Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Nelson de Figueiredo Barroso

Instrumentacão Virtual Aplicada à Automação de um Sistema Térmico para Experimentação Via Web

Divinópolis
2014

Nelson de Figueiredo Barroso

Instrumentacão Virtual Aplicada à Automação de um Sistema Térmico para Experimentação Via Web

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Projeto e Automação, Circuitos Elétricos e Eletrônicos, Programação de Computadores e Computação Aplicada e Controle de Processos.

Orientador: prof. Dr. Valter Júnior de Souza Leite
Co-orientador: prof. Dr. Márcio Fantini Miranda

Divinópolis
2014

Nelson de Figueiredo Barroso

Instrumentacão Virtual Aplicada à Automação de um Sistema Térmico para Experimentação Via Web

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Projeto e Automação, Circuitos Elétricos e Eletrônicos, Programação de Computadores e Computação Aplicada e Controle de Processos.

Comissão Avaliadora:

Prof. Dr. Valter Júnior de Souza Leite
CEFET-MG / *Campus* Divinópolis

Prof. Dr. Márcio Fantini Miranda
COLTEC/UFMG

Prof. M.Se Marlon Henrique Teixeira
CEFET-MG / *Campus* Divinópolis

Prof. Lucas Silva Oliveira
CEFET-MG / *Campus* Divinópolis

Divinópolis
2014

Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Monografia intitulada "Instrumentacão Virtual Aplicada à Automação de um Sistema Térmico para Experimentação Via Web" de autoria do graduando Nelson de Figueiredo Barroso, aprovada pela banca constituída pelos seguintes professores:

Prof. Dr. Valter Júnior de Souza Leite - CEFET-MG / *Campus* Divinópolis Divinópolis - Orientador

---

Prof. Dr. Prof. Dr. Márcio Fantini Miranda - COLTEC/UFMG - Co-orientador

---

Prof. M.Se Marlon Henrique Teixeira - CEFET-MG / *Campus* Divinópolis

---

Prof. Lucas Silva Oliveira - CEFET-MG / *Campus* Divinópolis

Prof. Dr. Valter Júnior de Souza Leite
Coordenador do Curso de Engenharia Mecatrônica
CEFET-MG / *Campus* Divinópolis

Divinópolis, Fevereiro de 2015

Dedico este trabalho à minha mãe Maria das Graças pelo amor incondicional e por não medir esforços para que eu pudesse completar mais uma etapa de minha vida. Dedico também à minha irmã Kellen Barroso Pais Figueiredo por continuar sempre presente.

Agradecimentos

Agradeço,

A Deus por me dar a consciência de que preciso me tornar uma pessoa melhor a cada dia e por me fornecer meios (exemplos e desafios) para alcançar tal objetivo.

À minha mãe pelo amor e apoio incondicionais dedicados a mim e às minhas irmãs e por nos ensinar a sermos honestos, dignos, corajosos, paciêntes e principalmente a termos fé em Deus e confiança em nós mesmos.

Às minhas irmãs Ana Lídia e Kátia pelo carinho, apoio e por sempre confiarem em minha capacidade e aos meus sobrinhos Gabriel Ângelo e Davi por me fazerem mais feliz.

Ao meu pai por sempre me impulsiona para frente não importa se através de seus bons ou maus exemplos.

À minha menininha Mariana de Fátima Madureira por estar o tempo todo ao meu lado, pelo companheirismo, cumplicidade e compreenção e por me motivar a seguir em frente sempre. Agradeço também aos seus pais Erlane e Maurício Madureira pelo apoio e incentivo durante todos esses anos.

Ao meu padrinho Antônio Cursage por me colocar no ponto de partida de toda essa trajetória e por nunca me recusar qualquer ajuda. Agradeço também à minha tia Maria do Socorro por me ajudar durante o tempo que precisei e aos meus tios Sálvio e Estela pelo apoio e carinho com que me receberam em minha nova trajetória.

A todos os professores e servidores do CEFET-MG / *Campus* Divinópolis por contribuírem para minha formação como aluno e como pessoa.

Em especial ao meu professor orientador Valter Júnior de Souza Leite por ser sempre uma inspiração a todos os alunos e por não poupar esforços, juntamente com outros grandes professores, para ajudarem a tornar o CEFET-MG / *Campus* Divinópolis uma instituição cada dia melhor. Espero poder um dia contribuir também para esse fim.

Agradeço ao amigo Álan Cristoffer Sousa por sua grande ajuda e participação no desenvolvimento do modelo proposto para o WebLab. Da mesma forma, agradeço ao amigo Jônathas V. V. Silva pela pela parceria na implantação das melhorias na planta do sistema de aquecimento de ar utilizada neste trabalho

Finalmente agradeço a todos os meus amigos, que com toda certeza fizeram parte desta conquista. Eles se identificarão!

A fé robusta dá a perceverança, a energia e os recursos que fazem com que se vençam os obstáculos assim nas pequenas coisas como nas grandes. Da fé vacilante resultam a incerteza e a hesitação de que se aproveitam os adversários que se tem de combater. Essa fé não procura os meios de vencer, porque não acredita que possa vencer. Contudo, a fé inabalável só é a que pode encarar de frente a razão, em todas as épocas da humanidade.

O Evangelho Segundo o Espiritísmo

Resumo

BARROSO, Nelson de Figueiredo. Instrumentacão Virtual Aplicada à Automacao de um Sistema Térmico para Experimentação Via Web. Trabalho de Conclusão de Curso. Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG *Campus* Divinópolis, 2014.

Este trabalho aborda os temas Instrumentação Virtual e Laboratórios Remotos aplicados ao ensino e pesquisa em Controle e Automação. O objetivo principal é automatizar um sistema de aquecimento de ar, por meio de recursos de instrumentação virtual e ferramentas para comunicação Web, possibilitando que usuários remotamente distribuídos o acessem para realizarem experimentos. Primeiramente foram implementados os circuitos necessários ao funcionamento da planta. Para instrumentação do sistema foram utilizados sensores de temperatura *LM35* e circuitos condicionadores de sinais para termopares. O acionamento dos atuadores foi obtido por meio de relés de estado sólido utilizando controle por ciclo integral. Em seguida foi desenvolvido um software aplicativo em LabVIEW® para comunicação com a planta. O aplicativo foi construído seguindo padrões de projeto que permitiram a obtenção de um programa modular, de fácil leitura e fácil manutenção. A comunicação entre a planta e o aplicativo foi realizada utilizando a placa de aquisição de dados modelo *NI PCI 6229* e os VIs *DAQ Assistant*. Por último, foram utilizados recursos do *database connectivity toolkit* para comunicação com banco de dados *MySQL* e também as ferramentas para comunicação através do protocolo *TCP/IP* para o comando dos atuadores. Para a comunicação via Web e a interface com o usuário remoto foram utilizados *Java* com apoio do servidor Apache. Ao final do projeto foi obtido um sistema onde o usuário remoto pode agendar experimentos para levantamento de características para obtenção de modelos e projeto de controladores e posteriormente, para teste dos controladores projetados. As entradas de comando são construídas através da determinação de perfis pelo usuário, podendo haver diversas combinações que permitem a realização tanto de testes mais simples como um degrau aplicado a uma determinada resistência e utilizando-se determinado sensor quanto a testes mais elaborados combinando-se diversos sensores e resistências em diferentes configurações.

Palavras-chave: Controle e Automação. Instrumentação Virtual. Web.

Abstract

This paper addresses the issues Virtual Instrumentation and Remote Laboratories applied to teaching and research in Control and Automation. The main objective is to automate an air heating system through virtual instrumentation features and Web communication tools, enabling remotely distributed users to access and perform experiments. First the circuits necessary for the plant operation have been implemented. For system instrumentation, $LM35$ temperature sensors and conditioners signals circuits for thermocouple were used. The drive actuators was obtained by solid state relays using full-cycle control. It was then developed LabVIEW® software application for communication with the plant. The application was built following design patterns that allow the production of a modular program, easy to read and easy to maintain. The communication between the plant and the application was performed using the data acquisition board *NI PCI 6229* model and DAQ Assistant VIs. Finally, database connectivity toolkit resources and tools for communication using TCP/IP protocol to control the actuators, were used. For Web Communication and remote user interface were used Java with Apache server support. At the end of the project was obtained a system where the remote user can schedule experiments to survey characteristics, develop models and design controllers and later to test the designed controllers. Control inputs are constructed by setting user profiles, and there may be several combinations that allow you to perform both simple tests as a step applied to a determined resistance and using certain temperature sensor as the more elaborated tests by combining several sensors and elements in different configurations.

Key-words: Control and Automation. Virtual Instrumentation. Web.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ANSI	*American National Standards Institute* (Instituto Nacional Americano de Padrões)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
ARC	*Automation Research Corporation* (Corporação de Pesquisa em Automação)
ASCII	*American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação)
ATM	*Asynchronous Transfer Mode* (Modo de Transferência Assincrona)
BD	Bando de Dados
CA	Corrente Alternada
CC	Corrente Contínua
CLP	Controlador Lógico Programável
CPE	Coordenação de Política Estudantil
CR	*Carriage Return* (Retorno do cursor para a primeira posição da linha onde se encontra)
CSS	*Cascading Style Sheets* (Folhas de Estilo em Cascata)
DAQ	*Data Acquisition* (Aquisição de Dados)
DB	*Data Base* (Banco de Dados)
DCL	*Data Control Language* (Linguagem de Controle de Dados)
DDL	*Data Definition Language* (Linguagem de Definição de Dados)
DML	*Data Manipulation Language* (Linguagem de Manipulação de Dados)
DMS	*Distributed Measurement Systems* (Sistemas de Medição Distribuídos)
DNS	*Domain Name System* (sistema de Nome de Domínio)
EEPROM	*Electrically-Erasable Programmable Read-Only Memory* (Memória Programável Elétricamente Apagável Somente de Leitura)

EMR	*Electro Mechanical Relay* (Relé Eletromecânico)
FCEIA	*Faculdad de Ciencias Exactas Ingenieria y Agrimensura* (Faculdade de Ciências Exatas, Engenharia e Agrimensura)
FK	*Foreing Key* (Chave Estrangeira)
FTP	*File Transfer Protocol* (Protocolo de Transferência de Arquivo)
HTML	*Hyper Text Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
I/O	*Input/Output* (Entrada/Saída)
IANA	*Internet Assigned Number Authority* (Autoridade para Atribuição de Números de Internet)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento Integrado)
IES	Instituições de Ensino Superior
IP	*Internet Protocol* (Protocolo de Internet)
JSP	*Java Server Pages*
LabVIEW®	*Laboratory Virtual Instrument Engineering Workbench*
LAR	Laboratórios de Acesso Remoto
LF	*Line Feed* (Nova Linha)
LMS	*Learning Management Systems* (Sistemas de Gestão da Aprendizagem)
MIMO	*Multiple Input Multiple Output* (Sistema com Múltiplas Entradas e Múltiplas Saídas)
MIT	*Massachusetts Institute of Technology* (Instituto de Tecnologia de Massachusetts)
NI	*National Instruments*
ODBC	*Open Database Connectivity* (Conectividade Aberta de Banco de Dados)
OLE	*Object Linking and Embedding*
OPC	*OLE for Process Control* Vinculação e Incorporação de Objetos (OLE para Controle de Processos)
PAC	*Programmable Automation Controller* (Controladores Programáveis para Automação)
PC	*Personal Computer* (Computador Pessoal)
PCI	*Peripheral Component Interconnect* (Interconexão de Componentes Periféricos)
PI	Proporcional Integrativo
PIC	*Programmable Interface Controller* Controlador de Interface Programável
PID	Proporcional Integral Derivativo

PK	*Primary Key* (Chave Primária)
PLC	*Programmable Logic Controller* (Controlador Lógico Programável)
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
REST	*Representational State Transfer* (Transferência de Estado Representacional)
RTOS	*Real Time Operation System* (Sistema Operacional em Tempo Real)
SCR	*Silicon Controlled Retifier* (Retificador Controlado de Silício)
SGBD	Sistema Gerenciador de Banco de Dados
SISO	*Single Input Single Output* (Sistema com Única Entrada e Única Saída)
SMPS	*Switched Mode Power Supplier* (Fonte de Alimentação em Modo Chaveada)
SMTP	*Simple Mail Transfer Protocol* (Protocolo de Transferência de Correio Simples)
SONET	*Synchronous Optical Network* (Rede Ótica Sincrona)
SPARC	*Standards Planning And Requirements Committee* (Comitê de Planejamento de Padrões e Requisitos)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
SRAM	*Static Random Access Memory* (Memória Estática de Acesso Aleatório)
SSR	*Solid State Relay* (Relé de Estado Sólido)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
TICs	Tecnologias da Informação e Comunicação
UDP	*User Datagram Protocol* (Protocolo de Datagramas de Usuário)
UNR	Universidade Nacional de Rosário
UPS	*Uninterruptible Power Suppliers* (Fonte de Alimentação Ininterrupta)
URI	*Uniform Resource Identifier* (Identificador Uniforme de Recursos)
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)
VI	*Virtual Instrument* (Instrumento Virtual)
VISIR	*Virtual Instrument Systems in Reality* (Sistemas de Instrumentação Virtual em Realidade)
WWW	Word Wide Web
XML	eXtensible Markup Language (Linguagem Extensível de Marcação)

NoSQL	Termo genérico para uma classe definida de banco de dados não-relacionais
SubVI	Sub programa desenvolvido em LabVIEW®
Weblab	Termo usado como sinônimo de laboratório remoto
WebLab	Nome dado ao laboratório remoto desenvolvido neste trabalho

Capítulo 1

Introdução

No presente capítulo será apresentada a relevância do tema escolhido assim como a motivação que resultou em sua proposição. Em seguida, será definido o problema em estudo, serão expostos os objetivos do trabalho proposto e, posteriormente a metodologia utilizada para concepção do projeto. Ao final do capítulo é dada uma descrição da organização do documento.

1.1 Relevância

A evolução dos computadores nos últimos 20 anos, acompanhado pelas mudanças nos processos produtivos, acelerou a busca por novas formas de produzir. Sendo assim, a presença dos computadores nos ambientes industriais tornou-se fundamental à medida que os processos passaram a ser mais complexos, exigindo soluções inteligentes, como por exemplo, na detecção de falhas e tomada de decisões (LOPES, 2007; NATIONAL INSTRUMENTS, 2009).

Dentre as vantagens dos sistemas baseados em PCs, destacam-se a flexibilidade e modularidade possibilitada pelos softwares aplicativos e a capacidade de integração com outras tecnologias, incluindo funções de conectividade com redes e Web. Tais características se tornaram essenciais na concepção de produtos mecatrônicos e de soluções em automação por que permitiram que esforços fossem voltados para a aplicação, diminuindo o tempo gasto no desenvolvimento de protótipos e aumentando a produtividade (OLIVEIRA, 2008).

Os sistemas medição e controle de instrumentos de forma remota, por meio de redes, conhecidos como Sistemas de Medição Distribuídos do inglês *Distributed Measurement Systems - DMS*, têm sido tópico de interesse para muitos pesquisadores em aplicações tanto na indústria quanto no campo educacional. Para aplicações industriais, esses sistemas são utilizados geralmente quando os instrumentos estão localizados em ambientes insalubres ou distantes do local de monitoramento, ou ainda quando configuram

vários nós em uma área extensa, como é o caso da internet. Já em aplicações educacionais, eles são utilizados para maximizar o treinamento experimental de estudantes, oferecendo a eles experiências práticas em sistemas reais através uma conexão virtual (GRIMALD; RAPUANO; LAOPOULOS, 2005).

O presente trabalho, abrange os dois cenários acima. Tendo como tema a Instrumentação Virtual e Laboratórios Remotos também conhecidos como WebLabs. O objetivo principal é automatizar o protótipo de um sistema de aquecimento de ar permitindo acesso remoto via Web para agendamento e realização de experimentos à distância caracterizando-se assim como um sistema distribuído para medição e controle.

No laboratório de Sinais e Sistemas do Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) *Campus* Divinópolis existe um sistema de aquecimento de ar cuja dinâmica é semelhante a de fornos industriais utilizados para reaquecimento de tarugos em processos de trefilamento ou mesmo para tratamento térmico de metais. Indústrias dessa natureza estão presentes na região do centro-oeste mineiro onde localiza-se esse unidade do CEFET-MG. Sistemas térmicos em geral, possuem constante de tempo alta, o que torna difícil a avaliação de seu desempenho durante o processo produtivo. Além disso, o ambiente industrial onde esses equipamentos estão presentes faz com que tarefas de aquisição de dados e controle em loco sejam dificultadas pelas altas temperaturas e condições de insalubridade. Mesmo não sendo do escopo deste trabalho, pelo seu nível inicial de aplicação, a possibilidade de empregar as idéias do projeto ao ambiente industrial está presente.

O intuito deste trabalho está voltado portanto, para o ensino e pesquisa em Controle e Automação. A realização de experimentos automatizados permitirá a utilização da planta sem a presença física do usuário, aumentando a taxa de utilização do equipamento disponível. Dessa forma, as limitações de tempo e espaço serão reduzidas. A planta poderá ser agendada, por exemplo, para realizar experimentos à noite ou nos finais de semana. Diferentes técnicas de controle, com diferentes variações de parâmetros, poderão ser implementadas, testadas e validadas, permitindo verificar e analisar os seus desempenhos, inclusive em relação ao consumo de energia. Dessa forma, este projeto também poderá ser utilizado como apoio didático nas disciplinas que compõem o eixo de Modelagem e Controle de Processos do curso de Engenharia Mecatrônica.

1.2 Motivação

A escolha do tema do presente trabalho de conclusão de curso surgiu a partir da participação deste autor em um projeto de trabalho do programa bolsa de complementação educacional, gerida pela Coordenação de Política Estudantil (CPE) do CEFET-

MG/Divinópolis no período de 2008 a 2010. O projeto em questão, tinha como principal objetivo, o apoio financeiro ao aluno, aliado à complementação técnico-científica do mesmo por meio de estudos coordenados aplicados à área de Instrumentação Virtual, apoiados na utilização do software LabVIEW® do inglês *Laboratory Virtual Instrument Engineering Workbench*. Naquela oportunidade, a proposta de trabalho foi ofertada e supervisionada pelo professor Valter Júnior de Souza Leite.

Diante das possibilidades oferecidas por esse tipo de instrumentação e de sua aplicabilidade nas engenharias em geral e em especial na Engenharia Mecatrônica, surgiu o interesse em se empregar os conhecimentos adquiridos naquela ocasião e principalmente aqueles adquiridos ao longo da graduação em um projeto que pudesse englobar grande parte da estrutura curricular do curso dando origem assim, ao trabalho proposto.

1.3 Definição do Problema

As Tecnologias da Informação e Comunicação (TICs), correspondem a todas as tecnologias que interferem e mediam os processos informacionais e comunicativos dos seres. Dentro do conjunto de recursos tecnológicos que definem o termo, estão inseridos os Laboratórios Remotos ou Weblabs, que utilizam técnicas modernas de engenharia para construção de arquiteturas que permitem o monitoramento e controle de equipamentos e dispositivos remotamente distribuídos, podendo ser acessados de qualquer lugar com conexão à internet.

O problema aqui abordado, consiste na automação de um sistema de aquecimento de ar, através de instrumentação virtual e ferramentas Web, para construção de um laboratório remoto, em que o cliente, como é chamado o utilizador do sistema, acesse a página do laboratório e agende determinado experimento. Os experimentos consistem basicamente, na aquisição de dados para identificação de modelos e projeto de controladores e posteriormente, teste dos controladores projetados. A solução proposta baseia-se no estudo e seleção de estruturas e softwares para programação de uma arquitetura que possibilite a comunicação, gerenciamento e coordenação do Weblab.

1.4 Objetivos do Trabalho

O presente trabalho de conclusão de curso tem como objetivo principal automatizar um sistema de aquecimento de ar, por meio de recursos de instrumentação virtual e ferramentas para comunicação Web, permitindo que usuários acessem a planta remotamente para agendar experimentos a fim de obter modelos do sistema sob diferentes configurações, projetar controladores e posteriormente testar os controladores projetados.

Os objetivos secundários, consistem em projetar e implementar os circuitos elétricos

necessários ao funcionamento da planta; promover interface entre o sistema de aquecimento de ar e computador; projetar e desenvolver instrumentos virtuais em LabVIEW® para leitura e escrita de dados e comunicação remota; configurar servidores Web para comunicação direta e indireta com cliente; criar banco de dados para comunicação indireta entre cliente e servidor e desenvolver página Web para interface com o usuário remoto.

1.5 Metodologia

Este trabalho partilha de três desenvolvimentos complementares: o primeiro refere-se à preparação física do processo a ser controlado e trata do projeto e implementação dos circuitos e dispositivos que compõem seus módulos. O segundo diz respeito ao software aplicativo a ser usado para controle local do processo e destaca os padrões de projeto utilizados, os estados do sistema e os principais recursos aplicados. O terceiro e último, descreve os passos seguidos para o desenvolvimento da arquitetura para agendamento e acionamento remoto do processo e dos experimentos a serem realizados. Por isso, adotou-se um conjunto de procedimentos para cada um desses desenvolvimentos. Cada um desses conjuntos de procedimentos é descrito, de forma geral, a seguir.

Preparação física do processo

O presente trabalho usou como base para seu desenvolvimento a planta de um sistema de aquecimento de ar já existente no laboratório de Sinais e Sistemas do CEFET-MG *Campus* Divinópolis. No início da proposição, tal sistema apresentava diversos problemas em sua parte elétrica/eletrônica. Dessa forma, o primeiro passo a ser realizado foi analisar as condições em que este se encontrava.

Foi verificado que as resistências da planta estavam queimadas; os circuitos de aquisição de dados apresentavam acúmulo de erros nas medições e os circuitos de potência apresentavam uma pequena tensão de offset, ou seja, já forneciam, mesmo que em um nível baixo, uma certa potência às cargas quando estas deveriam estar totalmente desligadas. Outro problema encontrado, estava no circuito de controle que apresentava mau contato desligando subitamente quando comutado para modos manual ou automático. Acredita-se que a maioria desses problemas decorreu do mal uso e do transporte da planta das antigas instalações do CEFET para o novo *Campus*.

Feita uma análise das condições físicas da planta, o próximo passo foi propor algumas modificações e melhorias. A primeira delas foi em relação ao circuito de acionamento, incluindo os dispositivos a serem utilizados para fornecer potência às cargas, o modo de controle de potência CA e o sistema de alimentação. A outra, foi a adição de transdutores de corrente para medição da energia consumida pelas resistências, com a finalidade de

avaliar o desempenho de controladores em relação a tal parâmetro. Também foram acrescentados ao projeto circuitos termopares para medição de temperatura na parte externa do sistemas em posições não fixas.

Devido ao mau funcionamento do circuito de controle e às novas funcionalidades necessárias ao projeto, decidiu-se por desenvolver outro circuito de controle, também baseado em microcontroladores. O objetivo principal foi descentralizar o processamento de dados do computador deixando-o responsável em sua maior parte, por receber sinais condicionados da planta e realizar a comunicação de dados com o usuário remoto.

Para acionamento das resistências foram utilizados relés de estado sólido (*Solid State Relays - SSRs*) e o sistema de alimentação CA foi modificado passando a ser trifásico ao invés de monofásico. Os SSRs foram escolhidos devido à sua ampla aplicação industrial, principalmente em sistemas térmicos. O comando desses dispositivos foi feito por meio de controle por ciclo integral. Esse tipo de controle é ideal para sistemas com dinâmica lenta. Para medição de corrente nas resistências foram utilizados transdutores de corrente por efeito *Hall*. Como o acionamento do sistema é feito remotamente, foi criado também um circuito de acionamento geral da planta. Para alimentação dos sensores, transdutores e microcontroladores foram construídas fontes de alimentação CC.

Após o projeto e implementação dos circuitos necessários ao funcionamento da planta, o passo seguinte foi interfacear a planta com o computador. Essa interface foi estabelecida através da placa de aquisição de dados *NI PCI* 6229 da National Instruments.

Desenvolvimento do software aplicativo

Com a parte física do sistema funcionando e a interface entre computador e planta estabelecida, iniciou-se a segunda fase da metodologia. Percebeu-se nesse momento, a necessidade de se determinar um método consistente para projeto e desenvolvimento de *softwares*. O ponto de partida para esta ação, foi a idéia inicial de construir um programa modular, permitindo portanto, modificar o código facilmente, possibilitando posterior evolução do projeto através da fácil implementação de novas funcionalidades e melhorias nas funções existentes.

De acordo com tais preceitos, o primeiro passo foi realizar um estudo sobre práticas de desenvolvimento de *softwares*. Através desse estudo, ficou evidente a importância do planejamento, principalmente em relação à definição dos requisitos de projeto, recursos e prazos. Foram abordados as vantagens e desvantagens de cada método. Dessa forma, para o presente trabalho, decidiu-se por seguir uma metodologia híbrida, reunindo os principais aspectos de cada um.

A técnica de projeto adotada extendeu a intenção de promover um *software* modular, acrescentando os conceitos de fácil leitura e fácil manutenção. O próximo passo foi então,

definir as especificações de projeto. Para tanto, foram listados todas as funções desejáveis, ou seja, aquelas que o usuário remoto poderia realizar na planta. Essa tarefa foi feita com o cuidado de não extrapolar a definição inicial de construir uma base, ou seja, uma estrutura fundamental para desenvolvimentos posteriores.

Após definir as especificações de projeto, o passo seguinte foi pesquisar sobre as estruturas de programação do LabVIEW® para manter o programa fiel aos propósitos expostos anteriormente e analisar os recursos disponíveis para atendê-las. Recorreu-se assim, aos Padrões de Projeto do LabVIEW®, construindo o programa sobre as estruturas *produtor-consumidor*, *máquina de estados* e *loops paralelos*.

Os estados do sistema foram definidos de acordo com as funções desejáveis, ou seja, *verificação de registros*, responsável por inicializar os dados do usuário e do experimento; *atuação direta*, que permite que o usuário controle os atuadores da planta e leia os valores dos sensores de temperatura diretamente; *perfil de testes*, que recebe parâmetros para realização de testes a partir de perfis pré-programados pelo usuário; *teste de controladores*, que além do perfil criado pelo usuário, que são as referências do sistema de controle, também recebe os parâmetros do controlador a ser aplicado ao sistema e, por último, o estado responsável por finalizar o experimento.

Das ferramentas do LabVIEW® para desenvolvimento das funções, foram utilizadas essencialmente: *NI DAQ Assistant*, responsáveis por gerenciar os recursos da placa de aquisição de dados (leitura dos canais de entradas analógicas referentes ao sensores de temperatura, escrita nos canais de saídas analógicas para comandos dos atuadores e escrita em linha de saída digital para acionamento geral da planta); *Database Connectivity toolkit* para comunicação com o banco de dados e *Data Communication TCP toolkit* para comunicação direta com a planta. Esses dois últimos conjuntos de ferramentas foram definidos após a determinação dos recursos a serem apresentados seção seguinte.

Desenvolvimento da arquitetura do sistema

A interface de usuário remoto é onde culmina todo o esforço sintetizado na metodologia apresentada até agora. Essa fase do projeto foi a que exigiu maior dedicação principalmente por envolver assuntos fora da grade curricular do curso de Engenharia Mecatrônica, embora tais assuntos estejam ganhando espaço no curso a partir da oferta de disciplinas optativas.

Os conceitos relacionados à Tecnologia da Informação são muitas vezes abstratos e isso ocorre ao passo que novos conceitos e arquiteturas são implementadas. Além disso, ao se aventurar nessa área, o aluno se depara com um processo lento de aprendizagem em que um determinado assunto pode levar a tantos outros, que se torna difícil filtrar o que realmente importa para se desenvolver determinada aplicação. Diante desses fatos,

decidiu-se por focar na aplicação proposta e através dela, buscar os meios necessários para desenvolver o projeto.

Tendo-se em mente os requisitos de projeto, foi realizada uma pesquisa em trabalhos com o tema laboratórios remotos, com o objetivo de verificar as arquiteturas utilizadas e seus componentes, incluindo os *softwares* para desenvolvimento e linguagens de programação. Uma vez decidido sobre o uso do LabVIEW® como principal plataforma de desenvolvimento, foram analisados os recursos de conectividade do software que poderiam ser usados.

Comparando-se os recursos do LabVIEW® com as arquiteturas normalmente utilizadas em projetos semelhantes, optou-se por construir um sistema baseado no padrão Cliente-Servidor. A definição desse padrão levou em conta dois aspectos principais: o agendamento dos experimentos juntamente com o armazenamento das informações necessárias à interpretação dos resultados pelo cliente e a comunicação "direta" entre o cliente e o sistema de aquecimento de ar. Para dar apoio ao desenvolvimento da interface de usuários foi imprescindível a realização de uma revisão bibliográfica abordando conceitos de redes para Web e banco de dados.

Para que o usuário pudesse controlar a planta remotamente foi criada uma interface Web. Nessa interface, o cliente utiliza um navegador ou *browser* padrão para contatar o servidor que atua como intermediário entre o cliente e o banco de dados e entre o cliente e o LabVIEW®, nesse último caso, apenas durante a função de atuação direta. O cliente se comunica com o servidor Web por uma questão de segurança, pois isso impede que ele tenha acesso ou consiga informações acerca do LabVIEW® que podem ser utilizadas por exemplo, para realizar um ataque. O servidor utilizado foi o *Apache*.

Os servidores projetados especificamente para Web não apenas são mais versáteis em suas configurações relacionadas à redes e permissões de acesso, como são capazes de utilizar varias tecnologias padrão da indústria para proteger o conteúdo e as comunicações. Além disso, os servidores referência no mundo são gratuitos e de código aberto. Outra vantagem, é a possibilidade de utilizar tecnologias variadas na construção das páginas Web. Esta capacidade foi amplamente explorada por este projeto através do uso de várias tecnologias que não são suportadas pelos instrumentos virtuais do inglês *Virtual Instruments - VIs*, do LabVIEW® que servem para prover conteúdo Web, como o uso da linguagem *Java*, e bibliotecas como *Twitter Bootstrap* e *jQuery*.

Para criar a interface utilizada pelo cliente, foram empregadas tecnologias de uso comum por profissionais da área, como o *framework Twitter Bootstrap*, que permite a criação de interfaces otimizadas para aparelhos móveis, como celulares e tablets, e provê componentes básicos que não são fornecidos pela definição HTML/CSS padrão, e *jQuery*, que simplifica o uso do *Javascript* através de APIs mais fáceis de usar, além de ser uma

biblioteca que implementa varias funcionalidades não presentes na definição padrão do *Javascript*. No servidor foi utilizado a linguagem Java, através das tecnologias *Java Server Pages* (JSP) e *Servlet*. A primeira reúne HTML e Java em um mesmo arquivo, permitindo a fácil manipulação do HTML pelo Java. A segunda consiste em classes que respondem à requisições Web. Para escrever o código foi utilizada a IDE *NetBeans*.

Com o cliente se comunicando apenas com o servidor Web, ficou a cargo deste se comunicar com o LabVIEW®. Esta comunicação foi feita de forma indireta, através de um banco de dados. A necessidade de utilização de um banco de dados foi estabelecida levando-se em consideração as informações referentes aos registros de usuários e a necessidade de especificar o que cada usuário poderá realizar ao acessar o sistema. Tal banco foi criado utilizando-se a base de dados *MySQL* e suas tabelas foram definidas de acordo com cada função disponível para o usuário remoto. Tanto o servidor quanto o LabVIEW® se comunicam com a mesma instância do *MySQL*, assim, as leituras e escritas de um são visíveis para o outro. A comunicação entre o banco e o LabVIEW®, foi feita através do conjunto de ferramentas disponíveis no *Database Connectivity toolkit* do LabVIEW®.

Na função de atuação direta como citado anteriormente, o usuário comunica-se com a planta no sentido cliente-planta sem intermédio do banco de dados. Já no sentido contrário, os dados obtidos como potência aplicada aos atuadores e temperatura, são armazenados no banco de dados para posterior envio ao cliente. A interface entre o servidor e o LabVIEW® na função de atuação direta, foi estabelecida através das ferramentas do *Data Communication TCP toolkit* do LabVIEW®.

O envio dos resultados dos experimentos para o *e-mail* do usuário remoto foi feito a partir da criação de um *e-mail* e configuração deste como servidor SMTP (*Simple Mail Transfer Protocol*).

1.6 Organização do Documento

No presente capítulo, foram apresentadas a relevância do trabalho, a motivação para desenvolvê-lo, a definição do problema em questão, os objetivos principal e secundários e finalmente a metodologia utilizada para alcançar tais objetivos. Os parágrafos a seguir, resumem o que será encontrado nos próximos capítulos.

No capítulo 2 são apresentados os fundamentos que sustentaram o desenvolvimento do projeto. Inicialmente, as seções 2.1 e 2.2, são abordados os temas instrumentação virtual e laboratórios remotos, dando uma visão geral sobre esses dois assuntos e apresentando os seus principais conceitos. Em seguida a seção 2.3 traz alguns conceitos sobre práticas de desenvolvimento de *software*, que são utilizadas para estruturar o projeto e sugerem metodologias para facilitar o planejamento de suas etapas e principais pontos a serem

atendidos. Posteriormente, são abordadas na seção 2.4 as definições básicas para o entendimento da Web, destacando o seu funcionamento e os protocolos de rede que a sustenta. Por último, a seção 2.5 trata do assunto banco de dados, abordando, definições, padrões e tipos de conexão.

O capítulo 3, descreve o desenvolvimento do WebLab. Primeiramente, na seção 3.1 serão descritos os requisitos a serem atendidos pelo projeto. Esses requisitos definem o que se deseja obter ao final do trabalho, na visão de quem irá usar o WebLab. Em seguida, a seção 3.2 define o modelo obtido para o sistema, facilitando o entendimento sobre os seus módulos e suas relações. Finalmente, as seções de 3.3 a 3.6, descrevem detalhadamente cada módulo da estrutura geral criada para o WebLab apresentando os recursos e ferramentas utilizadas.

No capítulo 4, são expostos os resultados experimentais realizados para validar o projeto. Nesse capítulo, são apresentados fluxogramas que ajudam a entender, passo a passo, a dinâmica de todas as funções disponíveis para o usuário remoto. Cada função definida nos requisitos de projeto será testada a partir do agendamento dos experimentos e da análise dos resultados.

No capítulo **??** são feitas as considerações finais, apresentando os pontos da proposta de trabalho que foram ou não atendidos. Em seguida, são apresentados algumas propostas de modificações e continuidade do projeto.

Capítulo 2

Fundamentos

Neste capítulo será feita uma breve revisão da literatura abordando os conceitos e o cenário atual da instrumentação virtual e dos laboratórios remotos. Esses dois assuntos são apresentados sob as perspectivas industrial e educacional dando ênfase sobre como essas ferramentas estão sendo utilizadas para disseminar o conhecimento e promover aprendizagem. Em seguida, serão apresentas as principais bases teóricas que fundamentaram o desenvolvimento deste trabalho.

A seção 2.3 introduz conceitos de práticas de desenvolvimento de *softwares*. Tais conceitos, sugerem metodologias para estruturação de projetos, facilitando o planejamento de suas etapas e principais pontos a serem atendidos. A seção 2.4, refere-se aos principais conceitos relacionados às redes para Web, destacando o funcionamento de clientes e servidores e os protocolos de rede que sustentam a Web. A seção 2.5 trata de banco de dados, abordando definições, padrões e tipos de conexões.

2.1 Instrumentação Virtual

A Instrumentação Virtual consiste na utilização de computadores equipados com *software* aplicativo e *hardware* de propósito geral para medição e controle. Um instrumento virtual é constituído por um computador industrial ou estação de trabalho (Workstation) equipado com *software* aplicativo, *hardwares*, como placas *plug-in* e *drivers*, que juntos desempenham as funções dos instrumentos tradicionais. Instrumentos virtuais representam uma revolução nos sistemas de instrumentação tradicionais focados em *hardware* para sistemas centrados em *software* que exploram o poder computacional, a produtividade, a visualização gráfica e as funcionalidades de conectividade dos populares computadores de mesa (*PCs desktops*) e estações de trabalho (NATIONAL INSTRUMENTS, 2009).

A instrumentação virtual esteve por muito tempo dedicada apenas às áreas de testes e medições nas fases de projeto e desenvolvimento. No entanto, a busca das empresas por soluções que atendam às necessidades do mercado, que exige com crescente intensidade, o

aumento da produtividade com menor custo, impulsionou o desenvolvimento e evolução dos sistemas produtivos que com o passar do tempo deixam de ser dedicados para se tornarem mais flexíveis e inteligentes. Assim, a instrumentação virtual, vem ganhando espaço também na área de controle industrial.

No campo dos testes, as plataformas devem incluir ferramentas para rápido desenvolvimento de sistemas adaptáveis para serem utilizadas durante o fluxo de produção. Disponibilizar produtos de forma rápida e produzi-los de maneira eficiente, requer um sistema de testes de alta velocidade e com medições precisas e sincronizadas. Em relação a projetos, a instrumentação virtual, devido à flexibilidade proporcionada pelo *software*, e no cenário atual, a modularidade também de *hardware*, permite a integração com diversos sistemas e com diversos dispositivos de diferentes fabricantes disponíveis no mercado, melhorando a interface entre as fases de projeto e teste/validação, acelerando o ciclo de desenvolvimento (OLIVEIRA, 2008).

Sistemas de controle baseados em PCs vem ganhando nas últimas décadas, um espaço cada vez maior no meio industrial. Sua utilização aumenta à medida que se necessita de atender a aplicações mais complexas que exigem elevadas taxas de repetição, algoritmos de controle avançados, melhores características analógica e melhor integração com a rede corporativa. Tais aplicações segundo estimativas da *Automation Research Corporation* (ARC) representam 20% das soluções industriais, sendo que este número tende a aumentar ao passo que os sistemas produtivos se tornam mais inteligentes incluindo tomadas de decisões (NATIONAL INSTRUMENTS, 2012).

Para atender a indústria, os sistemas baseados em PCs tiveram que evoluir para garantir confiabilidade e determinismo. Tais características eram até então atendidas unicamente por sistemas baseados em CLPs, que por sua vez, dificilmente atendiam às funcionalidades mais avançadas permitidas pelos PCs tais como poder computacional, modularidade e conectividade. Neste contexto, engenheiros que precisavam de soluções mais complexas se uniram a fornecedores da área de controle para criarem um novo conceito em sistemas de controle. Surgiram assim os Controladores Programáveis para Automação, (PAC - Programmable Automation Controllers) que combinam a flexibilidade e potencialidades dos PCs com a robustez e confiabilidade dos CLPs .

Dentre as principais vantagens das novas controladoras podem ser citadas: funcionalidade múlti-domínio, com pelo menos 2 funçóes de lógica, controle de movimento, controle PID, drives e processos em uma única plataforma; plataforma de desenvolvimento única e multi disciplinar incorporando configuração simples de tags e uma única base de dados para acesso a todos os parâmetros e funções; ferramentas de *software* que possibilitam o projeto processando seu fluxo através de várias máquinas ou unidades de processo, junto com o padrão IEC61131-3 que trata dos padrões para o usuário e gerenciamento de da-

dos; arquiteturas abertas e modulares que espelham as aplicações industriais de layouts de máquinas em fábricas a unidades de operação em plantas de processos e finalmente utilização de padrões para interfaces de rede, linguagens, etc., tais como TCP/IP, OPC e XML, e SQL.

2.2 Laboratórios Remotos

Os computadores são ferramentas altamente flexíveis e adaptáveis a novas tecnologias. Tais características se devem principalmente à flexibilidade e modularidade possibilitada pelos *softwares*. Os *softwares* para instrumentaçao virtual utilizam o mesmo ambiente de desenvolvimento tanto para aplicações em computadores pessoais quanto em aplicações industriais utilizando sistemas operacionais em tempo real do inglês *Real Time Operation Systems - RTOS*. Considerando ainda a conectividade com redes e Web, permitida pelos PCs, é possível configurar, por exemplo, um servidor Web integrado a uma plataforma de controle industrial permitindo assim acesso sob os dados do processo e atuação em seus parâmetros de forma remota, sendo o controle determinístico de tal processo executado localmente. Esse tipo de aplicação é conhecido como instrumentação virtual distribuída (NATIONAL INSTRUMENTS, 2013).

O conceito de instrumentação virtual distribuída permitiu a criação dos Laboratórios de Acesso Remoto (LAR), comumente chamados de Weblabs. Nos últimos anos, avanços nos computadores e nas tecnologias de rede permitiram uma grande evolução desses sistemas que abordam aplicações que vão desde ensino/aprendizagem até aplicações industriais de controle, monitoramento, diagnóstico de falhas, dentre outros (NGOLO, 2009).

Um laboratório remoto consiste em uma série de equipamentos instrumentados com sensores e atuadores que podem ser remotamente monitorados e controlados através da internet. Na educação, eles possibilitam que estudantes possam conduzir experimentos e interagir com equipamentos físicos reais e realizarem experimentos autênticos, não simulados, de qualquer lugar com acesso à internet a qualquer hora. De acordo com KOSTULSKI; MURRAY (2010) os Weblabs oferecem vantagens em relação à flexibilidade, utilização, economia de espaço e questões relacionadas à segurança.

A utilização de laboratórios remotos no ensino e aprendizagem segue um caminho natural de adequação e adaptação dos meios de se promover a educação às novas tecnologias. Em muitas Instituições de Ensino Superior (IES) as aulas expositivas nas áreas das ciências, tecnologia e engenharias são frequentemente complementadas por laboratórios de experimentação remota onde os estudantes podem observar fenômenos dinâmicos que são muitas vezes difíceis de explicar através de material escrito. As plantas de experimentação remota aumentam a motivação dos alunos e também desenvolvem uma abordagem

realista para resolver problemas. Diferentemente dos laboratórios virtuais, onde todos os processos são simulados, o laboratório de experimentação remota possibilita a interação com processos reais permitindo ao utilizador uma análise dos problemas práticos do mundo real (SILVA *et al.*, 2013).

Em vista do alto custo dos laboratórios de ciências e engenharia devido às despesas com instalações, equipamentos técnicos caros e gastos com manutenção, os Weblabs tem sido uma solução para muitas universidades ao redor do mundo que podem compartilhar seus laboratórios criando uma rede que potencializa o uso destes, levando conhecimento a um número maior de pessoas. Geralmente, esses projetos são desenvolvidos em consórcios envolvendo várias universidades de diversos países e outras organizações inclusive com apoio governamental.

Como exemplo destes consórcios e dos esforços para o desenvolvimento e implantação dos Weblabs podem ser citados o *MIT iLab team* do Instituto de Tecnologia de Massachusetts (HAWARD *et al.*, 2004) nos Estados Unidos pioneiro nesse campo; VISIR do inglês *Virtual Instrument Systems Reality* da universidade tecnológica de *Blenkinge* (*Blenkinge Institute of Technology*) na Suécia (GUSTAVISSON *et al.*, 2007), *Labshare* da Universidade de Tecnologia em Sidney na Austrália (LOWE *et al.*, 2009); WebLab - Deusto da Universidade de Deusto em Bilbal na Espanha (ZUBIA *et al.*, 2009) ; Laboratório Remoto FCEIA da Faculdade de Ciências Exatas, Engenharia e Agrimensura da Universidade Nacional de Rosário (UNR) na Argentina (LERRO *et al.*, 2008); dentre outros. Todos esses exemplos configuram iniciativas dedicadas à criação de plataformas de ensino a distância. São os LMSs do inglês *Learning Manegement Systems*, ou seja, Sistemas de Gestão da Aprendizagem que visam concentrar experimentos desenvolvidos por diversos colaboradores em um único lugar disponibilizando um número maior de experimentos para um número maior de usuários.

2.3 Práticas de Desenvolvimento de *Software*

O desenvolvimento de um projeto é um processo que exige planejamento detalhado para garantir que todos os processos necessários estejam devidamente mapeados e documentados. Por meio de técnicas de desenvolvimento de projetos, é possível determinar o que está e o que não está incluso no projeto, agrupando características e funcionalidades para que todo o sistema funcione de maneira correta e eficiente, economizando tempo e evitando esforços desnecessários. Os fundamentos abaixo sobre Práticas de Desenvolvimento de *Software* foram baseados na bibliografia (NATIONAL INSTRUMENTS, 2008).

No projeto de *softwares*, programadores adotam métodos de desenvolvimento para resolver problemas utilizando programas. Esses métodos, ajudam o programador a construir

códigos que possuem grande potencial para solucionar satisfatoriamente um problema proposto se comparado com um código escrito sem planejamento. A Figura 2.1 apresenta um esquema das fases do processo de desenvolvimento de *softwares* denominada de Mapa de Desenvolvimento de *Softwares*.

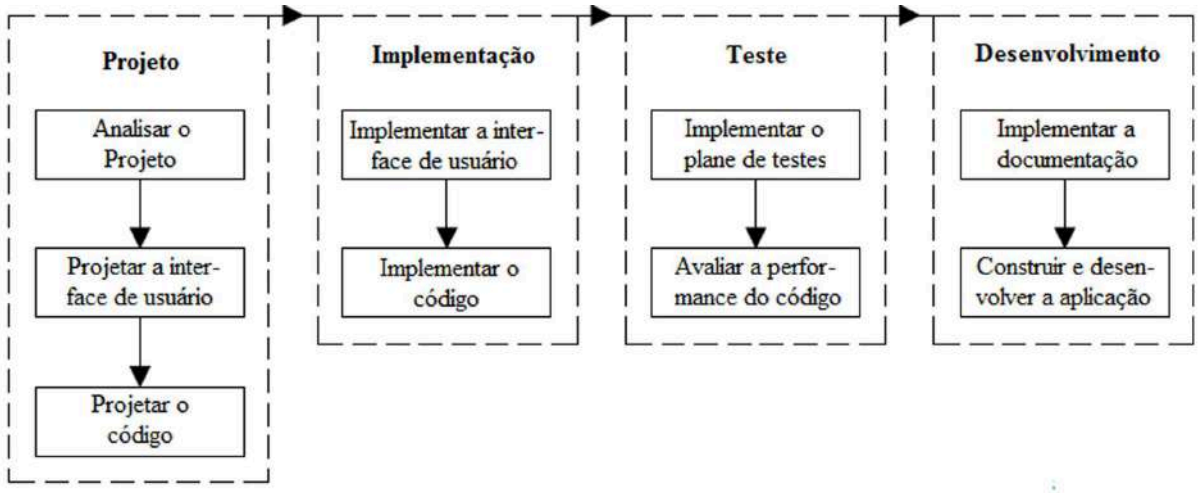

Figura 2.1: Mapa de desenvolvimento de *software*.
Fonte: (NATIONAL INSTRUMENTS, 2008).

2.3.1 *Softwares* modulares, de fácil leitura e fácil manutenção

Os métodos de desenvolvimento de *software* possibilitam que os códigos sejam mais legíveis, escaláveis e modificáveis. A importância desses princípios se tornam mais evidentes à medida que as aplicações se tornam mais complexas.

