Value',[0],'VerticalAlignment','
middle','Visible','on','Tag','kp','Callback','kp')
handles.ki=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.4539443,0.7578788,0.055127
6,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','0','Style','edit','Value',[0],'VerticalAlignment','
middle','Visible','on','Tag','ki','Callback','ki')
handles.kd=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.5826218,0.7578788,0.055127
6,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','0','Style','edit','Value',[0],'VerticalAlignment','
middle','Visible','on','Tag','kd','Callback','kd')
handles.setpoint=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.7212993,0.7578788,0.055127
6,0.0662879],'Relief','default','SliderStep',[0.01,0.1],'String','20','Style','edit','Value',[0],'VerticalAlignment'
,'middle','Visible','on','Tag','setpoint','Callback','setpoint')

handles.obj17=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[14],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[0.12,0.56,1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'
Position',[0.8596288,0.9306818,0.0800464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Inp
ut','Style','text','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','obj17','Callback','')
handles.entrada=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.8596288,0.8636616,0.080
```

```
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','
','Style','edit','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','entrada','Callback','entrada')
handles.obj19=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[14],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.8596288,0.7866414,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Output','Style','text','Value',[0],'VerticalAli
gnment','middle','Visible','on','Tag','obj19','Callback','')
handles.saida=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.8596288,0.7196212,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','
','Style','edit','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','saida','Callback','saida')
handles.obj20=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[14],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.8596288,0.6496212,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Flow','Style','text','Value',[0],'VerticalAlign
ment','middle','Visible','on','Tag','obj19','Callback','')
handles.vazao=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.8596288,0.5796212,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','0','Style','edit','Value',[0],'VerticalAlignme
nt','middle','Visible','on','Tag','saida','Callback','vazao')
handles.obj21=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[14],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.2796288,0.8636616,0.150
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Step/Tem.Amb','Style','text','Value',[0],'Ve
rticalAlignment','middle','Visible','on','Tag','obj19','Callback','')
handles.degrau=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.4196288,0.8636616,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','0','Style','edit','Value',[0],'VerticalAlignme
nt','middle','Visible','on','Tag','saida','Callback','vazao')
handles.obj22=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[14],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.5296288,0.8636616,0.150
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','Gain','Style','text','Value',[0],'VerticalAlign
ment','middle','Visible','on','Tag','obj19','Callback','')
handles.ganho=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight',
'normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.6696288,0.8636616,0.080
0464,0.0606061],'Relief','default','SliderStep',[0.01,0.1],'String','1','Style','edit','Value',[0],'VerticalAlignme
nt','middle','Visible','on','Tag','saida','Callback','vazao')

//////////
top_axes_bounds = [0.0754060,0.4299242,0.8665893,0.5321970];
bottom_axes_bounds = [0.0754060,0.4299242,0.8665893,0.5321970];
//limites do gráfico
minTempDisplay = 0;
maxTempDisplay = 80;
```

```
timeBuffer = 200;
//Grafico do sensor 1
subplot(222);
a = gca();
a.axes_bounds = top_axes_bounds;
a.tag = "minuteAxes";
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), color("blue"));
a.title.text="Last 5 minutes";
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
e = gce();
e = e.children(1);
e.tag = "minutoSensor1";
//Grafico do sensor 2
a = newaxes();
a.y_location = "right";
a.filled = "off"
a.axes_bounds = top_axes_bounds;
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), color("black"));
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
a.axes_visible(1) = "off";
a.foreground=color("black");
a.font_color=color("black");
e = gce();
e = e.children(1);
e.tag = "minutoSensor2";
//Grafico da atuação do controlador
a = newaxes();
a.y_location = "right";
a.filled = "off"
a.axes_bounds = top_axes_bounds;
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), color("orange"));
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
a.axes_visible(1) = "off";
//a.foreground=color("orange");
//a.font_color=color("orange");
e = gce();
e = e.children(1);
e.tag = "atuador";
//Linha de SetPoint
%Setpoint=20;
plot([0, 200], [%Setpoint, %Setpoint]);
e = gce();
e = e.children(1);
e.tag = "instantMinTemp";
e.line_style = 5;
e.thickness = 2;
e.foreground = color("green");
e.data(:,2) = %Setpoint;
lastErr=0;

//Funções secundárias

function
   global %serial_port1
   saida= 'L;'+'L;';
   writeserial(%serial_port1,saida);
   closeserial(%serial_port1);
```

```
endfunction

function
  e = findobj("tag", "minuteSensor");
  e.data(:, 2) = 0;
  clc
  clear
endfunction

function
  Parar();
  global %serial_port1
  f = findobj("tag", "mainWindow");
  delete(f);
  mclose;
endfunction

function
  global %vazao
  Vazao1=0;
  Vazao1=get(handles.vazao,'String');
  ganho=get(handles.ganho,'String');
  ganho=evstr(ganho);
  %Vazao1=evstr(Vazao1);
  %Vazao2=255-%Vazao1;
  Vazao1=string(%Vazao2);