Os códigos legíveis expandem a funcionalidade de uma aplicação sem que seja necessário reprojetá-la completamente. Esse princípio evita que o código seja desestruturado, difícil de ler e difícil de entender.

Os códigos escaláveis possibilitam inspecionar visualmente o projeto de uma aplicação e rapidamente entender seu propósito e funcionalidade. No desenvolvimento de códigos escaláveis ou modulares, é necessário pensar sobre as partes da aplicação antes de pensar no processo como um todo. Ao se projetar uma aplicação deve-se considerar o propósito da aplicação e como gerenciar modificações quando a escala da aplicação vai além das especificações originais.

Já os códigos modificáveis são aqueles que permitem modificar facilmente o código escrito pelo desenvolvedor original ou por qualquer outro desenvolvedor sem afetar o propósito inicial. É necessário manter em mente que outros programadores poderão precisar usar e modificar o código no futuro. Dessa forma, códigos de fácil manutenção permitem que novos recursos sejam adicionados sem que a aplicação seja completamente reescrita.

2.3.2 *Software lifecycles*

Desenvolver *softwares* complexos é tarefa difícil. Para lidar com essa complexidade, muitos desenvolvedores aderem a um conjunto de princípios de desenvolvimento. Tais princípios definem o campo da engenharia de *software*. A principal referência deste campo é o modelo de desenvolvimento de *software lifecycle*. O modelo *lifecycle* descreve os passos a serem seguidos para desenvolvimento de um projeto de *software* desde a sua concepção, passando por manutenções até os estágios subsequentes de atualização.

Existem diversos modelos *lifecycles* e cada um deles possui suas vantagens e desvantagens em termos de prazo, qualidade e gestão de riscos. Além dos modelos existentes, é possível customizar determinados modelos para atenderem requisitos específicos de um projeto. No entanto, é recomendável que todos os passos sejam atendidos. Devem ser considerados aspectos sobre como se decide quais requisitos e especificações do projeto devem ser atendidas e como lidar com mudanças. Deve ser considerado também quando os requisitos serão atendidos e o que fazer se o prazo final não for cumprido.

O modelo *lifecycle* é a base para o desenvolvimento do *software* e permite que boas decisões resultem no aumento da qualidade e redução do tempo de desenvolvimento.

2.3.3 Modelo de código e correção

O modelo de código e correção é provavelmente a metodologia mais utilizada na engenharia de *software*. Ele inicia com nenhum ou quase nenhum planejamento. O desenvolvimento começa imediatamente e os problemas são resolvidos assim que ocorrem até que o projeto esteja completo. Portanto, é um método de tentativa e erro onde o desenvolvedor se depara com um cronograma apertado já que inicia o código imediatamente e vê resultados também imediatos.

A grande desvantagem do método é que se o desenvolvedor encontrar mais tarde maiores problemas na arquitetura do projeto, assim, ele terá que reescrever grande parte da aplicação. Modelos de desenvolvimento alternativos, podem ajudar a prevenir esses problemas em estágios anteriores em que mudanças podem ser feitas com mais facilidade e com menos custo. Este modelo é mais indicado para projetos pequenos e que não serão utilizados como base para desenvolvimentos futuros.

2.3.4 Modelo em cascata

O modelo em cascata é um método clássico utilizado pela engenharia de *software*. Este modelo é um dos mais antigos e é largamente utilizado por exemplo em projetos governamentais e por grandes empresas. Tal modelo enfatiza o planejamento em etapas anteriores ao desenvolvimento e permite detectar falhas de projeto antes que elas aconteçam. Como

é feito intensiva documentação e planejamento, o modelo se enquadra bem em projetos que exigem maior controle de qualidade.

O modelo em cascata puro consiste em várias etapas que não podem ser sobrepostas. Como pode ser observado na Figura 2.2, a etapa inicial consiste no estabelecimento dos requisitos do sistema, seguida pela definição dos requisitos de *software*. Posteriormente estão os estágios de projeto de arquitetura, detalhamento do projeto, código, teste e manutenção.

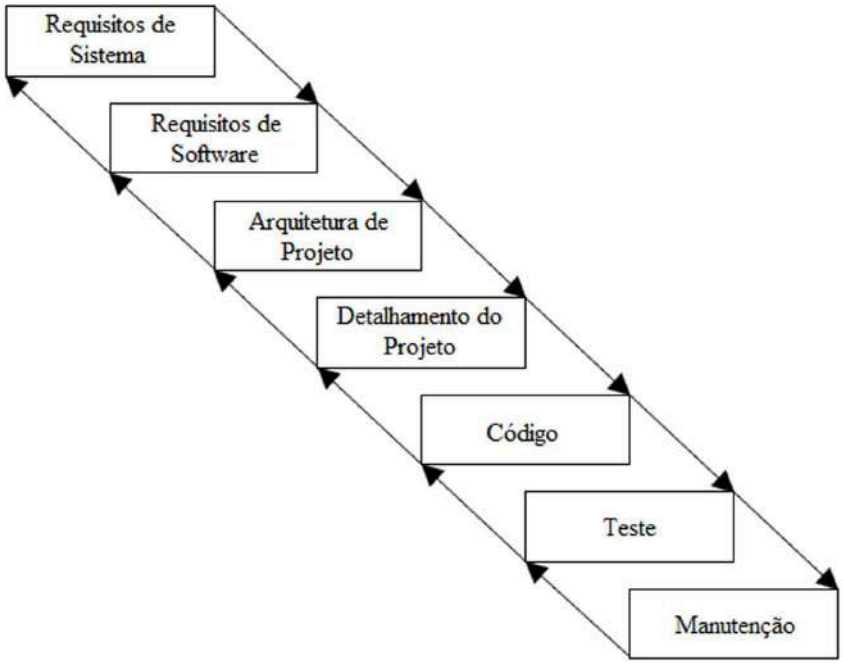

Figura 2.2: Modelo em cascata puro.
Fonte: (NATIONAL INSTRUMENTS, 2008).

Os passos que descrevem o modelo em cascata são descritos a seguir:

1. **Requisitos do sistema:** estabelecer os componentes para configurar o sistema, incluindo requisitos de *hardware*, ferramentas de *software* dentre outros componentes. Devem ser incluídas também decisões em *hardware* tais como as configurações na placa (número de canais, velocidade de aquisição, etc.) e decisões externas ao *software* como banco de dados ou bibliotecas.

2. **Requisitos de** ***software*****:** estabelecer as expectativas para a funcionalidade do *software* e identificar quais os requisitos dos sistema são afetados pelo *software*. A análise deve incluir e determinar interações com outras aplicações e banco de dados, requisitos de performance, interface de usuários, dentre outros.

3. **Arquitetura de projeto:** determinar o *framework* do *software* de um sistema para atender as necessidades especificadas. O projeto define a maioria dos componentes

e suas interações, mas o projeto não define a estrutura de cada componente. É definido também a interface interna e ferramentas para serem usadas no projeto.

4. **Detalhamento do projeto:** examinar os componentes de *software* definidos no estágio de arquitetura de projeto e produzir especificações sobre como cada componente deve ser implementado.

5. **Código:** programar as especificações de projeto detalhado.

6. **Teste:** determinar se o código atenderá às especificações de projeto e identificar possíveis erros presentes no código.

7. **Manutenção:** identificar problemas e implementar melhorias no código criado.

Cada mudança que ocorre na fase de manutenção deve levar em conta todas as partes do sistema. Considerando então o modelo em cascata, ao corrigir um problema ou promover uma melhoria, todas as fases do modelo anteriores à manutenção devem ser reaplicadas.

Cada estágio deve ser devidamente documentado. Devem ser criados documentos que explicam os objetivos e descrevem os requisitos de cada fase. No final de cada estágio, uma revisão deve ser feita para determinar se o projeto pode prosseguir para a próxima fase. Protótipos também podem ser incorporados a cada fase do projeto.

O modelo em cascata não pode ser aplicado a todas as situações. Por exemplo, no modelo em cascata puro, os requisitos devem ser listados antes do projeto e o projeto completo deve ser definido antes do desenvolvimento do código. Além disso, as fases não podem ser sobrepostas. No entanto, no mundo real algumas questões podem surgir durante a fase de projeto ou codificação e essas questões podem apontar erros ou negligências referentes aos requisitos de projeto.

O modelo em cascata não proíbe o retorno às fases anteriores como por exemplo da fase de projeto para a fase de requisitos. No entanto, isto envolve custos de retrabalho. Cada fase completa requer revisão formal e extensiva documentação. Assim, falhas que ocorrerem em etapas anteriores se tornarão caras para corrigir mais tarde.

Outra restrição no modelo em cascata é que como a fase de desenvolvimento propriamente dita demora a acontecer, os resultados demoram a aparecer. Isso faz com que o processo seja cansativo, podendo desmotivar gestores e clientes. Muitos desenvolvedores também acham a quantidade de documentação excessiva e inflexível. Apesar disso, pode-se considerar que as fases do modelo enfatizam importantes estágios no desenvolvimento de projetos. Dessa forma, mesmo que o modelo não seja aplicado, é importante que suas fazes sejam consideradas, sendo uma boa base para estruturar o planejamento e desenvolvimento do projeto.

2.3.5 Modelo em cascata modificado

Muitos engenheiros recomendam a versão modificada do modelo em cascata. Essas modificações têm o intuito de permitir que alguns estágios sejam sobrepostos. O resultado é a redução do volume de documentação e do custo de retorno para revisão dos estágios anteriores. Outra modificação comum é incorporar protótipos nas fases de requisitos.

A sobreposição de etapas, integrando, por exemplo, requisitos e projeto, permite que sejam dados *feedbacks* de uma fase para outra. Porém, um inconveniente é que sobrepondo estágios, pode ser difícil identificar o início e o final de cada um. Consequentemente, é mais difícil progredir com as demais fases. A não distinção dos estágios podem causar a negligência de decisões importantes que serão tomadas mais tarde no processo quando eles se tornam mais caros para serem corrigidos.

2.3.6 Prototipagem

Um dos principais problemas no modelo em cascata é que os requisitos frequentemente não são completamente entendidos nos estágios iniciais de desenvolvimento. Quando se chega aos estágios de projeto ou codificação, começa-se a enxergar como tudo funciona em conjunto, a descobre-se que é necessário fazer ajustes nos requisitos.

A prototipagem é uma ferramenta efetiva para demonstrar como um projeto atende uma série de requisitos. Um protótipo pode ser construído e os requisitos podem ser ajustados e revisados quantas vezes for necessário até que se obtenha uma figura clara de todos os objetivos. Além disso, para tornar os requisitos claros, o protótipo naturalmente define várias áreas do projeto simultaneamente.

O modelo em cascata puro permite que a prototipagem seja feita nos estágios finais da arquitetura de projeto e nos estágios subsequentes, porém não é permitido que seja construídos nos estágios iniciais. No entanto, a prototipagem tem suas desvantagens. Pelo fato de parecer que se tem um sistema funcionando, clientes podem esperar um sistema completo antes da hora.

Um cuidado deve ser tomado na prototipagem para evitar que o protótipo disfarce um código e pare o desenvolvimento. Antes do início da prototipagem, devem ser reunidos requisitos claros e criado um planejamento de projeto. O tempo gasto para prototipagem também deve ser limitado antes do início para evitar exageros na prototipagem. À medida que mudanças são incorporadas, os requisitos e o projeto devem ser atualizados. A prototipagem deve ser considerada em uma das fases do modelo em cascata como na fase de requisitos ou de projeto.

2.3.7 Modelo em espiral

O modelo em espiral é uma alternativa popular para o modelo em cascata. Ele enfatiza o gerenciamento de riscos, dessa forma, é possível identificar problemas nas fases iniciais do ciclo de desenvolvimento. No modelo em cascata, é necessário completar a fase de projeto para que a fase de codificação seja iniciada. Com o modelo em espiral, o projeto pode ser dividido em um conjunto de riscos com os quais é preciso lidar. É feita uma série de iterações em que se deve fazer análises sobre os riscos mais importantes, avaliando as opções para resolvê-los, lidar com eles, acessar os resultados e planejar as próximas iterações. A Figura 2.3 ilustra o modelo em espiral.

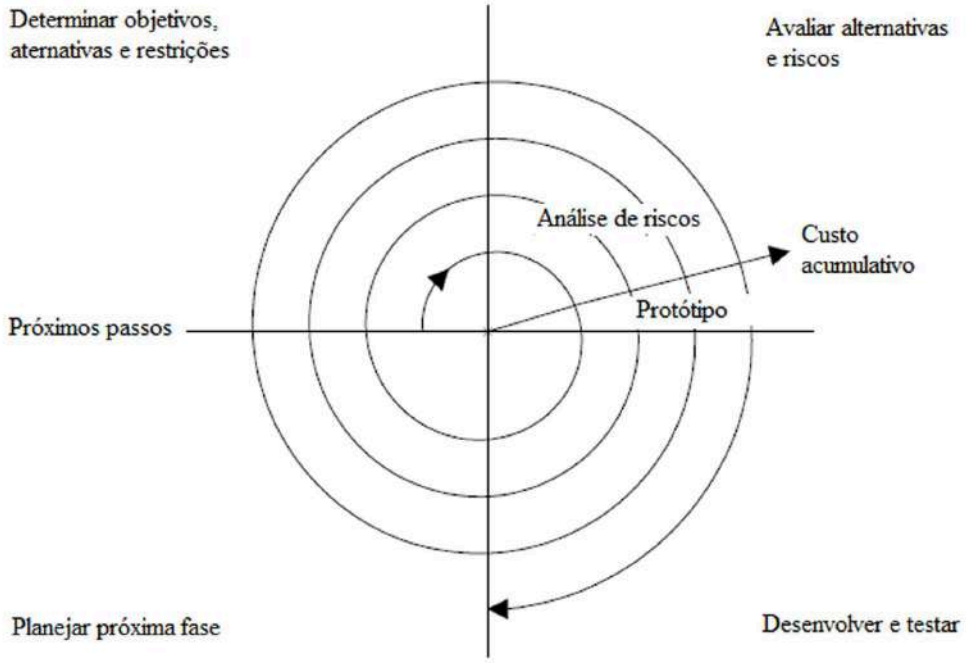

Figura 2.3: Modelo em espiral.
Fonte: (NATIONAL INSTRUMENTS, 2008).

Os riscos são quaisquer questões que não são claramente definidas ou que possuem potencial para afetar o projeto de forma adversa. Para cada risco, deve ser considerada duas coisas: a semelhança entre as ocorrências dos riscos (probabilidade) e a severidade do efeito do risco ao projeto (perdas).

Os riscos e perdas podem ser avaliados em uma escala de 1 a 10, onde 1 representa a menor probabilidade ou perda e 10 representa a maior. A exposição aos riscos é o produto destas duas condições. Um quadro como o representado na Tabela 2.1 pode ser usado para acompanhar os principais itens de risco do projeto.

Em geral, deve-se atender os riscos que possuem maior exposição. No exemplo do quadro da Tabela 2.1, primeiro a espiral lida com a possibilidade da taxa de aquisição

Tabela 2.1: Quadro de riscos e perdas.

ID	**Riscos**	**Probabilidades**	**Perdas**	**Exposição aos riscos**	**Gestão dos riscos**
1	Taxa de aquisição alta	5	9	45	Desenvolver protótipo para demonstração
2	Formato do arquivo pode não ser eficiente	5	3	15	Desenvolver pesquisa para mostrar a velocidade de manipulação de dados
3	Incerteza da interface de usuário	2	5	10	Consultar cliente; Desenvolver protótipo

Fonte: (NATIONAL INSTRUMENTS, 2008).

de dados ser alta. Se após a primeira espiral, seja constatado que as taxas são realmente altas, é possível mudar para uma configuração de *hardware* diferente para atender aos requisitos de aquisição. No entanto, cada iteração pode identificar novos riscos. Neste exemplo, usando um *hardware* melhor pode introduzir maior custo como sendo um novo risco.

Por exemplo, assumindo que se esteja projetando um sistema de aquisição de dados com uma placa de aquisição de dados *plug-in*. Neste caso, o risco é: o sistema fará a aquisição, analisará e mostrará os dados rápido o suficiente? Algumas restrições neste caso são o custo do sistema em contrapartida aos requisitos para uma taxa de amostragem específica e também quanto à precisão das medições.

Após determinar as opões e restrições, os riscos devem ser avaliados. Neste exemplo, pode ser criado um protótipo ou *benchmark* para testar a taxa de aquisição. Após isso, é possível avaliar se vale a pena continuar o a solução ou escolher uma opção diferente. Isso é feito reacessando os riscos baseado nas novas informações adquiridas construindo o protótipo.

Na fase final, os resultados são avaliados junto ao cliente. Baseado nas respostas deste, é possível reavaliar a situação e decidir os próximos passos, decidindo sobre os maiores riscos e iniciando o ciclo novamente.O processo continua até que o *software* seja finalizado ou até que se decida que os riscos são grandes e o desenvolvimento seja interrompido. É possível que nenhuma opção seja viável devido ao custo, tempo ou o não cumprimento dos requisitos pré-estabelecidos.

A vantagem do modelo em espiral em relação ao modelo em cascata é a possibilidade de

avaliar com quais riscos lidar em cada ciclo. Além disso, é possível avaliar os riscos a partir de protótipos muito mais cedo do que no modelo em cascata. É possível também vencer maiores obstáculos e selecionar alternativas em estágios iniciais, o que é mais barato. Com o modelo em cascata padrão, suposições sobre componentes de riscos podem se espalhar por todo o projeto, e quando se descobre o problema, o retrabalho envolvido pode ser muito caro.

2.4 Fluxo de Comunicação Web

O conceito atual de Web pode ser interpretado como a realização da ideia exposta por Vannevar Bush em 1945. Preocupado com a velocidade com que o conhecimento estava sendo criado pelas publicações e a capacidade com que a humanidade era capaz de navegar por essas informações, ele propôs uma forma de aproveitar os meios mecânicos para "estender a memória humana". O *Memex*, como foi denominado o conceito estabelecido por Bush, seria um dispositivo em que um indivíduo armazena todos os seus livros, registros e comunicações, e que é mecanizado de modo que possa ser consultado com alta velocidade e flexibilidade. Os textos a seguir, sobre comunicação Web foram baseados na bibliografia (KRISHNAMURTHY; REXFORD, 2001).

A *Word Wide Web - WWW* ou simplesmente Web, é o universo de informações acessíveis por meio de computadores em rede, onde uma interface gráfica intuitiva permite que os usuários naveguem por uma coleção de páginas sem se preocuparem com o formato ou o local onde o conteúdo está armazenado. Dessa forma, ela é uma aplicação em rede que liga usuários e serviços distribuídos por computadores em todo mundo.

Embora inicialmente desenvolvida para oferecer acesso público às informações, a Web é cada vez mais usada dentro das instituições para conectar usuários com dados privados ou patenteados. Muitas empresas possuem *sites* internos que oferecem acesso a bancos de dados com informações, por exemplo, sobre salários e benefícios ou ainda dados para desenvolvimento colaborativo. O número de *sites* Web internos, cresceu consideravelmente nos últimos anos, possivelmente ultrapassando o número de páginas públicas em geral. A presença crescente de *sites* da Web internos destaca a diferença entre a Web e a internet. Uma empresa pode por exemplo, conectar seus computadores a uma rede privada sem oferecer acesso ao restante da internet, ao passo que os computadores que se comunicam via internet não atuam necessariamente como clientes ou servidores da Web.

Ainda assim, a internet está bastante ligada à Web. Na verdade, ela fornece uma infraestrutura de comunicação global, permitindo que clientes da Web acessem uma grande variedade de servidores Web em todo mundo.

O entendimento sobre o funcionamento da Web parte da compreensão de seus com-

ponentes semântico principais: *Uniform Resource Locator* (URL) são os mecanismos de nomeação universais para identificar os recursos na Web. *Hypertext Markup Language* (HTML) é uma linguagem padrão para a criação de documentos de hipertexto e *Hypertext Transfer Protocol* (HTTP) é a linguagem para a comunicação entre clientes e servidores da Web. A Figura 2.4 ilustra de forma simplificada os componentes para comunicação Web.

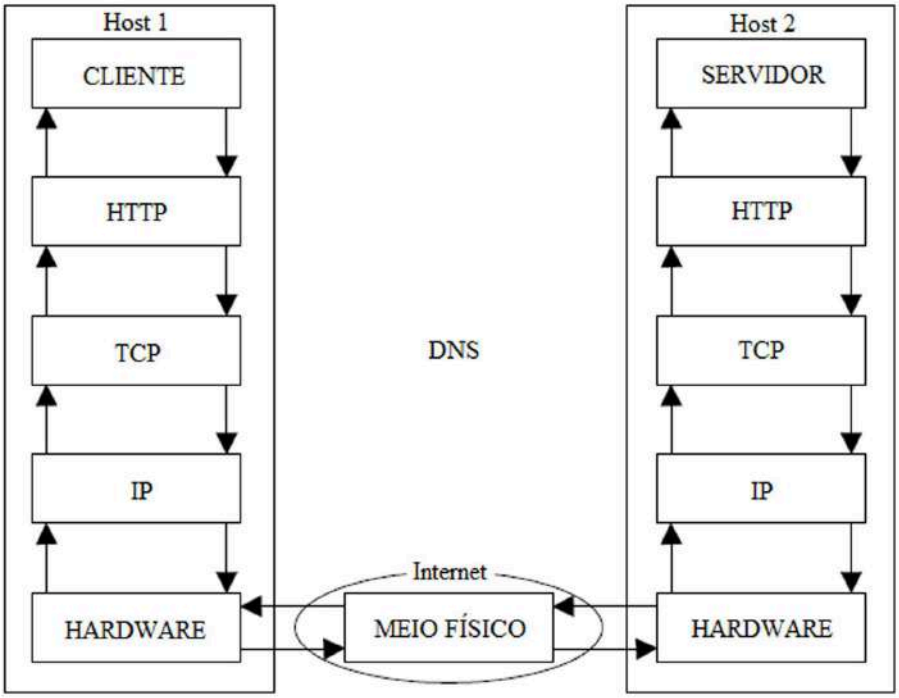

Figura 2.4: Esquema de comunicação Web.

O acesso e a manipulação de recursos distribuídos ao longo da Web exige um modo para identificá-los. Um recurso da Web é identificado por um *Uniform Resource Identifier* (URI) que pode ser imaginado como um ponteiro para uma caixa preta à qual métodos de pedido podem ser aplicados para gerar respostas potencialmente diferentes em diferentes momentos. Um método de pedido é uma operação simples como a busca, troca ou exclusão de recursos. Um URI representa um recurso independentemente de seu local ou valor atual. No nível mais alto, é simplesmente uma *string* formatada, como *http://www.weblab.com/figura.gif*. Nesse exemplo, identifica-se três partes: o protocolo para a comunicação com o servidor (*http*); o nome do servidor (*www.weblab.com*) e o nome do recurso nesse servidor (*figura.gif*). O URL é o formato mais popular do URI.

O HTML oferece uma representação de padrão para documentos de hipertexto em formato ASCII e permite que os autores formatem texto, referenciem imagens e incluam *links* de hipertexto para outros documentos. A sintaxe HTML é relativamente simples e fácil de se aprender e o documento mais simples é pouco mais que um arquivo ASCII

puro, sem formatação especial, com texto ou referências a outros recursos. Aplicações mais complexas, podem ser desenvolvidas utilizando *Java*, juntamente com HTML. Os arquivos HTML servem para ser analisados por programas de computadores, como *Web browsers*, ao invés de serem lidos diretamente pelos usuários.

A operação da Web depende de se ter um modo padronizado e bem definido para os seus componentes se comunicarem. O HTTP é o modo mais comum de se transferir recursos. Esse protocolo define o formato e o significado das mensagens trocadas entre os componentes como clientes e servidores. Ele define a sintaxe das mensagens e como os campos em cada linha da mensagem devem ser interpretados. É um protocolo pedido-resposta em que o cliente envia uma mensagem de pedido e depois o servidor responde com uma mensagem de resposta. Os pedidos de cliente normalmente são disparados por ações de usuários, como um clique em um *link* de hipertexto ou a digitação de uma URL na janela do *browser*. O HTTP é um protocolo sem estado, ou seja, clientes e servidores tratam cada troca de mensagem independentemente, e não precisam manter qualquer informação de estado entre pedidos e respostas.

A Web consiste em uma coleção de recursos ou objetos distribuídos ao longo da internet. Cada recursos é um documento ou serviço acessível pela rede, que pode estar disponível em diferentes formatos (como HTML ou PostSript). Um recurso pode ser um arquivo estático em uma máquina ou gerado dinamicamente no momento do pedido. Cada transferência HTTP refere-se a um recurso único, conforme identificado pela URL na mensagem de pedido. Uma página Web consiste em um recurso de container, que pode incluir *links* para um ou mais recursos embutidos, como imagens ou animações. O *dowload* de uma página da Web envolve transferências HTTP separadas para o container e cada um dos recursos embutidos. Cada transferência HTTP consiste em duas mensagens: a mensagem de pedido, primeiro enviada pelo cliente, e a mensagem de resposta correspondente vinda do servidor. Dessa forma, o cliente é o emissor da mensagem de pedido e o receptor da mensagem de resposta, enquanto o servidor é o receptor da mensagem de pedido e o emissor da mensagem de resposta.

O programa cliente que inicia o pedido, denominado agente de usuário, inicia um pedido HTTP e trata a resposta. O exemplo mais comum de agente de usuário é o *browser da Web*, que gera pedidos em favor de um usuário e realiza uma variedade de outras tarefas, como exibir páginas e armazenar os marcadores do usuário. Na prática, o cliente pode enviar seu pedido para um intermediário - outro componente da Web no caminho para o servidor de origem. O servidor de origem é o programa que oferece ou gera um recurso. Por exemplo, funcionários de uma empresa poderiam configurar seus *browsers* para passarem os pedidos por um *proxy* compartilhado. Um *proxy* é um programa intermediário que funciona como um servidor para um cliente e como um cliente para um

servidor. Os *proxies* podem realizar diversas funções, como filtrar pedidos para *sites* da Web indesejáveis, oferecer um certo grau de anonimato para os clientes e realizar o *caching* de recursos populares. Como o HTTP é um protocolo sem estado, o servidor não tem que manter quaisquer informações sobre o pedido uma vez que a resposta foi enviada. O servidor no entanto, pode instruir o agente do usuário a reter o estado durante uma série de pedidos e respostas, armazenando um *cookie* no cliente. *Cookies* são informações do estado passadas entre o agente do usuário e o servidor de origem. Um *cache* de Web é a capacidade de armazenar e utilizar recursos utilizados anteriormente.

Os programa cliente e servidor normalmente são executados em computadores ou *hosts* separados, residindo normalmente em locais diferentes na internet. O envio e o recebimento de mensagens HTTP exigem uma maneira de os dois *hosts* se identificarem para trocarem informações. O desenvolvimento da Internet levou à definição de protocolos padrão para dar suporte a um série de serviços conectados, como por exemplo *File Transfer Protocol* (FTP) e a própria Web. No nível mais baixo, a Internet fornece um serviço de comunicação muito básico, ou seja, a remessa de um pacote de dados (unidade básica de comunicação na internet) de um *host* para outro. O *Internet Protocol* (IP) coordena a remessa de pacotes individuais, na qual os *hosts* de envio e recepção são identificados por endereço IP.

O simples serviço de entrega de pacotes não satisfaz as necessidades da maioria das aplicações interligadas em rede, como a Web. Um cliente da Web identifica o servidor por um nome de *host* por exemplo, *www.weblab.com* ao invés de um endereço IP e os dois aplicativos trocam mensagens HTTP no lugar de pacote IP. O *Domain Name System* (DNS) e o *Transmission Control Protocol* (TCP) fazem a ponte. Antes de contatar o servidor da Web, o cliente primeiro traduz o nome do *host* em um endereço IP associado. O cliente da Web faz uma chamada do sistema para contatar um servidor de DNS que retorna o endereço de IP. Usando esse endereço IP, o cliente inicia a comunicação com o servidor, ou seja, eles estabelecem uma conexão TCP, um canal lógico bidirecional de comunicação entre as duas aplicações. Implementado no sistema operacional dos dois *hosts*, o TCP oculta os detalhes do envio e da recepção de dados via internet.

2.4.1 Protocolos da Web

As transferências da Web dependem de um conjunto de protocolos de comunicação. Um protocolo define a sintaxe e a semântica das mensagens trocadas entre emissores e receptores. Por exemplo, o HTTP define o formato e o significado dos pedidos enviados por clientes e as respostas enviadas pelos servidores. Os protocolos de rede, geralmente são desenvolvidos em camadas, com cada uma tratando de um aspecto específico da comunicação. O conjunto de protocolos para a internet consiste em quatro camadas

principais como ilustra a Figura 2.5.

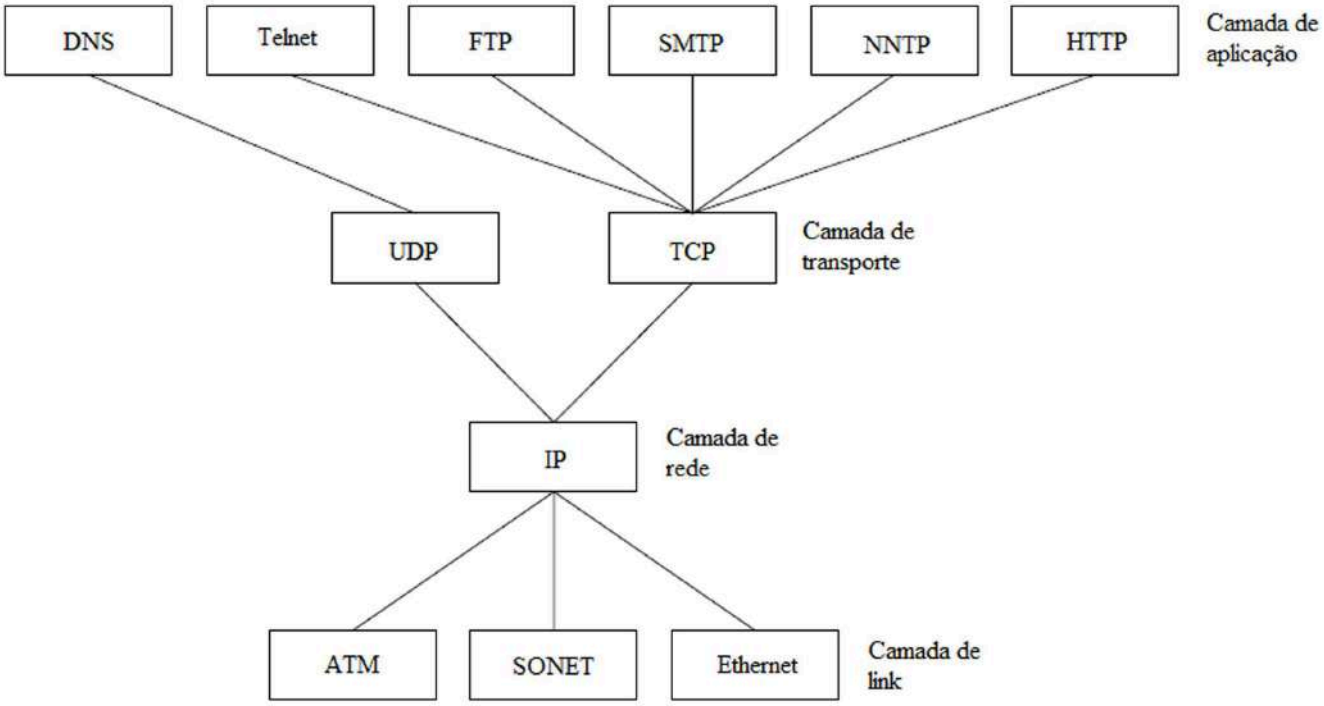

Figura 2.5: Disposição de protocolos em camadas.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

A camada de *link* trata dos detalhes de *hardware* da interação com o meio físico de comunicação, como um rede Ethernet, *Asynchronous Transfer Mode* (ATM) ou *Synchronous Optical Network* (SONET). A camada de rede cuida da remessa de pacotes individuais de dados através da rede. Um protocolo da camada de rede é em roteadores e *hosts* finais. A camada de transporte coordena a comunicação entre os *hosts* em favor da camada de aplicação. Na prática, um protocolo da camada de transporte normalmente é implementado no sistema operacional do *host* final. Finalmente, a camada de aplicação trata dos detalhes das aplicações específicas. Na prática, um protocolo dessa camada normalmente é implementado como parte do *software* de aplicação, como em *browser* da Web ou um servidor Web.

A divisão da funcionalidade permite que cada protocolo focalize a realização de uma única tarefa com interfaces bem definidas para os protocolos nas camadas adjacentes. A padronização dos protocolos permite a interoperabilidade entre os componentes desenvolvidos por diferentes fornecedores. O IP, o TCP e o DNS são os três principais protocolos envolvidos na transferência de mensagens HTTP, ou seja, são considerados protocolos de suporte ao HTTP.

2.4.2 Protocolo IP

O *Internet Protocol* (IP) é o protocolo em nível de rede que dá suporte à internet, cujo propósito é rotear datagramas, ou seja, um conjunto de pacotes através de um conjunto de redes inter-conectadas. As redes consistem em roteadores que direcionam o tráfego por uma coleção de *links*. O roteador por sua vez, é um computador especializado dedicado a direcionar o tráfego dos seus *links* de entrada para seus *links* de saída.

Ao receber um tráfego em um *link* de entrada, um roteador precisa selecionar um *link* de saída para transportar esse tráfego para o seu destino final. Um conjunto de protocolos de roteamento coordena a seleção de rotas para o tráfego do IP seguir. Dentro de uma única rede, os roteadores se comunicam utilizando protocolos intradomínio e entre redes, os interdomínios.

A idéia do IP é manter a rede relativamente simples e colocar qualquer inteligência necessária nos *hosts* finais. No nível mais básico, o IP oferece uma estrutura para enviar pacotes individuais. Um pacote é uma unidade de informação. Ao trafegar de um *host* de envio para um *host* de recepção, um pacote atravessa uma coleção de roteadores que se comunicam por IP. Os roteadores na internet tratam cada pacote independentemente, e não precisam reter os estado entre os pacotes sucessivos. Uma sequência de pacotes IP trafegando de um *host* para outro pode não seguir o mesmo caminho pela rede. Além disso, eles podem ser perdidos, adulterados ou entregues fora de ordem.

O tráfego na internet é dividido em pacotes IP que podem ser enviados conforme a necessidade do usuário. Dessa forma, como esse tráfego pode ser gerado a qualquer momento, cada pacote precisa incluir um cabeçalho identificando o destino. O cabeçalho de um pacote IP inclui todas as informações necessárias para o roteador entregar o conteúdo ao destino apropriado.

Os IPs, que identificam os *hosts* da internet, são endereços numéricos. O campos do cabeçalho IP normalmente são definidos pelo sistema operacional na máquina emissora. Embora o roteador IP encaminhe um pacote com base no endereço de destino, o cabeçalho inclui campos adicionais que são importantes para a comunicação bem sucedida entre o emissor e o receptor. A Figura 2.6 ilustra o cabeçalho de um pacote IP.

Quando a rede está pouco carregada, um pacote recebido prossegue diretamente para um *link* de saída em cada roteador em seu caminho. No entanto, devido à ausência de um circuito dedicado, a rede nem sempre possui recursos suficientes para transportar cada pacote ao seu destino. Sendo assim, quando um *link* está sobrecarregado, o roteador armazena temporariamente em uma fila os pacotes que estão aguardando. Durante o congestionamento, o roteador pode ter que descartar um ou mais pacotes para evitar o estouro da fila.

A maioria das aplicações da internet não pode tolerar dados incompletos, sendo assim,

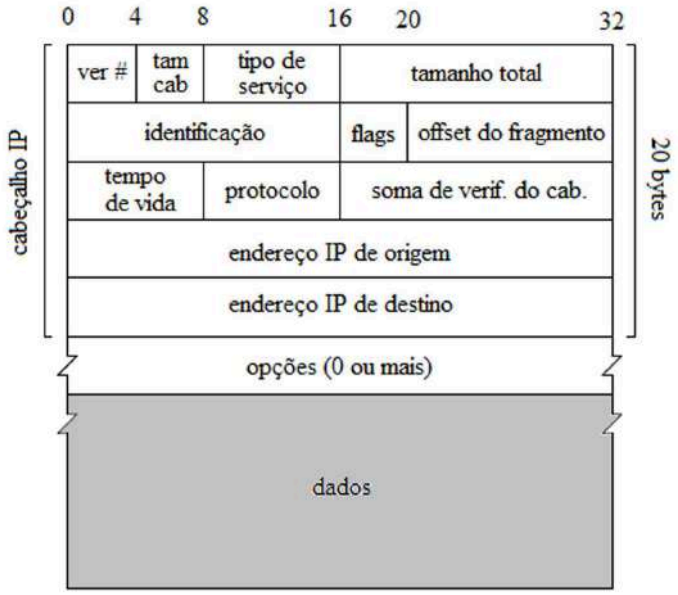

Figura 2.6: Formato de um pacote IP.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

os serviços mais rigorosos são fornecidos por protocolos da camada de transporte, rodando nas máquinas de origem e destino, e não pelos roteadores da rede. Os dois principais protocolos de transporte são o TCP e o UDP. O TCP oferece a principal abstração necessária para a maioria das aplicações da internet, uma sequência lógica que oferece uma sequência de bytes do emissor para o receptor de forma ordenada e confiável. O UDP oferece uma abstração simples da remessa de datagrama pouco confiável.

2.4.3 Protocolos TCP e UDP

O TCP coordena a transmissão de dados entre um par de aplicações. As aplicações se comunicam lendo e escrevendo em um *socket* ou soquete, que apresenta dados como um fluxo de *bytes* ordenado e confiável. Os sockets podem ser entendidos como programas que atuam como uma interface entre um programa aplicativo e os protocolos de comunicação em um sistema operacional, permitindo que um processo troque mensagens com outro processo, tanto local, quanto remotamente. Assim, são conhecidos como Interface de Programa Aplicativo (*Application Program Interface* - API) (COSTA, 2005).

O TCP oferece uma conexão lógica entre duas extremidades pela montagem em cima do serviço de remessa de pacote do IP. O emissor do TCP divide os dados em segmentos e transmite cada um deles em um pacote IP juntamente com o cabeçalho TCP. Antes de transmitir os dados, as duas extremidades precisam estar coordenadas para estabelecer a conexão. Durante a transferência de dados, as extremidades cooperam para controlar o fluxo de dados e retransmitir os pacotes IP perdidos. Além disso, cada extremidade

adapta sua taxa de transmissão em resposta ao congestionamento, para evitar a sobrecarga da rede. O cabeçalho TCP inclui as informações necessárias para coordenar a remessa ordenada e confiável dos segmentos. A Figura 2.7 ilustra o cabeçalho de um protocolo TCP.

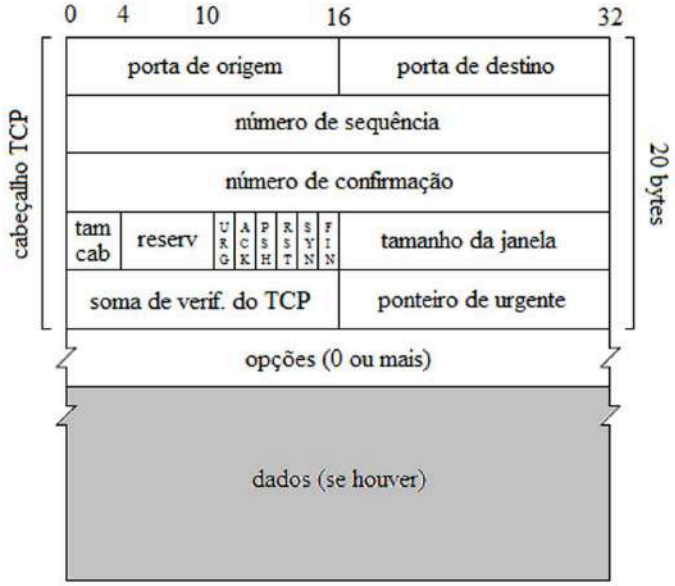

Figura 2.7: Formato de um segmento TCP.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

O protocolos em nível de transporte coordenam a transferência de pacotes IP para oferecer uma semântica de serviço mais rica, ou seja, onde as aplicações de envio e recepção devem assumir uma comunicação por um canal que oferece confiabilidade. Como o IP não oferece tal características, essa abstração é fornecida pelo TCP. Uma aplicação utiliza o TCP criando um *socket*, de modo semelhante à abertura de um arquivo. As duas extremidades precisam de um modo exato para identificar o *socket*. No entanto, conhecer os endereços IP de cada máquina não é o suficiente. Uma única máquina pode rodar várias aplicações e uma única aplicação, como um servidor da Web, pode ter vários *sockets*.

Cada *socket* está associado a um número de porta em cada extremidade. O número de porta, é um inteiro de 16 bits (0 a 65535), onde os números abaixo de 1024 são portas reservadas para os protocolos específicos em nível de aplicação como a porta 80 para o HTTP. Já os números de portas restantes, podem ser utilizados para qualquer aplicação. Se um novo serviço da internet que não tenha uma porta bem conhecida é desenvolvido, essa aplicação poderá selecionar um número de porta não reservado resultando em um padrão de fato para essa aplicação.

Por definição, um cliente da Web (como um *browser*), cria um *socket* que se conecta à porta 80 na máquina servidora. No entanto, o servidor poderia ser configurado para

ouvir pedidos do cliente na porta 8000. Nesse caso, o cliente deveria ser configurado para solicitar uma conexão com a porta 8000, ao invés da porta 80. Tal conexão é feita colocando-se o número da porta na URL. Sendo assim, se o usuário solicitasse por exemplo o recurso *http://www.weblab.com:8000/*, o cliente solicitaria um *socket* que se conecta à porta 8000 no servidor correspondente. O cliente também precisa de um número de porta a sua extremidade de conexão. Caso contrário, o servidor não saberia como direcionar os dados para o cliente. Como parte da criação do *socket*, o sistema operacional na máquina do cliente atribui um número de porta efêmero (entre 1024 e 65535) à aplicação do cliente.

Portanto, o *socket* é identificado por cinco informações diferentes; dois endereços IP para as máquinas executando as duas aplicações, dois números de porta para as duas extremidades de aplicação e o protocolo, no caso, o TCP. A atribuição de números de porta às aplicações é controlada pela *Internet Assigned Number Authority* (IANA). Já as aplicações, criam *sockets* por meio de chamadas implementadas no sistema operacional. A Figura 2.8 ilustra o diagrama de comunicação entre cliente e servidor para o *socket datastream* (TCP).

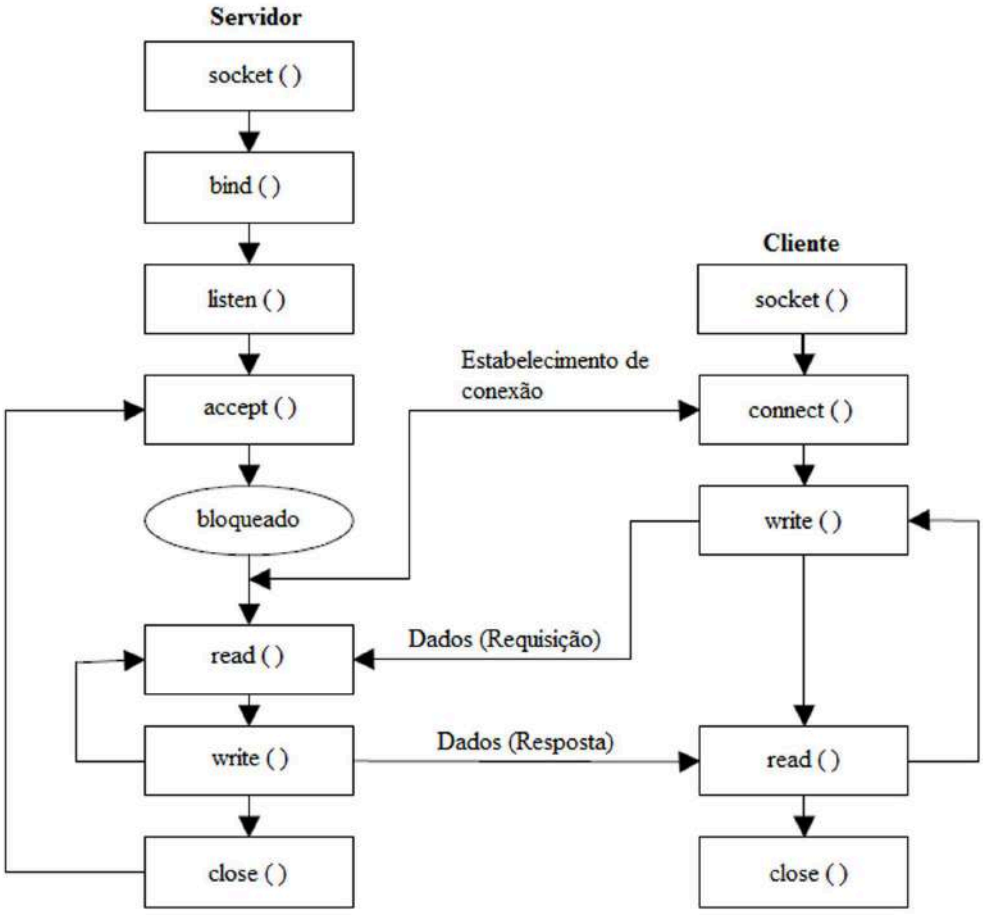

Figura 2.8: Socket datastream (TCP).
Fonte: (COSTA, 2005).

Considerando-se por exemplo uma aplicação A, representando um cliente Web estabelecendo uma conexão via *socket* com uma aplicação B, representando um servidor da Web. A aplicação A inicia a criação *socket* invocando uma chamada do sistema. Sendo assim, a função *socket( )* é usada para criar um novo *socket*. Em seguida, a aplicação invoca a chamada *connect( )* para associar o soquete ao endereço IP e número de porta para a aplicação B. Como parte da execução da chamada *connect( )*, o sistema operacional também seleciona um número de porta incomum (de 1024 a 65535) para a aplicação A. Nesse ponto, o sistema operacional executando a aplicação A conhece os dois endereços IP e dois números de porta que identificam exclusivamente a conexão bidirecional entre as duas aplicações. Depois, o sistema operacional inicia o estabelecimento da conexão TCP com a aplicação B. Quando a conexão for estabelecida, a chamada *connect* retorna e a aplicação A começa a ler e escrever dados no *socket*.

Ao contrário, a aplicação B inicialmente desempenha uma papel passivo no estabelecimento do *socket*. A aplicação B ouve os pedidos em uma porta específica para estabelecer uma conexão. Isso envolve a criação de um *socket* e a chamada à função *bind( )* para atribuir um número de porta local com por exemplo a porta 80. Em seguida, a aplicação B invoca a chamada do sistema *listen( )*, significando a intenção de B de esperar pelas conexões das aplicações remotas, o que aciona o sistema operacional para que responda a quaisquer pedidos para estabelecer conexões TCP com essa porta. A aplicação descobre a respeito dessas novas conexões TCP chamando a função *accept( )*. Por definição, a chamada *accept( )* espera até que pelo menos uma nova conexão esteja disponível. Quando houver uma conexão disponível, a chamada do sistema completa a criação do *socket*. Assim, a aplicação B pode iniciar a leitura dos dados do *socket* e a escrita dos dados para o *socket*.

Quando a conexão estiver sido estabelecida, qualquer aplicação poderá ler ou gravar dados. Como os *sockets* estabelecem um canal de comunicação bidirecional, as duas aplicações poderiam ler e escrever ao mesmo tempo. Na Web, o cliente (Aplicação A) inicia a comunicação escrevendo o pedido HTTP no *socket*. O servidor (Aplicação B), espera então que os dados cheguem ao seu *socket*. Então, o servidor lê o pedido HTTP e após processar o pedido, ele retorna uma resposta HTTP no *socket*. Nesse intervalo, o cliente espera que os dados cheguem ao seu *socket* e lê a resposta do *socket* do servidor. O TCP oferece um serviço muito mais genérico, que aceita a comunicação bidirecional arbitrária entre as duas aplicações. Sendo assim, em geral, tanto A quanto B poderiam escrever os dados primeiro ou as duas aplicações poderiam ler e escrever ao mesmo tempo.

O sistema operacional cuida dos detalhes do estabelecimento de uma conexão lógica entre as duas aplicações e coordenação da transferência de pacotes IP. Para a aplicação A, o sistema operacional executa as funções *socket( )* e *connect( )*. Se o *host* remoto não

responder, o sistema operacional informará à aplicação A que o pedido para criar um *socket* falhou. Para a aplicação B, o sistema operacional executa a função *socket( )*, *bind( )*, *liste( )*, e *accept( )*. Como parte da criação do *socket*, cada *host* aloca memória para transmitir e receber pacotes.

Quando a aplicação escreve no *socket*, o sistema operacional direciona os dados para o endereço IP e porta remota. Da mesma forma, quando os pacotes chegam, o sistema operacional direciona os dados para o *socket* correto, com base no número de porta. A aplicação pode então ler os dados do *socket*. Debaixo das duas aplicações que se comunicam, o sistema operacional coordena o envio e a recepção dos pacotes para criar a abstração de um fluxo de bytes ordenado e confiável.

A Figura 2.9 mostra um diagrama que ilustra o procedimento para conexão entre cliente e servidor usando *socket datagram* (UDP). A utilização das funções *sockets* é mais simples de se implementar, no entanto, a utilização de comunicação baseada em datagramas determina que as rotinas de recuperação das sequências dos pacotes entregues a rede devem ser programadas, aumentando assim a possibilidade de falhas na implementação.

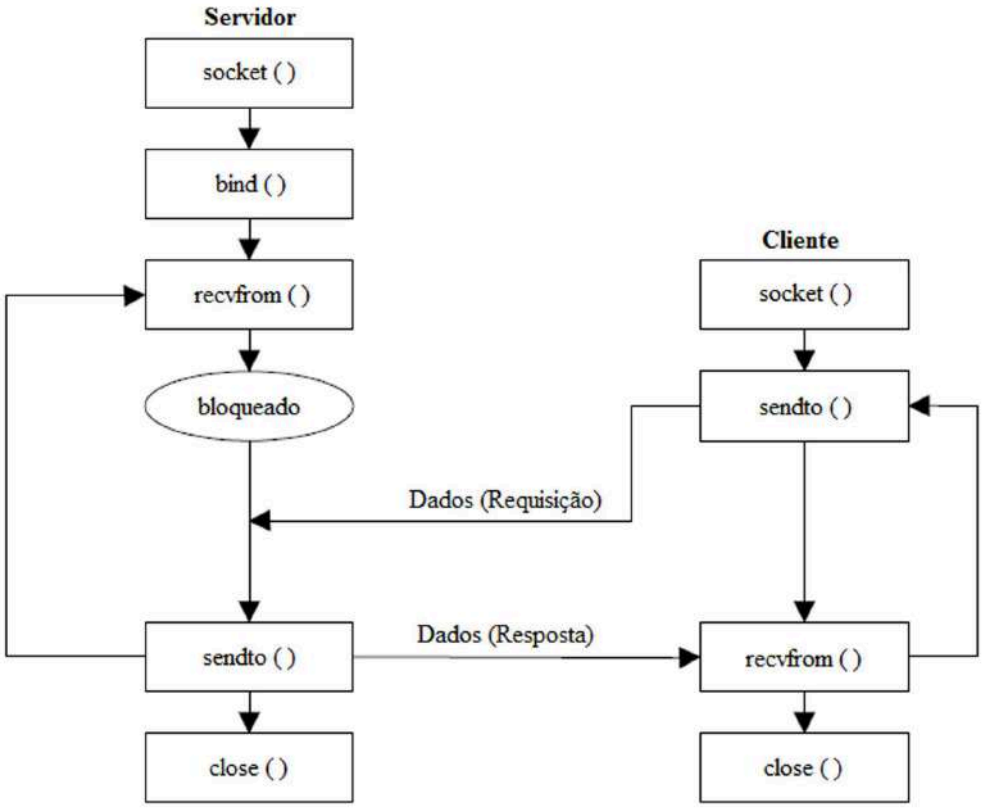

Figura 2.9: Socket datagram (UDP).
Fonte: (COSTA, 2005).

A sequência e as funções definidas para uma conexão *datagram* são as mesmas da conexão *stream*, excluindo apenas as funções *Listen( )*, *Accept( )* e *Connect( )*. Isso

ocorre porque o protocolo baseado em datagrama não utiliza uma conexão real entre as máquinas de origem e destino, implementando automaticamente dentro das funções de troca de mensagens, *Sendto( )* e *Recvfrom( )*, um cabeçalho com o endereço IP da máquina no qual será feita a comunicação (COSTA, 2005).

Caso ocorra uma perda de conexão, tanto por parte do cliente quanto pelo servidor sem antes ter sido efetuada a função *Close( )*, que ainda estiver na linha não poderá reutilizar a porta que estava sendo utilizada até que o seu fechamento se dê de forma correta. Para evitar que problemas dessa natureza possam vir a ocorrer, é necessário o desenvolvimento de uma função de espera (*time-out*), de modo a proporcionar um nível de segurança ainda maior para as conexões existentes, e permite que a função *Close( )* seja ativada automaticamente ao se verificar a inexistência de qualquer tipo de troca de mensagem após um determinado tempo, possibilitando assim a otimização das portas do protocolo TCP/IP.

2.4.4 Protocolo DNS

Os roteadores IP encaminham pacotes baseados no endereço de destino no cabeçalho IP. No entanto, os endereços numéricos são inconvenientes para os usuários e para as aplicações. Ao invés disso, as máquinas na internet possuem nome de *host*, como por exemplo, www.weblab.com, que consistem em *strings* separadas por ponto, associado a um IP. O endereço IP associado a um nome pode mudar com o tempo. Por exemplo, o endereço IP de um *site* da Web pode depender da empresa que está hospedando o conteúdo, dessa forma, se o *site* da Web mudar para um novo serviço de hospedagem, o endereço IP poderá mudar. Em contrapartida, se as URLs do *site* da Web incluissem o endereço IP, como, *http://10.187.56.3/algo.html* as URLs teriam que mudar toda vez que o endereço IP mudasse. Finalmente, o mesmo *site* da Web pode estar disponível em vários *hosts*, cada um com um endereço IP diferente.

A tradução dos nomes de *host* para endereços IP e virse-versa, é coordenada pelo DNS. As aplicações da internet, como os *browsers* da Web, acessam o DNS por meio de um tradutor, ou seja, uma biblioteca de *software* que está ligada à aplicação. Esse tradutor realiza duas funções principais: a função *gerhostbyname( )* que converte um nome de *host* em um endereço IP e a função *gethostbyaddr( )* que faz a operação inversa. O tradutor interage com um ou mais servidores de DNS para realizar suas funções em favor da aplicação. Para iniciar o processo, ele precisa saber contatar pelo menos um servidor de DNS. Uma máquina usando a internet normalmente está configurada com uma lista de endereços IP, cada um correspondendo a um servidor de DNS local.

Supondo que um usuário digite a URL *http://www.weblab.com* em um *browser* da Web, o *software* do *browser* extrairia o nome de domínio *www.weblab.com* da URL. Para

contatar o servidor da Web, o *browser* invocaria a função *gethostbyname( )*, que entraria em contato com um dos servidores de DNS locais. A réplica do servidor de DNS inclui o endereço de IP do servidor da Web, que é retornado para a função *gethostbyname( )*. Depois, o *browser* pode iniciar a comunicação com *www.weblab.com*.

O DNS é então, um banco de dados distribuído, que consiste em um conjunto hierárquico de servidores de nomes, cada um responsável por uma parte dos nomes de domínio e espaço de endereços. A arquitetura do DNS reflete a herarquia dos nomes de *host* e endereços IP. A responsabilidade da tradução dos nomes de *hosts* para endereços IP é atribuída com base nas *strings* no nome do host. De modo semelhante, a responsabilidade pela tradução do endereço IP em nomes de *host* é atribuída com base nos números do endereço IP. A Figura 2.10 ilustra a arquitetura do DNS, ela é composta por um servidor raiz, um servidor de domínio de alto nível e um servidor de domínio de segundo nível. Além disso, um domínio *.arpa* separado controla a tradução de endereços IP par nomes de domínio KRISHNAMURTHY; REXFORD (2001).

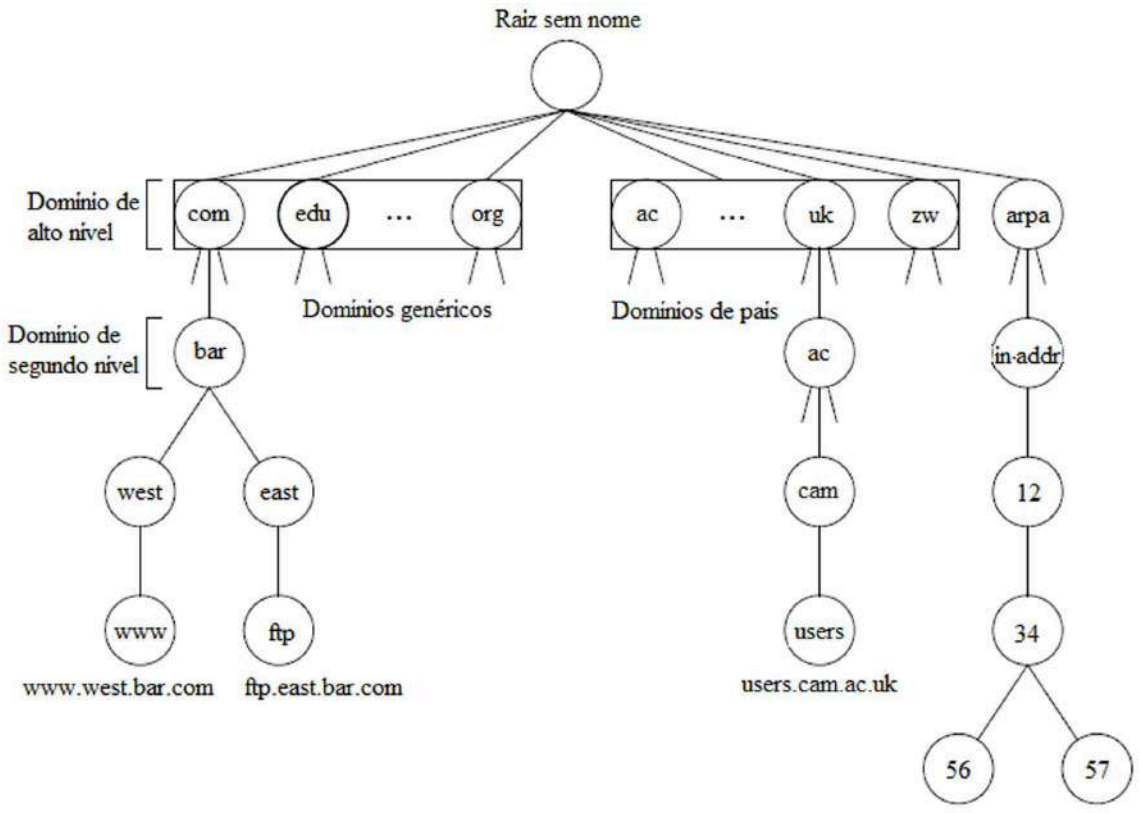

Figura 2.10: Hierarquia do DNS.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

O protocolo DNS controla a comunicação entre um cliente DNS e um servidor de DNS. Um cliente DNS envia uma consulta por informações (por exemplo, o endereço de IP associado a um nome de host qualquer) a um servidor de DNS, e o servidor retorna

a resposta com a informação solicitada (por exemplo, o endereço IP). O servidor DNS local envia respostas para os tradutores e emite consultas a outros servidores de DNS. Os servidores de DNS raiz enviam respostas às consultas geradas por outros servidores de DNS e não emitem consultas próprias.

Supondo que uma aplicação invoque a chamada *gethostbyname( )* para determinar o endereço IP de *www.foo.bar.com*. O tradutor entra em contato com o servidor DNS local, que emite uma consulta a um servidor DNS raiz para descobrir o endereço IP do servidor de DNS *.com*. Depois o servidor de DNS local envia uma consulta ao servidor de DNS *.com* para descobrir o endereço IP do servidor bar.com. O servidor DNS local em seguida envia uma consulta ao servidor de DNS para a zona *bar.com*. Se essa zona tiver sido subdividida, uma consulta adicional pode ser emitida para o domínio algo.bar.com, que responde com o endereço IP para *www.foo.bar.com*. De modo semelhante, o mapeamento do endereço IP 12.34.56.78 para um nome de *host* resulta em uma série de consultas do DNS a vários servidores de DNS na parte *in-arpa* da hierarquia.

A Figura 2.11 ilustra como ocorrem as consultas ao DNS. Tais consultas podem ser recursivas ou iterativas. Uma consulta recursiva solicita que o servidor de DNS receptor traduza o pedido inteiro por sua conta. Por exemplo, o tradutor emite uma consulta recursiva ao servidor de nomes local para traduzir um nome de *host* para um endereço IP. O tradutor é invocado por uma chamada do sistema na aplicação (etapa 1). Em seguida, o tradutor envia uma consulta de DNS ao servidor de DNS local (etapa 2) e espera uma resposta (etapa 9). O servidor de DNS local trata das etapas envolvidas no cumprimento da consulta do tradutor. Uma consulta iterativa requer que o servidor DNS receptor responda diretamente ao cliente de DNS com o endereço IP do próximo servidor DNS na hierarquia de servidores de nome. Os servidores raiz tratam apenas de consultas iterativas. O servidor de DNS local envia uma consulta ao servidor raiz de DNS (etapa 3) para descobrir os nomes e endereços IP do(s) servidor(es) de DNS para a zona no próximo nível (etapa 4). Depois, o servidor de DNS local pode enviar uma consulta ao próximo servidor de DNS na cadeia (etapas 5 e 6 e etapas 7 e 8). Por fim, o servidor de DNS local responde ao tradutor (etapa 9) e o tradutor oferece o endereço IP para a aplicação (etapa 10).

2.4.5 Protocolo HTTP

O *Hypertext Transfer Protocol* (HTTP) é o protocolo a nível de aplicação de pedido e resposta que oferece suporte para a WWW. Como visto anteriormente, a Web consiste basicamente em três partes semânticas principais, o protocolo HTTP; a *Hypertext Markup Language* (HTML) e o esquema de nomeação URI. A Web possui alguns componentes de *software* como *browsers*, proxies e servidores de origem que se comunicam entre si usando

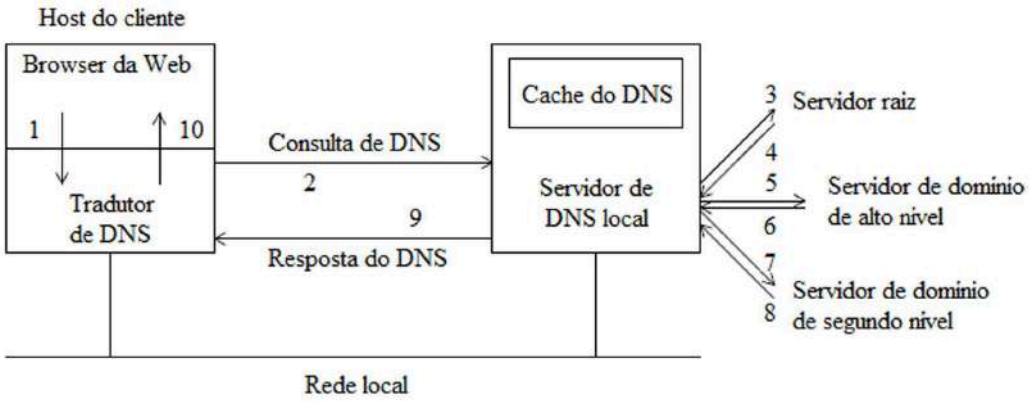

Figura 2.11: Tradutor do DNS e servido de DNS local.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

o protocolo HTTP.

O HTTP utiliza URIs em todas as suas transações para identificar os recursos da Web. O mecanismo de nomeação permite que os recursos residam em qualquer lugar na internet e separa a noção de um recurso e a de uma resposta. Um recurso pode conter o mesmo URI conectado a ele para sempre, embora a representação e o conteúdo do recurso mudem várias vezes durante seu tempo de vida. Um URI portanto, é uma *string* formatada do ponto de vista do protocolo, que indica um recurso independente do seu local ou nome pelo qual é conhecido.

A forma mais comum do URI é o URL. Um URL começa geralmente com a indicação do protocolo que deve ser utilizado, como o *http:*, acompanhado por uma *string* representando o recurso que pode ser obtido por meio desse protocolo. Embora um nome de *esquema*, como é conhecido tal indicação, seja mapeado para um protocolo específico, mais de um protocolo pode estar envolvido no acesso a um recurso nomeado por um URL.

O protocolo HTTP especifica a sintaxe e a semântica pelas quais os componentes da Web, como clientes e servidores, se comunicam uns com os outros. Uma mensagem HTTP, é uma coleção estruturada de octetos em uma sintaxe específica. O protocolo específica um conjunto de métodos de pedido extensíveis, que são usados pelo cliente para realizar operações como pedido, alteração, criação e exclusão de recursos, ou seja, objetos, serviços ou coleção de entidades que podem ser claramente identificados e localizados em qualquer lugar da rede. O HTTP pode usar qualquer protocolo de transporte básico para transmitir a mensagem do emissor para o receptor. No entanto, praticamente todas as implementações conhecidas do HTTP utilizam o TCP como seu protocolo em nível de transporte KRISHNAMURTHY; REXFORD (2001).

Uma das principais características do protocolo HTTP é a sua falta de estado, ou seja, ausência de manutenção de estado entre os pares de pedido e resposta. Cada pedido por

um recurso aciona uma aplicação separada do método de pedido sobre a URL do recurso, com uma nova resposta sendo gerada. Manter o estado a respeito dos pedidos e respostas anteriores dependem dos componentes que usam o HTTP. Um *browser* enviando vários pedidos em sequência pode acompanhar a espera entre cada resposta. Um servidor poderia manter informações sobre o endereço IP do cliente que enviou os doze últimos pedidos. No entanto, o protocolo em si não tem qualquer ciência do pedido ou da resposta anterior. A falta de estado do HTTP foi uma decisão de projeto para garantir sua facilidade de expansão.

Outra característica do HTTP são os metadados. Esses, são informações relacionadas a um recurso, mas que não fazem parte do recurso em si. Os metadados podem incluir dentre outros, o tamanho de um recurso; o tipo do conteúdo e a hora da última modificação do recurso. A inclusão ou não dos metadados, dependem o recurso. Por exemplo, as respostas geradas dinamicamente provavelmente não terão seu tamanho de conteúdo incluído, pois isso aumentaria a latência da resposta. Da mesma forma, incluir a última modificação pode fazer sentido para um recurso estático, mas não para um recurso dinâmico.

Elementos da linguagem HTTP

A seguir são apresentados alguns termos básicos usados na especificação do HTTP são eles: mensagem; entidade; recurso e agente do usuário.

Mensagem

Uma mensagem HTTP é uma sequência de octetos enviada por uma conexão de transporte. Uma mensagem é a unidade de comunicação fundamental no HTTP e pode ser um pedido enviado de um cliente para um servidor ou uma resposta do servidor para um cliente. As mensagens de pedido e resposta podem ter zero ou mais cabeçalhos de mensagem, separadas de um corpo de mensagem opcional por dois caracteres - *carriage return* (CR) e *linefeed* (LF).

A Figura 2.12 mostra uma mensagem de pedido HTTP. Ela possui o seguinte formato sintático:

```
Linha de pedido
Cabeçalho(s) geral(is)/pedido/entidade
CRLF
Corpo da mensagem opcional
```

Como pode-se observar, a mensagem de pedido começa com uma linha de pedido e é seguida por um conjunto de cabeçalhos e um corpo de mensagens opcionais. A linha de pedido consiste em um método de pedido, a URL solicitada e a versão do protocolo

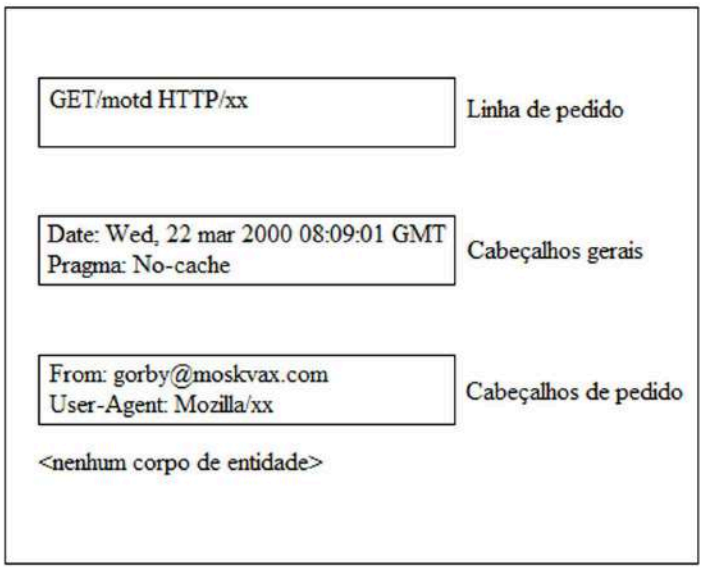

Figura 2.12: Mensagem de pedido HTTP.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

do cliente. O método de pedido em questão é o GET e o recurso solicitado é o */motd*. Também aparece a versão do protocolo, no caso, *HTTP/xx*. Os cabeçalhos *Date* e *Pragma* são cabeçalhos gerais, e podem estar presentes em pedidos e respostas. Os cabeçalhos *From* e *User-Agent* são cabeçalhos de pedido e só aparecem em mensagens deste tipo. A mensagem de pedido, termina com uma sequência de caracteres de quebra de linha *CLRF*, que consiste em caracteres de CR e LF.

A Figura 2.13 mostra uma mensagem de pedido com corpo de mensagem. O método *PUT* é usado para criar um recurso */motd*. O método do pedido é seguido por um único cabeçalho geral (*Date*) e dois cabeçalhos de pedido (*From e User-Agent*). A mensagem possui dois cabeçalhos de entidade (*Content-Length*e *Allow*). O primeiro deles indica o tamanho do conteúdo e o segundo especifíca quais métodos podem ser aplicados ao recurso */motd*. O corpo da mensagem, consistindo em uma *string* vem em seguida. Uma mensagem de resposta possui o seguinte formato sintático:

```
Linha de status
Cabeçalho(os) geral(is)/resposta/entidade
CRLF
Corpo da mensagem opcional
```

A mensagem de resposta começa com uma linha de *status*, que inclui a versão HTTP do servidor e um código de *status* da resposta. Esta é acompanhada por cabeçalhos gerais e de respostas opcionais e um corpo de mensagem também opcional. Devido à presença de intermediários, a mensagem final recebida não refletirá necessariamente o número de versão do protocolo do servidor de origem.

A Figura 2.14 mostra a mensagem de resposta para o pedido GET descrito anteriormente. A linha de *status* na mensagem indica que o servidor aceita *HTTP/xx*, e o código de resposta *200 OK* indica que o pedido teve sucesso. A mensagem de resposta inclui o cabeçalho de resposta geral *Date* e um cabeçalho de resposta *Server*. O tamanho do corpo da entidade é apresentada no cabeçalho *Content-Length*. Depois da sequência de quebra de linha *CRLF* vem o corpo da entidade, que no caso, é uma *string*.

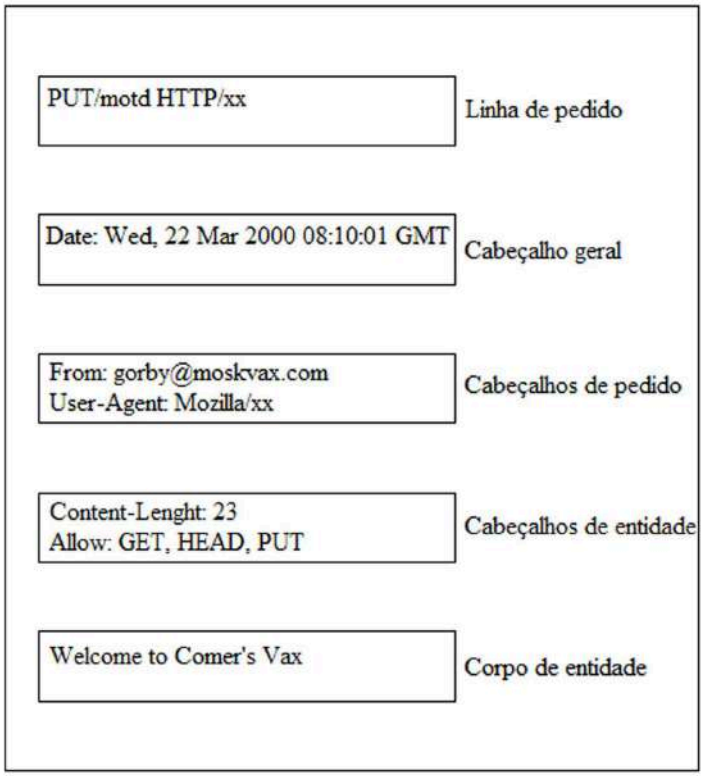

Figura 2.13: Mensagem de pedido HTTP com corpo de mensagem.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

Entidade

A entidade é a representação de um recurso que está delimitado em uma mensagem de pedido ou resposta. Uma entidade é formada por cabeçalhos de entidade e um corpo de entidade opcional. Um corpo de entidade, é de certa forma, a parte mais importante da mensagem HTTP. No caso de uma mensagem de pedido, o corpo da entidade poderia ser os dados que o usuário digitou em um formulário HTML. Já em uma mensagem de resposta, o corpo de entidade é o corpo da mensagem, ou seja, o conteúdo da resposta sem os cabeçalhos de resposta.

Recurso

Figura 2.14: Mensagem de resposta HTTP.
Fonte: (KRISHNAMURTHY; REXFORD, 2001).

Um recurso pode ser definido como um objeto de dados ou serviço da rede que pode ser identificado por um URI. O objeto de dados ou o serviço pode estar localizado em qualquer lugar no mundo, desde que seja acessível por uma conexão de rede. Um objeto de dados é gerado estática ou dinamicamente, mas um serviço pode implementar uma aplicação arbitrária que simplesmente utiliza a Web como um meio de transferência para iniciar o serviço e tratar a resposta.

Agente do usuário

Um agente do usuário é o cliente que inicia o pedido, podendo ser um *browser* ou outra ferramenta envolvida na geração do pedido. A distinção entre um agente do usuário e um cliente geral é muito importante. Em geral, requisitos adicionais são impostos ao componente que inicia. O agente do usuário só é conectado parcialmente ao usuário se o pedido foi iniciado pelo usuário. O agente do usuário pode ignorar mensagens de erro ou apresentá-las, oferecer opções como tentar um pedido novamente após uma falha de autenticação, redirecionar os pedidos ou escolher um local alternativo específico, ou ainda negociar a melhor representação de uma resposta. Normalmente, a informação sobre o agente do usuário é enviada como parte do pedido no cabeçalho *User-Agent*. O cabeçalho inclui informações como o nome do *browser* ou o sistema operacional da máquina.

Métodos de pedido do HTTP

Um método de pedido notifica um servidor HTTP quanto à ação que deve ser realizada sobre o recurso identificado pelo URI especificado na lista de pedido. O método mais utilizado é o GET, que apanha o valor atual que apanha o valor atual do recurso identificado pela URI. É difícil descrever métodos, campos de cabeçalho e código de resposta de uma forma sequencial pois qualquer discussão sobre um desses conceitos necessariamente envolve os outros dois. Um método de pedido está incluído no pedido de um cliente junto com vários cabeçalhos e um URI. O método é aplicado ao recurso pelo servidor de origem e a resposta é gerada. Tal resposta consiste em um código de resposta, metadados sobre o recurso e outros cabeçalhos de resposta.

Duas propriedades importantes de um método são segurança e coerência. Um método de pedido que simplesmente examina o estado de um recurso é visto como um método seguro. Já um que pode alterar o estado do recurso, não é seguro. Um método coerente, por outro lado, possui a propriedade de que os efeitos colaterais de um pedido são iguais aos de vários pedidos idênticos. Dessa forma, se vários pedidos idênticos são emitidos em sequência, o resultado da aplicação do método sobre o recurso ou não produz o mesmo efeito em qualquer caso. A seguir, são descritos os métodos de pedido do HTTP.

GET

Um pedido GET é aplicado ao recurso especificado no URI, e a resposta gerada é o valor atual do recurso. Essa resposta é retornada ao cliente solicitante. Se o URI se referisse a um arquivo estático, um pedido GET resultaria na leitura do arquivo e no retorno do seu conteúdo. Se o URI se referisse a um programa, então os dados são retornados como corpo da resposta. O método GET é seguro e coerente. O método de pedido GET não possui corpo de pedido. Se um corpo de pedido estiver presente, ele será ignorado pelos servidores.

HEAD

O método HEAD foi introduzido para se obter apenas os metadados associados a um recurso. Nenhum corpo de mensagem é retornado como resultado de um pedido HEAD. No entanto, os metadados que um servidor retorna deverão se os mesmos metadados que seriam retornados se o método de pedido fosse o GET.

A aplicação do método HEAD inclui depurar a implementação do servidor com overhead relativamente baixo e determinar se um recurso mudou recentemente sem transferi-lo realmente. Os metadados podem ser colocados em *cache* ou usados para atualizar os valores em *cache* existentes se uma mudança no recurso puder ser detectada. O método HEAD também não possui um corpo de pedido.

POST

Ao contrário do método GET e HEAD, que são utilizados para ler informação, o método POST é utilizado principalmente para atualizar um recurso existente ou para se oferecer entrada para um processo manipulando dados. O corpo do pedido inclui os dados. O servidor de origem, dependendo do URI do pedido, permite a execução de ações específicas. O método POST poderia alterar o conteúdo de um recurso e, portanto, não pode ser visto como um método seguro. Como os efeitos colaterais de um conjunto de pedidos POST idênticos poderia diferir, POST não é considerado um método coerente.

Para modificar um recurso existente, o usuário precisa ter a autenticação necessária. Nem todos os usuários podem ter direitos de acesso para alterar um recurso. No entanto, se ele tiver permissão, o servidor de origem aceitará uma nova versão do recurso a partir de cliente. Outro uso para o método POST é apanhar o corpo de pedido e usá-lo como entrada para um programa identificado pelo URI do pedido. É possível que nenhum recurso seja afetado ou criado como resultado de um pedido POST. Nesse caso, o corpo do pedido é tratado como entrada para um programa que simplesmente consome os dados.

PUT

O método PUT é semelhante a POST porque o processamento do método normalmente resultaria em uma versão diferente do recurso sendo identificada pelo URI do pedido. Se o URI do pedido não existe, ele seria criado. Se já existe, ele é modificado. No entanto, o recurso identificado somente no método PUT modificaria como resultado o pedido. O método PUT não é um método seguro. Ele é coerente porque uma sequência de pedidos PUT idênticos teria que incluir a mesma entidade e, portanto, os efeitos colaterais seriam iguais em cada caso.

DELETE

O método DELETE é usado para se excluir o recurso identificado pelo URI do pedido. Esse método é fornecido como uma conveniência para a exclusão de recursos remotamente. No entanto, pela natureza da ação, os servidores de origem possuem controle sobre se e quando a ação solicitada será realmente realizada. O servidor pode enviar uma resposta de sucesso sem realmente excluir o recurso.

Existem dois tipos de respostas de sucesso: uma que indica aceitação do pedido para processamento posterior e uma que é a realização real do pedido. Essa flexibilidade é fundamental para os servidores de origem decidirem quando e como agendarão a ação, não sendo forçados a manter a conexão com o cliente aberta até que a ação tenha sido realmente completada. O método DELETE não é um método seguro, porém, assim como o PUT, ele é coerente.

2.5 Banco de Dados

Banco de dados ou base de dados são conjuntos de dados com uma estrutura regular que tem como objetivo organizar informações. Um banco de dados agrupa informações utilizadas para um mesmo fim de forma que possam representar coleções de informações que se relacionam entre si de forma a criar um sentido. Entende-se por dados, todas as informações que podem ser armazenadas e que apresentam algum significado dentro do contexto ao qual se aplicam.

Segundo MACHADO (2008) citado por ISSA (2011), para que um conjunto de dados seja considerado como um banco de dados é necessário que tenha significado no contexto. Dessa forma, uma disposição desordenada de dados não pode ser considerada um banco de dados. Além disso, sua finalidade é o armazenar dados para um propósito específico, ou seja, deve possuir um conjunto pré definido de usuários e aplicações. Por fim, ele representa um aspecto do mundo real também conhecido como minimundo, onde qualquer alteração nesse minimundo irá refletir nos dados armazenados no banco.

Um banco de dados informatizado é usualmente mantido e acessado por meio de um Sistema Gerenciador de banco de dados (SGBD). Esses sistemas são projetados para lidar com grandes quantidades de informações e dentre as suas funções, se destacam a definição de estruturas de armazenamento de dados, mecanismos de controle de acesso aos dados, manipulação de informações, compartilhamento de dados entre diversos usuários e certificação de integridade dos dados. A Figura 2.15 mostra um esquema genérico de um sistema de base de dados. Tal sistema é composto basicamente por uma base de dados e *softwares* para gerenciamento.

2.5.1 Modelos de banco de dados

O modelo de dados de um banco de dados é a forma de descrever os dados, seus relacionamentos, sua semântica e restrições de consistência. Um modelo descreve o projeto do banco nas formas física, lógica e de visualização. Dentre os modelos existentes destacam-se o modelo relacional, o modelo entidade/relacionamento e modelo orientado a objetos. Dentre esses, o mais utilizado para representar e armazenar dados em um SGBD é o modelo relacional, onde as estruturas têm a forma de tabelas, compostas por linhas e colunas (SILBERSCHATZ; KORTH; SUDARSHAN, 2006) citado por ISSA (2011).

O modelo relacional utiliza tabelas para representar os dados e os relacionamentos existentes entre as várias instâncias destas tabelas. Cada tabela possui diversas colunas, que são atributos desta entidade que serão armazenados. O modelo entidade/relacionamento fundamenta-se na filosofia de que o banco de dados é baseado na percepção do mundo real, que é formado por vários objetos básicos, ou seja, as entidades e os relacionamentos

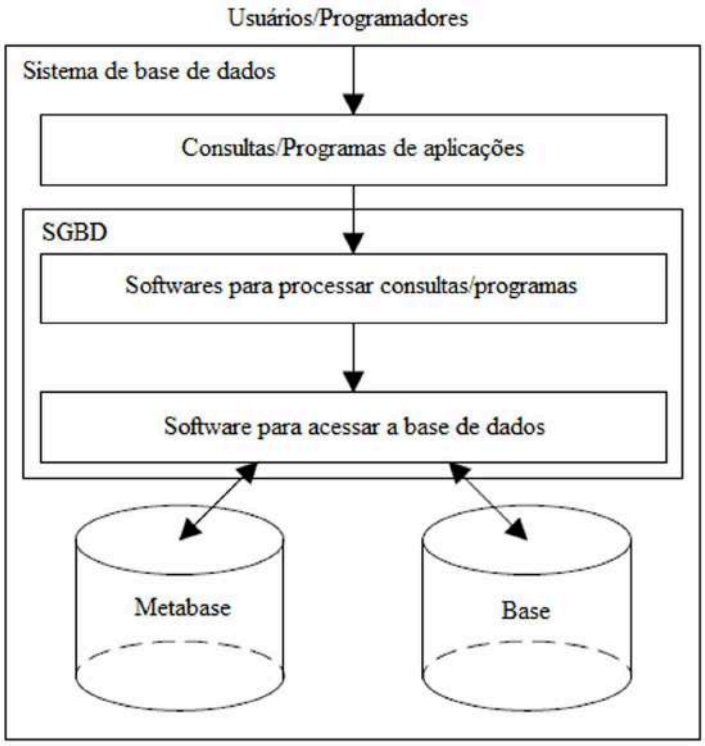

Figura 2.15: Sistema de banco de dados.
Fonte: (TAKAY; ITALIANO; FERREIRA, 2005).

existentes entre estas entidades. Já o modelo orientado a objetos pode ser visto como um modelo entidade/relacionamento mais avançado, onde é acrescentado funções de encapsulamento, métodos ou funções, e identidade de objeto.

2.5.2 Bancos de dados relacionais

Os bancos de dados relacionais utilizam a metodologia de prover um meio de descrever o dado apenas com sua estrutura natural, ou seja, sem impor estruturas adicionais com propósitos de otimização computacional. Outra vantagem desta abordagem é a boa fundamentação em consistência de relacionamentos e redundância de dados.

Um banco de dados relacional é formado por várias tabelas, cada uma com um nome único atribuído a ela. Os cabeçalhos segundo a normalização dos modelos relacionais são chamados de atributos. Para cada atributo, há um conjunto de valores, denominados domínio, que são armazenados nessas colunas.

A Tabela 2.2 ilustra uma tabela de um banco de dados relacional referente ao registro de alunos. Para o atributo *Nome de Aluno*, por exemplo, o domínio do atributo é o conjunto de todos os alunos registrados pela instituição. O mesmo ocorre com o atributo curso, cujo domínio é o nome dos cursos ofertados e assim por diante. Qualquer linha

desta tabela, precisa consistir em um conjunto de quatro elementos (de a_1 até a_4) que se referem aos atributos. Para cada atributo portanto, existe um domínio (de D_1 a D_n) que identifica o tipo de dados registrado. Assim, o atributo a_1, está no domínio D_1, que é formado pelos números de matrícula registrados. Portanto, todo registro de alunos conterá penas subconjuntos dos conjuntos possíveis para cada atributo da tabela.

Tabela 2.2: Tabela relacional Alunos.

Matrícula	**Nome do Aluno**	Curso	Telefone
14001	Aluno A	Curso 1	$3212-0001$
14002	Aluno B	Curso 2	$3212-0002$
14003	Aluno C	Curso 3	$3212-0003$
14004	Aluno D	Curso 4	$3212-0004$
14005	Aluno E	Curso 5	$3212-0005$

Fonte: (ISSA, 2011).

Em geral, uma tabela de n atributos como mostra a Equação (2.1) é um sub conjunoto de:

$$D_1 \times D_2 \times \cdots \times D_{n-1} \times D_n \tag{2.1}$$

É exigido que, para todos os registros da tabela, os domínios dos atributos sejam atômicos. Para tanto, ele deve ser formado apenas por elementos indivisíveis. Por exemplo, o conjunto dos números inteiros é um domínio atômico, porém o conjunto de todos os conjuntos de números inteiros não é atômico, pois o subconjunto é composto por diversos números inteiros. Esta exigência é necessária porque apesar de *Nome do Aluno* e *Curso* serem ambos armazenados como conjunto de caracteres no esquema físico (armazenamento em arquivo), no nível lógico (onde são representadas as tabelas) a lógica impõe que o *Nome do Aluno* e o *nome do curso* sejam de conjuntos distintos.

Em resumo, e como citado por MEIRA (2011), de acordo com a arquitetura ANSI/SPARC os bancos de dados relacionais consistem em três componentes:

1. Uma coleção de estruturas de dados, formalmente chamadas de relações, ou informalmente tabelas, compondo o nível conceitual;

2. Uma coleção dos operadores, a álgebra e o cálculo relacionais, que constituem a base da linguagem SQL;

3. Uma coleção de restrições da integridade, definindo o conjunto consistente de estados de base de dados e de alterações de estados.

De acordo com o princípio de informação, toda informação tem de ser representada como dados; qualquer tipo de atributo representa relações entre conjuntos de dados. Nos

bancos de dados relacionais os relacionamentos entre as tabelas não são codificados explicitamente na sua definição. Em vez disso, se fazem implicitamente pela presença de atributos chave. As bases de dados relacionais permitem aos utilizadores (incluindo programadores) escreverem consultas (*queries*), reorganizando e utilizando os dados de forma flexível e não necessariamente antecipada pelos projetistas originais (MEIRA, 2011).

Chaves primárias e chaves estrangeiras

No modelo relacional a única forma de relacionar dados que existem em uma tabela com dados que existem em outra tabela é através de atributos comuns. Assim, vão existir atributos especiais (chaves estrangeiras) que servem para fazer a ligação com outras tabelas, onde esses mesmos atributos são usados para identificar, unicamente, cada uma das linhas (chaves primárias). Dessa forma, uma chave primária identifica cada tupla (linha) e a chave secundária é um atributo ou conjunto de atributos de uma relação, que é chave primária em outra relação.

É preciso ter um modo de definir como as linhas dentro de uma tabela são identificadas e distinguidas umas das outras. Essa identificação é feita utilizando os atributos. Sendo assim, os valores de um registro precisam ser criados de modo que possam identificar unicamente aquela linha da tabela. Em outras palavras, nenhuma linha em uma tabela pode ter todos os valores para todos seus atributos.

O termo *chave primária* é utilizado para identificar o atributo que foi escolhido pelo projetista do banco de dados para identificar unicamente os registros que são armazenados em uma determinada tabela. Uma chave é propriedade de toda a tabela, e não de um registro em específico. Nenhum registro pode ter o mesmo valor no atributo chave primária de uma determinada tabela do banco de dados.

Uma chave primária (atributo identificador) é uma coluna ou um grupo de colunas que assegura a unicidade das linhas dentro de uma tabela. Uma chave primária que tenha mais de uma coluna é chamada de chave primária composta. Chaves primárias são geralmente indicadas pela sigla PK do inglês *primary key*.

Uma chave estrangeira é uma coluna ou grupo de colunas que pode ou não ser chave primária da tabela em questão, mas, com certeza é chave primária de outra tabela. Uma chave estrangeira formada por mais de uma coluna é chamada de chave estrangeira composta. Chaves estrangeiras são indicadas pela sigla FK do inglês *Foreign Key*.

Um esquema de tabela $T1$ pode ter entre seus atributos a chave primária de outra tabela $T2$. Este atributo é chamado de chave estrangeira de $T1$, referenciando $T2$. Por exemplo, no esquema da Figura 2.16, o atributo *nome_agência* na tabela *conta* é uma chave estrangeira desta tabela, que referencia a tabela agência. Em qualquer conjunto de registros do banco de dados, para cada linha existente na tabela *conta*, é necessário que

nome_agência possua uma referência a um registro válido da tabela *agência*. Na Figura 2.16 é apresentado um esquema de uma instituição bancária. As chaves primárias de cada entidade são apresentadas antes dos outros atributos (parte destacada em cinza).

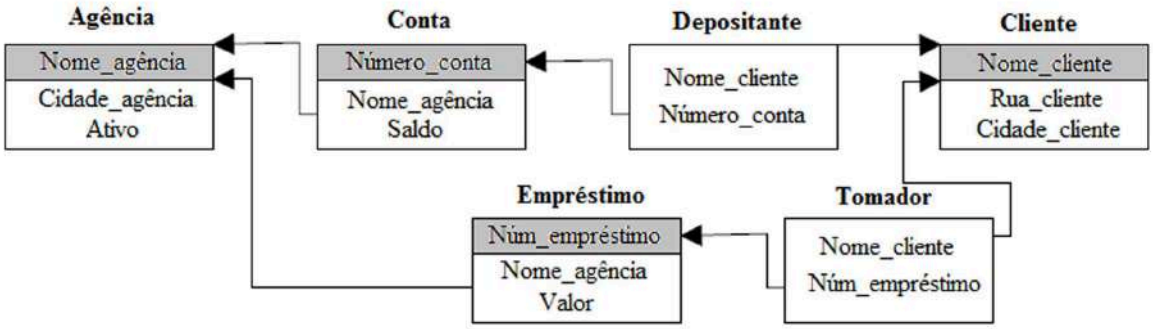

Figura 2.16: Banco de dados relacional.
Fonte: (ISSA, 2011).

Normalização

Normalização é o processo que permite a simplificação da estrutura de um banco de dados de modo que este se apresente em um ótimo estado. Esse processo é bastante simples e tem por principal objetivo remover grupos repetidos de informações. Normalmente, ao normalizar aumenta-se o número de tabelas e o número total de campos. A teoria da normalização é baseada no conceito de formas normais, que são regras aplicada as estruturas das tabelas com o objetivo de minimizar ou eliminar a redundância de informações. Embora existam seis formas normais, normalmente se considera que um esquema de banco de dados está em um bom nível quando ele se encontra na terceira forma normal.

1ª Forma normal

Uma relação está na primeira forma normal quando:

- Não contém atributos multivalorados (grupos de atributos)
- Não contém grupos repetitivos

2ª Forma normal

Uma relação está na segunda forma normal quando:

- Está na primeira forma normal.
- Todos os atributos não chave dependem da totalidade da chave.

3ª Forma normal

Uma relação está na terceita forma normal quando:

- Está na segunda forma normal.
- Todos os seus atributos não chave não são identificados por um outro também não chave.

Uma propriedade indesejável que um projeto de banco de dados possa ter é a repetição desnecessária de informações. Considerando, por exemplo que o Aluno A, do esquema da Tabela 2.2, se matricule para um novo curso na instituição de ensino e uma nova matrícula seja gerada para registro dele no curso, é desejável que não haja repetição de informações no projeto do banco de dados, porque isto aumenta o custo de armazenamento e dificulta a tarefa de atualização dos dados. Supondo que o número do telefone deste aluno seja alterado, a alteração deve refletir nas duas linhas, sendo mais custosa nesta tabela do que em uma tabela normalizada. Aplicando a normalização a este projeto, o atributo curso fica independente do registro do aluno, evitando a repetição de informações desnecessárias. Nas Tabelas de 2.3 a 2.5 observa-se que os dados representados são os mesmos (com adição do registro do novo curso do aluno 14001), porém sem a duplicação desnecessária de dados.

Tabela 2.3: Tabela Alunos.