  //agora precisa calcular o a vazao equivalente para cada valor do 0 ao 255
  //foi encontrado 2 equações para intervalos distintos do pwm relacionado a vazão
  Vaztemp=0;
  if %Vazao1>=20 & %Vazao1<=112 then
    Vaztemp=873.73*log(%Vazao1)-2570.6;
    Vaztemp=Vaztemp*ganho;
    end
  if%Vazao1>112 & %Vazao1<=255 then
    Vaztemp=-0.0191*%Vazao1*%Vazao1+8.7598*%Vazao1+753.47;
    Vaztemp=Vaztemp*ganho;
  end
endfunction

function
  exec(vazao);
  global %Output;
      if data2< %Setpoint then
        Degrau=get(handles.degrau,'String');
        Output = string(Degrau);
        saida= Output+';'+Vazao1+';';
        writeserial(%serial_port1,saida);
      else
        Output = '0';
        saida= Output + ';'+Vazao1+';';
        writeserial(%serial_port1,saida);
      end
  timeChange=timer();   //mede o tempo entre cada medição da temperatura
  timeChange = evstr(timeChange);
  t=timeChange;
  t=string(timeChange);
```

```
    e=string(data1);
    s=string(data2);

endfunction

function
    global %Output;
    global %Q;
    global %TempAmp;
    Degrau=get(handles.degrau,'String');
    exec(vazao);
    %TempAmp=evstr(Degrau);
    if %Setpoint>data1 & %TempAmp<%Setpoint then
        Qperd=-22.973269*(%TempAmp - %Setpoint);
        Qprecisa=180*Vaztemp*(%Setpoint-data1)*0.00116299354;
        xx=15.18-Qperd-Qprecisa;
        p=poly([xx,87.439,- 1.1124,0.009,- 0.00003,0.00000004],'x','coeff');
        raizes=roots(p);
        posicao=1;
        while posicao<=5 do
            raiz1=evstr(string (real(raizes(posicao:posicao))));
            raiz2=evstr(string (imag(raizes(posicao:posicao))));
            if raiz2==0 then
                if raiz1<=255 & raiz1>=0 then
                  %Output=round(raiz1);
                  %Output=round(%Output);
                end
             end
            raiz1=0;
            raiz2=0;
            posicao = posicao + 1;
       end
       else
          %Output=0;
    end

    Output=round(%Output)
    Output=string(%Output)
    timeChange=timer();   //mede o tempo entre cada medição da temperatura
    timeChange = evstr(timeChange);
    t=timeChange;
    t=string(timeChange);
    ee=string(data1);
    ss=string(data2);

endfunction

//Função responsavel pelo controle PID
function
     exec(kp);
     exec(ki);
     exec(kd);
     exec(vazao);
     global %Output;
     timeChange=timer();   //mede o tempo entre cada medição da temperatura
     timeChange = evstr(timeChange);
     //Equação do controlador PID digital simples
```

```
    erro = %Setpoint - data2;
    errSum = (erro * timeChange);
    errSum = errSum + (erro * timeChange);
    dErr = (erro - lastErr) / timeChange;
    Output = Kp1 * erro + Ki1 * errSum + Kd1 * dErr;
   //conversão dos sinais
    Output=round(Output);
    Output=string(Output);
    ee=string(data1);
    s=string(data2);
    t=string(timeChange);
    lastErr = erro;
    data2=0;
    //restringindo saída do controlador
    %Output=evstr(Output);
      if(%Output > 255) then
        Output = '255'
      end
      if(%Output < 0) then
        Output = '0'
      end

endfunction

Degrau=get(handles.degrau,'String');

function
  global %onoff
  onoff=0;
  onoff=get(handles.OnOff,'Value');
  %onoff=onoff;
endfunction

function
  global %Feedforward
  Feedforward=0;
  Feedforward=get(handles.feedforward,'Value');
  %Feedforward = Feedforward;
endfunction

function
  global %Feedback
  Feedback=0;
  Feedback=get(handles.feedback,'Value');
  %Feedback=Feedback;

endfunction

function
  global %Obj23
  Obj23=0;
  Obj23=get(handles.obj23,'Value');
  %Obj23=Obj23;

endfunction

function
```

```
  global %kp
  Kp=get(handles.kp,'String');
  Kp1=evstr(Kp);
endfunction

function
  global %ki
  Ki=get(handles.ki,'String');
  Ki1=evstr(Ki);
endfunction

function
  global %kd
  Kd=get(handles.kd,'String');
  Kd1=evstr(Kd);
endfunction

function
  global %Setpoint
  Setpoint=get(handles.setpoint,'String');
  %Setpoint=evstr(Setpoint);
  if  %Setpoint>50 then
    messagebox(["TEMPERATURA CRÍTICA" "Selecione um valor menor"], "AVISO!!")
    %Setpoint=50;
    set(handles.setpoint,"string",'50');
  end