Matrícula	**Nome do Aluno**	Telefone
14001	Aluno A	$3212-0001$
14002	Aluno B	$3212-0002$
14003	Aluno C	$3212-0003$
14004	Aluno D	$3212-0004$
14005	Aluno E	$3212-0005$

Fonte: (ISSA, 2011).

Tabela 2.4: Tabela Curso.

Curso id	**Nome do Curso**
1	Curso 1
2	Curso 2
3	Curso 3
4	Curso 4
5	Curso 5

Fonte: (ISSA, 2011).

Tabela 2.5: Tabela relacionamento Aluno × Curso.

Matrícula	**Curso id**
14001	Curso 2
14002	Curso 2
14003	Curso 3
14004	Curso 4
14005	Curso 1

Fonte: (ISSA, 2011).

2.5.3 Bancos de dados não-relacionais

NoSQL é um termo utilizado para identificar bancos de dados de código aberto que não possuíam interface SQL. Bancos de dados não relacionais, incluindo hierárquicos, gráficos e orientados a objetos estão em uso desde os anos 60. Entretanto, novos tipos de bancos de dados não relacionais estão sendo desenvolvidos, como os bancos colunares. Diferentes bancos de dados NoSQL apresentam diferentes abordagens, porém o que todos eles possuem em comum é que não são relacionais. A principal vantagem de um tipo de dados não relacional é que, diferente da abordagem tradicional, eles lidam bem com dados desestruturados, tais como *e-mails*, dados multimídia e dados provenientes de mídias sociais.

O NoSQL lida bem com distribuição horizontal, ou seja, consegue ser distribuído entre várias estações para armazenamento de maior quantidades de dados. Estes novos servidores não precisam ser de alto desempenho. A nova abordagem, é principalmente utilizada por empresas da Web, com quantidades gigantescas de dados e infra-estrutura, como *Google* e *Amazon*.

O armazenamento de dados destes bancos evoluiu conforme a necessidade da aplicação, com diferentes modelos de dados e capacidades. Os elementos básicos podem ser considerados “objetos”, “documentos” ou “registros”. Apesar dessas variações, algumas características podem ser identificadas:

- A não existência de um esquema pré definido. Os dados podem possuir qualquer número de atributo de qualquer tipo. A falta de um esquema global facilita o desenvolvimento de um banco de alta disponibilidade, pois não é necessário desativar o banco para efetuar alterações no esquema.

- Estes sistemas possuem uma API de buscas simples, o que, diferente dos bancos relacionais, encoraja buscas simples, sem necessidade de grandes uniões de tabelas.

- A escalabilidade proporcionada pelos nós da rede. Diferente dos bancos de dados

relacionais, que utilizam as propriedades ACID, bancos NoSQL promovem uma "consistência eventual", ou garante a consistência dentro de um único objeto (ou documento ou registro).

- Esta metodologia também permite alta disponibilidade, que é necessária para fazer com a que a escalabilidade entre diversas máquinas seja utilizável. Isto é alcançado através da identificação de falhas e de recuperação utilizando cópias de nós espelhos.

2.5.4 A linguagem SQL

O SQL é uma linguagem de programação para a criação de banco de dados, tabelas e atributos em um SGBD. Essa linguagem é compostas de comandos de manipulação, definição e controle de dados. A SQL estabeleceu-se como linguagem padrão de banco de dados relacionais.

Uma das principais características da linguagem SQL é a sua capacidade de gerenciar índices, sem a necessidade de controle individualizado de índice corrente, algo muito comum nas linguagens de manipulação de dados do tipo registro a registro. Outra característica muito importante disponível em SQL é sua capacidade de construção de visões, que são formas de visualizarmos os dados na forma de listagens independente das tabelas e organização lógica dos dados.

A SQL apresenta uma série de comandos que permitem a definição dos dados, chamada de DDL (*Data Definition Language*), composta entre outros pelos comandos *Create*, que é destinado a criação do banco de dados, das Tabelas que o compõe, além das relações existentes entre as tabelas. Como exemplo de comandos da classe DDL temos os comandos *Create*, *Alter* e *Drop*.

Os comandos da série DML (*Data Manipulation Language*), destinados a consultas, inserções, exclusões e alterações em um ou mais registros de uma ou mais tabelas de maneira simultânea. Como exemplo de comandos da classe DML podem ser citadas os comandos *Select*, *Insert*, *Update* e *Delete*. Uma subclasse de comandos DML, a DCL (*Data Control Language*), dispõe de comandos de controle como *Grant* e *Revoke*.

Comandos DDL

A maioria dos SGBDs (inclusive o SQL Server e o MySQL) disponibiliza de ferramentas gráficas que permitem a criação de bancos de dados, mas é possível criar o próprio banco de dados a partir de um comando SQL.

A função CREATE DATABASE permite criar um banco de dados em um SGBD escolhido, ao passo que a função DROP DATABASE permite remover um determinado banco de dados, apagando todas as tabelas e estruturas associadas e, consequentemente, todos os dados existentes nelas.

```
CREATE DATABASE nome_do_banco_de_dados
```

```
DROP DATABASE nome_do_banco_de_dados
```

O comando CREATE TABLE é o principal comando DDL da linguagem SQL. A criação de tabelas é realizada em SQL utilizando este comando. Para execução do comando CREATE TABLE é necessário indicar qual o nome da tabela e, para cada uma das colunas, o nome da coluna e o tipo de dados. No entanto, podem ser indicadas as características próprias de cada uma das colunas, tais como: Que valores podem admitir? Qual o valor padrão? O campo representa um atributo identificador (chave primária)?

```
CREATE TABLE nome_da_tabela(
campo1 Tipo,
campo2 Tipo,
campo3 Tipo)
```

Para que uma coluna não admita valores nulos, usamos a cláusula NOT NULL. Caso um valor não seja inserido em uma coluna o valor padrão (*default*) armazenado nela é NULL. No entanto, é possível associar um outro valor default através da cláusula DEFAULT. Na criação de tabelas, também podem ser definidas o uso de restrições. As restrições são regras a que os valores de uma ou mais colunas devem obedecer. A utilização de restrições é a única garantia que temos de que os dados existentes nas colunas estão de acordo com as regras especificadas no projeto do banco de dados.

O comando ALTER TABLE permite alterar a estrutura de uma tabela. Através deste comando é possível adicionar uma nova coluna, modificar uma coluna já existente ou eliminar uma coluna.

```
ALTER TABLE Nome_Da_Tabela ADD Nome_Coluna Tipo_Coluna
```

```
ALTER TABLE Nome_Da_Tabela MODIFY Nome_Coluna Tipo_Coluna
```

```
ALTER TABLE Nome_Da_Tabela DROP Nome_Coluna Tipo_Coluna
```

Comandos DML

Os comandos de manipulação de dados em SQL são representados por: INSERT, permite a inclusão de novas linhas nas tabelas; UPDATE, que altera os valores de dados já cadastrados; DELETE, que remove dados já cadastrados e SELECT, usado para consultar o banco de dados e retornar dados que satisfazem a determinada expressão em um comando. A sintaxe desses comandos são mostradas a seguir.

```
INSERT INTO NOME DA TABELA(coluna1,coluna2,coluna3) VALUES (valor1, valor2, valor3)
```

O comando UPDATE é usado para mudar valores de linhas de dados que já foram cadastrados anteriormente e que obedecem a determinados critérios, especificados em condições. Este comando pode alterar mais de uma linha ao mesmo tempo, caso mais de uma linha obedeça a determinada condição. As condições podem também ser representadas utilizando os operadores: AND,OR e NOT.

O comando UPDATE, contém a cláusula WHERE, de forma a restringir o conjunto dos registros que serão processados pelo comando. Se não for colocada a cláusula WHERE no comando UPDATE, as alterações serão realizadas em todos os registros da tabela.

```
UPDATE NOME DA TABELA
SET coluna1 = valor1, coluna2 = valor2
WHERE condições
```

O comando DELETE é usado para remover linhas de uma tabela. Este comando pode remover mais de uma linha ao mesmo tempo, caso mais de uma linha obedeça a uma certa condição. As condições podem ser representadas utilizando os operadores AND, OR e NOT. O comando DELETE, contém a cláusula WHERE, de forma a restringir o conjunto dos registros que serão processados pelo comando. Se não for colocada a cláusula WHERE no comando DELETE, serão apagados todos os registros de uma tabela.

```
DELETE FROM NOME DA TABELA
WHERE <condições>
```

O comando SELECT é usado para consultar o banco de dados e retornar dados recuperados que satisfazem a determinada condição expressa no comando. o símbolo $*$ na cláusula SELECT indica que deverá ser selecionado todos os campos de uma tabela.

```
SELECT <lista de atributos>
FROM NOME DA TABELA
WHERE <condições>
```

Comandos DCL

Para controlar o acesso à informação, podem ser criados um conjunto de regras para acesso ao SGDB, bem como a conceção/eliminação de uma determinada permissão ou privilégio a um usuário. Os comandos de SQL para definir mecanismos de segurança são conhecidos como linguagem de controle de dados (*Data Control Language* - DCL).

Os bancos de dados permitem a criação e a manutenção de usuários através de comandos da linguagem SQL. Para criar, alterar ou remover um usuário e senha para acesso a dados podem ser utilizadas as funções CREATE, ALTER e DROP.

```
CREATE USER Nome_do_usuario
IDENTIFIED BY Senha_do_usuario

ALTER USER Nome_do_usuario
IDENTIFIED BY Senha_do_usuario

DROP USER Nome_do_usuario
```

Além da segurança de dados em nível de usuário, podem ser atribuídos privilégios diferentes a cada um dos usuários do SGBD. Para isso é usado o comando GRANT. O comando REVOKE permite retirar os privilégios concedidos através do comando GRANT.

```
GRANT Privilegio ON Objeto TO Usuario

REVOKE Privilegio ON Objeto TO Usuario
```

Capítulo 3

Desenvolvimento do Sistema

Neste capítulo, serão abordadas as questões referentes ao desenvolvimento do WebLab. Inicialmente serão descritos na seção 3.1 os requisitos de projeto a serem atendidos em um nível mais alto de abstração, ou seja, tais requisitos determinam o que se deseja obter ao final do projeto na perspectiva de quem irá usar o WebLab. Em seguida, na seção 3.2, será apresentado o modelo definido para o sistema a fim de se ter uma visão geral do projeto, facilitando o entendimento sobre as interações entre as partes que o compõe. Posteriormente, cada parte dessa estrutura será descrita separadamente. Na Seção 3.3, será apresentada a planta do sistema de aquecimento de ar abordando as condições em que ela se encontrava antes do início do projeto, assim como as modificações e melhorias realizadas. Será apresentado também um diagrama esquemático dando uma visão geral das entradas e saídas de cada módulo descrevendo em seguida os circuitos e dispositivos que compõem cada um deles: alimentação; potência; aquisição de dados e condicionamento dos sinais. Na Seção 3.4 será descrito o programa desenvolvido em LabVIEW® para controlar o sistema de aquecimento de ar. Serão apresentados os padrões de projeto utilizados e posteriormente, serão detalhados cada módulo do programa, criados a partir das funções definidas como requisitos de projeto. Por fim, a Seção 3.6 descreve o desenvolvimento da interface de usuário remoto. Nela, serão apresentados os recursos utilizados para criação das páginas Web, assim como as questões referentes ao banco de dados, ao servidor Web e ao servidor de *e-mail*.

3.1 Requisitos de Projeto do Ponto de Vista do Usuário

Todos os recursos necessários para o desenvolvimento do WebLab foram definidos a partir da idéia que se tinha sobre o produto final. No desenvolvimento de projetos de *software* isso ocorre, por exemplo, quando o cliente contrata um desenvolvedor para criar uma determinada aplicação. Dessa forma, entender o que o cliente deseja é o primeiro

passo para que o projetista consiga definir como atender os requisitos especificados por ele. O produto final consiste então em um conjunto de requisitos de projeto, que no caso do WebLab, podem ser representados, do ponto de vista do usuário, como no esquema da Figura 3.1.

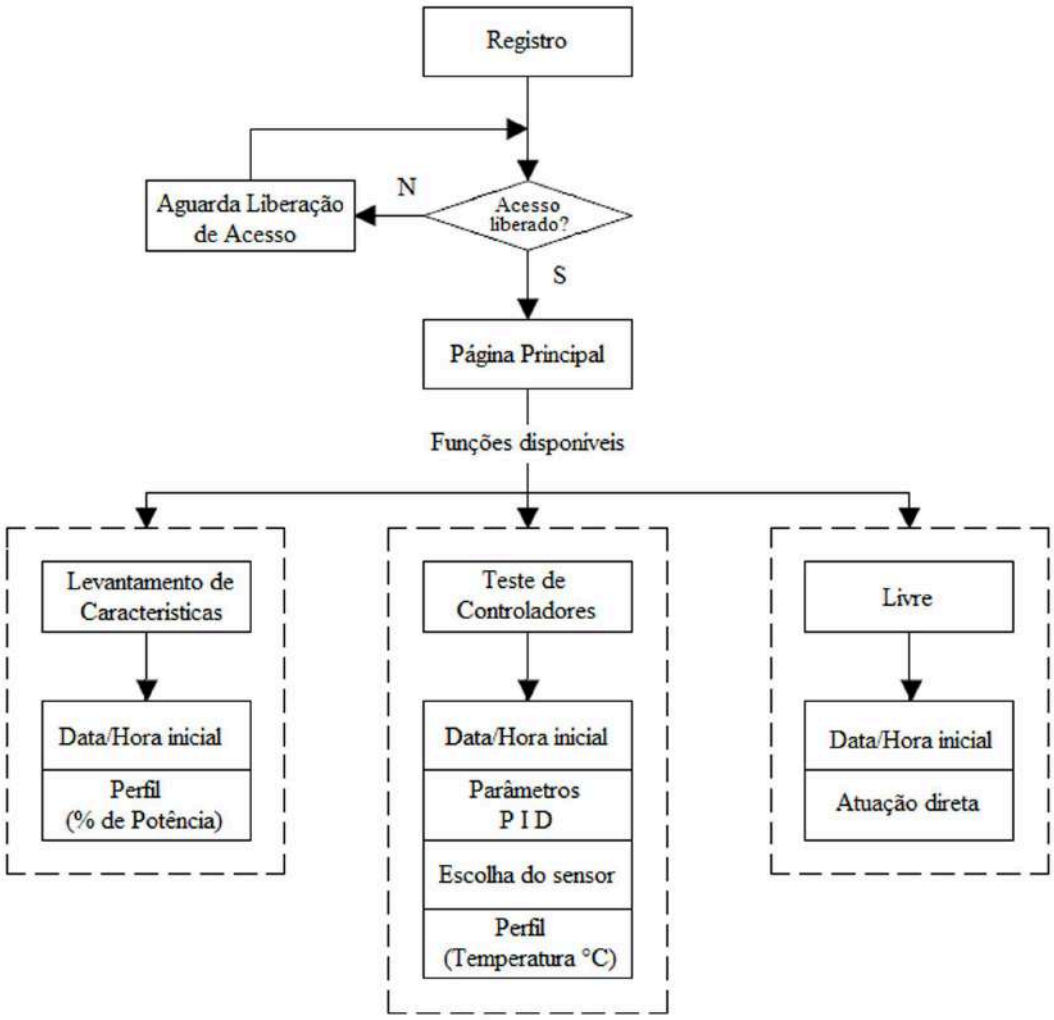

Figura 3.1: Requisitos de projeto do ponto de vista do cliente.

Primeiramente o usuário acessa a página do WebLab e realiza o seu registro no sistema. Nesse registro, ele informa o seu nome completo; endereço de *e-mail*; nome de usuário; número de *matrícula* e senha como apresentado na Figura 3.2. Tais informações são utilizadas pelo administrador do sistema para controlar quem pode ou não acessá-lo. Em seguida, o usuário aguarda que o administrador libere o seu acesso, que pode ser editado como **aluno**, **professor** ou **administrador**.

Após ter acesso liberado, o usuário pode escolher entre as funções *levantamento de características*; *teste de controladores* ou **Livre**. O administrador pode acessar qualquer dessas três ações e editar o acesso de professores e alunos. Já o professor, tem acesso às duas primeiras ações e pode editar o acesso de alunos. Finalmente os alunos, tem acesso

Figura 3.2: Página de registro de usuários.

às duas primeiras ações e não edita o acesso de ninguém. A Figura 3.3 mostra a página de edição de usuários.

Figura 3.3: Página de edição de usuários.

Levantamento de características

A função *levantamento de características* permite que o usuário agende um experimento para identificar o comportamento do sistema de aquecimento de ar a partir de entradas conhecidas. Nessa função, o usuário desenha perfis de teste para serem aplicados aos atuadores, nesse caso, os perfis consistem em porcentagens de potência a serem aplicados nas resistências da planta (o sistema permite a criação de degraus e/ou rampas).

A modelagem do sistema, dependerá da configuração da planta, ou seja, como o sistema possui quatro atuadores e onze sensores de temperatura, além de um *damper* para restrição do fluxo de ar, o usuário pode utilizar, por exemplo, um atuador e uma resistência em uma configuração de única entrada e única saída (*Single Input Single Output* - SISO), ou pode combinar mais de um atuador e mais de um sensor em uma configuração de múltiplas entradas e múltiplas saídas (*Multiple Input Multiple Output* - MIMO). Na data e hora programada, o sistema executa o experimento e envia como resposta ao cliente com todos os dados necessários à identificação.

A Figura 3.4 ilustra a criação de um perfil típico para levantamento de características, aplicado à Resistência 1. Como se trata de um sistema térmico é utilizado uma série de degraus ascendentes e descendentes para identificação da dinâmica de aquecimento e resfriamento da resistência. Não há necessidade de se escolher um sensor específico pois quando os dados forem enviados ao cliente, serão enviados os valores de todos os sensores e o próprio usuário escolhe os que lhe interessa. Em relação aos atuadores, no caso de um sistema SISO, basta desenhar o perfil no que é de interesse e deixar os outros em zero pois não serão utilizados. Da mesma forma, os valores de todas as resistências serão enviadas ao usuário, que deverá filtrar a(s) de interesse.

Teste de controladores

Feita a identificação do sistema sob as condições escolhidas pelo usuário, ele pode modelar o sistema, projetar controladores e em seguida, agendar um novo experimento no WebLab para aplicá-los ao sistema de aquecimento de ar. Nessa função, o usuário desenha as entradas de referência, que agora são os valores de temperatura a serem alcançados, informa os parâmetros do controlador projetado e o sensor de temperatura a ser monitorado. Neste caso, informar qual o sensor está sendo considerado é importante porque o controlador precisa dessa informação para calcular o erro.

A Figura 3.5 ilustra a aplicação de uma entrada de referência e uma sequência de degraus para analisar um controlador PI na Resistência 2. Essa sequência pode ser observada no gráfico à direita. Já o gráfico da esquerda mostra um perfil aplicado à Resistência 1 que pode ser utilizado para gerar uma perturbação no sistema. Novamente, os parâmetros dependem do tipo de controlador utilizado. Atualmente, estão implementados no

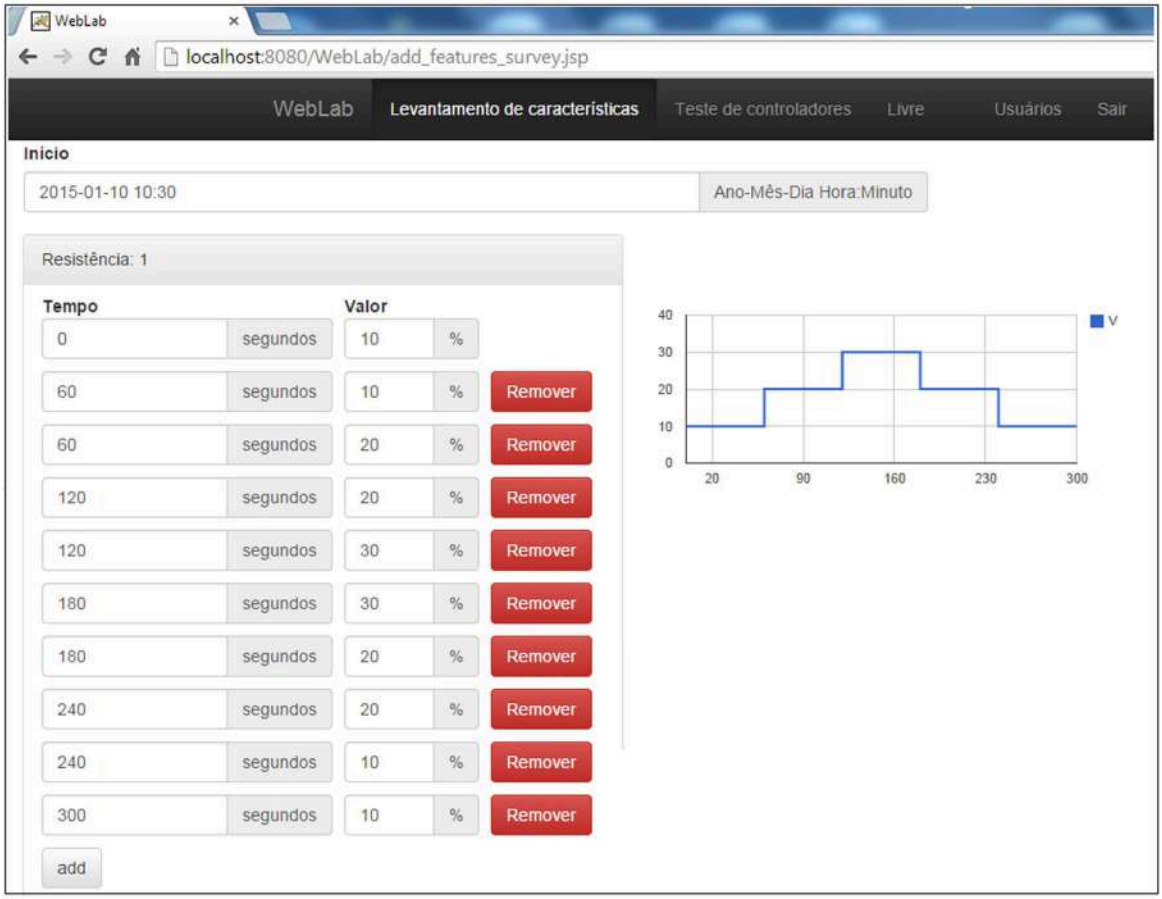

Figura 3.4: Perfil em degraus para levantamento de características.

WebLab controladores PID em paralelo. Esses controladores são independentes para cada atuador. Assim, o usuário pode projetar um controlador para cada resistência. Da mesma forma, quando chegar a data e hora programada, o sistema executa o experimento e no final, envia ao cliente todos os dados coletados durante o ensaio do experimento.

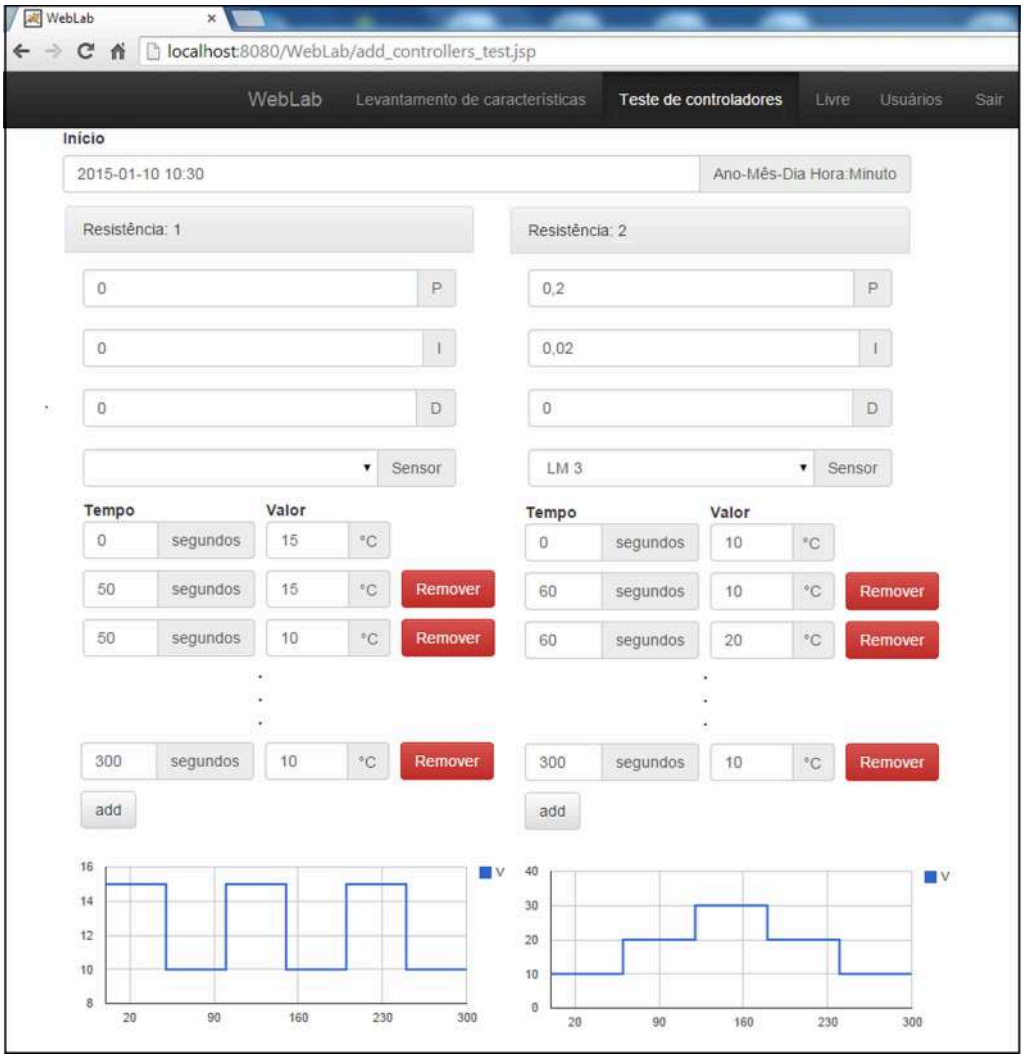

Figura 3.5: Perfil em degraus para teste de controladores.

Livre

Essa função permite que o usuário atue diretamente na planta com o objetivo de verificar se está tudo funcionando como se espera. O usuário informa a data de início do experimento e a duração do teste em minutos conforme janela de interface com o usuário, apresentada na imagem da Figura 3.6. Em seguida, ele pode observar se os comandos estão sendo aplicados nas resistências atuando diretamente. A temperatura é atualizada a cada 1 segundo e através de sua variação é possível verificar se as resistências estão funcionando. A Figura 3.7 apresenta a interface com o usuário remoto, vale observar que a sigla LM refere-se aos sensores $LM35$, enquanto que a sigla TP refere-se aos termopares.

Figura 3.6: Agendamento para função livre.

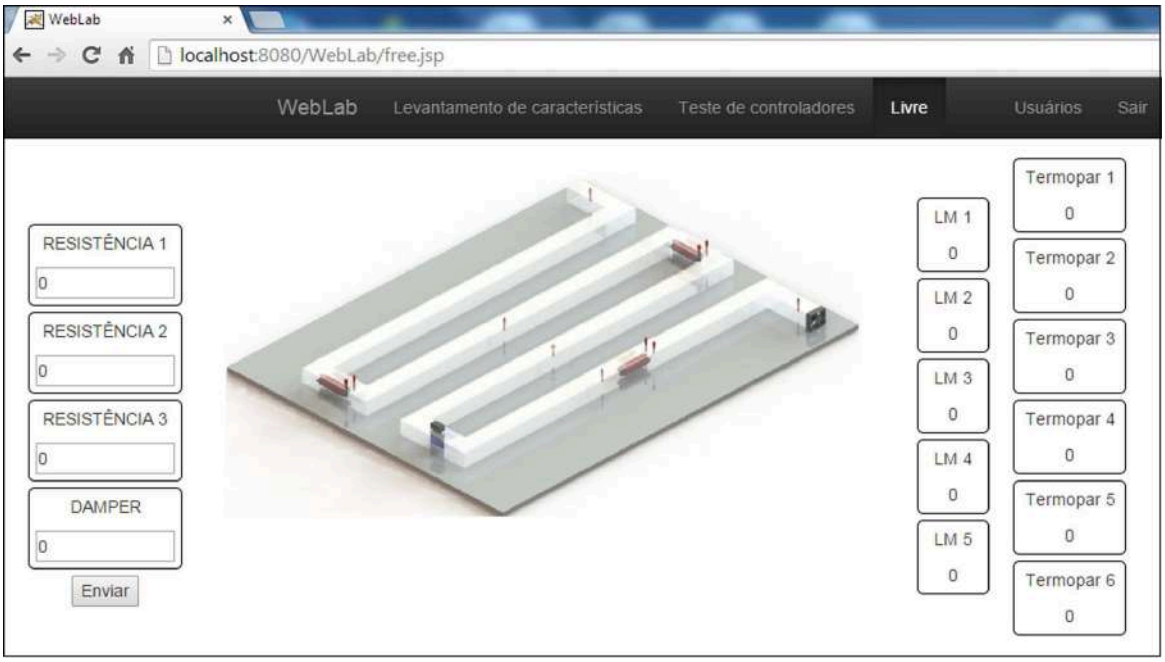

Figura 3.7: Interface de usuário para função livre.

3.2 Definição do Modelo do WebLab

O esboço dos requisitos de projeto do ponto de vista do usuário ou cliente, como aquele ilustrado na Figura 3.1, permite avançar para a parte técnica do desenvolvimento. Nesse momento, preocupa-se então com os recursos a serem empregados para alcançar o objetivo proposto, ou seja, o projeto passa a ser visto em um nível um pouco mais baixo. Pensando-se então nas funções descritas anteriormente, verificou-se a necessidade de se utilizar uma arquitetura que permitisse acessar o sistema de aquecimento de ar remotamente. A escolha da arquitetura deveria no entanto possibilitar o uso do LabVIEW®, pois esse já havia sido difinido como principal plataforma de desenvolvimento devido à sua disponibilidade no laboratório de sinais e sistemas do CEFET-MG/Divinópolis.

A primeira possibilidade, seria utilizar uma ferramenta do LabVIEW® denominada *Web Publishing Tool*. Tal ferramenta permite a publicação do painel frontal do programa desenvolvido em LabVIEW® sem a necessidade de criação de outras aplicações através de outras linguagens e programas. No entanto, para utilizar esse recurso é necessário a instalação de *plug-ins* adicionais do lado do cliente para habilitar o *Web browser* além de um programa denominado *LabVIEW Run Time Engine* para possibilitar que o cliente visualize o painel frontal. Essa ferramenta foi testada e dentre outras desvantagens, verificou-se que através dela, apenas o primeiro usuário teria acesso aos controles do aplicativo em LabVIEW® sendo que outros usuários não poderiam atuar no sistema.

A segunda possibilidade, seria utilizar outro recurso do LabVIEW® denominado *LabVIEW Web Service*. Um *Web service* no LabVIEW® consiste em um conjunto de instrumentos virtuais que permitem a invocação de um método em um *target* remoto usando protocolos padrão baseados na Web. Um cliente envia uma solicitação para um servidor remoto, que a processa e retorna uma resposta, que é então interpretada e apresentada pela aplicação cliente. Para utilizar essa ferramenta do LabVIEW®, basta configurar a aplicação desenvolvida para que essa seja vista como um conjunto de métodos a serem invocados por uma requisição do cliente, feita através de uma URL em um *browser* de internet.

O LabVIEW® conta com um servidor próprio e a estrutura utilizada é baseada na arquitetura RESTful que fica invisível ao desenvolvedor. A configuração consiste basicamente na definição dos métodos dentro da aplicação principal (*target*). Cada método definido terá uma URL associada a ele que é usada pelo cliente para fazer uma solicitação utilizando HTTP padrão. O servidor então comunica-se com a aplicação em LabVIEW® que por sua vez realiza as ações necessárias e retorna a resposta ao servidor que a retransmite para o cliente.

Ao contrário da ferramenta *Web Publishing Tool*, o LabVIEW® *Web Service* não necessita que nenhum programa adicional seja instalado do lado do cliente, sendo possível

a utilização de qualquer tecnologia de cliente baseada na Web, incluindo por exemplo, linguagens comuns como HTML e *Javascript*. No entanto, como as configurações do *LabVIEW Web Service* são padronizadas, as URLs criadas também são, e dessa forma, o cliente deve respeitar a sua estrutura conhecida como URL amigável. Essa característica tornaria o desenvolvimento do programa cliente muito restrita. Além disso, a utilização do servidor proprietário limitaria ao uso exclusivo do LabVIEW® como único programa para desenvolvimento dos métodos.

Com base nos estudos realizados em trabalhos semelhantes, considerando as possibilidades apresentadas anteriormente e ainda as vantagens do LabVIEW® em relação à sua flexibilidade e facilidade no desenvolvimento de aplicações para aquisição de dados e conectividade, foi resolvido mantê-lo como plataforma para criação dos métodos. Porém, decidiu-se utilizar um modelo baseado na arquitetura cliente-servidor possibilitado através de recursos para desenvolvimento livre (*Open Source*). Chegou-se então à estrutura da Figura 3.8. As etapas de desenvolvimento que resultaram em tal estrutura são descritas nas próximas seções. A descrição será feita de baixo para cima, iniciando com a planta do sistema de aquecimento de ar e finalizando com o programa cliente.

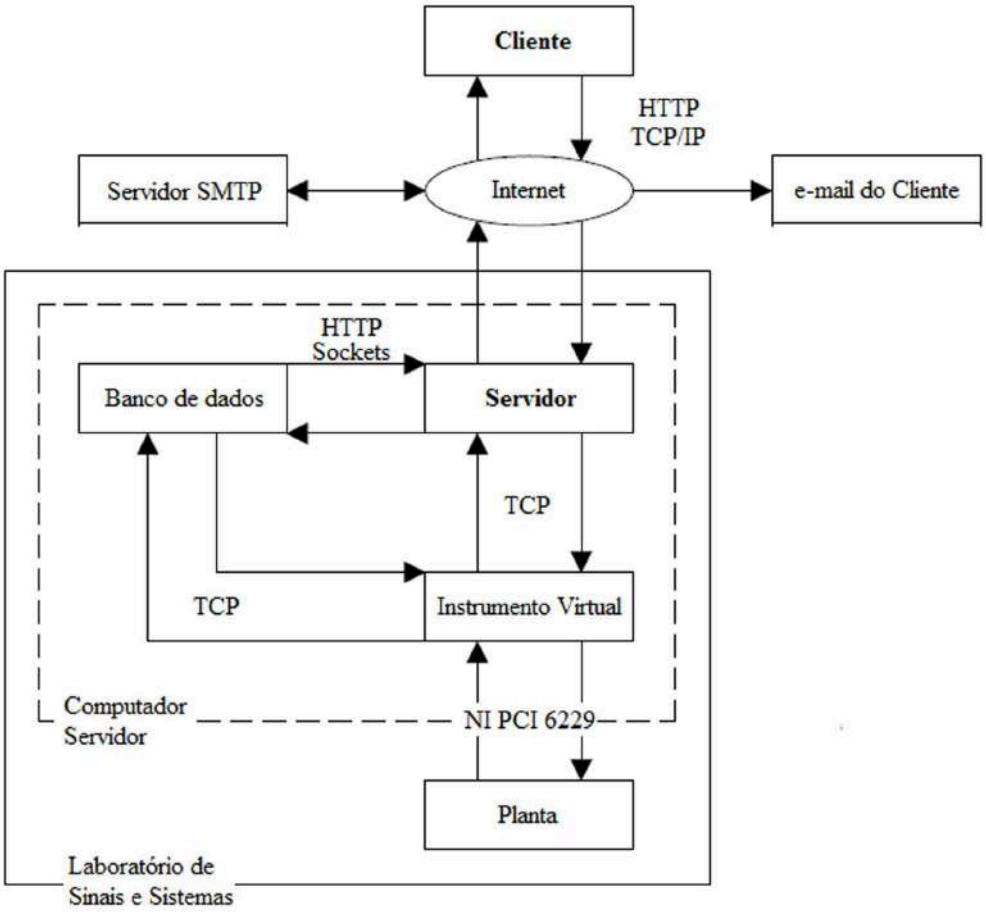

Figura 3.8: Modelo proposto para o WebLab.

3.3 Sistema de Aquecimento de Ar

A planta do sistema de aquecimento de ar, se encontra no laboratório de Sinais e Sistemas do CEFET-MG/Divinópolis. Este protótipo foi desenvolvido por SIMEÃO (2009) como objeto de estudo para seu trabalho de mestrado e consiste em um túnel de acrílico por onde o ar circula insuflado por um ventilador axial; três atuadores (resistências elétricas); cinco sensores de temperatura ($LM35$) e uma válvula de controle de fluxo (*damper*). A Figura 3.9 mostra o desenho esquemático do sistema, sendo possível observar as seções nas quais a estrutura é dividida assim como a disposição de cada componente. De acordo com o desenho têm-se:

- Seção 1: Entrada forçada de ar e sensor de temperatura 1 ($LM1$);
- Seção 2: Resistência elétrica 1 ($R1$) e sensor de temperatura 2 ($LM2$);
- Seção 3: Restrição para controle de fluxo de ar;
- Seção 4: Sensor de temperatura 3 (LM3);
- Seção 5: Resistência elétrica 2 ($R2$);
- Seção 6: Sensor de temperatura 4 ($LM4$);
- Seção 7: Resistência elétrica 3 ($R3$);
- Seção 8: Não utilizada;
- Seção 9: Sensor de temperatura 5 ($LM5$).

No início da proposição do presente trabalho, o protótipo em questão, apresentava diversos problemas em sua parte elétrica/eletrônica. Suas resistências se encontravam queimadas; os circuitos de aquisição de dados apresentavam acúmulo de erros nas medições e os circuitos de potência apresentavam uma pequena tensão de *offset*, fornecendo mesmo que em um nível baixo, uma certa potência às cargas quando estas deveriam estar totalmente desligadas. Outro problema presente no sistema, estava no circuito de controle que apresentava mau contato, desligando subitamente quando era comutado para os modos manual ou automático.

A situação em que a planta se encontrava, também foi uma das motivações na decisão de escolhê-la como objeto de pesquisa no presente trabalho. No entanto, a motivação principal está relacionada ao fato de que devido às suas proporções, o sistema possui resposta temporal lenta, fazendo com que alguns experimentos possam levar horas, o que justifica mais uma vez a realizá-los remotamente, otimizando os fatores tempo e espaço.

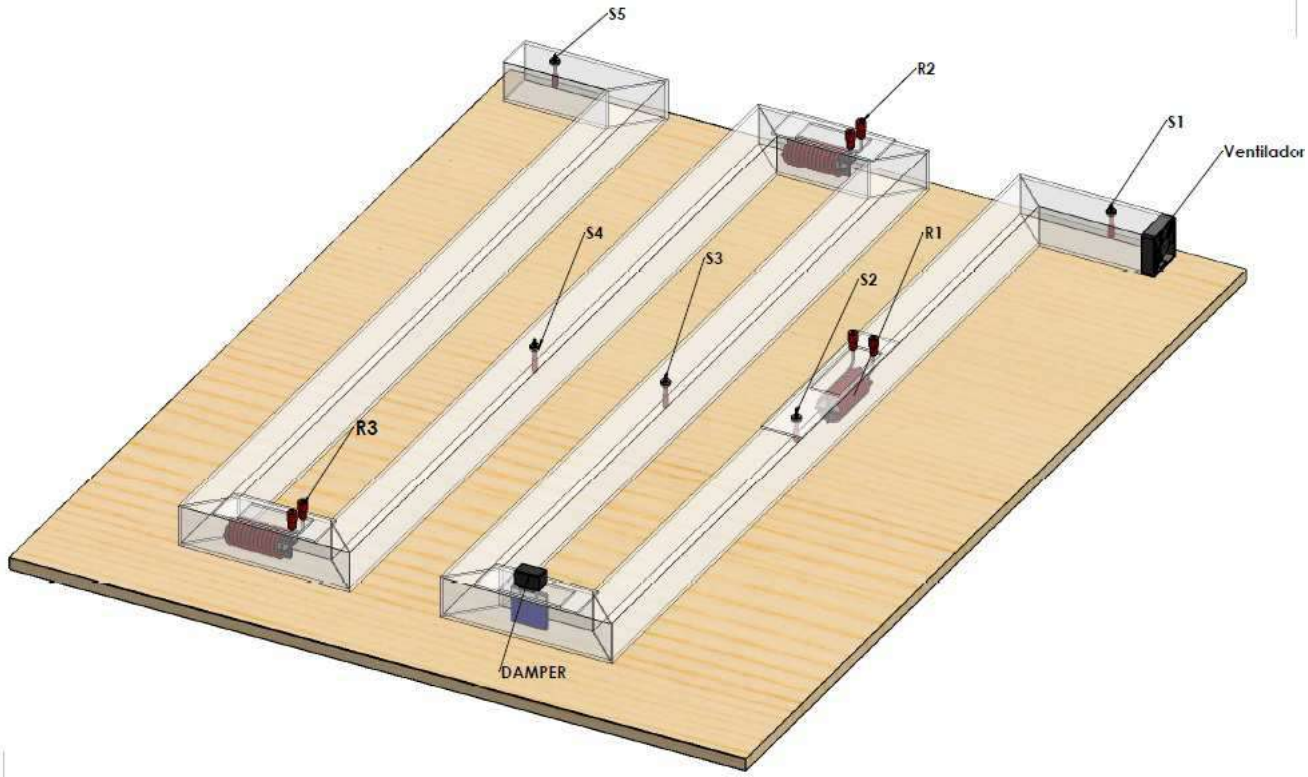

Figura 3.9: Esquema da planta do sistema de aquecimento de ar.

Após uma análise das condições da planta, foram realizadas algumas modificações e melhorias. Foram reprojetados os circuito de acionamento, incluindo os dispositivos utilizados para fornecer potência às cargas, o modo de controle de potência *AC* e o sistema de alimentação, adicionando-se a este, um circuito para acionamento geral da planta. Foram acrescentados ao sistema circuitos para medição da energia elétrica consumida pelas resistências, com a finalidade de avaliar o desempenho de controladores em relação a tal parâmetro. Além disso, as medições de temperatura foram reforçadas por circuitos termopares projetados e implementados pelo ex-aluno Jônathas Vinícius do Valle Silva em seu trabalho de conclusão de curso intitulado *Modelagem Matemática a Parâmetros Distribuídos e Controle em Malha Fechada de um Sistema de Aquecimento de Ar* (SILVA, 2013).

Devido ao mau funcionamento do circuito de controle e às novas funcionalidades necessárias ao projeto, decidiu-se por desenvolver outro, também baseado em microcontroladores, com objetivo principal de descentralizar o processamento de dados, como cálculos de consumo de energia e geração de pulsos de comando, deixando o computador sendo responsável em sua maior parte, por receber sinais condicionados da planta e realizar a comunicação de dados com o usuário remoto via Web. A Figura 3.10 mostra o sistema de aquecimento de ar após as modificações realizadas.

Figura 3.10: Sistema de aquecimento de ar após modificações.

3.3.1 Estradas e saídas do sistema

A Figura 3.11 é um diagrama esquemático construído a partir da análise dos sinais de entrada e saída desejados em cada módulo que compõe o sistema. Toda a eletrônica da planta, depende direta ou indiretamente dos comandos enviados pelo computador através do *software* LabVIEW® por meio da placa de aquisição de dados $DAQ\ NIPCI-6229$ da *National Instruments*.

A placa $NI\ PCI-6229$ possui 32 canais de entrada analógica e 4 canais de saída analógica com resolução de $16bits$ e escala de $-10V$ a $+10V$; 3 portas digitais de $32bits$ com 32 linhas cada, que podem ser configuradas como entrada ou saída (é possível trabalhar com toda a porta ou apenas com determinado número de linhas).

O circuito de acionamento geral da planta recebe um sinal de liga/desliga do programa em LabVIEW® por meio da placa de aquisição de dados, permitindo ou não a alimentação das fontes CC, que por sua vez alimentam os circuitos eletrônicos, assim como o ventilador axial e as resistências. Estas porém, dependerão do circuito de comando para que sejam ou não alimentadas.

Os circuitos de comando, dependem do circuito microcontrolado, que recebe um sinal de referência de 0 a $5V$ e gera um pulso com largura variável do inglês *Pulse Width Modulation - PWM* para comandar os relés de estado sólido que por sua vez permitem ou não que as cargas sejam alimentadas com a tensão da rede. Outro circuito microcontrolado é responsável por ler os sinais analógicos dos transdutores de corrente e por calcular o consumo de energia, enviando os sinais para o computador a cada ciclo de leitura dos sensores de temperatura $LM35$ e termopares.

Tendo-se uma visão geral do sistemas com suas entradas e saídas, as subseções seguintes descrevem os módulos desenvolvidos.

3.3.2 Circuito de acionamento geral

Como os experimentos a serem realizados são agendados, foi construído um circuito de acionamento geral da planta. Tal circuito é responsável por energizar ou desenergizar todos os módulos e fornecer ou não alimentação para as cargas de acordo com o agendamento feito e com os sinais do circuito de comando. A Figura 3.12, mostra o circuito desenvolvido para acionamento geral. Ele é composto por uma contatora $CNU16$ que interrompe ou não as fases RST da alimentação da rede.

Para o acionamento da contatora, foi utilizado um circuito auxiliar. Tal circuito é constituído basicamente por um optoacoplador ($MOC3021$)que recebe um sinal de $5V$ do LabVIEW®, por intermédio da placa de aquisição de dados, acionando um $TRIAC$ $BT139$ permitindo a energização da bobina da contatora fechando seus contatos principais. O optoacoplador é responsável pela isolação entre o sinal enviado pela placa de aquisição

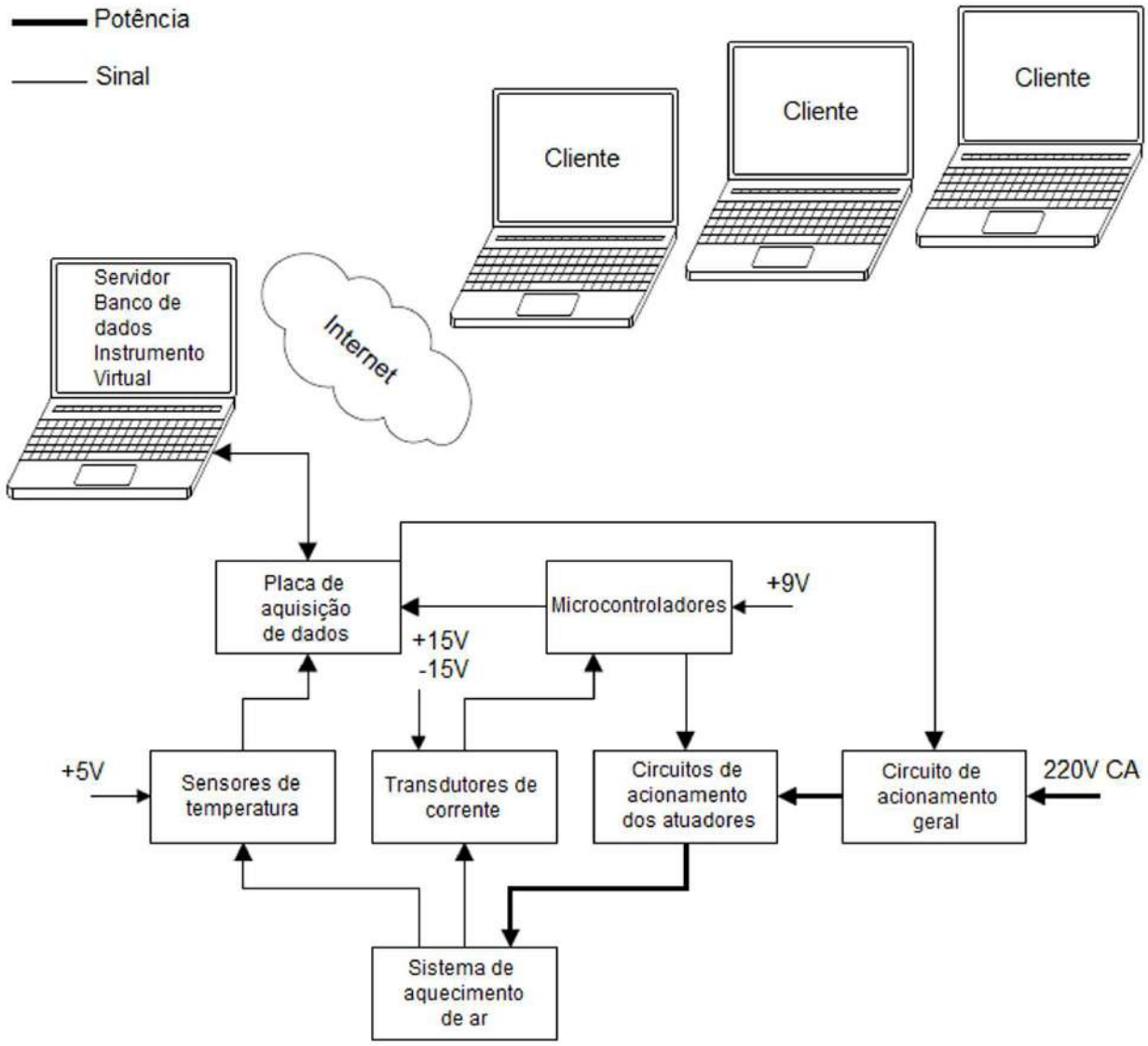

Figura 3.11: Diagrama esquemático do sistema.

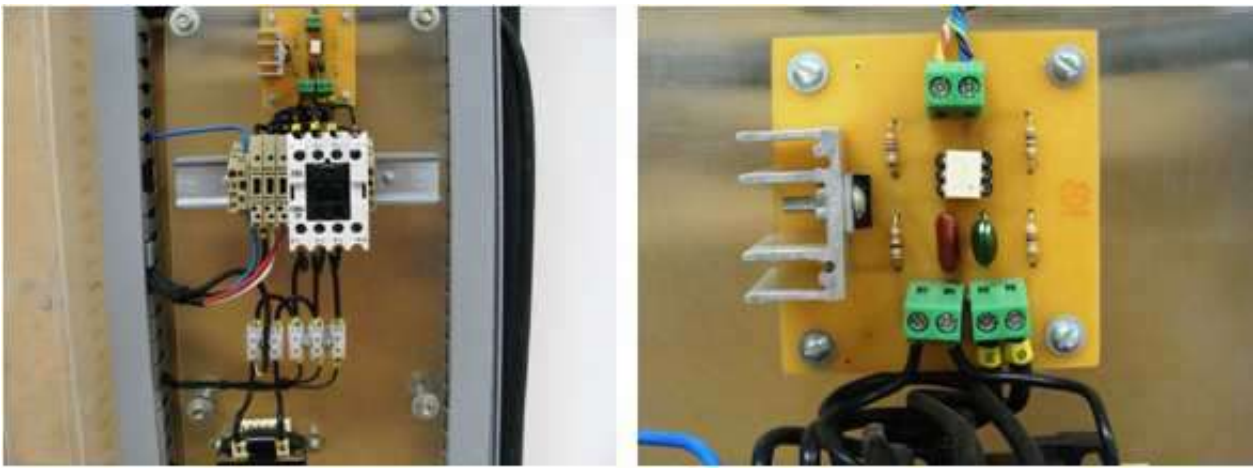

Figura 3.12: Circuito de acionamento geral da planta.

de dados e o circuito de potência que aciona a bobina da contatora.

Todos os circuitos da planta são protegidos por fusíveis de $10A$ instalados na entrada de alimentação da rede. Além disso, o laboratório conta com um disjuntor trifásico de $16A$ no quadro de distribuição que reforça a proteção da planta.

3.3.3 Circuito de comando

O circuito de comando do sistema de aquecimento de ar foi construído a partir de relés de estado sólido. A escolha destes para acionamento das resistência do forno se deu principalmente por sua ampla aplicabilidade industrial, principalmente em processos que envolvem controle de temperatura.

A Figura 3.13, apresenta a imagem do SSR utilizado. A inscrição $D2425$ em seu corpo, é o seu código de identificação. A letra D, indica que ele é controlado por um sinal de tensão DC, os dois primeiros números indicam a tensão máxima em seu terminal de potência, no caso, o número 24 significa uma tensão de saída máxima de $240V_{ac}$ e os dois últimos números indicam a corrente nominal suportada por ele ($25A$). Além dessas informações, no corpo do SSR também são indicados como deve ser o sinal de comando e a faixa de tensão que pode ser aplicada. Para o $D2425$, o sinal pode ser contínuo ou pulsado e a faixa de tensão é de $+3V$ a $+32V$. Essas são as especificações necessárias ao dimensionamento do dispositivo.

Figura 3.13: Relé de estado sólido.

Os relés de estado sólido utilizados foram adquiridos anteriormente ao projeto e dessa forma, as especificações para compra não foram necessárias. Ao contrário, foi então verificado se suas especificações atendiam ao projeto. Quanto à instalação, foi indispensável dimensionar os dispositivos de proteção contra curto-circuito e sobre-aquecimento.

Antes de se prosseguir com os cálculos de dimensionamento para proteção dos relés, foi feito um estudo para se definir o sistema de alimentação das cargas. As resistências utilizadas no protótipo são de secador de cabelo e possuem potência nominal de $1400W$, podendo ser alimentadas com $127V$, ou $220V$. Além disso, elas são constituídas por dois filamentos, podendo ser ligados em série ou em paralelo. Medindo o valor da resistência

elétrica, foi verificado aproximadamente $27,6\Omega$ para cada filamento.

A Figura 3.14 apresenta esquematicamente as possíveis formas de ligação das resistências. Foram calculados os valores da corrente e da potência para cada caso, levando-se em consideração o valor da resistência elétrica medida. A Tabela 3.1 indica os valores das potências máximas calculados.

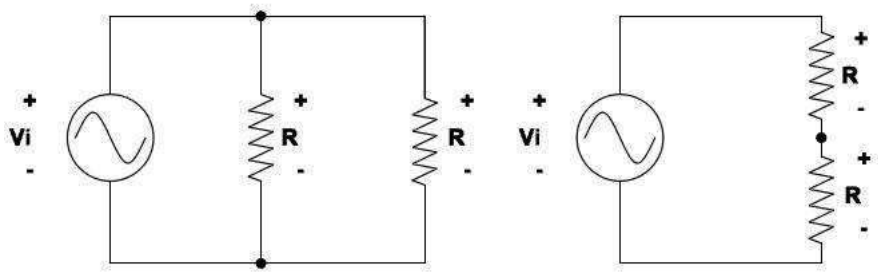

Figura 3.14: Possíveis formas de ligação das resistências.

Tabela 3.1: Correntes e potências calculadas.

v_i	i(Série)	P(Série)	i(Paralelo)	P(Paralelo)
$127V$	$2{,}3A$	$292W$	$9{,}2A$	$1168W$
$220V$	$3{,}98A$	$874W$	$15{,}94A$	$3506W$

Pela análise dos valores obtidos, verificou-se que a melhor forma de ligação foi aquela utilizando tensão de alimentação de $220V$ com os filamentos da resistência em série. Esse tipo de ligação exige da rede uma corrente relativamente baixa, o que permite a utilização simultânea de outras plantas no laboratório. Em relação à potência máxima, verificou-se que o valor de aproximadamente $875W$ é suficiente ao propósito do trabalho. A Tabela 3.2 mostra os valores de tensão e corrente medidos com voltímetro e alicate amperímetro, em cada uma das resistências com filamentos ligados em série. Tais valores confirmam e dão maior precisão ao cálculo realizado. Vale lembrar que a corrente máxima fornecida pelas instalações do laboratório é de $16A$ por fase.

Tabela 3.2: Correntes e potências medidas.

Resistência	Tensão (V_{rms})	Corrente (A_{rms})	$Potenciamxima(W)$
1	219	3,92	854,48
2	217	3,85	835,45
3	219	3,80	832,20

Como o acionamento das três resistências devem ser individual, existiam duas possibilidades em relação ao sistema de alimentação, ou seja, poderia ser utilizado o sistema

monofásico, ligando as resistências em paralelo, ou trifásico, ligando-as em triângulo. Contudo, pela primeira opção, a corrente total seria maior tanto para $127V$ quanto para $220V$, exigindo mais das instalações do laboratório. Sendo assim, optou-se pelo esquema de ligação em triângulo conforme representado pelo desenho esquemático da Figura 3.15. Por meio desses cálculos, verificou-se que os SSR disponíveis atendiam ao projeto, inclusive sendo possível trocalos a qualquer momento por $SSRs$ com corrente máxima menor.

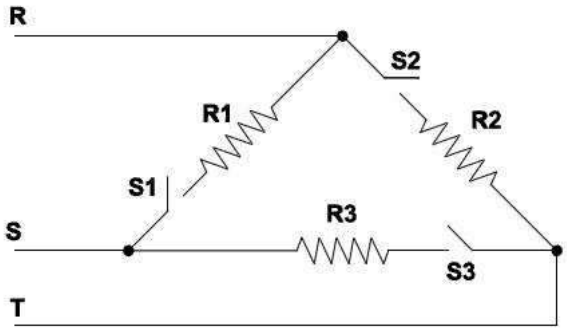

Figura 3.15: Esquema de alimentação das resistências.

Para proteção dos relés de estado sólido contra curto-circuito devem ser utilizados fusíveis ultra rápidos. Porém, esses fusíveis não foram adquiridos. No entanto, como a corrente máxima calculada foi de aproximadamente $4A$, ou seja, seis vezes menor que a corrente nominal do relé, foi utilizado um fusível comum no valor de $10A$. Os fusíveis recomendados para $SSRs$ são do tipo *Silized*. A Figura 3.16 apresenta os fusíveis utilizados para proteção dos $SSRs$.

Figura 3.16: Fusíveis para proteção dos relés de estado sólido.

A instalação dos relés requer também que eles sejam protegidos contra sobre-aquecimento. Com a corrente de carga circulando, há geração de intenso calor sobre a chave. Este calor dever ser transferido rapidamente para o ambiente para evitar a queima do dispositivo. A temperatura na base do relé não pode exceder $75°C$. O cálculo para o dimensionamento do dissipador de calor é feito segundo a Equação (3.1).

$$R_{thha} = \frac{75°C - T_{amb}}{I_L V_{ssr}} \tag{3.1}$$

em que:
R_{thha} = resistência térmica do dissipador.
T_{amb} = temperatura ambiente.
I_L = corrente de carga.
V_{ssr} = queda de tensão no SSR quando conduzindo.

A queda de tensão nos $SSRs$ utilizados é de $1{,}6V$ e a corrente de carga calculada anteriormente foi de aproximadamente $4A$. Com isso, o dissipador a ser utilizado, teria resistência térmica de aproximadamente $5{,}47°C/W$ (considerando a temperatura ambiente igual a $40°C$), sendo até mesmo dispensável seu uso. Para realizar função de dissipador, os relés de estado sólido foram parafusados sobre uma chapa lisa de alumínio onde foi também utilizado pasta térmica. Esse procedimento foi realizado principalmente para ilustrar a instalação correta do dispositivo e é apresentado na Figura 3.16. A posição correta de funcionamento da planta é na vertical, e portanto, os $SSRs$ operarão nesta posição, possibilitando a passagem fluxo de ar sobre os mesmos.

O tipo de controle de potência escolhido para comandar os relés de estado sólido foi o controle por ciclo integral, descrito no Apêndice A. Segundo ASHFAQ (2000), este tipo de controle é geralmente empregado em sistemas que possuam constante de tempo grande, como por exemplo, o controle de sistemas térmicos.

De forma a se obter um bom compromisso entre a resolução, ou seja, a escala da porcentagem de potência entregue à carga, com a característica de resposta do sistema, foi feito um estudo para se determinar a frequência adequada dos pulsos de comando. Por meio de testes, foi verificado que assumindo uma frequência de $1Hz$ o compromisso entre a precisão do acionamento e a percepção do chaveamento pelos sensores de temperatura seria atendido.

Uma vez que o controle é feito por semi-ciclos, dividindo-se a frequência de $1Hz$ pelo dobro da frequência da rede, constatou-se que a precisão do acionamento seria de 0,8%. Em outras palavras, isso quer dizer que ocorre uma variação de $\pm 0{,}02\%$ na potência entregue à carga.

3.3.4 Fontes de alimentação

Para alimentar os cinco sensores de temperatura $LM35$ e os seis termopares, o circuito transdutor de corrente (LA 55-P) e o microcontrolador do circuito de controle, foram construídas três fontes de alimentação do tipo linear, alimentadas com $15Vac$ capazes de fornecerem juntas até $1A$. Tais fontes possuem saídas positivas e negativas podendo variar de 0 a $15V$ na saída positiva e de 0 a $-15V$ na saída negativa.

O diagrama elétrico do circuito é apresentado na Figura 3.17 e foi adaptado de FRANCO (2012). A saída positiva é regulada pelo $LM317$ ($U1$) sendo o nível de tensão ajustado pelo resistor variável $RV1$. Já a saída negativa é regulada pelo $LM337$ ($U2$) sendo o nível de tensão ajustado pelo resistor variável $RV2$. A fonte de tensão construída é apresentada na Figura 3.18.

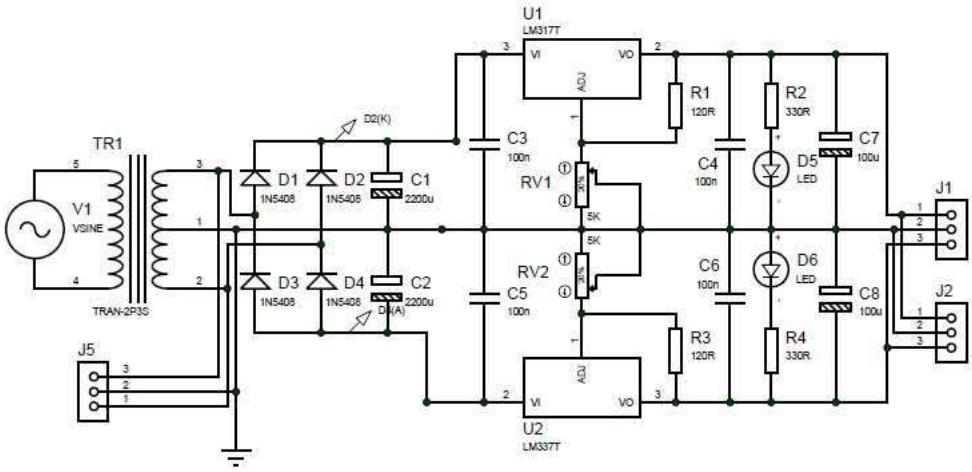

Figura 3.17: Diagrama elétrico da fonte de alimentação.

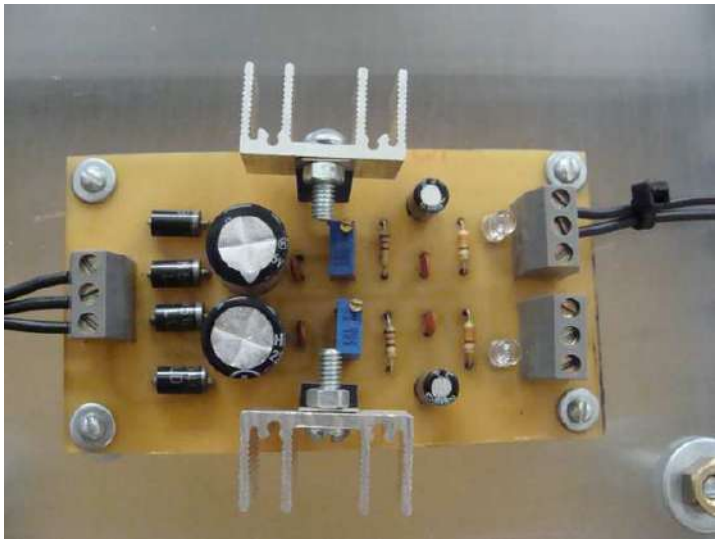

Figura 3.18: Fonte de alimentação $+15V$ e $-15V$.

3.3.5 Circuito transdutor de corrente

Um parâmetro importante a ser considerado no projeto de controladores, é o consumo de energia. Através dele, é possível analisar a viabilidade de determinado controlador em relação a outras especificações de projeto, tais como rapidez e precisão por exemplo. Dessa forma, para medir essa grandeza em cada resistência, foram projetados circuitos de medição de corrente. Tal circuito, é composto por um transdutor de corrente por efeito *Hall*, descrito a seguir, e um circuito microcontrolado para cálculo do consumo de energia descrito na Subseção 3.3.7.

O circuito para medição de corrente foi construído utilizando-se o transdutor de efeito *Hall* em *loop* fechado $LA55-P$ fabricado pela LEM (Figura 3.19).

Figura 3.19: Transdutor de corrente por efeito *Hall* LEM $LA55-P$.

A Figura 3.20 apresenta o diagrama de ligação do circuito. Quando um fio condutor com uma corrente I_P passa pelo orifício do transdutor, é gerado no secundário do mesmo uma corrente I_S proporcional. Por meio de uma resistência de medição R_M em série com o secundário, é criada uma tensão de saída que é também proporcional à corrente medida.

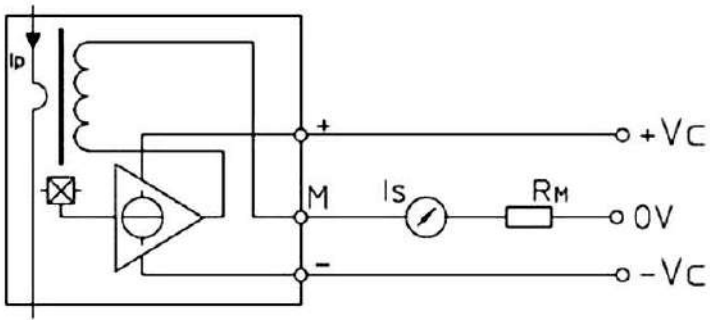

Figura 3.20: Esquema do transdutor de corrente por efeito *Hall* LEM $LA55-P$.
Fonte: (MICHIGAN UNIVERSITY, 2013).

A Figura 3.21 indica os dados elétricos do transdutor. Por meio da análise destes dados, verifica-se que ele foi projetado para medir correntes de até $50A$. Verifica-se também que a razão entre a corrente no primário e no secundário é dada pela constante de conversão

K_N (1 : 2000), ou seja,

$$I_S = K_N \cdot I_P. \tag{3.2}$$

Electrical data

I_{PN}	Primary nominal current rms			50			A
I_{PM}	Primary current, measuring range			0 .. ± 100			A
R_M	Measuring resistance		T_A = 70°C		T_A = 85°C		
			$R_{M\,min}$	$R_{M\,max}$	$R_{M\,min}$	$R_{M\,max}$	
	with ± 12 V	@ ± 50 A_{max}	0	215	0	210	Ω
		@ ± 100 A_{max}	0	35	0	30	Ω
	with ± 15 V	@ ± 50 A_{max}	0	335	30	330	Ω
		@ ± 100 A_{max}	0	95	30	90	Ω
I_{SN}	Secondary nominal current rms			25			mA
K_N	Conversion ratio			1 : 2000			
V_C	Supply voltage (± 5 %)			± 12 .. 15			V
I_C	Current consumption			10 (@ ± 15 V) + I_S			mA

Figura 3.21: Dados elétricos ($LEM\ LA55-P$).
Fonte: (MICHIGAN UNIVERSITY, 2013).

Como o fundo de escala do transdutor é de $50A$ e considerando que a máxima corrente a ser medida por ele é de $4A$, foi necessário estabelecer uma proporcionalidade entre os dois valores. A corrente no primário do transdutor é o produto do número de voltas (N) do condutor em torno dele pelo valor da corrente que passa pelo condutor (I_P). Sendo assim, para que a faixa de medição do sensor seja totalmente aproveitada, basta dividir o fundo de escala pelo máximo valor de corrente a ser medida. Portanto, fazendo-se essa divisão verificou-se que o número de voltas em torno do primário do transdutor deveria ser igual a 12, obtendo-se uma nova corrente $I_{P'}$.

$$I_{P'} = 12 \cdot I_P \tag{3.3}$$

$$I_{P'} = 12 \cdot 4A \tag{3.4}$$

$$I_{P'} = 48A \tag{3.5}$$

Esse valor de medição deve ser corrigido pelo microcontrolador para fazer o cálculo do consumo de energia.

A resistência de medição R_M depende da alimentação do transdutor. Para este projeto, decidiu-se por alimentá-lo com uma tensão de $\pm 15V$. De acordo com os dados da Figura 3.21, existe uma faixa de valores de resistência R_M para cada valor de alimentação. Recorrendo-se à Equação (3.2) e assumindo o valor de $R_M = 330\Omega$ e ainda o novo valor de I_P, ou seja, $I_{P'} = 48A$ obtém-se o valor de I_S e consequentemente o valor da tensão de medição V_M.

$$I_S = 5 \times 10^{-5} \cdot I_{P'} \tag{3.6}$$

$$I_S = 0{,}024A$$

$$V_M = R_M \cdot I_S \tag{3.7}$$

$$V_M = 7{,}92V$$

Essa é a tensão máxima a ser lida quando a resistência estiver operando em sua potência máxima. Neste ponto, foi observado um problema em relação ao microcontrolador utilizado. A faixa de leitura analógica do microcontrolador, varia normalmente de 0 a $5V$, ou seja, não seria possível ler a tensão quando a resistência estivesse operando em potência máxima. Dessa forma, a solução mais simples foi diminuir V_M para 200Ω. Assim, a tensão máxima a ser lida pelo microcontrolador passou a ser $4{,}8V$.

3.3.6 Sensores de temperatura

Como pode ser observado na Figura 3.10, a planta do sistema de aquecimento de ar é equipada com cinco sensores de temperatura $LM35$ e seis termopares. Para a utilização desses sensores foram realizadas calibrações a fim de se obter equações do tipo.

$$T(v_t) = av_t + b \tag{3.8}$$

Em que $T(v_t)$ é a temperatura real e v_t é a tensão retornada pelo sensor.

O procedimento para calibração dos sensores $LM35$ e termopares, são detalhados em SILVA (2013). No presente trabalho, os resultados obtidos por ele são utilizados para permitir a medição correta da temperatura. Os coeficientes da Equação (3.8) se encontram disponíveis nas Tabelas 3.3 para os sensores $LM35$ e 3.4 para os termopares, respectivamente. Nas mesmas tabélas são apresentados os resultados para o teste σ_T^2 para obtenção de intervalos de confiança de 99,7%.

Os parâmetros σ_a e σ_b são calculados de acordo com as Equações (3.9), (3.10) e (3.11), cujo método é proposto por DOEBELIN (1990).

$${\sigma_a}^2 = \frac{N{\sigma_T}^2}{N\sum_{i=1}^{N} v_{t^2} - \left(\sum_{i=1}^{N} v_t\right)^2} \tag{3.9}$$

$${\sigma_b}^2 = \frac{{\sigma_T}^2 \sum_{i=1}^{n} {v_t}^2}{N\sum_{i=1}^{N} v_{t^2} - \left(\sum_{i=1}^{N} v_t\right)^2} \tag{3.10}$$

$${\sigma_T}^2 = \frac{1}{N}\sum_{1}^{N} (av_t + b - T)^2 \tag{3.11}$$

Tabela 3.3: Resultado da calibração dos sensores $LM35$.

Sensor	Coeficiente a	σ_a	Coeficiente b	σ_b
1	99,355	0,1086	−0,4016	0,0526
2	101,0179	0,1438	−0,4121	0,0685
3	101,0749	0,0961	−0,3647	0,0457
4	97,1902	0,0972	0,4754	0,0473
5	100,5313	0,0944	−0,5503	0,0453

Fonte: (SILVA, 2013).

Tabela 3.4: Resultado da calibração dos Termopares.

Termopar	Coeficiente a	σ_a	Coeficiente b	σ_b
1	93,9109	0,1130	0,8182	0,0621
2	93,6383	0,1098	1,4822	0,0598
3	96,2809	0,1830	−0,0222	0,0997
4	94,8384	0,1311	0,2075	0,0727
5	93,6271	0,1257	−0,0263	0,0703
6	94,5403	0,1255	−0,0668	0,0695

Fonte: (SILVA, 2013).

3.3.7 Circuitos microcontrolados

Com o propósito de descentralizar o processamento realizado pelo computador, foram utilizados dois circuitos microcontrolados. Inicialmente, esses circuitos seriam projetados utilizando-se microcontroladores PIC. No entanto, com o objetivo de se concentrar esforços no desenvolvimento do *software* aplicativo em LabVIEW® e na comunicação Web, foram utilizadas placas prontas. A Figura 3.22 apresenta os microcontroladores utilizados.

O primeiro circuito foi programado para gerar os pulsos com largura variável para comando dos relés de estado sólido e *damper*. Foram utilizadas quatro entradas analógicas e quatro saídas digitais. Ao receber um sinal de referência de 0 a $5V$ do LabVIEW® através da placa de aquisição de dados DAQ NI $PCI-6229$, o microcontrolador utiliza o $Timer$ 1 para gerar um pulsos com período de $1Hz$ e largura variável.

O segundo microcontrolador seria programado para calcular o consumo de energia. Ele seria então responsável por ler os sinais analógicos gerados pelo circuito transdutor de corrente, e calcular a potência consumida pelas resistências. No entanto, não foi possível desenvolver essa aplicação devido à dedicação a outras partes mais significativas do trabalho.

A placas utilizadas foram Arduino Uno *R3* da Arduino®. Essas placas são compostas por um microcontrolador *ATmega* 328, com tensão de operação de $5V$, possuindo 14

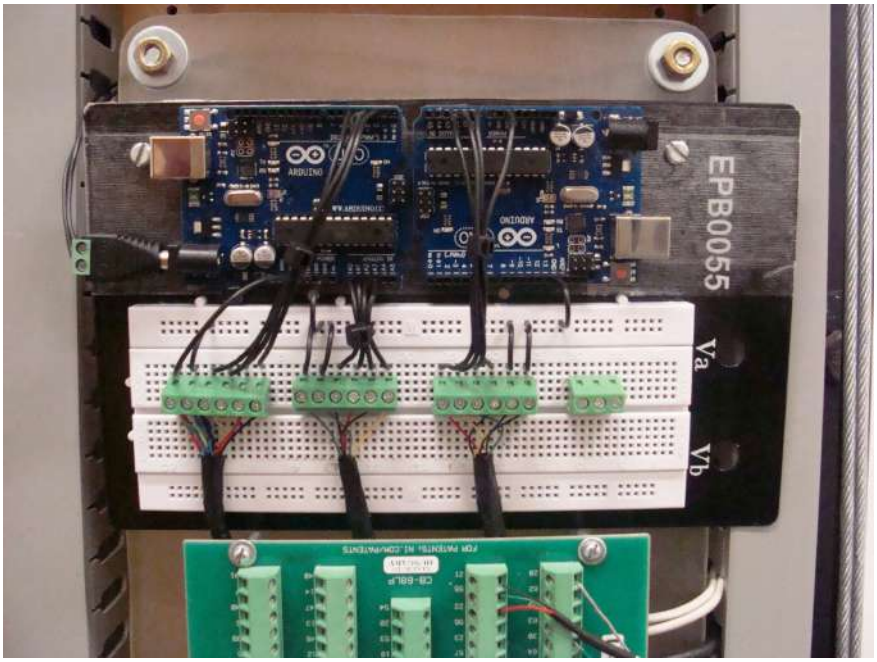

Figura 3.22: Circuitos microcontrolados Arduino® utilizados no projeto.

portas (Entrada/Saída) digitais e 6 portas de entrada analógicas. Possui també $32Kb$ de memória *flash*, SRAM de $2Kb$, EEPROM de $1Kb$ e velocidade de *clock* de $16MHz$.

3.4 Desenvolvimento da Aplicação em LabVIEW®

Para o desenvolvimento do programa aplicativo em LabVIEW® foram utilizadas um conjunto de arquiteturas denominadas padrões de projeto. Esses padrões consistem em técnicas comuns de desenvolvimento de *software* que proporcionam a criação de códigos mais eficientes, podendo ser aplicadas a qualquer linguagem de programação. A Figura 3.23 apresenta o esquema do código desenvolvido em LabVIEW® dando uma visão geral dos diversos padrões que o compõe. O código real pode ser consultado no manual do WebLab ou no CD em anexo.

Para o WebLab foram utilizados basicamente três padrões: produtor-consumidor; máquinas de estado e *loops* paralelos. Em geral, um padrão de projeto de um instrumento virtual ou VI do inglês *Virtual Instrument*, possui três parte principais: *inicialização*, *aplicação principal* e *finalização*. Cada fase pode conter códigos que segue outro tipo de padrão de projeto:

1. **Inicialização:** esta fase inicializa o *hardware*, lê as informações de configurações em arquivos ou espera para que a localização de um arquivo seja informada pelo usuário.

2. **Aplicação principal:** esta fase consiste em pelo menos um *loop* que repete até que

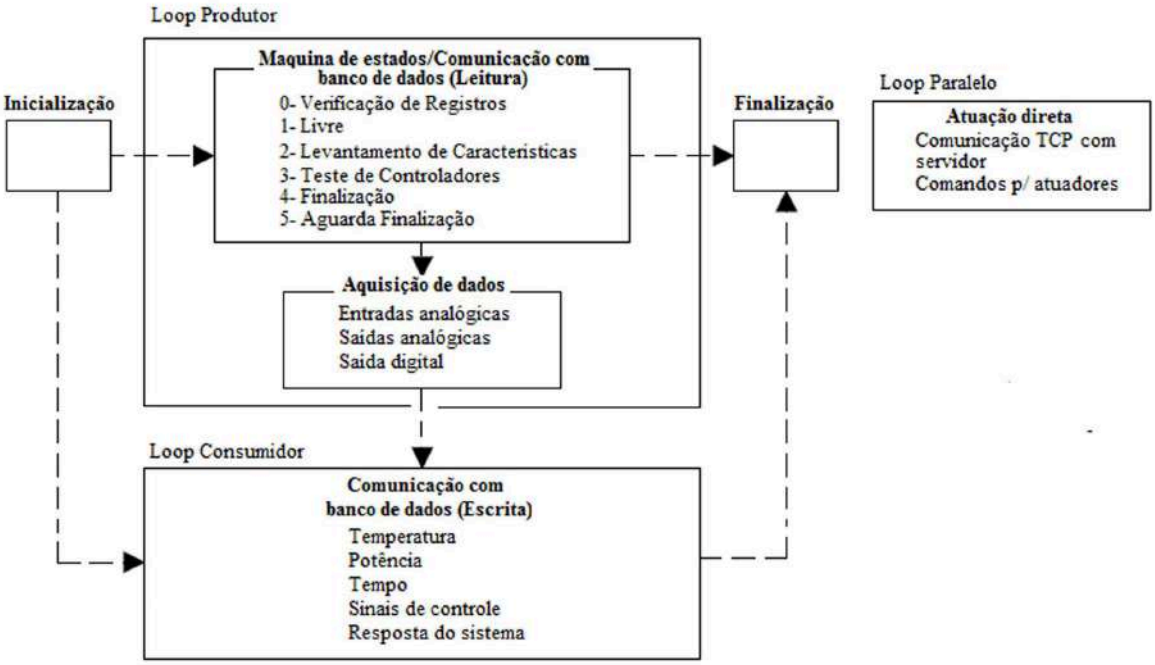

Figura 3.23: Esquema geral do código do instrumento virtual.

o usuário decida sair do programa ou que o programa encerre por outras razões, como por exemplo, a conclusão de entradas e saídas.

3. **Finalização:** esta fase fecha os arquivos, escreve informações de configurações no disco ou reinicializa as entradas e saídas para o estado padrão.

3.4.1 Produtor consumidor

A arquitetura *produtor-consumidor* consiste em dois *loops* paralelos que separam as tarefas que produzem os dados das tarefas que consomem os dados. A comunicação entre esses dois *loops* é feita através de filas, que por sua vez, armazenam em um *buffer* os dados produzidos pelo primeiro *loop* e os fornecem em ordem para o segundo. A vantagem dessa ação é que produzir os dados é mais rápido do que consumi-los, assim, as filas evitam que um dado se perca ou seja sobreposto por outro durante o processo leitura ou escrita.

A Figura 3.24 ilustra a estrutura do padrão *produtor-consumidor*. Esse padrão foi utilizado no WebLab para garantir que todos dos dados produzidos no *loop* produtor fossem enviados ao cliente de forma mais confiável possível, ou seja, sem perdas. O *loop* produtor foi então utilizado para realizar as tarefas de aquisição de dados e também para realizar as manipulações necessárias. Já o *loop* consumidor, foi utilizado para mostrar os resultados e principalmente para gravá-los no banco de dados.

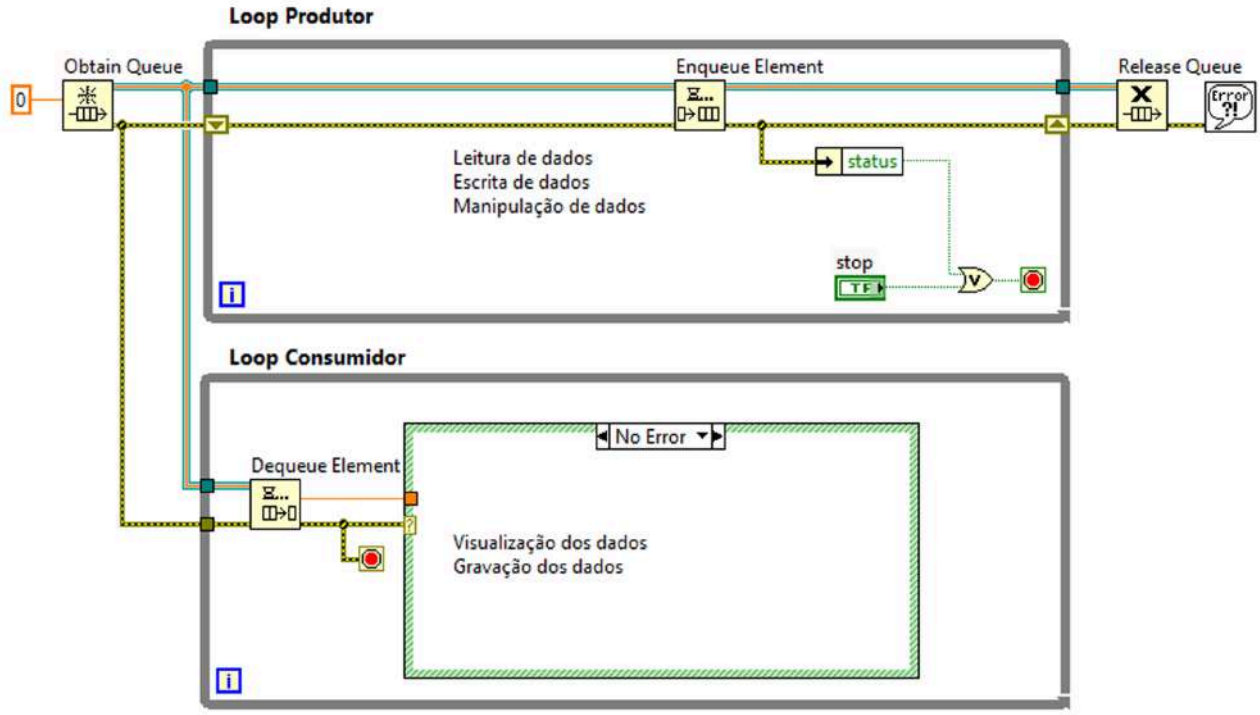

Figura 3.24: Padrão de projeto produtor-consumidor.

3.4.2 Máquina de estados

O padrão de projeto *máquina de estados* consiste em uma série de estados e uma função de transição que mapeia o próximo estado. Dessa forma, esse padrão é utilizado em aplicações que possuem estados distinguíveis. A máquina de estados verifica a entrada do usuário ou determinado cálculo realizado, para determinar qual será o próximo estado a ser executado.

No WebLab, foi utilizada uma máquina de estados para implementar as diversas funções que o usuário possa querer realizar. Nesse caso, as funções de transição são definidas pelo usuário quando ele deseja por exemplo, realizar algum dos teste disponíveis ou quando ocorre um teste de processo, ou seja, quando o estado atual provoca uma mudança de estado sob uma determinada condição.

A Figura 3.25 mostra o padrão *produtor-consumidor* juntamente com o padrão *máquina de estados*. O estado zero, *verificação de registros*, aguarda pelo agendamento de um experimento, se houver algum teste agendado para a Data/Hora atual, ele direciona o programa para um dos estados disponíveis informado pelo usuário. Para todas os outros estados, a transição ocorre por um teste de processo. Por exemplo, o estado *levantamento de características* direciona o programa para o estado Finalização quando o tempo do experimento acaba. Por sua vez, o estado Finalização direciona para o estado Aguarda Finalização que finaliza a operação e volta para o estado Verificação de Registro que fica

aguardando o próximo agendamento.

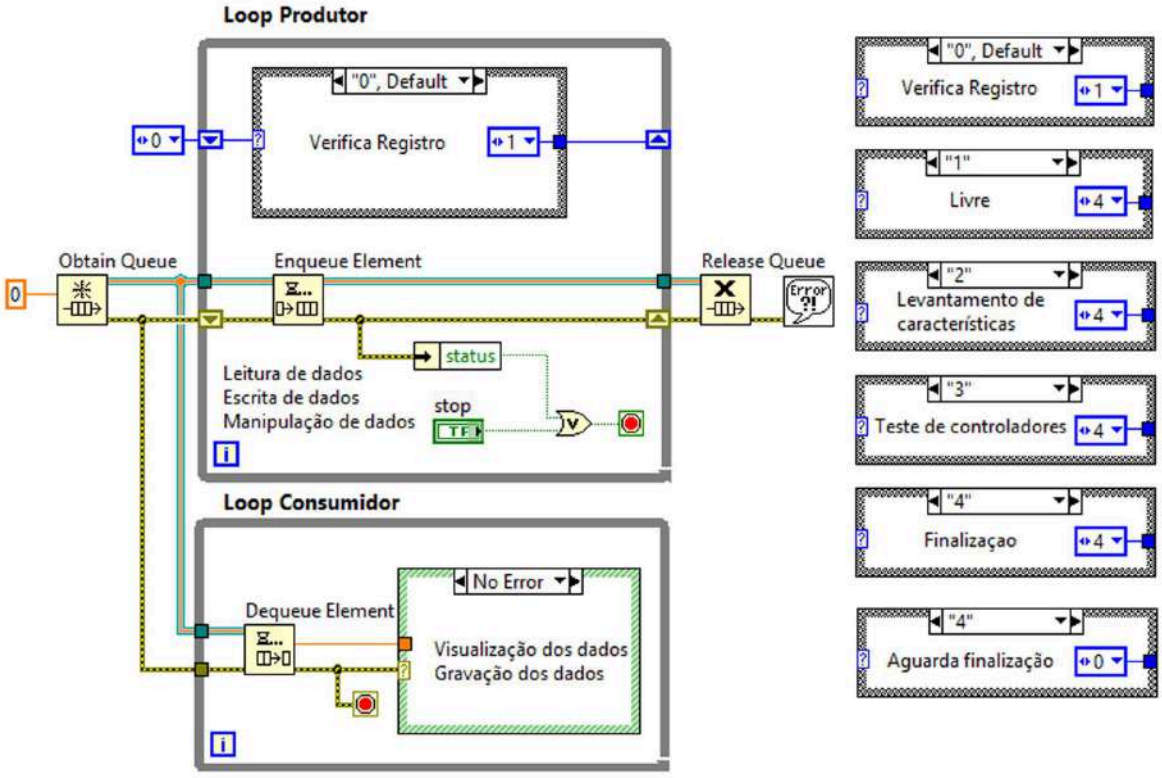

Figura 3.25: Padrão de projeto máquina de estados.

3.4.3 *Loops* paralelos

O padrão de projeto em *Loops* Paralelos é utilizado quando é necessário gerenciar tarefas múltiplas, simultâneas e independentes. Nesse padrão, a resposta de uma ação não impede que uma VI responda a outras ações. No WebLab, foi utilizado um *loop* paralelo para fazer a comunicação direta entre o servidor e o LabVIEW® via TCP/IP. Essa comunicação ocorrerá quando o estado *livre* estiver sendo executado. Como os dados são gerados no *loop* paralelo, ele se torna o *loop* mestre e o *loop* que contém o estado *livre* se torna escravo. Apenas nesse caso, aparece um outro padrão de projeto denominado *mestre-escravo*. Para todas as outras situações, o *loop* que faz a comunicação via TCP/IP é apenas um *loop* paralelo que fica executando independente do que ocorre na estrutura *produtor-consumidor*. A Figura 3.26 exemplifica a estrutura completa do *instrumento virtual* desenvolvido em LabVIEW®.

3.5 Módulos do Instrumento Virtual

Uma vez entendido a estrutura geral do programa, como ilustra a Figura 3.23, serão descritos a seguir cada um dos módulos ou sub-programas desenvolvidos. Esses módulos

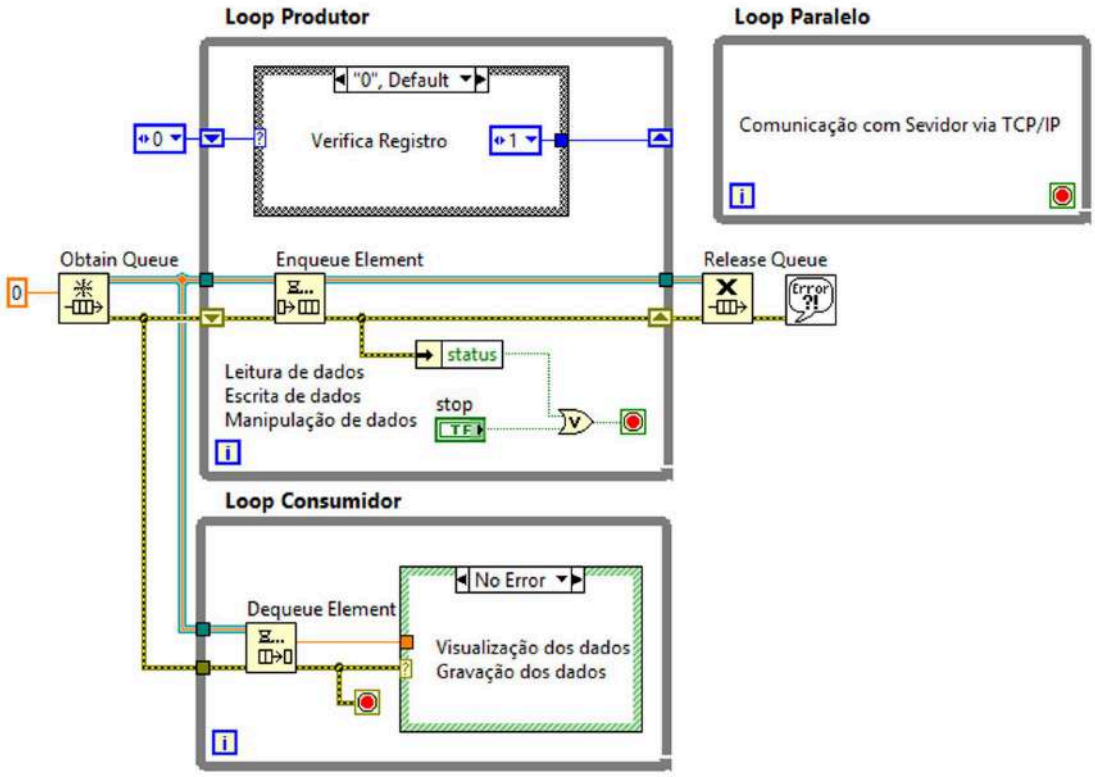

Figura 3.26: Padrão de projeto em *loops* paralelos.

constituem os métodos a serem executados quando o servidor receber as requisições do cliente. Como a maior parte deles fazem comunicação com o banco de dados, será data uma breve explicação sobre as ferramentas do LabVIEW® utilizadas para interagir com o banco de dados. As figuras que aparecerem a seguir, ilustram apenas pontos específicos do programa. Tanto o painel frontal, onde é criada a interface de usuário local quanto do diagrama de blocos, onde é criado o código do programa desenvolvido em LabVIEW® estão disponíveis integralmente no manual do WebLab ou no CD anexo.

3.5.1 Comunicação com banco de dados

De acordo com o esquema da Figura 3.8, a comunicação entre o LabVIEW® e o servidor pode ocorrer de duas formas. A primeira forma é indireta, onde o banco de dados envia ao LabVIEW® os dados ou informações de agendamento vindos das requisições feitas pelo cliente ao servidor e recebendo os resultados dos experimentos realizados pelo instrumento virtual. Na segunda forma, o servidor se comunica com o LabVIEW® diretamente através de uma conexão via TCP/IP. Como explicado anteriormente, esta última forma de comunicação só ocorre no estado *livre*.

A conexão entre o LabVIEW® e o banco de dados utiliza o padrão ODBC (*Open Database Connectivity*) e é iniciada sempre pelo LabVIEW®. Ao iniciar a aplicação a fim

de disponibilizar o sistema de aquecimento de ar para o usuário remoto, uma caixa de texto é aberta e o administrador pode configurar a conexão com a fonte de dados. Tal conexão só é desfeita quando o programa é interrompido ou para de funcionar. A Figura 3.27 mostra a conexão feita com o banco de dados criado e a caixa de texto para acessar a fonte de dados pelo LabVIEW®.