  e = findobj("tag", "instantMinTemp");
  e.data(:,2) = %Setpoint;
endfunction

//function obj24_callback(handles)
 //
//endfunction

//Função Prinncipal
function
  arq=x_dialog('Nome do arquivo: ',".txt")
  fd = mopen(arq,'wt');
  timer();
  tic();
  global %MaxTemp
  global %serial_port1
  global %Acquisition
  %Acquisition = %t;
  global %fanStatus
  %fanStatus = 0;
  %serial_port1=openserial(port_name,"9600,n,8,1");
 // Arduino toolbox
 lastErr=0;
  values1=[];
  value1=ascii(0);
  while %Acquisition
    while(value1~=ascii(13)) then
     value1=readserial(%serial_port1,1);
     values1=values1+value1;
     v1=strsubst(values1,string(ascii(10)),'')
```

```
    v1=strsubst(v1,string(ascii(13)),'')
    s1=part(v1,1:5)
    s2=part(v1,8:13)
    s3=part(v1,16:20)
    data1=evstr(s1);
    data2=evstr(s2);
    data3=evstr(s3);
    end
  set(handles.entrada,"string",string(s1));
  set(handles.saida,"string",string(s2));
  values1=[]
  value1=ascii(0);
  //plotando os dados do sensor 1
  e=findobj('tag',"minutoSensor1");
  dataanterior2 = e.data(:, 2);
  e.data(:, 2) = [dataanterior2(2:$) ; data1];
  //plotando os dados do sensor 2
  e=findobj('tag',"minutoSensor2");
  dataanterior2 = e.data(:, 2);
  e.data(:, 2) = [dataanterior2(2:$) ; data2];
  //executando as funções para as variáveis do controle
  exec(OnOff);
  exec(feedback);
  exec(feedforward);
  exec(setpoint);
  exec(kp);
  exec(ki);
  exec(kd);
  exec(obj23);

  //codigo responsavel pelo controle
  if %onoff == 1 & %Feedback == 0 & %Feedforward == 0 then
     set(handles.obj21,'String','Degrau');
     exec(Onoff1);
     xinfo("Selecione o tipo de controle");
     mfprintf(fd,'Setpoint %f,Output %s,Tempo %s,Entrada %s,Saida %s,Vazão %f,P %f,I %f,D
%f,TempAmp %s,%s,\n',%Setpoint,Output,t,e,s,%Vazao1,Kp1,Ki1,Kd1,Degrau,'OnOff');

end

  //Codigo do FeedBack
  if %onoff == 1 & %Feedback == 1 & %Feedforward == 0 then
     exec(PID);
  //Limites para visualização no grafico
     k=(80*%Output)/255;
  //Envia os dados do controlador para o gráfico
     e=findobj('tag',"atuador");
     dataanterior2 = e.data(:, 2);
     e.data(:, 2) = [dataanterior2(2:$) ; k];
     saida= Output+';'+Vazao1+';';
     writeserial(%serial_port1,saida);
     xinfo("Feedback");
     mfprintf(fd,'Setpoint %f,Output %s,Tempo %s,Entrada %s,Saida %s,Vazão %f,P %f,I %f,D
%f,TempAmp %s,%s\n',%Setpoint,Output,t,ee,s,%Vazao1,Kp1,Ki1,Kd1,Degrau,'Feedback');
     tic();
  end
```

```
  //Codigo do FeedForward
  if %onoff == 1 & %Feedback == 0 & %Feedforward == 1 then
     set(handles.obj21,'String','Tem.Amb');
     exec(REALIM);
  //Limites para visualização no grafico
     k=(80*%Output)/255;
  //Envia os dados do controlador para o gráfico
     e=findobj('tag',"atuador");
     dataanterior2 = e.data(:, 2);
     e.data(:, 2) = [dataanterior2(2:$) ; k];
     saida= Output+';'+Vazao1+';'
     writeserial(%serial_port1,saida);
     xinfo("Feedforward");
     mfprintf(fd,'Setpoint %f,Output %s,Tempo %s,Entrada %s,Saida %s,Vazão %f,P %f,I %f,D
%f,TempAmp %s,%s\n',%Setpoint,Output,t,ee,ss,%Vazao1,Kp1,Ki1,Kd1,Degrau,'Feedforward');
     tic();
  end

  if %Feedback == 1 & %Feedforward == 1 & %onoff == 1 then
     end
  //Caso nada tenha sido selecionado
  if %Feedback == 0 & %Feedforward == 0 & %onoff == 0 then
     set(handles.obj21,'String','Degrau/Tem.Amb');
     exec(vazao);
     xinfo("Nenhuma opção selecionada");
     saida= 'L;'+Vazao1+';';
     writeserial(%serial_port1,saida);
     end

 end

endfunction
```