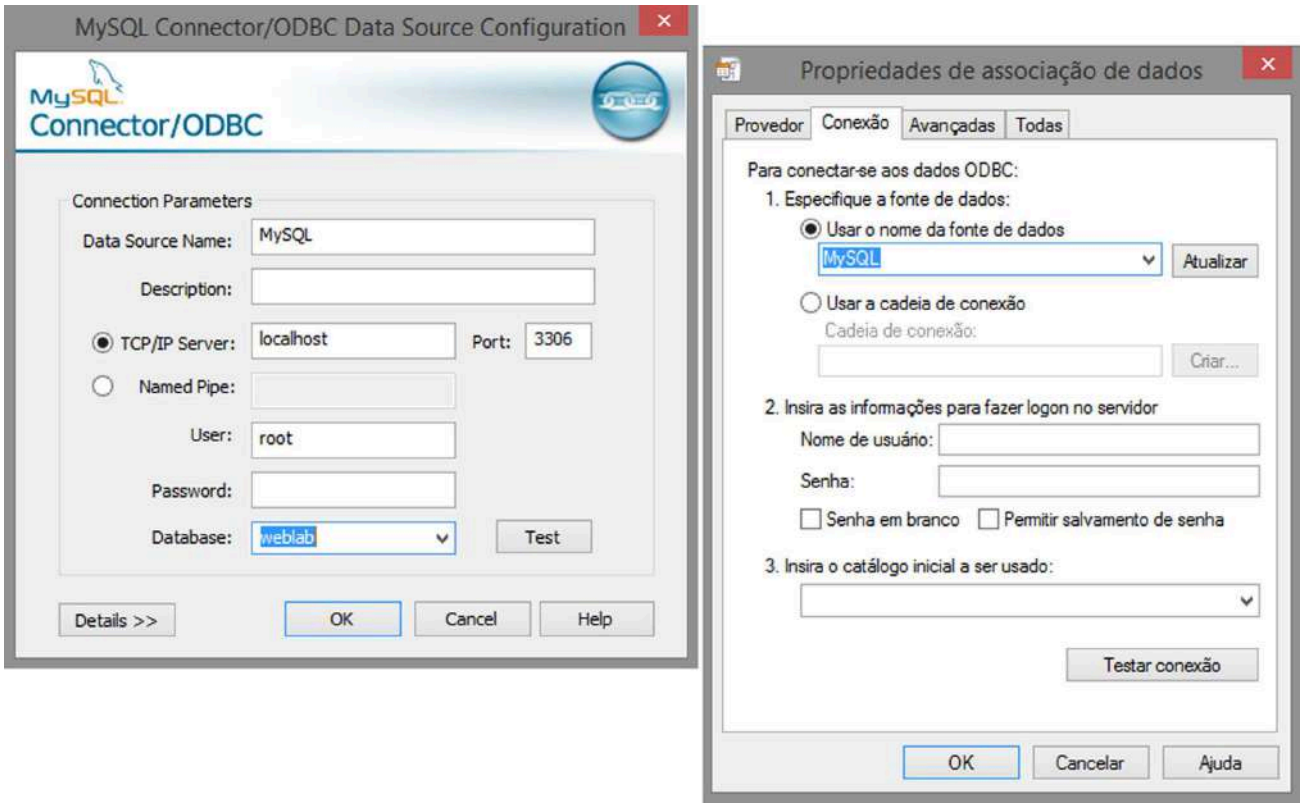

Figura 3.27: Conexão entre o instrumento virtual e o banco de dados.

Para conexão com o banco de dados, o LabVIEW® utiliza as ferramentas do *Database Connectivity Toolkit*. Essas ferramentas permitem a implementação de operações comuns em base de dados sem a utilização direta de programação em SQL. Como as tabelas do banco foram criadas no MySQL, foram utilizadas para o WebLab basicamente funções para selecionar e inserir dados no banco de dados.

3.5.2 Verificação de registros

Após a inicialização do instrumento virtual e da conexão estabelecida com o banco de dados, o programa fica aguardando um agendamento feito pelo usuário remoto. Na data e hora especificada, o aplicativo recebe do banco de dados as informações necessárias para iniciar o experimento. A Figura 3.28 apresenta a subVI, ou seja, um sub-código do código principal, responsável por esta ação. Tal ação define o estado *verificação de registros* e as informações recebidas nesse estado definem a data e hora inicial e final do experimento,

o estado seguinte, que determina o tipo de experimento a ser realizado, e o id que é a identificação para determinar a qual usuário, ou seja, a qual registro no banco de dados os resultados do experimento devem ser enviados.

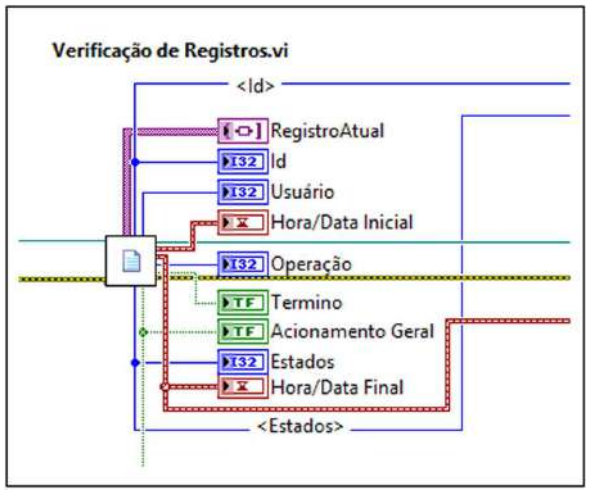

Figura 3.28: Verificação de registros.

3.5.3 Livre

O estado *livre* é a primeira função disponível na máquina de estados e corresponde à operação 1. Como citado anteriormente, quando o usuário remoto escolhe essa função, o instrumento virtual passa a comunicar diretamente com o programa servidor e assim o cliente pode atuar diretamente na planta. A Figura 3.29 mostra as VIs do *loop* paralelo utilizadas para comunicação via TCP/IP entre o servidor e o aplicativo em LabVIEW®. Essas VIs fazem parte do conjunto de ferramentas do *Data Communication TCP Toolkit*. Basicamente elas abrem uma conexão com o servidor e ficam "ouvindo" através de uma porta as informações passadas por ele.

3.5.4 Levantamento de características

A Figura 3.30 mostra as VIs responsáveis pela função *levantamento de características*. Essa função recebe do banco de dados os pontos que representam o perfil desenhado pelo usuário remoto. A Figura 3.31 mostra parte da VI que cria o perfil dentro do programa em LabVIEW® para serem aplicados ao sistema de aquecimento de ar. Para tanto é utilizada a estrutura *Formula Node* que compara os valores referentes ao tempo atual, valor atual, tempo corrente, tempo anterior e o valor anterior e gera na saída valores constantes, degraus ou rampas. As variáveis *Tempo atual*, *Valor atual*, *Tempo anterior* e *Valor anterior* correspondem aos valores lidos do banco de dados em um processo de varredura. Já o *Tempo corrente*, é o tempo contado à partir do momento que o programa entra no estado *levantamento de características*.

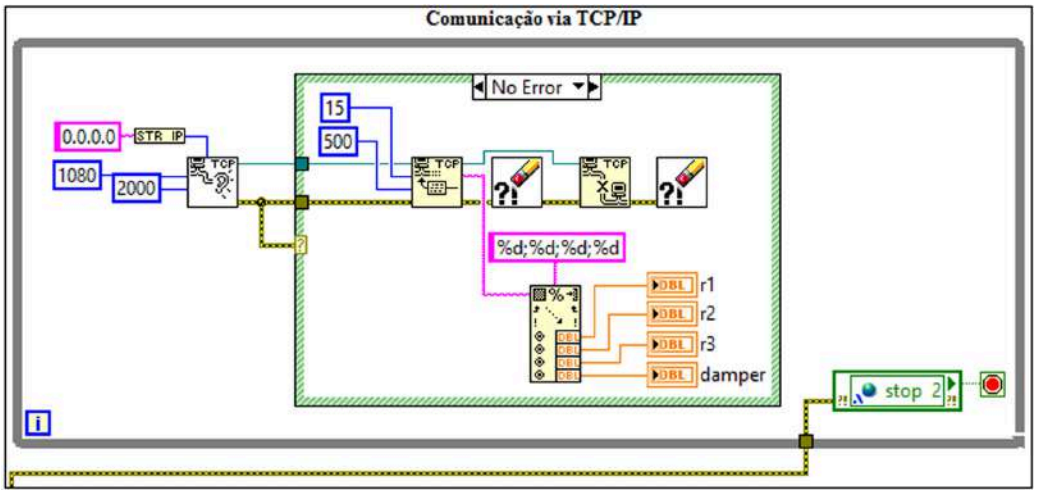

Figura 3.29: Comunicação via TCP/IP.

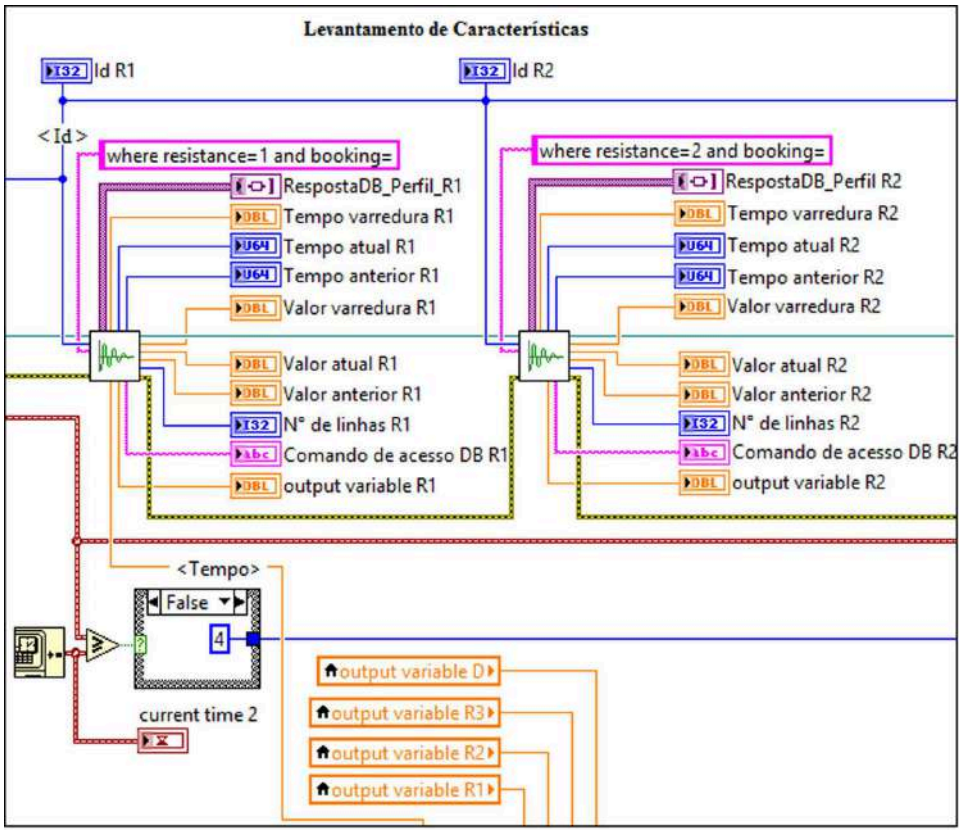

Figura 3.30: Visão parcial do código para levantamento de características.

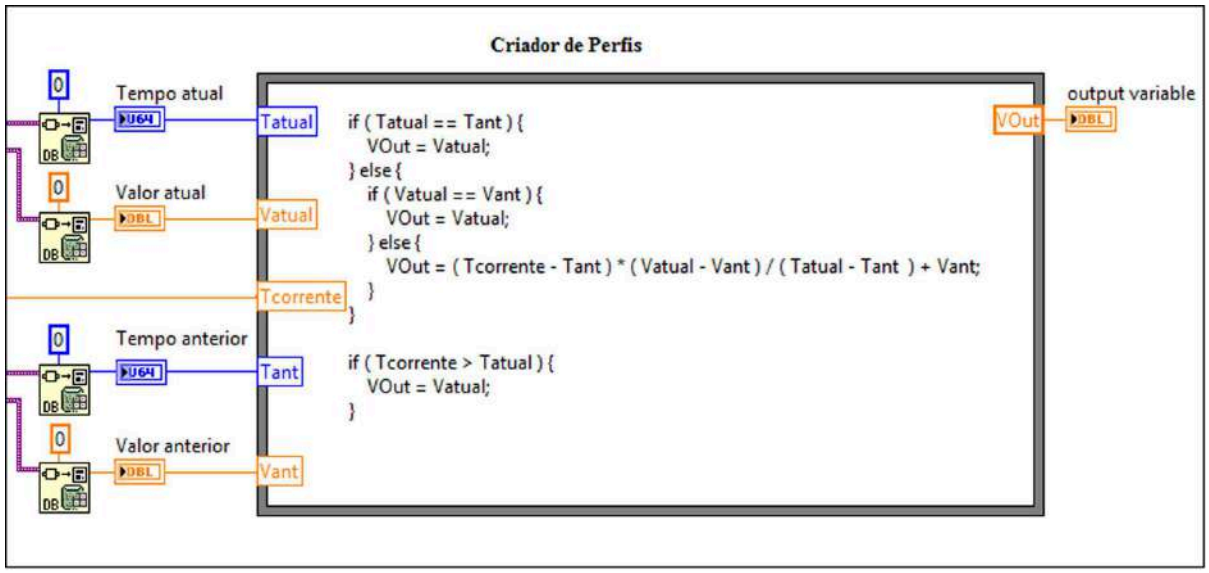

Figura 3.31: Parte do código para criação de perfil no levantamento de características.

3.5.5 Teste de controladores

Através da função *levantamento de características*, o usuário remoto pode configurar um experimento para obter o comportamento do sistema naquela configuração. Tendo em mãos a resposta em temperatura diante de uma entrada em porcentagem de potência conhecida, ele pode obter o modelo do sistema, projetar controladores e por meio da função *teste de controladores*, testar o desempenho desses controladores na planta.

Na função *teste de controladores* o usuário cria um perfil semelhante ao perfil criado para levantamento de características, no entanto, esse é um perfil de referência em valores de temperatura, os quais é desejável que o sistema siga, obedecendo as especificações de projeto, a partir da atuação do controlador. Na página do WebLab, o usuário deve informar o modelo de controlador que deseja utilizar e os seus parâmetros de acordo com o projeto realizado. Na aplicação atual, foram implementados controladores PID simples, um para cada resistência. O usuário deve informar também a resistência ou resistências em que deseja atuar e o sensor ou sensores de temperatura que deseja monitorar.

Apesar de ter sido implementado um modelo de controlador simples, têm-se todas as entradas e saídas da planta disponíveis, sendo assim, basta acrescentar outros tipos de controladores criando outros módulos de controle. Ou seja é possível expandir a função *teste de controladores* para oferecer outras opções ao usuário. Além disso, pode-se no futuro implementar um espaço na página de testes de controladores em que o usuário insira funções personalizadas.

A Figura 3.32 mostra a SubVI que recebe do banco de dados os valores referentes ao perfil de temperatura na função *teste de controladores* e a SubVI que recebe os ganhos PID dos controladores. A Figura 3.33 mostra a SubVI que contém a função do modelo

do controlador.

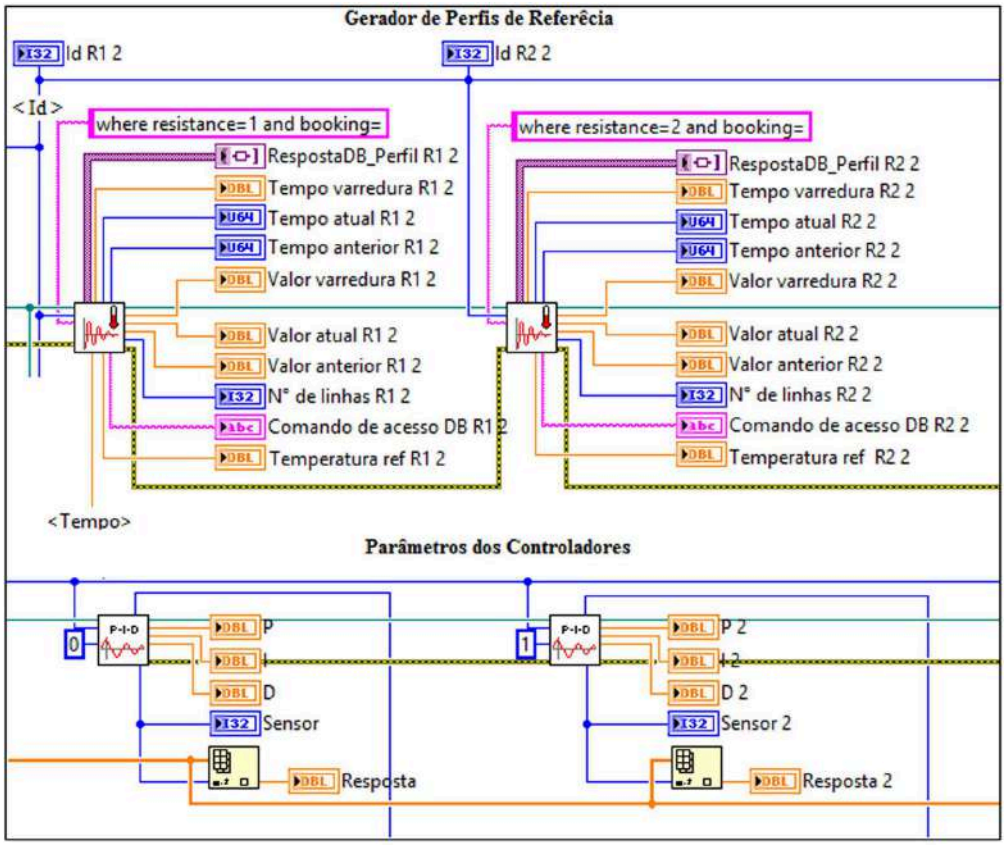

Figura 3.32: Perfis de referência de temperatura e ganhos PID.

3.5.6 Finalização

Ao término de cada uma das funções descritas acima o programa é direcionado para o estado Finalização. Nesse estado, o banco de dados é informado que o experimento chegou ao fim. Ao mesmo tempo, o LabVIEW® faz papel de cliente chamando uma URI do servidor Web, pelo método *GET* do HTTP, que tratará os dados salvos e enviará os resultados para o *e-mail* do usuário. Feito isso, o programa é direcionado para o estado Aguarda Finalização, que apenas espera um tempo para garantir que o agendamento foi finalizado e redireciona o programa para o estado Verificação de Registro, que por sua vez fica aguardando o próximo agendamento. A Figura 3.34 mostra as VIs do estado Finalização. Entre um agendamento e outro, foi assegurado, no código, um periodo de tempo para que a planta volte à condição ambiente antes de começar o próximo experimento. Esse tempo não foi estabelecido, sendo necessário um experimento para validar um intervalo suficiente para tal propósito

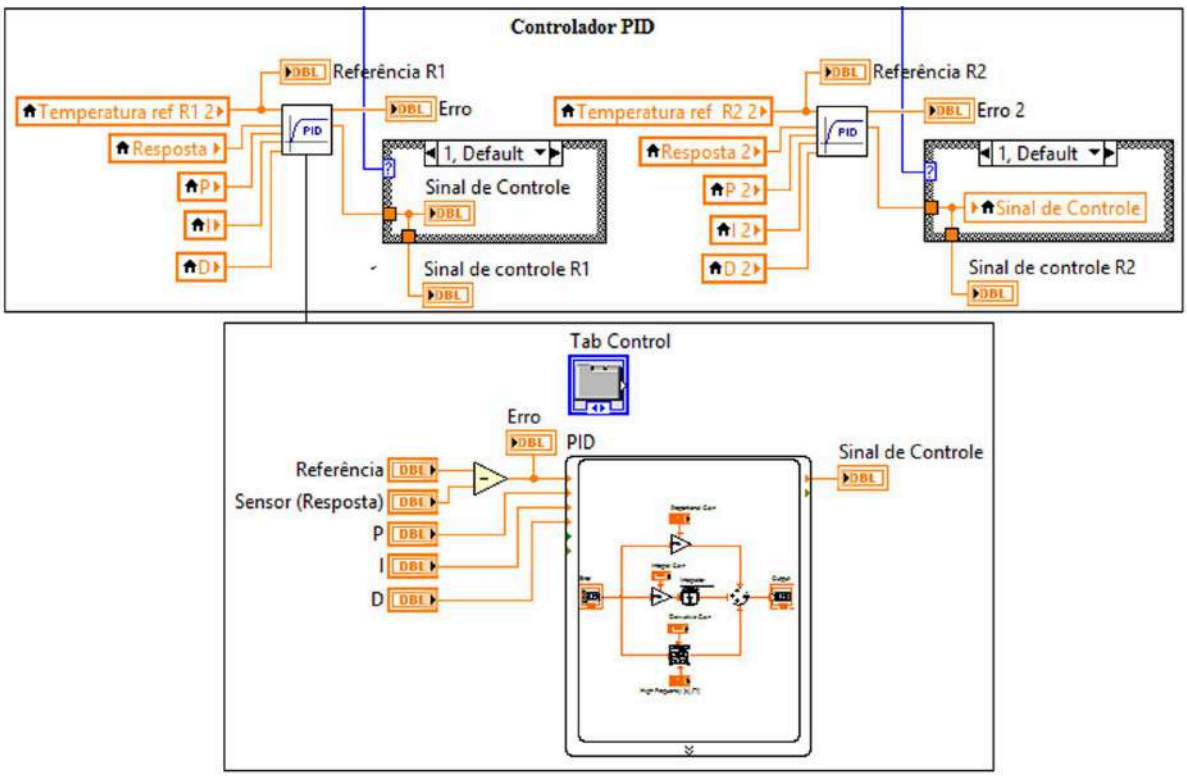

Figura 3.33: Desenho esquemático do controlador PID.

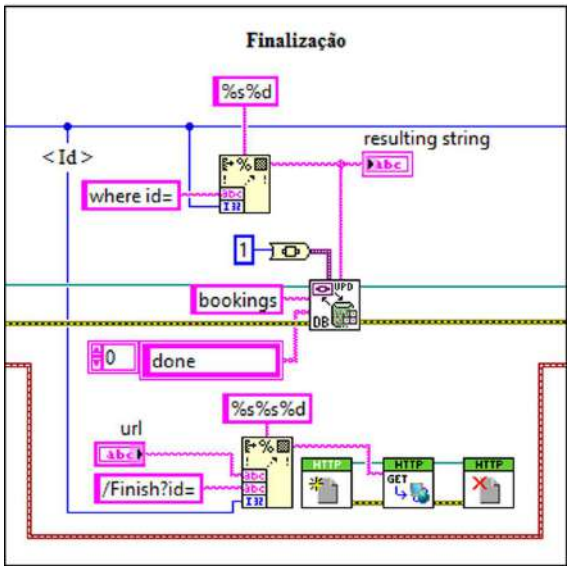

Figura 3.34: Desenho esquemático do estado de finalização do experimento.

3.5.7 Aquisição de dados

Para comunicação com o sistema de aquecimento de ar foi utilizada a VI *NI DAQ Assistant*. Essa VI é um código de alto nível que permite a configuração da placa de aquisição de dados para diversos tipos de aplicações. Por ser uma VI *menu-driven*, ou seja, configurada através de arquivos de menu, o seu uso diminui o risco de erro de programação e diminui o tempo de desenvolvimento.

A VI *DAQ Assitant* foi utilizada no WebLab para leitura dos sinais de temperatura dos sensores $LM35$ e termopares, para escrita dos sinais de tensão para comando dos atuadores e para envio de um sinal digital para ligar e desligar o sistema de aquecimento de ar. A Figura 3.35 apresenta as Sub-VIs em que a VI *DAQ Assistant* foram encapsuladas para exercerem as funções citadas acima.

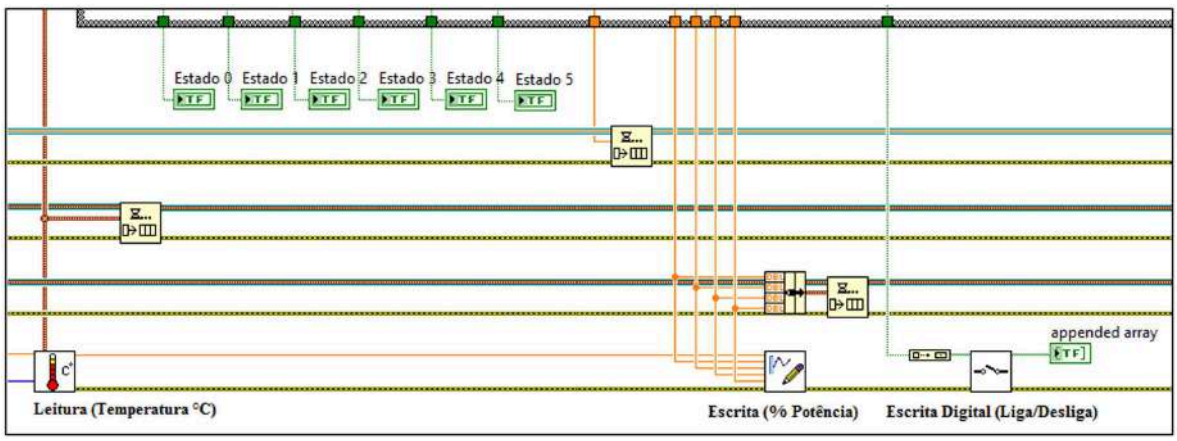

Figura 3.35: SubVIs para aquisição de dados.

A Figura 3.36 apresenta as VIs utilizadas para criação da função de aquisição de dados de temperatura. Nela, são configuradas a taxa de aquisição de dados e o número de amostras. Esses parâmentros sincronizam a velocidade de execussão de todo o programa. Já na Figura 3.37 é possível observar as VIs que compõem a função que comanda os atuadores através de sinas representados em porcentagem de potência. Como os relés de estado sólidos são acionados por sinais de tensão, a SubVI de escrita de dados envia ao microcontrolador sinais que variam de 0 a 5 Volts. Um microcontrolador então processa esses sinais e os interpretam como ordens para variar a largura do pulso de controle dos SSRs. Dessa forma, comando em porcentagem de potência é a variação do pulso de controle de 0 a 100% dos ciclos de tensão da rede fornecidos à carga.

A Figura 3.38 apresenta um diagrama esquemático das VIs responsáveis por comandar o acionamento e desligamento geral da planta. A SubVI de escrita digital está ligada a uma linha de saída digital de uma porta de 8 bits da placa de aquisição de dados que ativa ou desativa o circuito de potência, que por sua vez ativa ou desativa a contatora para alimentar os circuitos do sistema.

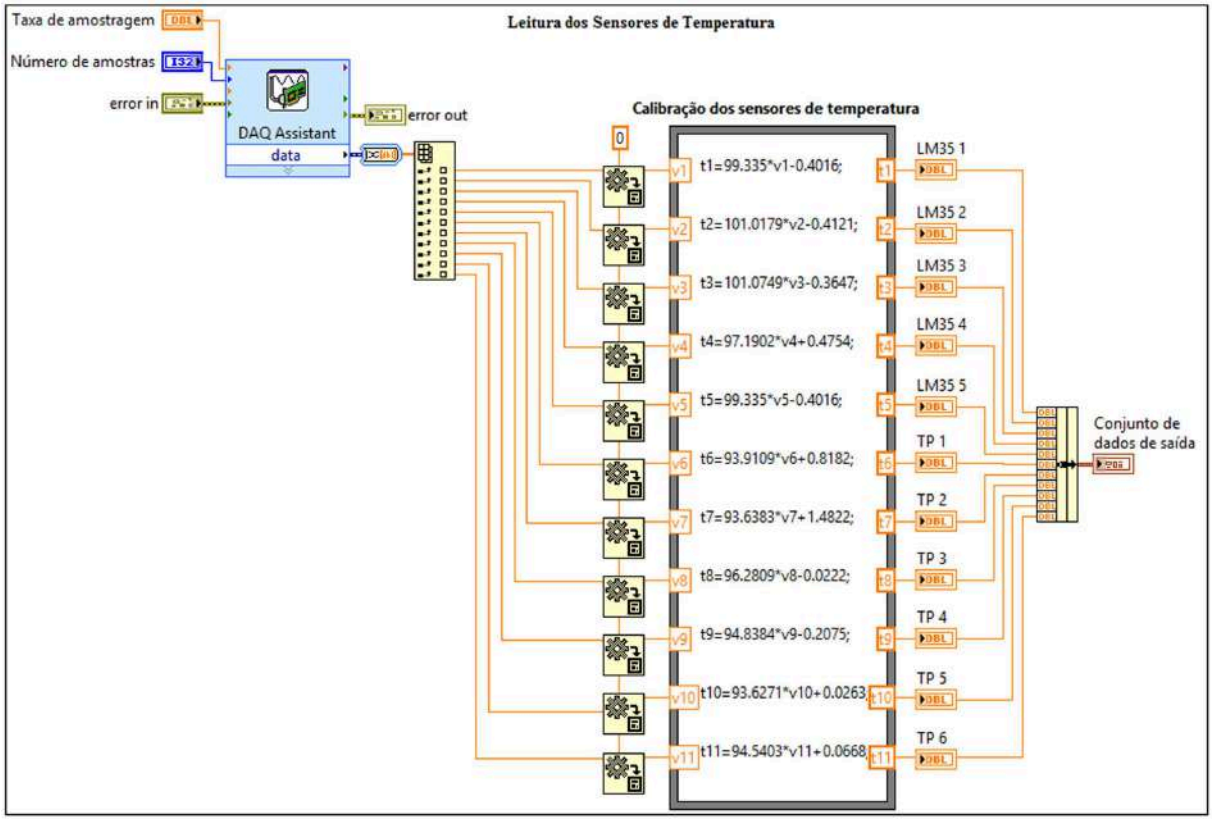

Figura 3.36: Desenho esquemático para leitura dos sensores de temperatura.

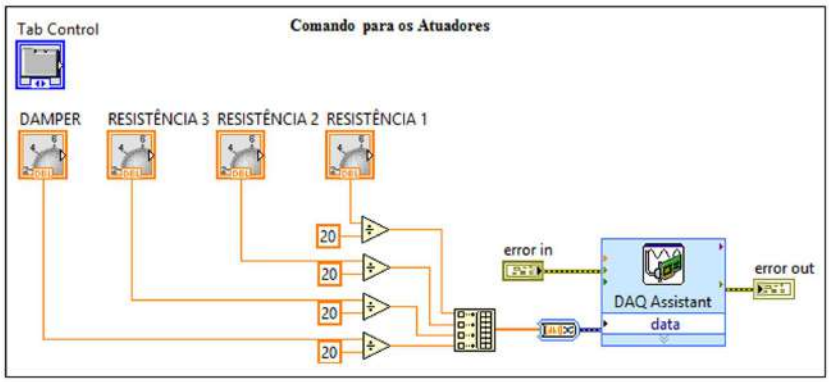

Figura 3.37: Desenho esquemático dos comandos dos atuadores.

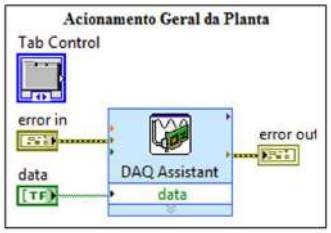

Figura 3.38: Desenho esquemático do sistema de acionamento geral.

3.5.8 Gravação dos dados no banco de dados

Durante qualquer uma das três fuções disponíveis para o usuário remoto os dados de tempertatura, de controle e tempo são gravados no banco de dados. Para tanto é utilizada a VI *Insert Data* do *Database Connectivity Toolkit*. A Figura 3.39 apresenta o modo como os dados que são gravados no banco de dados, equanto que na Figura 3.40 é possível observar o desenho esquemático da VI para comunicação e geração das informações no banco de dados. Para sincronizar a seleção e inserção de informações no banco de dados são utilizados semáforos.

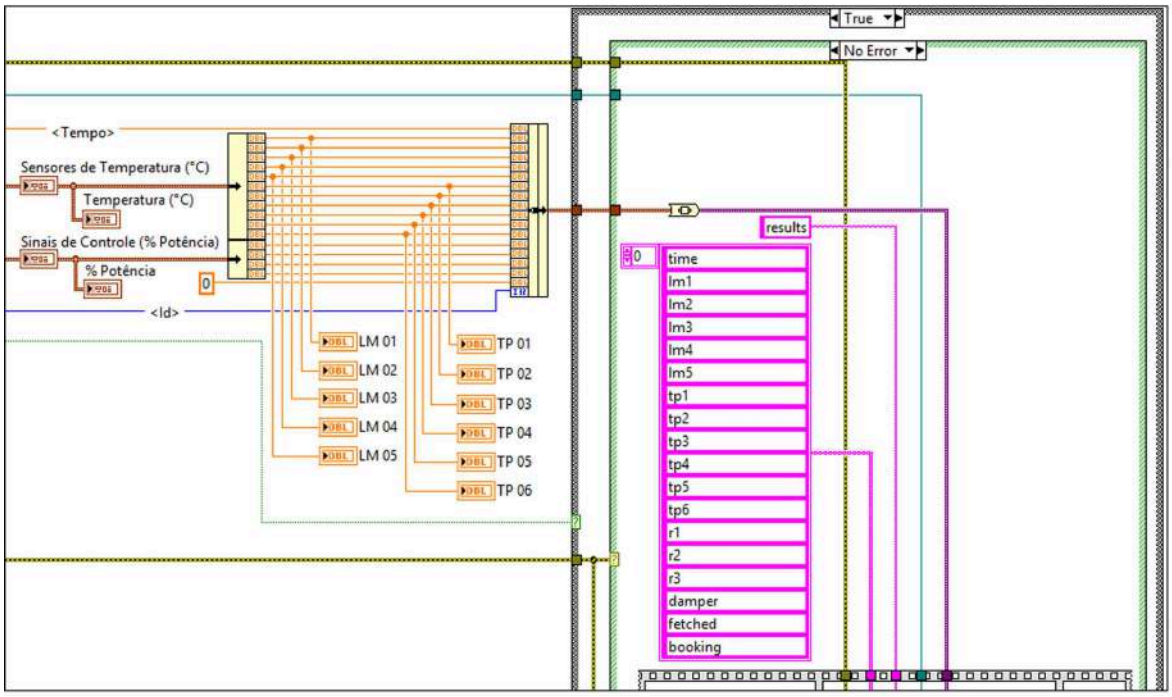

Figura 3.39: Desenho esquemático da estrutura do VI para gravação dos dados.

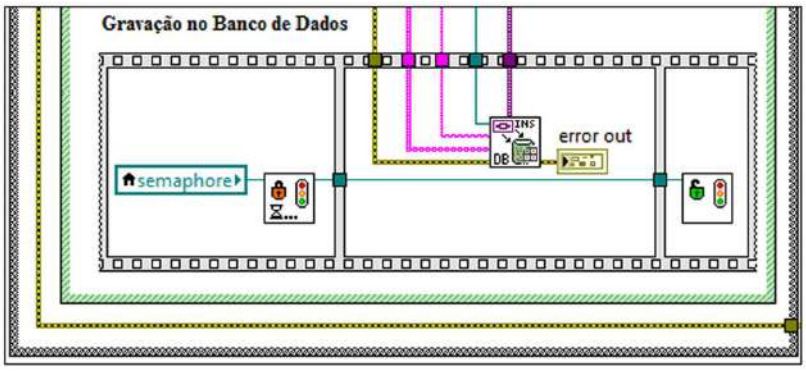

Figura 3.40: Desenho esquemático do VI para comunicação e gravação dos dados.

3.6 Desenvolvimento da Arquitetura Cliente-Servidor e do Banco de Dados

Foram vistos até aqui as questões referêntes ao blocos planta, instrumento virtual e parte do banco de dados, este último, na comunicação com o *instrumento virtual* como representado no modelo ilustrado na Figura 3.1. Na presente seção, serão abordadas as questões referentes aos blocos Cliente, Servidor, Servidor SMTP e outra parte do banco de dados, agora, na comunicação com o servidor.

3.6.1 Programas cliente e servidor

Na seção 2.4 foi visto que o cliente e o servidor do ponto de vista técnico são dois programas, e ambos rodam no computador servidor. Esses programas são desenvolvidos juntos e a separação entre eles só existe do ponto de vista do usuário, através da interface Web criada a partir de tecnologias como *JavaServer Pages* (JSP) e interpretada pelo navegador Web do lado do usuário. Ainda do ponto de vista mais alto do modelo apresentado na Figura 3.1, o servidor é o programa que interpreta os pedidos do cliente e retorna a resposta do que foi requisitado. Para estender as aplicações hospedadas nos servidores Web, existem os *Servlets* que podem ser entendidos como *Applet Java* que rodam dentro do servidor. No caso do WebLab, o programa servidor é o Apache.

Os servidores Web funcionam seguindo uma estrutura baseada em requisições. Nesta arquitetura, o programa servidor se conecta à uma porta de rede (normalmente 80 (HTTP) e/ou 443 (HTTPS)) com protocolo TCP e aguarda a conexão de um cliente. Quando o cliente se conecta, ele requisita um recurso. O servidor recebe esta requisição e executa uma ação de acordo com o recurso que foi pedido.

Na Figura 3.41 pode ser observada a barra de endereço de um *browser* que comunica com o servidor enviando requisições. A URL completa é http://localhost:8084/WebLab/index.jsp. No ambiente de produção essa URL se reduziria para, por exemplo, *http://localhost/index.jsp*, depois de removidas as partes adicionadas para desenvolvimento e testes. As partes da URL completa são descritas a seguir:

- *http*: é o protocolo utilizado. Este é o protocolo padrão de *Websites*.
- *localhost*: é nome do servidor. Neste caso, como o aplicativo está rodando em uma máquina de testes, ele não possui um nome de domínio (DNS), sendo assim, é utilizado o nome *localhost*, que é padrão para referenciar a própria máquina. Ele resolve para o IP 127.0.0.1, que fica atrelado à interface de "*loopback*" da rede.
- 8084: é a porta em que se deve conectar. Como dito anteriormente, utiliza-se por padrão para HTTP a porta 80, mas por se tratar de um ambiente de testes

Figura 3.41: Barra de endereços URL.

foi utilizada a porta 8084. Caso a porta 80 fosse utilizada, não seria necessário especificar a porta.

- */WebLab/index.jsp*: é o recurso requerido. Novamente, por se tratar de um ambiente de desenvolvimento, foi adicionado uma parte extra, *"/WebLab"*. O servidor, no entanto, sabe interpretar essa porção e que o recurso dentro do aplicativo é *"/index.jsp"*.

A Figura 3.42 ilustra o ambiente de desenvolvimento do *NetBeans* utilizados para criar a aplicação Web. A árvore projeto, do lado esquerdo mostra os módulos *.JSP* e *.java* que constituem o aplicativo. O código dos programas estão disponíveis no manual do WebLab ou no CD em anexo e a seguir será apresentado um resumo dos recursos implementados.

.JSP

- *addbooking.jsp*: utilizada apenas internamente para adicionar um novo agendamento.
- *addcontrollerstest.jsp*: formulário para entrar com dados de agendamento de testes de controlador.
- *addfeaturessurvey.jsp*: formulário para entrar com dados de agendamento de levantamento de características.
- *addfreebooking.jsp*: formulário para adicionar um agendamento livre e controlar um em andamento através do arquivo *controlfreebooking.jsp*.
- *controlfreebooking.jsp*: página que mostra os valores atuais dos sensores e permite mudar livremente os valores dos atuadores.
- *controllerstest.jsp*: lista de testes de controladores agendados e *link* para *addcontrollerstest.jsp*.

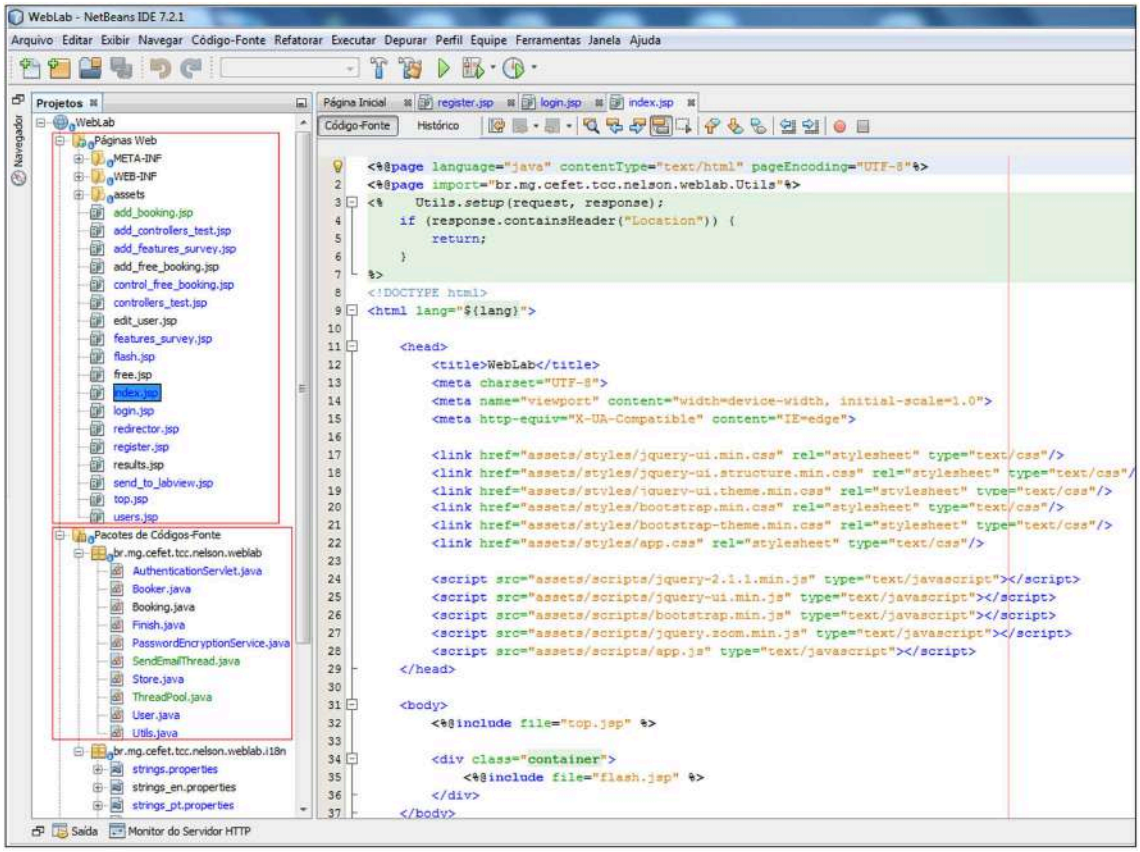

Figura 3.42: Ambiente de desenvolvimento *NetBeans*.

- *edituser.jsp*: formulário para editar as informações de usuário.
- *featuressurvey.jsp*: lista de levantamentos de características agendados e *link* para */addfeaturessurvey.jsp*.
- *flash.jsp*: exibe mensagens de erro/sucesso/aviso nas caixinhas vermehas/verdes/azuis que aparecem, por exemplo, quando as alterações em um usuário são salvas.
- *free.jsp*: "costura" *add_free_booking.jsp* e *control_free_booking.jsp* em um único lugar.
- *index.jsp*: página inicial, contendo o menu superior e uma mensagem de boas vindas.
- *login.jsp*: página onde o usuário digita seu nome de usuário e senha para entrar no sistema. Ele será redirecionado automaticamente para esta página caso tente acessar qualquer outra página sem ter sido autenticado.
- *register.jsp*: página de registro de novo usuário.
- *results.jsp*: página chamada pelo *javascript* em *controler_free.jsp* para obter do banco de dados os valores atuais dos sensores.

- top.jsp: página com o topo preto que aparece depois de logado, contendo os *links* das funções do WebLab.

- *users.jsp*: lista os usuários e dá a opção de remove-los.

.java

- *AuthenticationServlet.java*: responsável por autenticar o usuário. É aqui que o nome de usuário e senha são validados.

- *Booker.java*: classe que processa as informações dos formulários relativas a agendamentos. Prepara os dados para serem inseridos no banco de dados.

- *Finish.java*: utilizada ao fim de um agendamento, esta classe é responsável por processar os dados do banco (resultados do agendamento) e formata-los como CSV para ser enviado para o email do usuário que agendou o teste.

- *PasswordEncryptionService.java*: criptografa a senha de usuário. Utilizada em dois momentos: quando o usuário salva uma senha e quando ele se autentica no sistema. O algoritmo criptográfico utilizado é o $PBKDF2WithHmacSHA1$. Ele possui como vantagem a necessidade de um tempo para a geração da chave, o que impede um ataque de força bruta. Como é gerada uma chave e um "*salt*" para cada senha, esta não pode ser atacada com ataques de dicionário. Neste sistema este esquema criptográfico é utilizado apenas para proteger a senha, para proteger dados sensíveis é aconselhável um esquema mais elaborado.

- *SendEmailThread.java*: envia o *e-mail*. Feita pra ser chamada em outra *thread*, já que isso pode levar um tempo.

- *Store.java*: nesta classe são feitas todas as transações do banco de dados, por exemplo, cláusulas INSERT, DELETE, SELECT, UPDATE, CREATE, etc.

- *ThreadPool.java*: mantém uma *thread* rodando e enfilera o que tem que ser executado. Cada um é executado por vez, um atrás do outro, na mesma *thread*. Usado para enviar *e-mails*.

- *User.java*: Abstrai usuário.

- *Utils.java*: Alguns métodos de utilidade geral.

A seguir, um exemplo de dinâmica de uma requisições de um usuário que deseja agendar um teste de controlador (assumindo que ele já está autenticado):

1. O usuário acessa o aplicativo, requisitando a página com o formulário de cadastro de teste de controlador, *addcontrollerstest.jsp*.

2. O usuário preenche o formulário que foi retornado pelo servidor na requisição anterior e clica em "agendar". Ao clicar no botão o navegador irá enviar uma nova requisição ao servidor para o recurso */booking* contendo as informações do formulário anexadas.

3. O algoritmo no arquivo *Booker.java* será executado. Este por sua vez irá processar os dados anexados e decidir qual tipo de agendamento foi solicitado, bem como normalizar todos os dados, deixando-os prontos para serem inseridos no banco de dados. Para isso ele usa a classe *Store*.

4. A classe *Store*, implementada no arquivo *Store.java*, recebe os dados da classe *Booker*, já formatados, e os grava no banco usando instruções SQL. Após os dados serem gravados, ou caso algum erro aconteça, a execução retorna para a classe *Booker*.

5. A classe *Booker* verifica se a classe *Store* conseguiu ou não escrever os dados no banco, e responde à requisição inicial de acordo com a situação. Caso *Store* tenha sucedido, responde à requisição mandando o navegador fazer uma requisição para *controllerstest.jsp*. Caso tenha falhado, responde à requisição mandando o navegador fazer uma requisição para *addcontrollerstest.jsp* e guarda uma mensagem de erro internamente, que será exibida ao usuário após o redirecionamento.

Após este agendamento, o LabVIEW® será capaz de iniciar o teste no horário certo, já que ele lê o mesmo banco em que o aplicativo Web escreveu. Durante o teste ele escreve todas as informações necessárias à interpretação dos dados do teste no banco de dados. Ao fim do teste o LabVIEW® age como navegador e realiza uma requisição para *"finish"*, que executará a rotina do arquivo *Finish.java*, explicada anteriormente.

3.6.2 Banco de dados

Para desenvolver o banco de dados foi feito um levantamento das variáveis necessárias para todo o processo como explicado na seção 3.6. Por se tratar de um banco de dados relacional, o problema foi dividido em duas partes: usuário e agendamento.

A tabela usuários foi feita pensando-se em quais dados seriam necessários para a autenticação e identificação. No primeiro caso, percebeu-se a necessidade de um nome de usuário e senha. Já o segundo caso, foi subdividido em identificação mecânica e identificação pessoal. Na identificação pessoal, é requerido o nome da pessoa, o número de matrícula e o endereço de *e-mail* para envio dos resultados. Para a identificação mecânica

é necessário saber se o usuário esta ativo e qual o nível de acesso ele tem (*administrador*, *professor* ou *aluno*).

Para o agendamento é necessário conhecer qual o usuário, a data de início do experimento, assim como o tipo de operação, além dos dados do agendamento. Pela natureza dos dados do agendamento, foram criadas tabelas específicas para eles. Desta forma o número de dados relacionados a um agendamento é restringido apenas por limitações físicas (limite da memória). Era necessário também guardar os resultados, para tanto, foi criada uma tabela específica, seguindo a mesma lógica das tabelas de dados de agendamentos. Assim foram criadas as tabelas representadas no diagrama da Figura 3.43. A tabela *current booking* contém os dados do agendamento atual. Esses dados são informados ao LabVIEW® no estado *verificação de registros*. Portanto, quem controla o início do experimento é o próprio banco de dados, a partir das isntruções fornecidas pelo usuário.

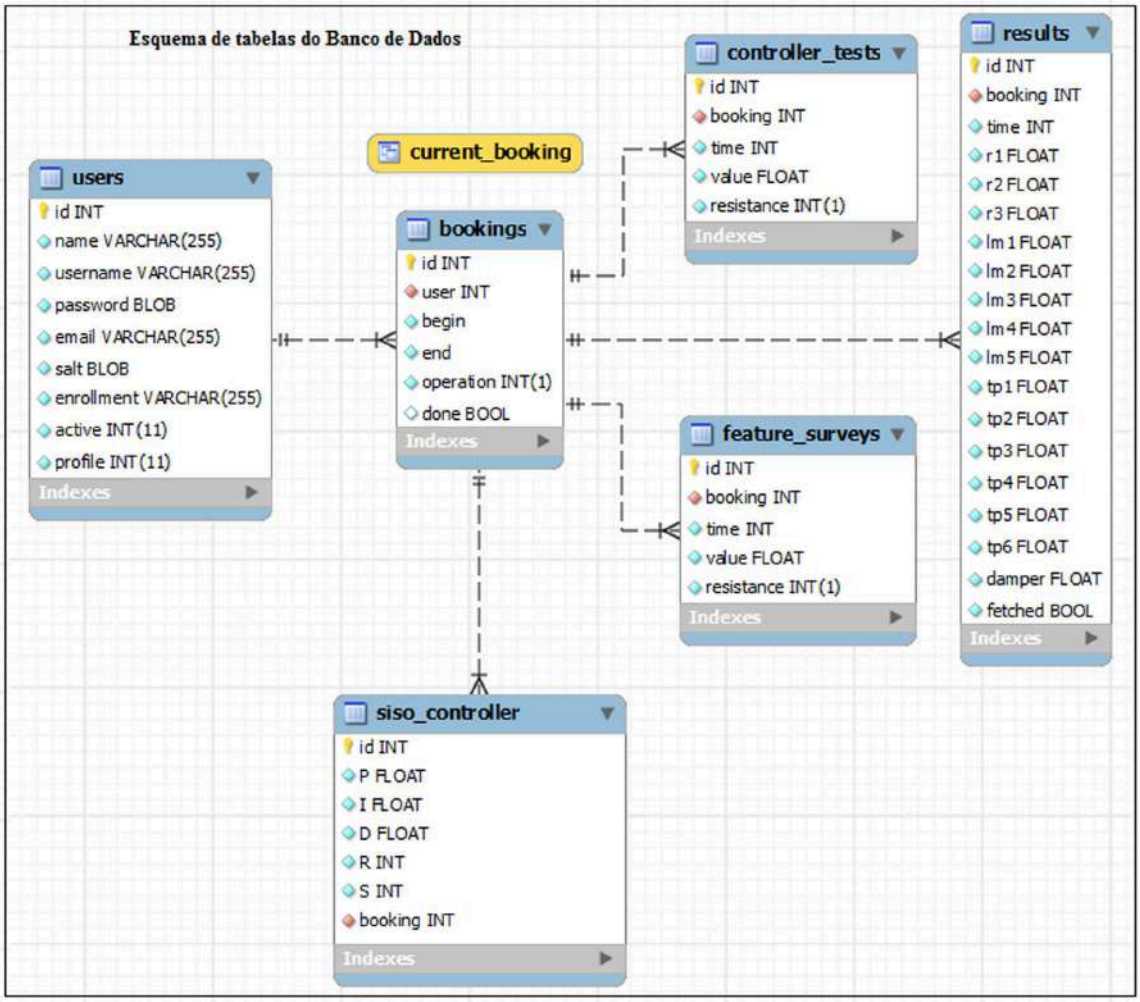

Figura 3.43: Diagrama das tabelas do banco de dados.

3.6.3 Servidor SMTP

O protocolo de transferência de correio simples do inglês *Simple Mail Transfer Protocol - SMTP*, é um protocolo padrão para envio de *e-mail* através da internet. No WebLab esse recurso foi utilizado para enviar os resultados do experimento para o *e-mail* do usuário remoto. Como explicado anteriormente, o *instrumento virtual* faz o papel de cliente e chama uma URI do servidor Web, que por sua vez comunica-se com o banco de dados, tratará os dados salvos e envia o resultado em formato CSV para um servidor de *e-mail* SMTP que o reenvia para o usuário.

O uso de um SMTP padrão associados a provedores gratuitos não garante a entrega correta do *e-mail*. Isso ocorre devido aos padrões de segurança de cada provedor. No entanto, existem serviços SMTP profissionais, que oferecem mais liberdade para configuração do recurso. Para o WebLab foi utilizado um provedor gratuito a fim de não gerar custos. A configuração padrão de um SMTP consiste basicamente em selecionar as configurações da conta, escolher a opção de servidor de saída SMTP e preencher os campos de configurações. A Figura 3.44 ilustra uma caixa de texto para configuração. Ela contém basicamente:

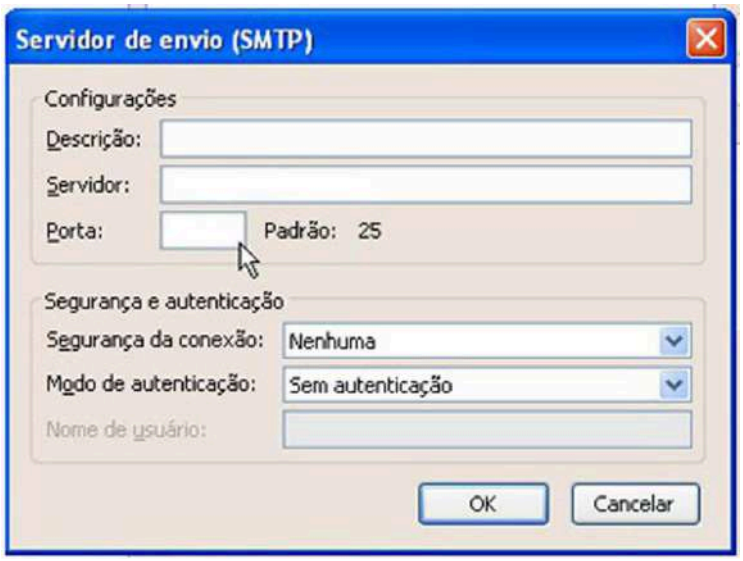

Figura 3.44: Configurações do servidor SMTP.

- **Descrição:** nome informal que irá identifica o servidor (será melhor utilizar o provedor de *e-mail*, como *Gmail* ou *Yahoo*).
- **Servidor:** especificação do servidor SMTP (URL e SMTP), por exemplo, *Gmail.com* e *Smtp.gmail.com* respectivamente.
- **Porta:** normalmente as portas 25 ou 587.

- **Segurança da conexão:** para uma ligação mais segura, pode-se utilizar uma extensão STARTTLS ou SSL/TLS, que utilizam uma porta separada para comunicação encriptada.
- **Modo de autenticação:** métodos de segurança. Sugere-se usar uma senha usando endereço de *e-mail* como nome de usuário.
- **Nome de usuário:** o próprio endereço de *e-mail*.

Capítulo 4

Resultados Experimentais

A seguir serão mostrados os resultados práticos obtidos. Tais resultados serão abordados de acordo com as funções disponíveis para o usuário remoto e serão discutidos com base em fluxogramas, seguidos da apresentação dos dados gravados no banco de dados e retornados por e-mail pelo sistema. Os fluxogramas dão continuidade a aquele representado na Figura 3.1 e têm como objetivo ilustrarem os programas desenvolvidos. Já a análise dos dados dos experimentos tem como objetivo principal avaliar as funcionalidades desses programas e de todos os módulos do sistema apresentados na Figura 3.8. Todas as aquisições foram feitas a $10Hz$ e considera-se que o cliente tenha acesso como administrador.

Levantamento de características

A Figura 4.1 ilustra um fluxograma que representa o programa desenvolvido para atender os requisitos de projeto para a função *levantamento de características*. Nele, estão destacados cada um dos módulos que deram origem ao modelo do WebLab (Figura 3.8). Além disso, é possível observar os estados envolvidos na operação de *levantamento de características* e as etapas (de 1 a 16) que ocorrem durante a sua execução. Nota-se que algumas dessas etapas ocorrem paralelamente e que todo o processo é coordenado pelo servidor. O programa servidor é responsável por receber as requisições do programa cliente e definir qual método será executado (seja pelo banco de dados ou pelo instrumento virtual), em seguida, o servidor recebe as respostas de tais métodos, as manipula e responde ao programa cliente.

Para validar o função *levantamento de características*, foi agendado um experimento no WebLab. Tal experimento foi feito com a intenção clara de que seus resultados seriam aplicados para obter o modelo do sistema e projetar um controlador para posteriormente testá-lo na função *teste de controladores*. Dessa forma, foi escolhido um ponto de operação e criado um perfil em degraus sequenciais.

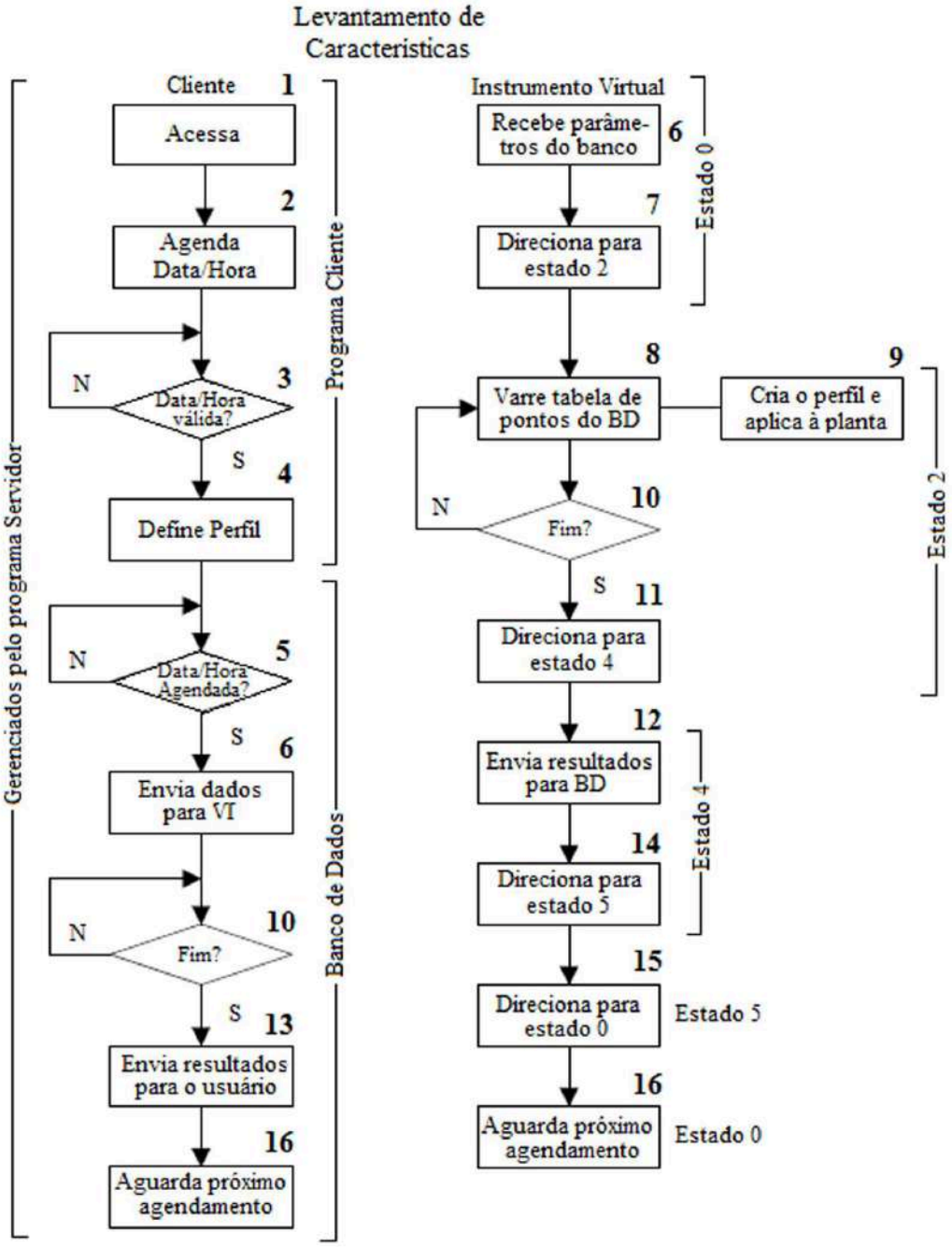

Figura 4.1: Função levantamento de características.

A Figura 4.2 mostra o perfil programado para o teste. O ponto de operação escolhido foi igual a 30% da potência total que pode ser aplicada nos atuadores. Dessa forma, os degraus sequenciais foram variados 10% acima e 10% abaixo, em torno desse ponto de operação, seguindo a sequência 20%, 30%, 40%, 30% e 20%. Como desejava-se projetar um controlador para um sistema tipo SISO, foi escolhida a resistência $R1$ e o sensor de temperatura $LM3$. A disposição desses componentes no sistema, pode ser visualizada na Figura 3.9.

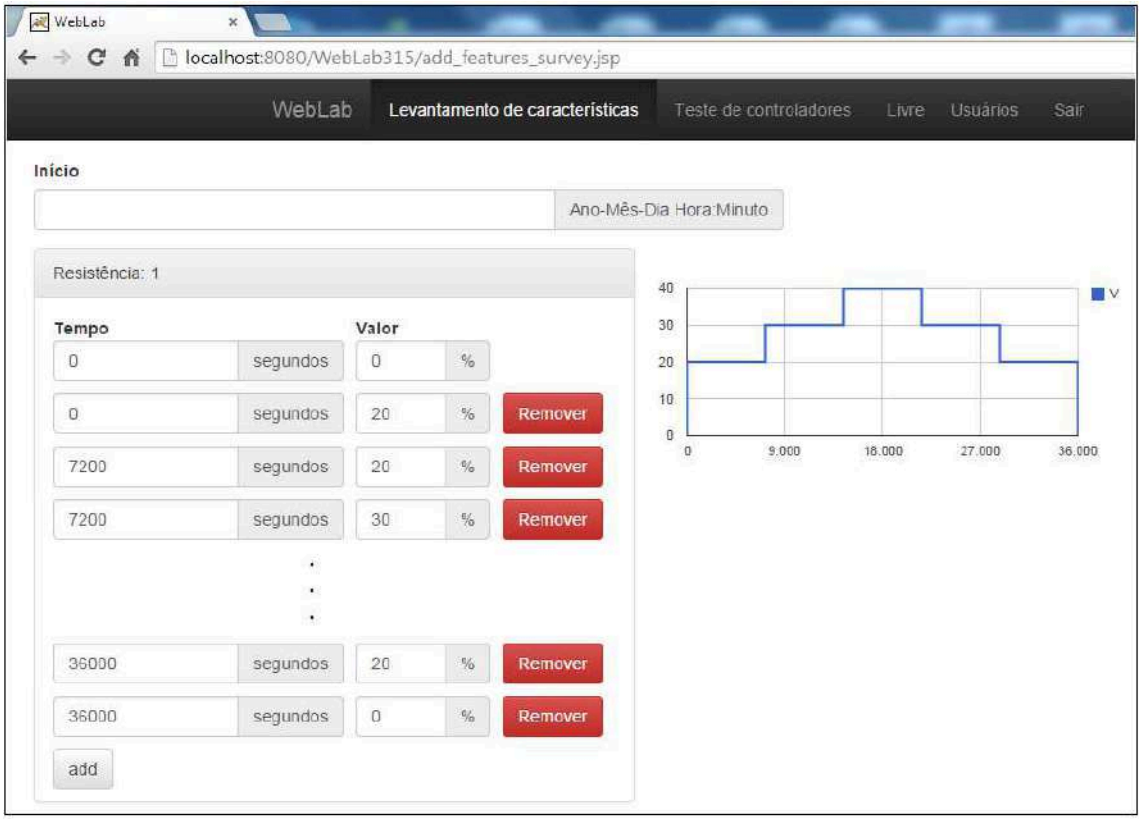

Figura 4.2: Perfil programado para levantamento de características.

A Figura 4.3 representa a resposta medida pelo sensor $LM3$ à sequência de degraus aplicada à resistência $R1$. Nota-se que, apesar do filtro analógico utilizado, como descrito na Subseção 3.3.6, o sinal ainda apresenta ruído. Acredita-se portanto, que tal ruído é proveniente do circuito de acionamento geral da planta, devido à presença de uma chave elétro-magnética (contator). Para contornar esse problema, foi utilizado um filtro digital passa-baixas com frequência de corte igual a $0{,}005Hz$ de acordo com a Equação (4.1). A Figura 4.4 mostra o sinal do sistema após ser aplicado o filtro. Verifica-se que ele não foi falseado, assim, a dinâmica do sistema foi coerentemente mantida.

$$F(s) = \frac{P}{s+P} = \frac{0{,}0314rad/s}{s+0{,}0314rad/s} \tag{4.1}$$

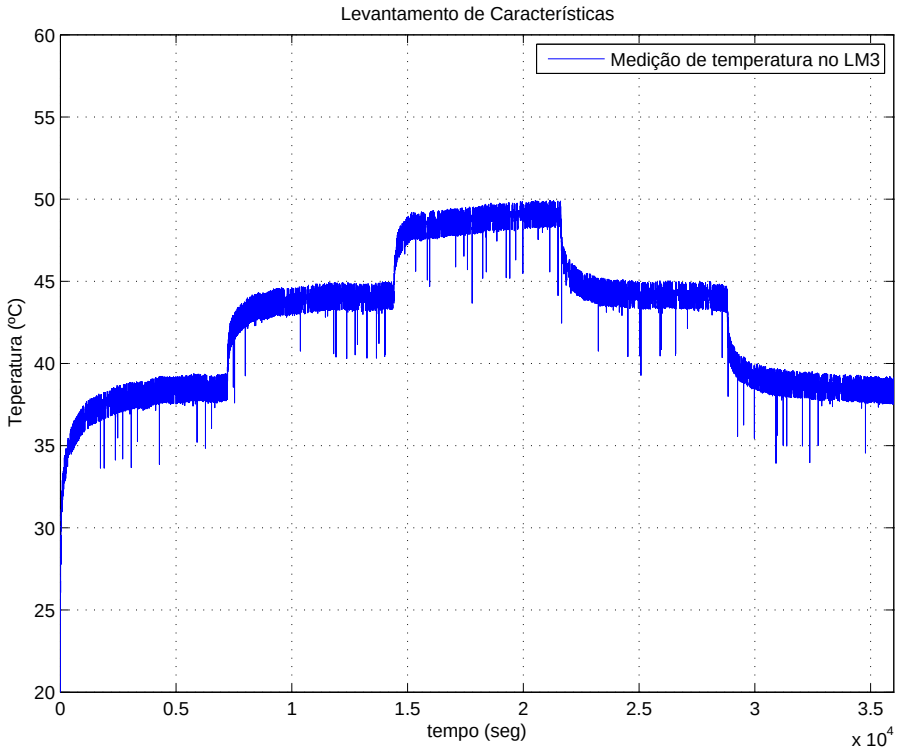

Figura 4.3: Sinal obtido durante o teste de levantamento de características.

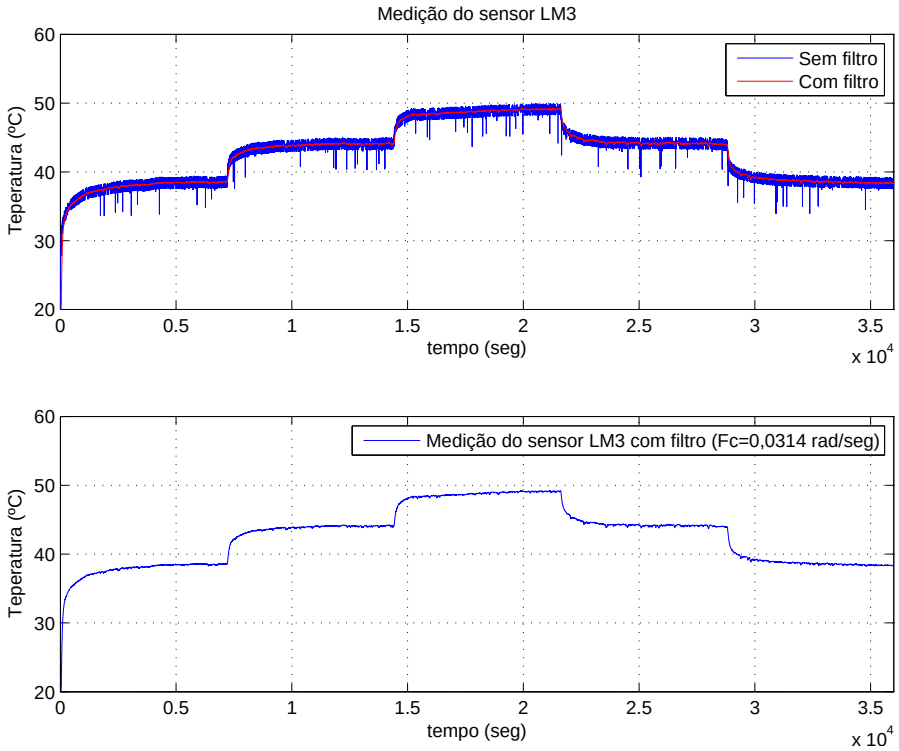

Figura 4.4: Sinal obtido durante o teste levantamento de características filtrado.

Uma vez conhecida a resposta do sistema, foi dado início à identificação. Para tanto, foi utilizado o método da resposta complementar para obtenção do modelo. O método da

resposta complementar é um método de identificação simples utilizado para obter modelos de baixa ordem. Tais modelos, obedecem a forma geral apresentada na Equação (4.2), em que K é o ganho do sistema, θ é o atraso e τ = é a constante de tempo. Esses modelos são aplicáveis a sistemas estáveis e superamortecidos. Além disso, assume-se que a entrada para excitação do sistema seja em degrau. Outro ponto importante, é que a coleta dos dados para identificação, deve partir sempre de uma condição de equilíbrio e o experimento deve ser longo o bastante para garantir que o sistema alcance regime permanente.

$$G(s) = \frac{Ke^{-\theta s}}{\tau s + 1} \tag{4.2}$$

O ganho do sistema é obtido pela análise da resposta ao degrau e calculado conforme a Equação (4.3), cujos parâmetros são indicados na Figura 4.5. Ele é a razão entre a variação do sinal de saída pela variação do sinal de entrada. Para o modelo do sistema, foi estabelecido que $K = 0{,}5346$, sendo que esse valor foi obtido pela média dos ganhos calculados para os quatro degraus, os dois primeiros de subida ($K = 0{,}5448$ e $K = 0{,}5131$) e os dois últimos de descida ($K = 0{,}5194$ e $K = 0{,}5612$).

$$K = \frac{\Delta y}{\Delta u} = \frac{y(\infty) - y(0)}{u(\infty) - u(0)} \tag{4.3}$$

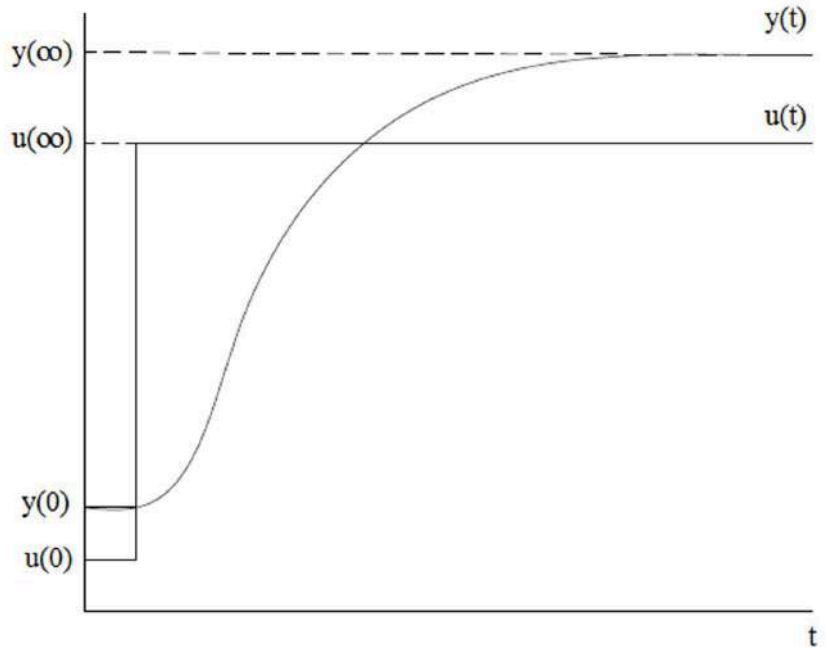

Figura 4.5: Definição do ganho K do sistema.

O modelo descrito pela Equação (4.2) pode ser representado no domínio do tempo pela Equação (4.4), em que $y(t)$ é o sinal de saída e $u(t)$ é o sinal de entrada.

$$y(t) = K(1 - e^{-(t-\theta)/\tau})u(t) \tag{4.4}$$

Assumindo que $y(\infty)$ é o valor do sinal de saída em regime permanente, substituindo $y(\infty)$ na Equação (4.4) e subtraindo $y(t)$ obtém-se a Equação (4.5), que pode ser reduzida à Equação (4.6).

$$y(\infty) - y(t) = K(1 - e^{-(t-\infty)/\tau})u(t) - K(1 - e^{-(t-\theta)/\tau})u(t) \tag{4.5}$$

$$y(\infty) - y(t) = K(1 - e^{-(t-\theta)/\tau})u(t) \tag{4.6}$$

Após algumas manipulações algébricas chega-se à Equação (4.7). Plotando o primeiro termo dessa equação em função de t, obtém-se uma curva cujos primeiros valores resultam em uma região quase linear de inclinação $-1/\tau$. Dessa forma, fazendo-se uma aproximação por uma reta nessa região, obtém-se o coeficiente angular, cujo valor é aproximado do valor do τ real. Para o modelo do sistema, o valor de τ foi estimado a partir da média dos valores de τ obtidos para os quatro degraus, que foram respectivamente 687,1, 481,9, 876,1 e 1419,6, resultando em $\tau = 866.2$. A Figura 4.6 ilustra o ajuste feito para o primeiro degrau.

$$\ln\left(\frac{y(\infty) - y(t)}{Ku(t)}\right) = \frac{-1}{\tau}t \tag{4.7}$$

Por último, o valor de θ foi obtido visualmente observando-se o início do degrau e o início da resposta do sistema. Da mesma forma, foi feita uma média dos quatro valores de θ referentes a cada degrau, sendo obtido os seguintes valores: 2,7, 3, 6,3, 7,8, resultando em um atraso médio de $\theta = 4{,}95$. A utilização da média dos parâmetros do modelo, foi uma tentativa de fazer com que o modelo seja aplicável a outras faixas de entradas de referência, quando estas se afastam do ponto de operação. Determinados os valores de K, τ e θ, foi obtido o modelo da Equação (4.8). A Figura 4.7 ilustra a resposta do modelo obtido. Como o foco do projeto é a automação do sistema para experimentação remota, não foi feita uma validação criteriosa do modelo.

$$G(s) = \frac{0{,}5346e^{-4{,}95s}}{866.2s + 1} \tag{4.8}$$

Teste de controladores

De posse do modelo do sistema, foi projetado um controlador PI (Proporcional Integral). Para tanto, utilizou-se o *Método da Síntese Direta*. Tal método é uma forma simples de síntese de controladores PI que permite ajustar o modelo para que, em malha

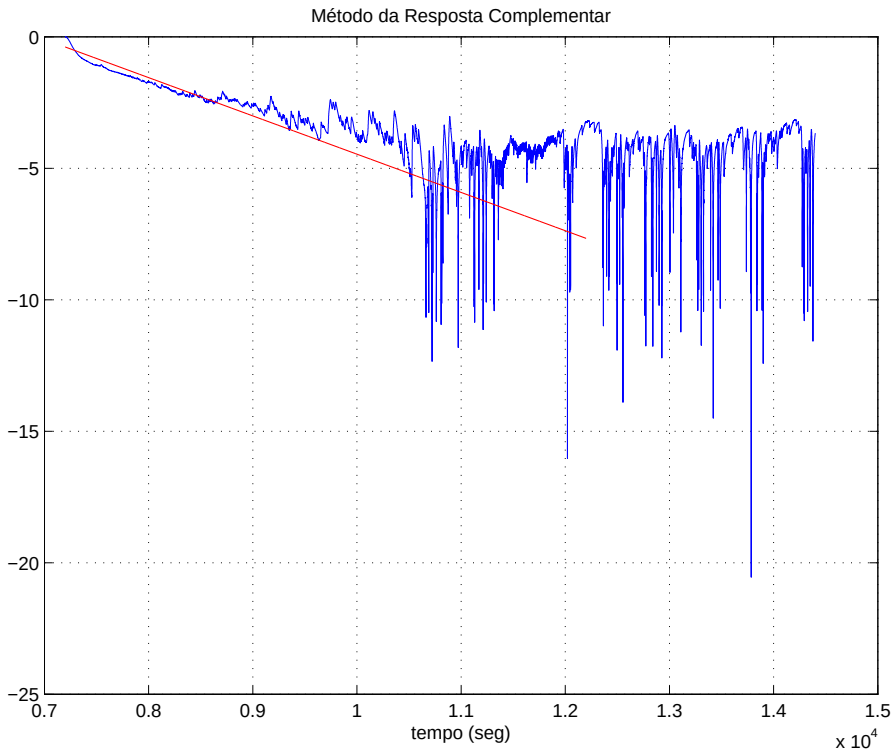

Figura 4.6: Determinação de τ para degrau de 20% para 30%.

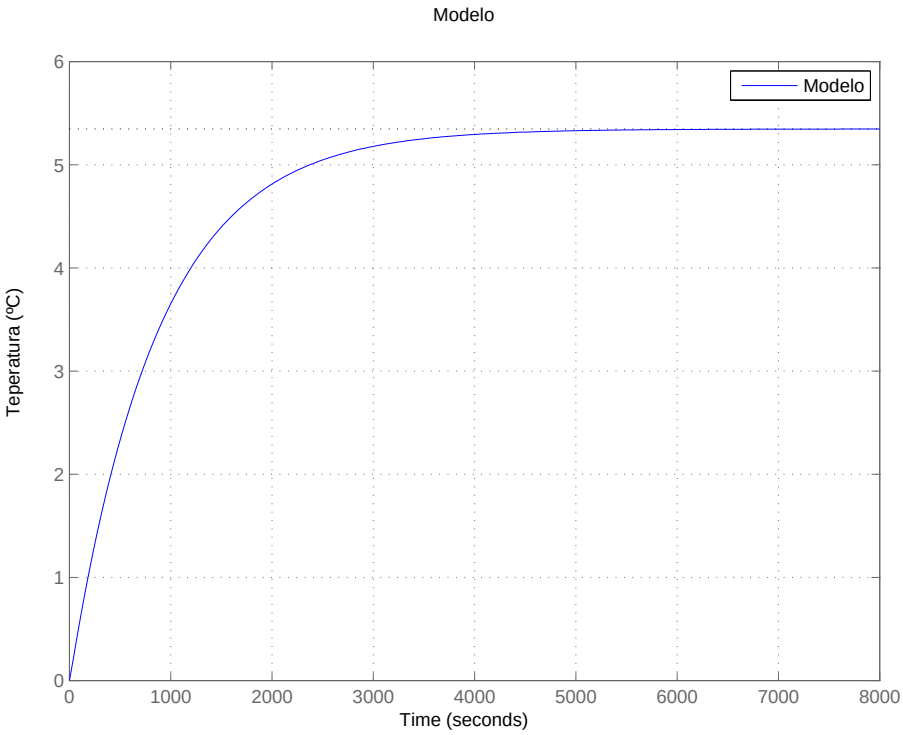

Figura 4.7: Resposta do modelo obtido.

fechada, possa assumir os parâmetros desejados pelo projetista. Portanto, o método faz com que o sistema em malha fechada, apresente um comportamento, conforme descrito pela Equação (4.9).

$$G_{MF}(s) = \frac{1}{\tau_d s + 1} e^{-\theta_d s} \tag{4.9}$$

O controlador que resulta na Equação (4.9) para o sistema em malha fechada é obtido da Equação (4.10). Utilizando-se a série de Taylor truncada em seu termo de primeira ordem tem-se que $e^{-\theta_d s} = 1 - \theta_d s$. Assim, substituindo em (4.10) obtém-se a Equação (4.11).

$$G_c(s) = \frac{1}{G(s)} \frac{e^{-\theta_d s}}{\tau_d s + 1 - e^{-\theta_d s}} = \frac{\tau s + 1}{K(\tau_d s + 1 - e^{-\theta_d s})} \tag{4.10}$$

$$G_c(s) = \frac{\tau s + 1}{K(\tau_d + \theta_d)s} \tag{4.11}$$

Para o controlador projetado, foi especificados que o erro em regime permanente fosse nulo e que a resposta do sistema se tornasse pelo menos duas vezes mais rápida. Portanto, aplicando-se o método da síntese direta, foi obtido o controlador indicado na Equação (4.12), que pode ser reescrito na forma indicada na Equação (4.13).

$$G_c(s) = \frac{\tau s + 1}{K(\tau_d + \theta_d)s} = \frac{\tau}{K(\tau_d + \theta_d)} \left(1 + \frac{1}{\tau s}\right) \tag{4.12}$$

$$G_c(s) = K_c \left[1 + \frac{1}{T_i s}\right] \tag{4.13}$$

Substituindo os parâmetros por seus respectivos valores, obteve-se a Equação (4.14). É importante lembrar que o valor do atraso desejado, deve ser maior ou igual ao valor de atraso real ($\theta d \geq \theta$), caso contrário, o sistema se torna não causal. As Figuras 4.8 e 4.9 permitem comparar as respostas do modelo obtido para o sistema em malha aberta e a resposta do sistema controlado. Deve-se, porém, se ater à forma do sinal, pois as entradas do sistema são distintas, em malha aberta, a entrada é dada em porcentagem de potência e em malha fechada a entrada é dada em valores de temperatura. Observa-se no entanto, que os requisitos de projeto foram atendidos, ou seja, resposta pelo menos duas vezes mais rápida e erro nulo em regime permanente.

$$G_c(s) = 7{,}3159 \left[1 + \frac{1}{866{,}2s}\right] \tag{4.14}$$

A Figura 4.10 mostra o fluxograma que representa o programa desenvolvido para atender os requisitos de projeto para a função *teste de controladores*. Esse fluxograma segue o mesmo padrão utilizado para criar o programa para a função *levantamento de características*, o que exemplifica a modularidade empregada no desenvolvimento de todo o sistema, ou seja, verifica-se que para a obtenção de uma nova função, basta acrescentar novas ações a um módulo já existente. No caso, foi acrescentado ao código as informações

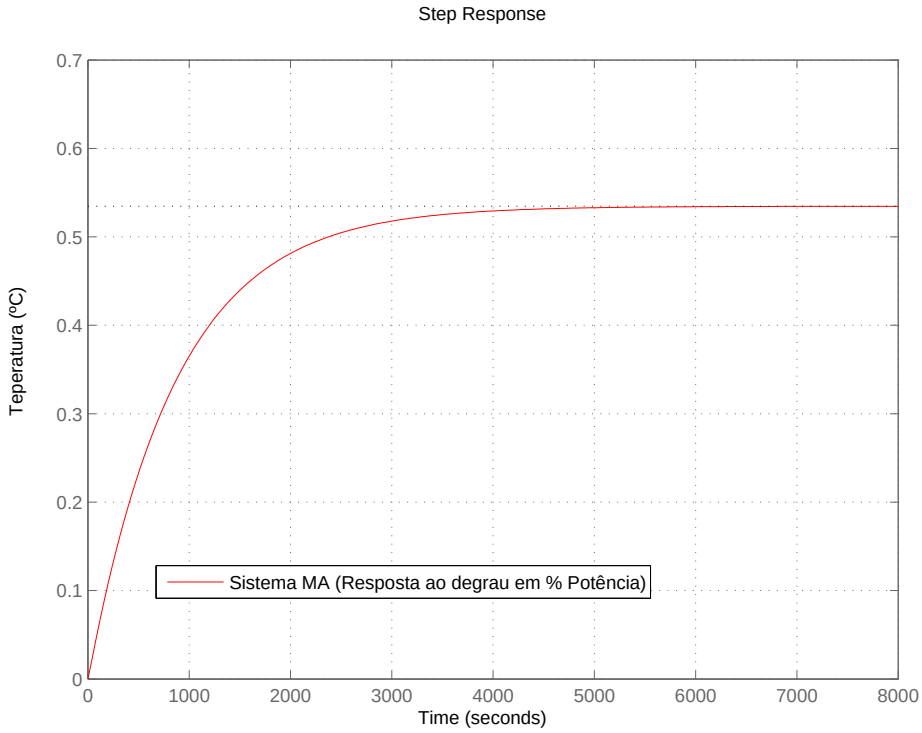

Figura 4.8: Resposta do sistema em MA a um degrau em escala de % de Potência.

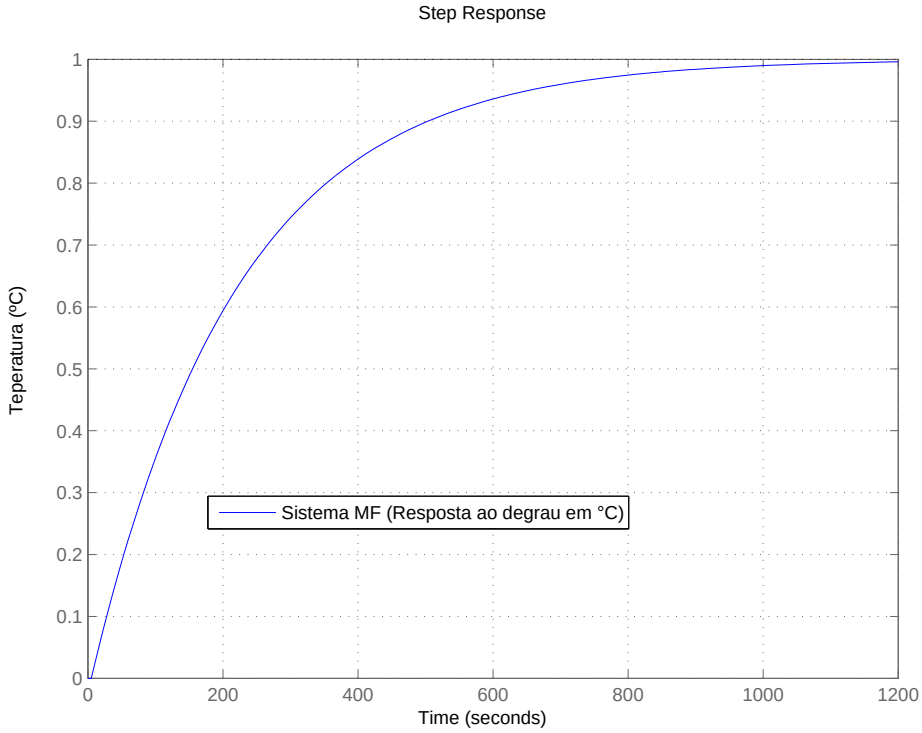

Figura 4.9: Resposta do sistema em MF a um degrau em escala $°C$.

referentes aos parâmetros do controlador e o perfil, que antes era aplicado diretamente aos atuadores como porcentagem de potência, passou a ser empregado como referência de temperatura (temperatura desejada) para o controlador.

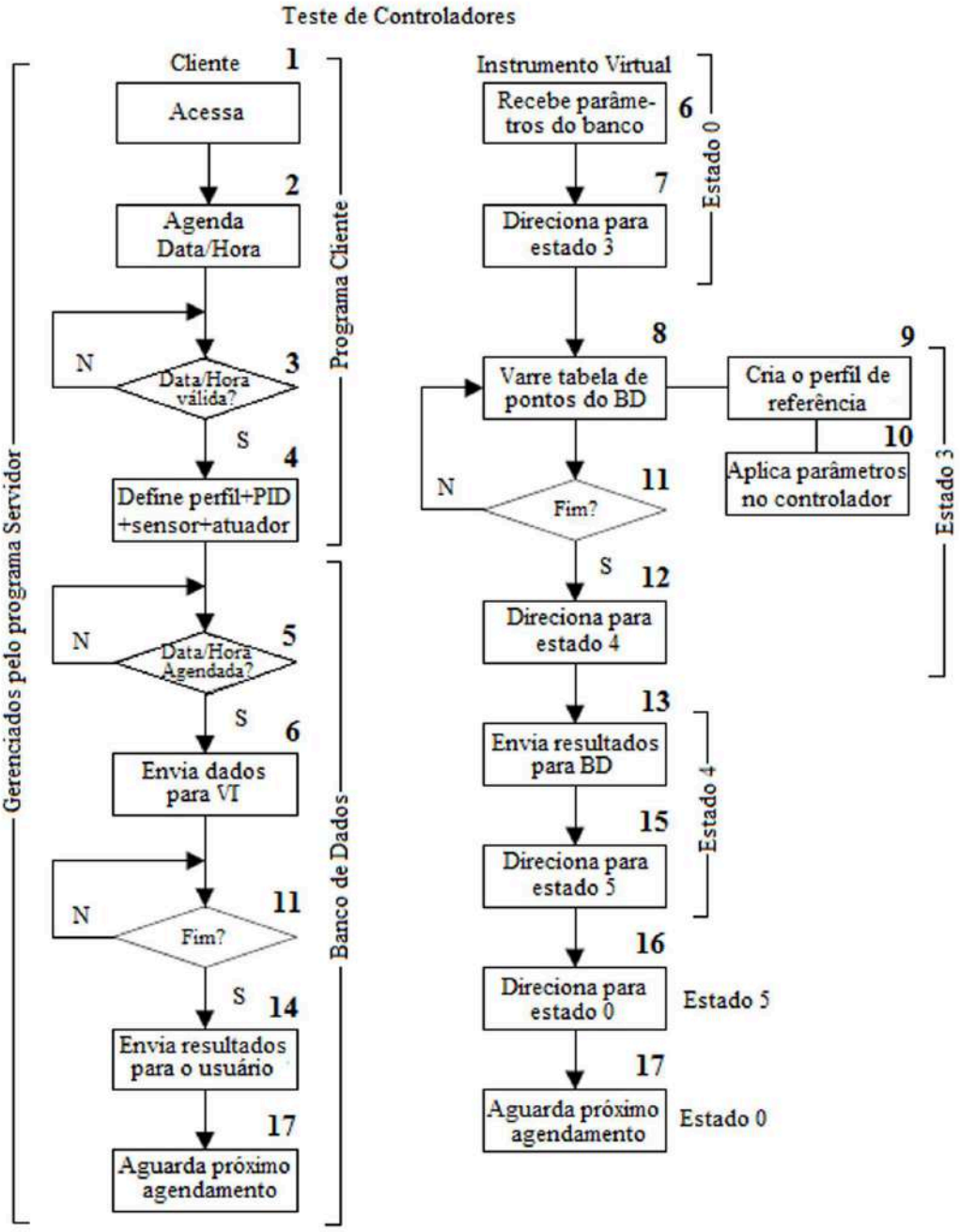

Figura 4.10: Função teste de controladores.

Utilizando-se o programa representado pelo fluxograma da Figura 4.10, presente no manual do WebLab e no CD em anexo, foi realizado um experimento para validar a função de teste de controladores. A Figura 4.11 ilustra o perfil em degrau de temperaturas de referência programado no WebLab. A aquisição começa com uma temperatura de referência de $30°C$. Essa valor foi escolhido por que a temperatura ambiente se encontrava em torno de $28°C$. Depois de duas horas, a temperatura de referência subiu para $40°C$ e posteriormente, após o mesmo intervalo, ela desceu novamente para $30°C$. O intervalo das referências foi escolhido nesse valor porque esse foi o mesmo intervalo utilizado na aquisição para identificação.

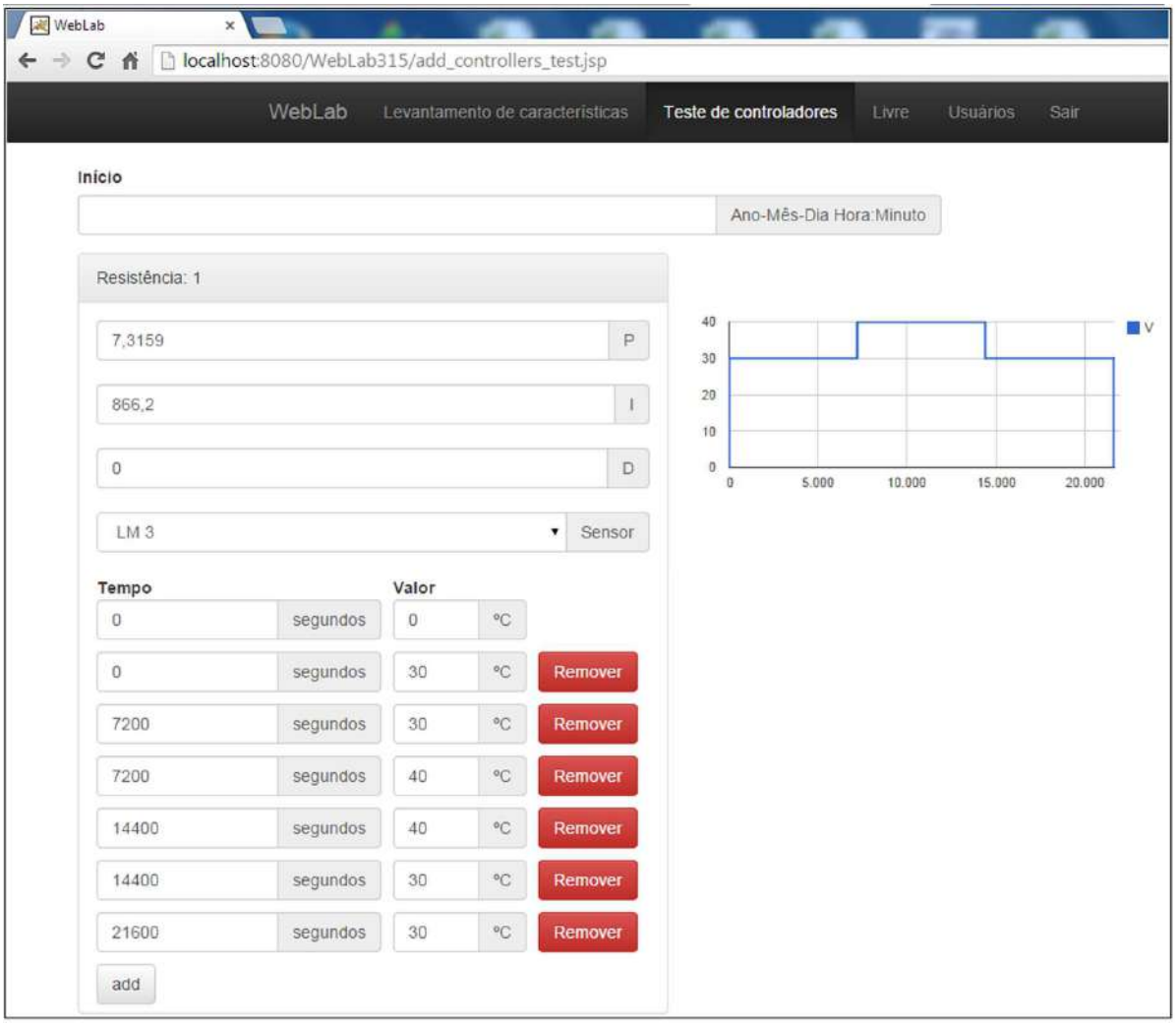

Figura 4.11: Perfil programado para teste de controladores.

Antes de testar o controlador na planta do sistema de aquecimento de ar, foi feito uma simulação do sistema em Simulink® para analisar o sinal de controle que seria aplicado nos atuadores. Isso foi feito para evitar que o controlador impute um valor de potência maior que o máximo permitido, ou seja, o controlador só pode atuar com uma potência de 0 a 100%, que corresponde à faixa de tensão de trabalho de 0 a $5V$ da placa de aquisição

de dados. A Figura 4.12 ilustra o diagrama de blocos do sistema construído.

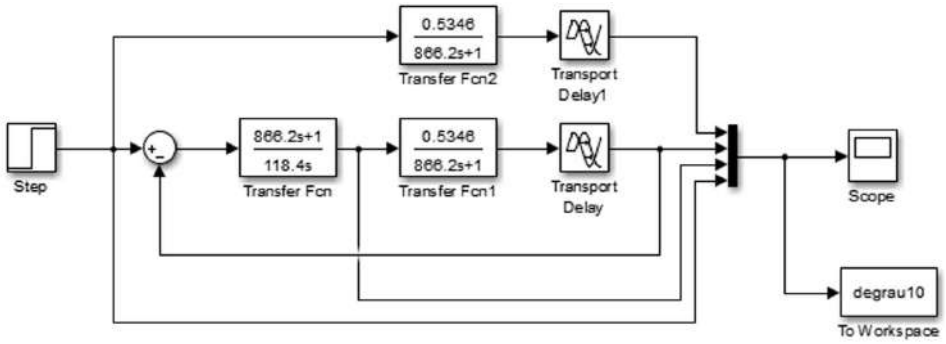

Figura 4.12: Diagrama de blocos do sistema.

A Figura 4.13 apresenta a resposta do sistema em malha aberta, onde é aplicado um degrau em escala de porcentagem potência. A Figura 4.14 apresenta a resposta do sistema em malha fechada, onde é aplicado um degrau em escala de temperatura. Na Figura 4.15, é apresentado o sinal de controle aplicado para uma entrada em degrau no sistema em malha fechada. Observa-se que o máximo valor do sinal de controle não excede o limite 100% aceitável, no entanto, se esse valor for excedido, ou seja, se o sinal de controle saturar, a VI onde o controlador é implementado, continua enviando o valor de referência máximo permitido para o comando dos atuadores, ou seja, 100% ou $5V$.

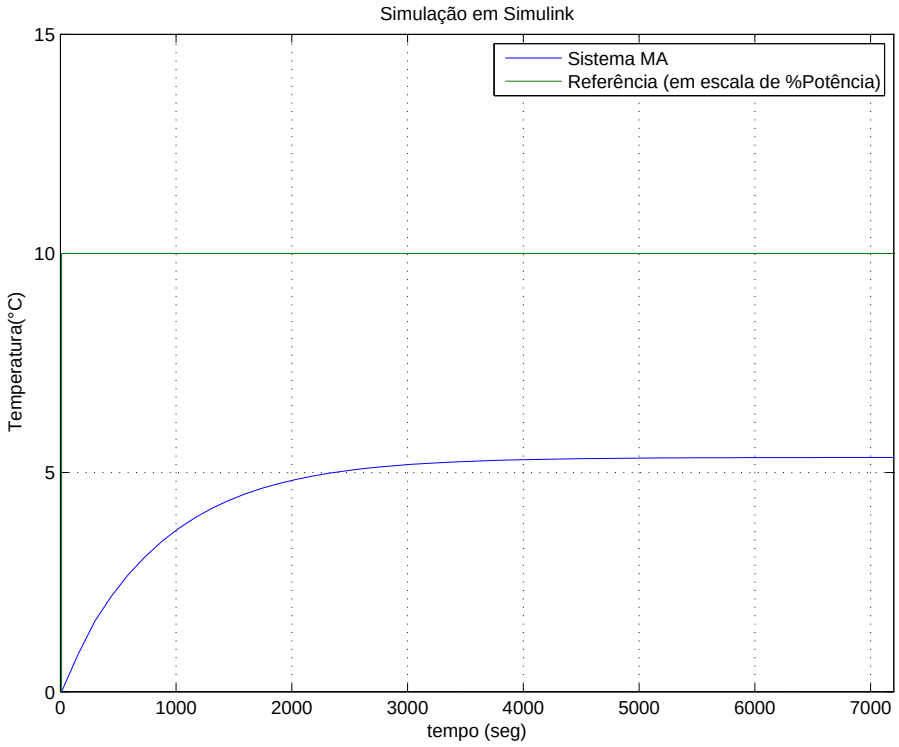

Figura 4.13: Simulação do sistema em MA no Simulink®.

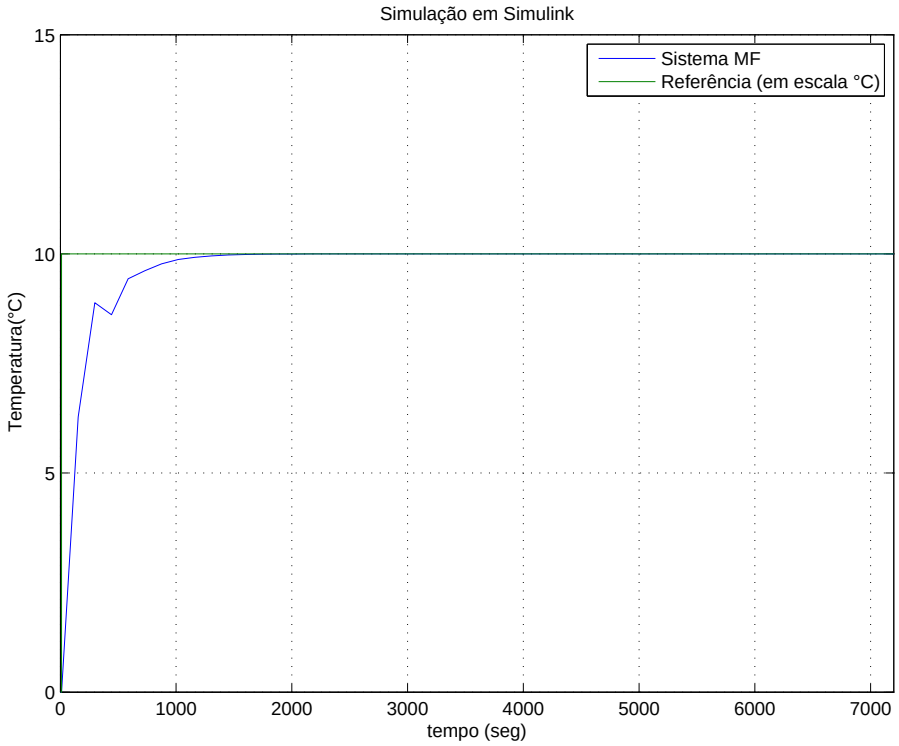

Figura 4.14: Simulação do sistema em MF no Simulink®.

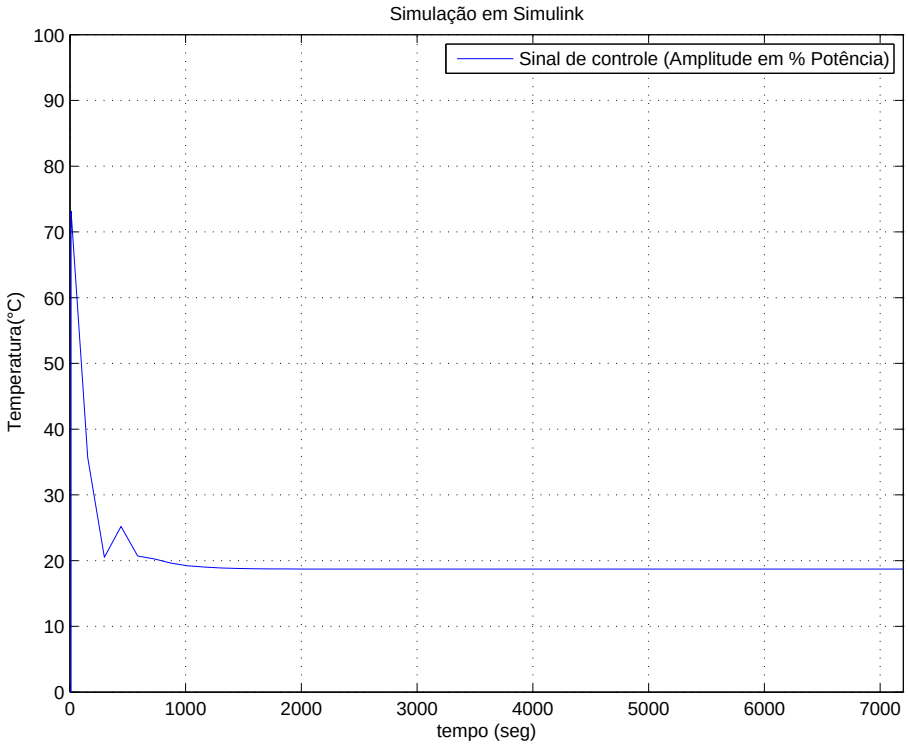

Figura 4.15: Simulação do sinal de controle do sistema no Simulink®.

Sabendo que o sinal de controle estava dentro da faixa de trabalho do sistema de aquisição de dados, deu-se continuidade ao experimento real para teste do controlador. A Figura 4.16 apresenta a resposta do sensor $LM3$ e o sinal de controle aplicado à resistência

$R1$ para manter o sistema controlado quando foi aplicado o perfil representado na Figura 4.11 com *setpoints* em $30°C$, $40°C$ e $30°C$. Na prática, verificou-se que o valor mais alto assumido pelo sinal de controle se manteve bem abaixo daquele simulado no Simulink®, dessa forma, acreditasse que degraus com amplitudes maiores possam ser aplicados.

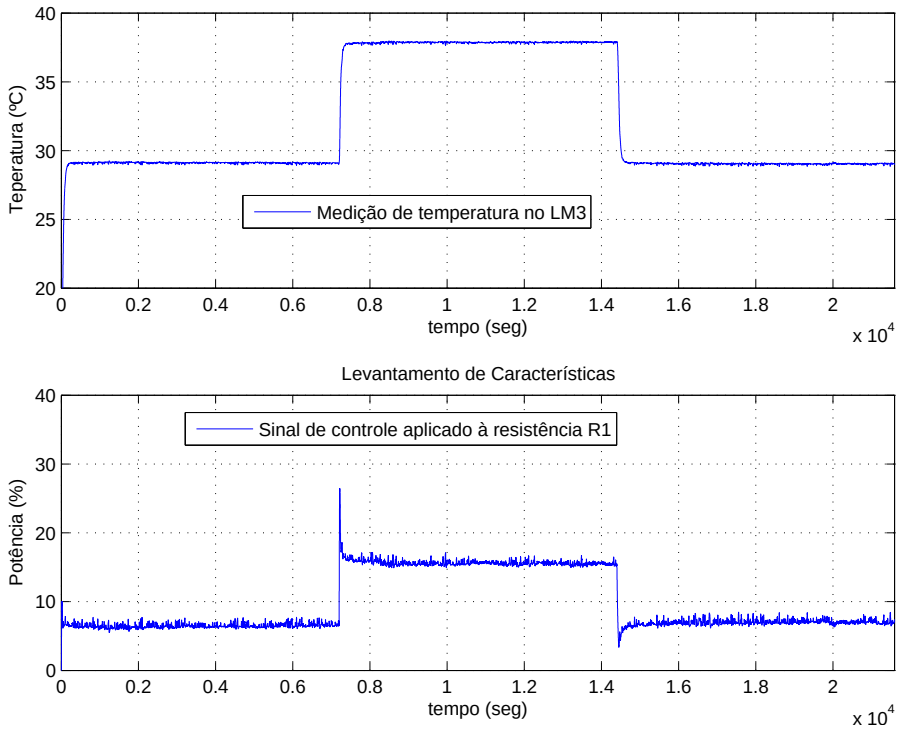

Figura 4.16: Sinal aquisitado na função teste de controladores.

A Figura 4.17, apresenta o sinal aquisitado na função *levantamento de características*, como resposta do sistema a um degrau de 10% da potência total que pode ser aplicada aos atuadores. Verifica-se que para um degrau de 30% para 40%, o sistema responde com uma temperatura variando de aproximadamente $44°C$ para $55°C$.

A Figura 4.18 apresenta o sinal aquisitado na função *teste de controladores*, como resposta do sistema a um degrau de de 10% da temperatura total que os atuadores conseguem atingir (de $30°C$ para $40°C$). É possível verificar que, na prática, o controlador projetado proporcionou uma resposta do sistema mais rápida, porém, ocorreu um erro em regime permanente, o que significa que não houve integração. Acredita-se que esse erro foi devido à VI utilizada para implementar o controlador, já que na simulação em Simulink® o controlador se comportou como esperado.

Outro fato a ser destacado, é que o teste de controladores foi feito em um ponto de operação diferênte daquele em que foi levantado o modelo do sistema. Para que o teste fosse realizado corretamente, seria necessário realizá-lo aplicando um valor de *offset* ao sinal de controle, fazendo com que este ficasse em torno do ponto de operação escolhido, ou seja, 30% da potência total. No entanto, o objetivo do teste feito, foi apenas ilustrar o

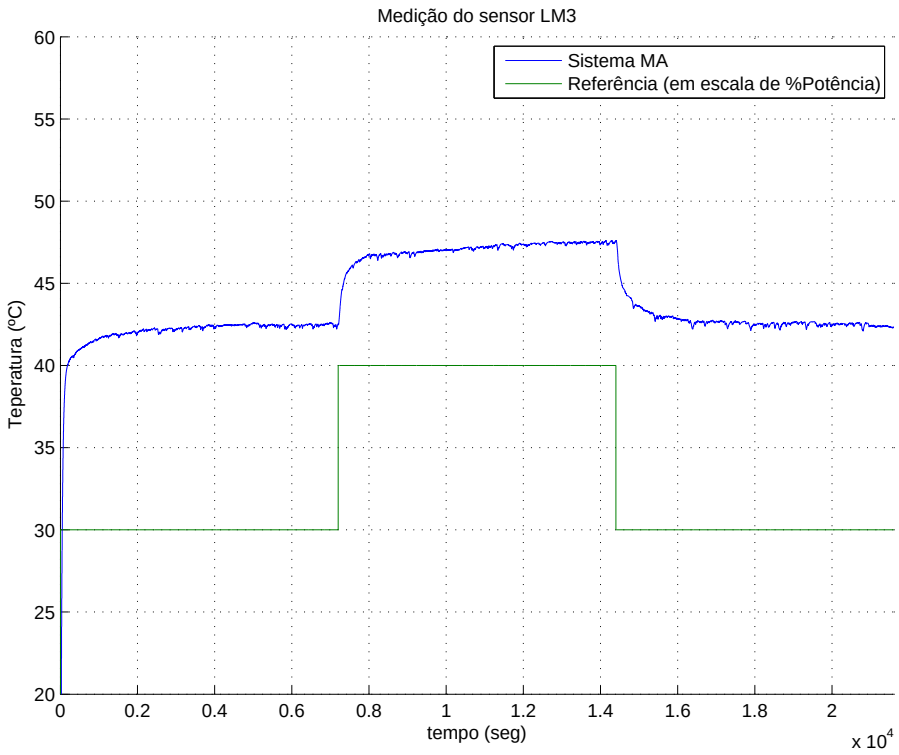

Figura 4.17: Resposta do sistema em MA para um degrau de 10% da potência total.

funcionamento da função *teste de controladores* e não projetar um bom controlador, nesse sentido, o objetivo foi alcançado.

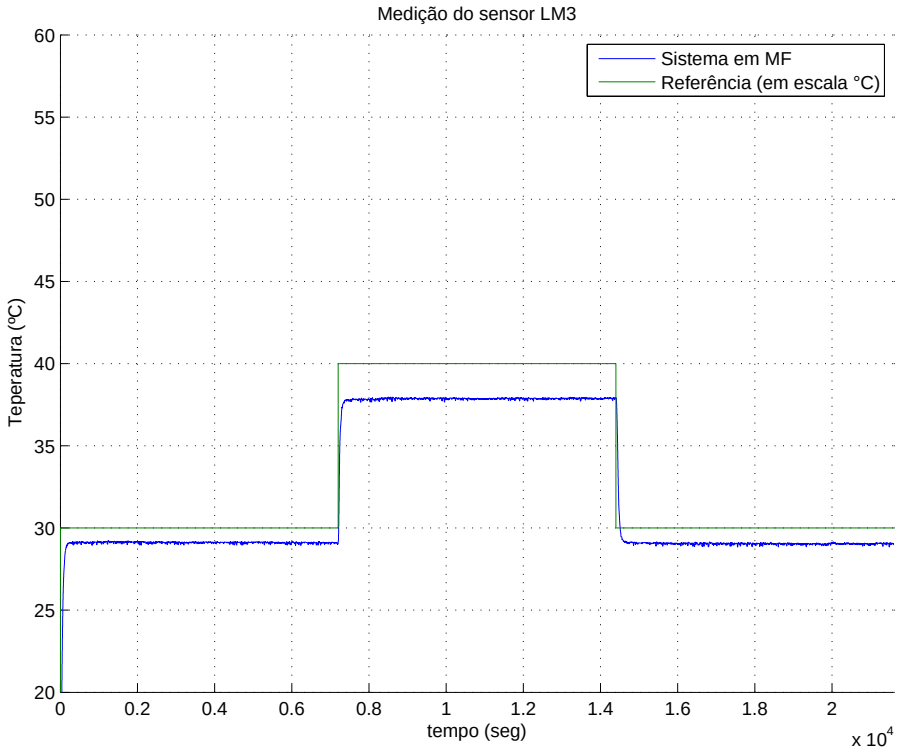

Figura 4.18: Resposta do sistema em MF para um degrau de 10% da temperatura total.

Livre

Como explicado anteriormente, a função *livre* tem como objetivo a verificação do funcionamento do sistema, ou seja, através dela é possível testar todas as conexões do WebLab e verificar o funcionamento dos componentes da planta. A Figura 4.19 descreve o fluxograma do programa criado para atender as especificações de projeto para o teste *livre*. A estrutura do programa é praticamente a mesma das outras duas funções. Porém, na função *livre*, é acrescentado um módulo para comunicação direta entre o servidor e a aplicação via TCP/IP. Além disso, a comunicação com o banco de dados ocorre somente para agendamento do experimento e gravação dos dados a serem enviados para o usuário já que os comandos são diretos e não são agendados.

Para demonstrar o funcionamento do teste *livre*, foi feito uma breve aquisição de dados utilizando a resistência $R1$ e o sensor $LM3$. para tanto, foi aplicado um degrau de 50% da potência durante aproximadamente cinco minutos. Em seguida, o comando foi retornado para zero permanecendo durante outros cinco minutos. A Figura 4.20 mostra o sinal aquisitado nessa ação.

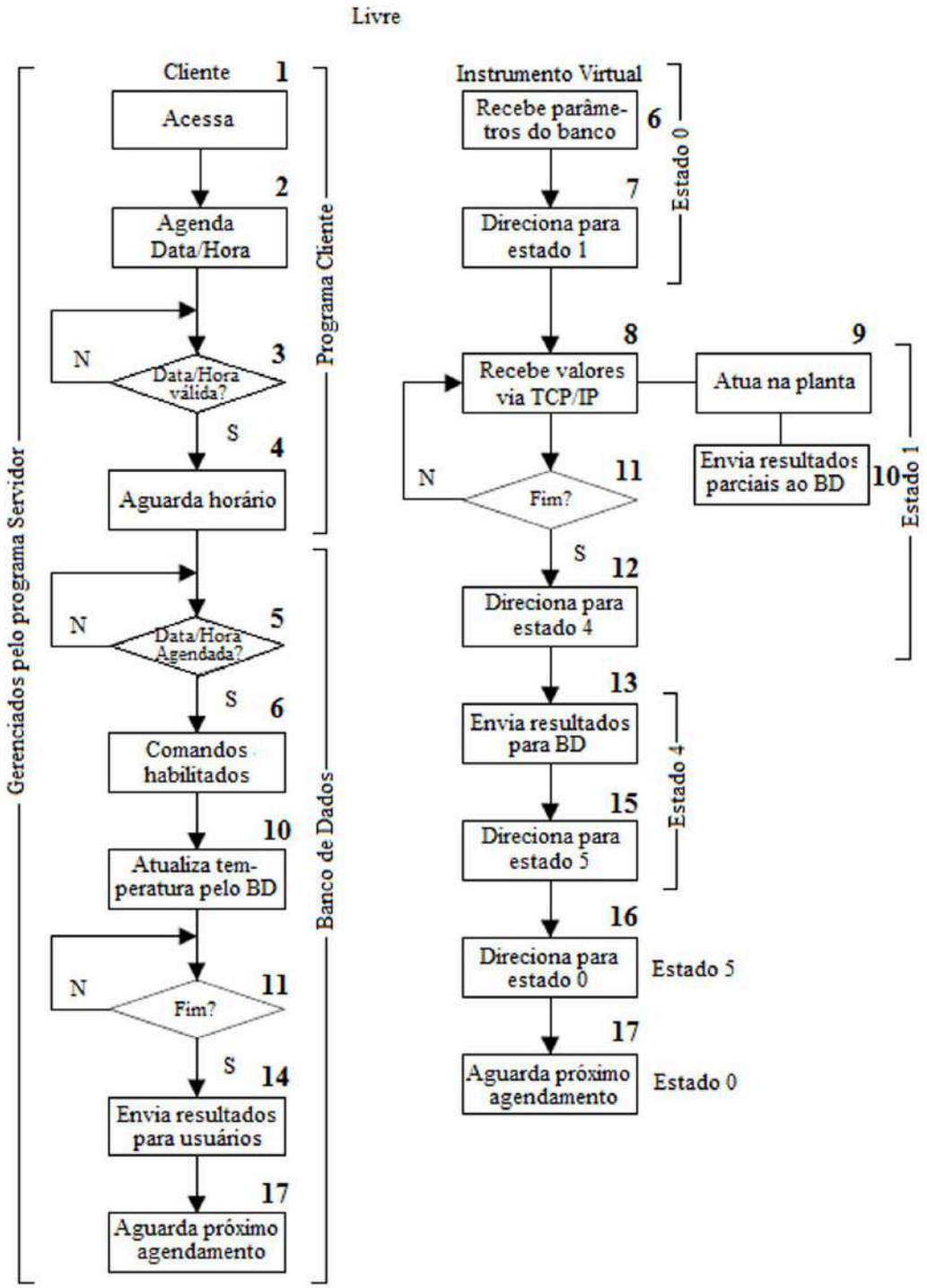

Figura 4.19: Função livre.

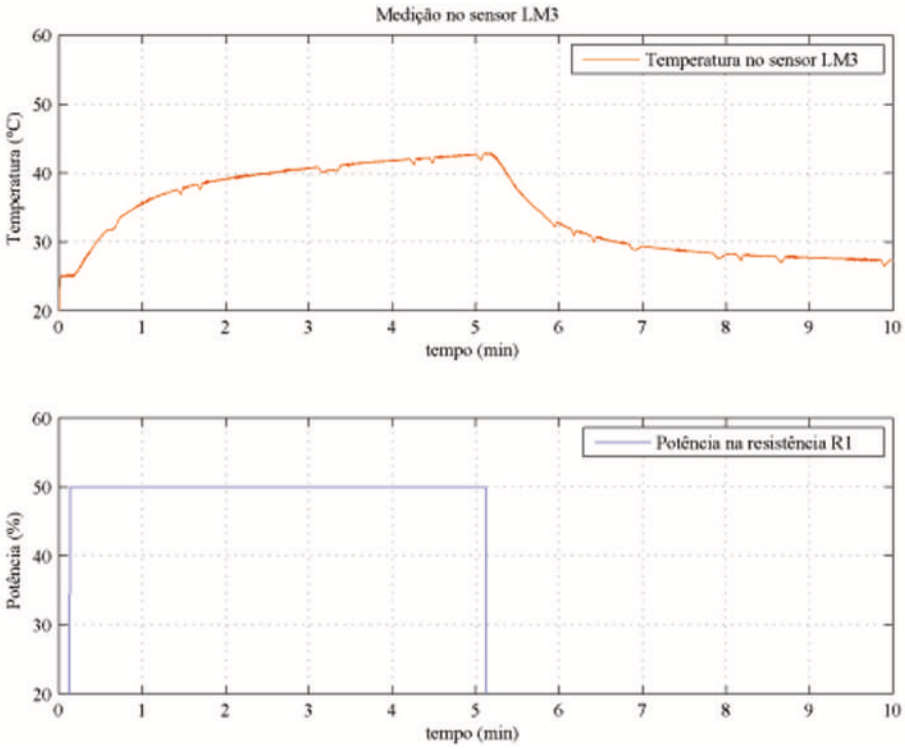

Figura 4.20: Sinal obtido durante o teste livre.

Capítulo 5

Considerações Finais e Perspectivas Futuras

5.1 Considerações Finais

No presente trabalho de conclusão de curso foi desenvolvido um Laboratório Remoto para o agendamento e realização de experimentos, via Web, em um sistema de aquecimento de ar. Para tanto, foram utilizados recursos de Instrumentação Virtual e ferramentas para comunicação Web que possibilitaram a criação de uma estrutura modular e facilmente expansível a outros sistemas.

A idéia inicial do trabalho era criar uma aplicação Web que permitisse que usuários remotamente distribuídos acessassem a página do WebLab na internet para realizarem experimentos a fim de identificarem o sistema sob diferentes configurações e testarem controladores projetados a partir dessa identificação. No entanto, durante a progressão do trabalho, constatou-se, através dos estudos realizados para fundamentação, que poderia ser criado um modelo base, que teria potencial para expandir o WebLab para outras plantas. O desenvolvimento do projeto tornou-se então, um pouco mais criterioso em relação aos padrões utilizados para programação das partes que o constitui. Todos os programas foram construídos de forma a serem modulares, de fácil leitura e fácil manutenção. Esses conceitos foram adaptados e aplicados também no projeto dos circuitos da planta do sistema de aquecimento de ar.

A planta do sistema de aquecimento de ar passou por diversas modificações. Foi trocado o tipo de alimentação, que passou a ser trifásico, com os atuadores ligados em triângulo, essa configuração diminuiu a corrente exigida pela planta e possibilitou que outras plantas do laboratório fossem utilizadas simultaneamente. Outra modificação significativa foi em relação ao tipo de acionamento das resistências, onde foram utilizados relés de estado sólido com acionamento por ciclo integral ao invés de disparo por fase. A planta ganhou também um circuito de acionamento geral. Em relação a instrumentação, foram acrescentados circuitos transdutores termopares para medição de temperatura externa ou em pontos não fixos da planta. Além disso, foram implementados circuitos para medição

de corrente nas resistências a fim de se monitorar o consumo de energia.

O modelo do WebLab foi desenvolvido baseado na arquitetura Cliente-Servidor. O controle do sistema de aquecimento de ar, foi realizado através de um aplicativo criado em LabVIEW®. Esse aplicativo funciona como um conjunto de métodos que são executados a pedido do usuário remoto por meio de um programa cliente, que envia requisições para o programa servidor, que por sua vez, é quem gerencia as ações do WebLab. A programação em LabVIEW® foi feita tendo como referencia padrões de projeto e técnicas de programação consistentes que permitiram a obtenção um programa bem estruturado.

Para armazenamento de dados de agendamento e resultados, foi utilizado um banco de dados. Ele também funciona como um método a ser executado a pedido do programa servidor. A utilização de banco de dados possibilitou o armazenamento de grande quantidade de dados e também o relacionamento entre eles.

Conclui-se portanto, que os objetivos do trabalho foram alcançados e que todas os requisitos de projeto foram atendidos já que todas as funções desejadas a priori foram desenvolvidas. Além disso, houve um ganho significante na proposta inicial, pois foi obtida uma estrutura que possui grande potencial para novos desenvolvimentos e melhorias.

O acesso ao sistema de aquecimento de ar via internet, requer que o servidor Web tenha um IP de internet fixo. Tendo este IP, qualquer máquina conectada à internet é capaz de se comunicar com o servidor e acessar o sistema de controle da planta. No entanto, IP's fixos custam caro e o CEFET-MG não permite que um ponto de internet chegue diretamente ao laboratório.

Uma solução alternativa para este problema, seria utilizar uma máquina auxiliar, com IP fixo, como proxy. Esta máquina, pode estar fisicamente em outro local. Assim, a máquina próxima da planta, que roda o servidor Web, se conectaria com esta outra utilizando uma técnica chamada Túnel SSH de proxy reverso, fazendo com que as requisições que chegassem pelo IP fixo na máquina de proxy fossem re-encaminhadas para a máquina servidor, que por sua vez, processaria os dados e devolveriam para a máquina de proxy, que se encarregaria de repassar a resposta à máquina que fez a requisição.

Uma solução para máquina de proxy, por exemplo, é um servidor da Amazon EC2. No entanto, esta solução foi descartada por envolver uma empresa. O ideal (conexão por IP fixo na máquina do servidor web) também é possível dentro do CEFET, usando túneis e redirecionamento de portas. No entanto o CEFET não autorizou a implantação da infraestrutura.

5.2 Perspectivas Futuras

Os resultados obtidos no desenvolvimento deste trabalho, permitem sugerir algumas propostas de melhorias:

Em relação ao sistema de aquecimento de ar, como os experimentos serão realizados à distância, devem ser tomados alguns cuidados em relação à segurança. Sugere-se portanto, a instalação de uma chave de emergência local para que a alimentação da planta possa ser imediatamente interrompida caso ocorra algum problema não detectado pelos fusíveis ou chave geral e tenha alguém no laboratório para acioná-la. A instalação de um relé de falta de fase também é indicada, pois se faltar uma das fases, o sistema não funcionará corretamente.

Além das proteções físicas, ações de segurança podem ser implementadas em diversas partes do *software* aplicativo, como por exempo, a definição de limites de temperatura e posteriormente, quando os circuitos transdutores de corrente estiverem funcionando, quanto a limites de corrente, ja que essas grandezas serão monitoradas o tempo todo.

Em relação à função de medição de consumo de energia, os circuitos transdutores de corrente estão devidamente instalados e a ligação física entre eles e o circuito microcontrolador já está estabelecida. Dessa forma, deve-se programar o microcontrolador para leitura dos sinais a serem medidos e cálculo do consumo de energia. Além disso, uma VI deve ser criada no *software* aplicativo para comunicação serial com o circuito microcontrolado, através do *toolkit* do LabVIEW® para Arduíno®, disponível gratuitamente para instalação. Os circuitos transdutores de corrente também devem ser calibrados.

Quanto ao *software* aplicativo, verificou-se que este pode ser constantemente melhorado. As funções programadas devem ser aprimoradas e novas funções podem ser criadas. Nesse aspecto, um destaque deve ser dado aos controladores a serem implementados. Sugere-se que sejam criadas VIs específicas para cada tipo de controlador e que as VIs prontas, sejam evitadas, a menos que seja feito um estudo bem cuidadoso sobre seu funcionamento para adequação correta à aplicação a ser desenvolvida.

Em relação ao banco de dados, sugere-se que este seja configurado para realizar cópias de segurança (*backup*), evitando assim que os dados dos experimentos sejam perdidos, pelo menos por um tempo de armazenamento a ser definido. Sugere-se também o desenvolvimento de uma ação que permita que determinado experimento seja continuado do ponto onde parou caso ocorra, por exemplo, um problema de falta de energia, ou seja, se o teste parar por algum motivo, assim que o sistema for re-estabelecido, volta a executar o experimento do ponto em que parou ou repete o experimento automaticamente no horário disponível seguinte.

Para interface de usuário remoto, segere-se o desenvolvimento de páginas que permitam a interação entre o cliente e o experimento de forma dinâmica e intuitiva, através de

gráficos que permitam o monitoramento dos testes em tempo real. Sugere-se também a instalação de cameras para filmagem do processo enquanto ele acontece.

De uma maneira geral, deseja-se que o presente projeto seja levado à diante para que possa ser construída uma rede de experimentos, com várias plantas, permitindo que o conhecimento prático seja facilitado e que chegue mais longe, atingindo um número maior de estudantes.

Apêndice A

Controle AC por Ciclo Integral

Relés de Estado Sólido

O relé de estado sólido é um dispositivo eletrônico usado no acionamento de cargas resistivas e/ou indutivas. O SSR (*Solid State Relay*), se difere do relé eletromecânico (*Electro Mechanical Relay*-EMR) por não apresentar partes mecânicas móveis. Sua estrutura interna é feita de semicondutores (componentes estáticos), o que confere a ele velocidades de operação bem mais altas que a do relé convencional, sendo esta apenas uma de suas vantagens.

Existem duas categorias de relés de estado sólido, que se diferenciam pela potência que atendem. A primeira delas é conhecida como módulo I/O e é empregada em acionamentos de baixa potência fazendo a interface entre comando digital e pequenas cargas tais como solenóides, lâmpadas, eletroválvulas, etc. A segunda, é denominada chave estática e é empregada para acionamento de cargas com alta potência, normalmente grandes motores, ou grandes bancos de resistores.

O SSR é construído em um encapsulamento plástico para proteção contra choques (*safety-cover*) e possui um terminal de controle e um terminal de potência. Um sinal de comando determina o acionamento da carga conectada aos terminais de potência do dispositivo. A Figura A.1 mostra o diagrama de blocos de um SSR monofásico. Ele é composto basicamente por um circuito de acoplamento, um circuito detector de passagem por zero, um *triac* e um circuito de proteção interno (*Snubber*).

A Figura A.2 ilustra as formas de onda do Relé de Estado Sólido. Ao receber um sinal de comando em seu terminal de controle a chave estática conduz e alimenta a carga. A condução ocorre efetivamente na próxima passagem por zero da tensão da rede. O mesmo acontece no desligamento, ou seja, o sinal de comando é retirado e na próxima passagem por zero da corrente elétrica de alimentação e o dispositivo é bloqueado. Isso implica, que para uma frequência de rede de $60Hz$, o atraso liga/desliga nunca é superior a $8{,}3ms$.

O fato de ligar e desligar a alimentação da carga sempre em um cruzamento por

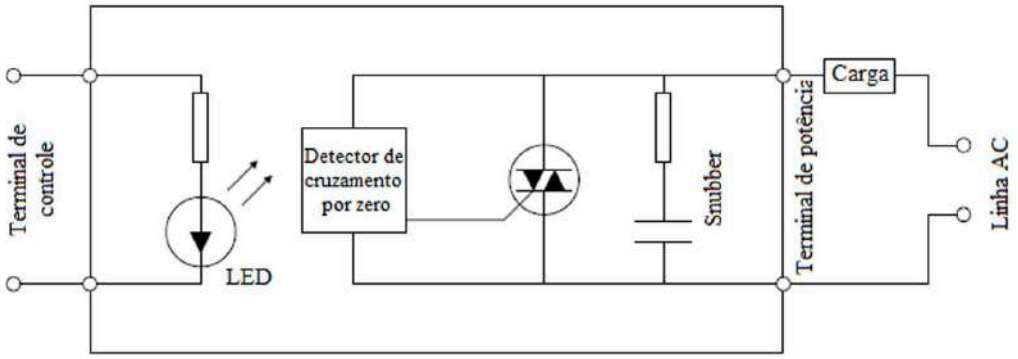

Figura A.1: Diagrama de blocos SSR monofásico.
Fonte: (ASHFAQ, 2000).

zero, trás outra importante vantagem do SSR, ou seja, praticamente não são geradas interferências elétricas na instalação e a chave não é submetida a condições severas de chaveamento (NOVUS AUTOMATION, 2013).

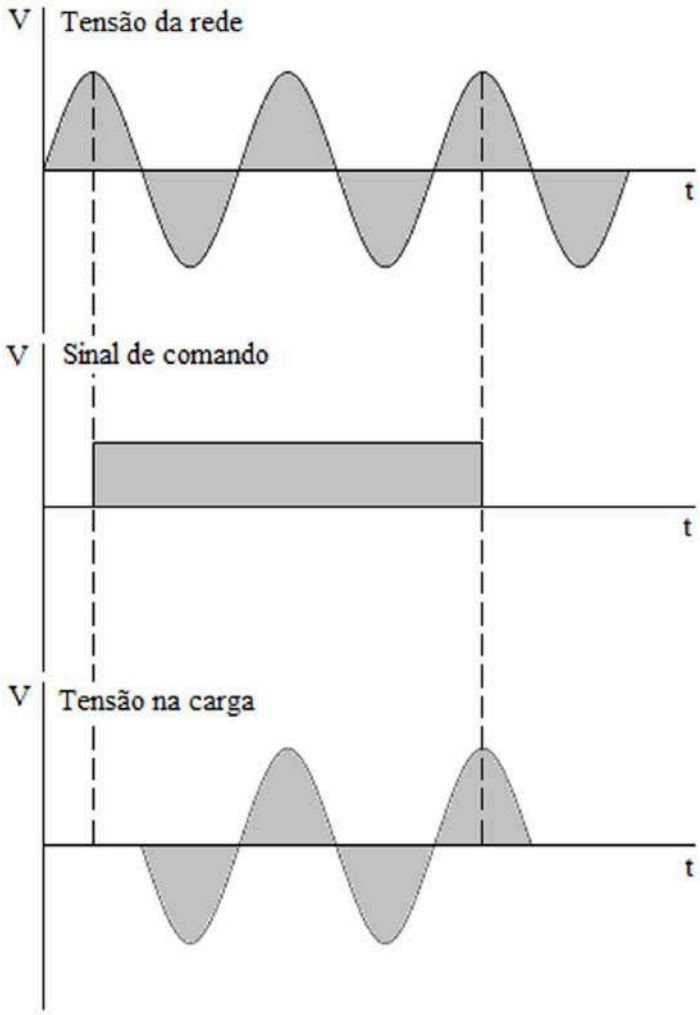

Figura A.2: Formas de onda do SSR.
Fonte: (ASHFAQ, 2000).

Os semicondutores que formam os $SSRs$ podem ser os mais diversos, incluindo *mosfets*, transistores bipolares e ponte de $SCRs$. No entanto, a forma mais comum é aquela que utiliza *triac*. Além disso, os relés de estado sólido também podem ser encontrados na versão trifásica e o seu acionamento pode ser por controle CA.

Os relés de estado sólido são largamente utilizados para controle de processos industriais, especialmente em controle de temperatura, motores, válvulas, solenóides, lâmpadas, transformadores, etc. Além disso ele são empregados na fabricação de máquinas (injetoras, extrusoras, sopradoras, embalagem), em sistema de segurança, em controladores de demanda de energia, controladores de tráfego, instrumentação eletrônica, equipamentos médicos e hospitalares, controle de elevadores, equipamentos meteorológicos, dentre outros.

Controle de Potência CA por Ciclo Integral

O controle de potência CA por ciclo integral, ou controle liga/desliga é um dos métodos básicos existentes para controle de potência. Este tipo de controle é geralmente empregado em sistemas que possuem constante de tempo grande, como por exemplo, o controle de sistemas térmicos.

A potência de uma carga pode ser controlada com a ligação e o desligamento da fonte que a alimenta, por alguns ciclos completos e depois com a repetição do chaveamento. A duração relativa dos períodos nos estados ligado e desligado, isto é, o ciclo de trabalho d é ajustado de modo que a potência média entregue à carga atenda a algum objetivo particular.

A Figura A.3 representa um controlador de tensão CA. O triac pode ser disparado pelo circuito de controle em um ângulo $\alpha = 0$ da tensão de alimentação do controlador, permitindo que os ciclos completos da tensão da fonte sejam aplicados à carga. Se não houver nenhum sinal de disparo em cada ciclo, então nenhuma tensão aparecerá sobre a carga e assim, é possível permitir que ciclos completos de tensão da fonte sejam aplicados à carga, seguidos de ciclos completos de extinção. A relação entre o tempo ligado e o ciclo total, ou seja, o período no qual a condução padrão se repete, permite que a potência média entregue à carga seja controlada de 0 a 100% (em circunstâncias ideais).

A **Figura** A.4 ilustra as formas de onda do controle por ciclo integral. T_{ON} é o número de ciclos para os quais a carga é energizada e T é o número de ciclos no período completo de operação. Durante T_{ON} a chave está ligada e a potência na carga é máxima. Durante o restante do ciclo ($T_{OFF} = T - T_{ON}$), a chave está desligada e a potência na carga é nula.

Para uma carga resistiva R, a potência média é dada por:

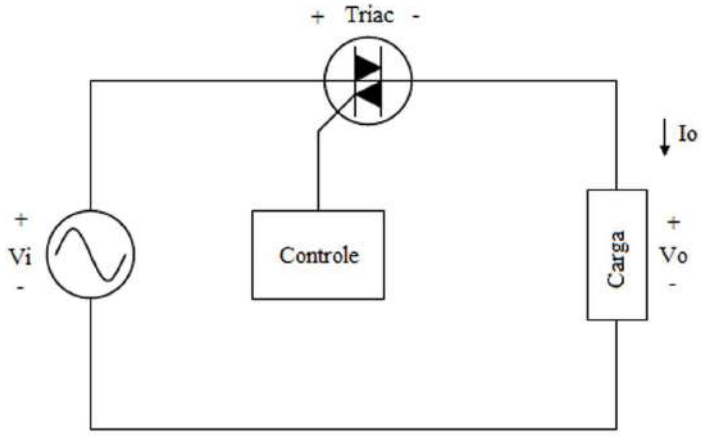

Figura A.3: Controlador de tensão CA.
Fonte: (ASHFAQ, 2000).

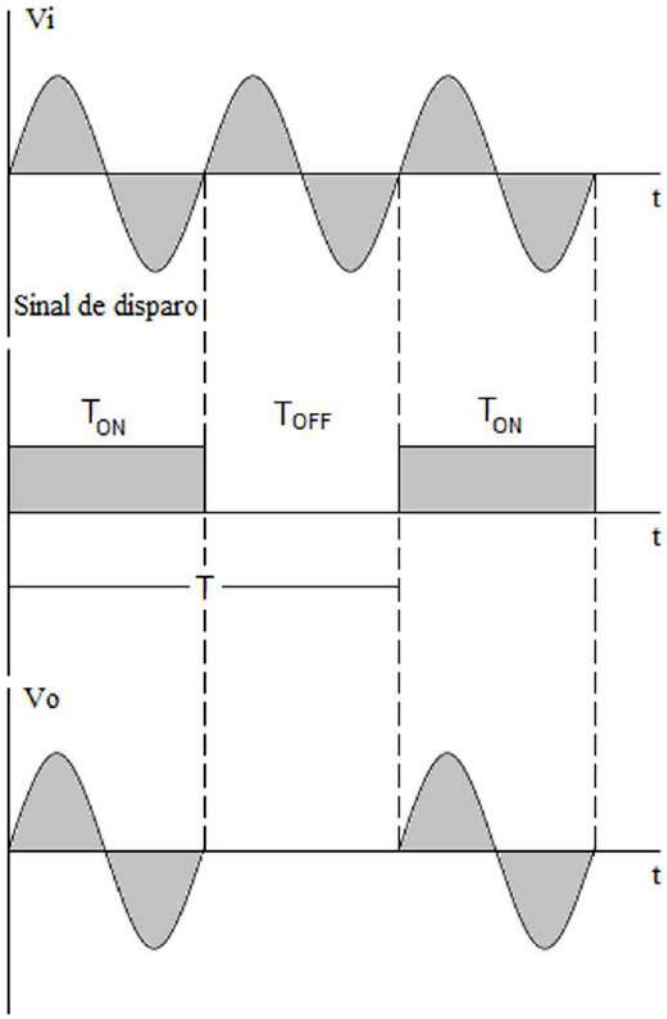

Figura A.4: Formas de ondas do controle por ciclo integral.
Fonte: (ASHFAQ, 2000).

$$P_{o(avg)} = \frac{{V_i}^2 T_{ON}}{RT} = \frac{V_i^2}{R} d = P_{o(max)} d \tag{A.1}$$

O valor RMS da tensão de saída é dado por:

$$V_{o(RMS)} = \frac{V_m}{\sqrt{2}} \sqrt{\frac{T_{ON}}{T}} = V_i \sqrt{d} \tag{A.2}$$

em que:

V_m = valor máximo da tensão de entrada.

V_i = valor RMS da tensão de entrada = $V_m/\sqrt{2}$.

Uma vez que T_{ON} pode variar somente como um número inteiro, o valor médio da potência na carga não é uma função contínua, tendo apenas níveis discretos. Dessa forma, o número de degraus disponíveis para a regulação da potência média depende do número total de ciclos incluídos no padrão de repetição.

A conversão de potência é a razão entre a potência média de saída ($P_{o(avg)}$) com a potência máxima de saída ($P_{o(max)}$). Essa razão é igual ao ciclo de trabalho (d).

$$d = \frac{T_{ON}}{(T_{ON} + T_{OFF})} = \frac{T_{ON}}{T} \tag{A.3}$$

O fator de potência é dado por:

$$PF = \sqrt{\frac{T_{ON}}{T}} = \sqrt{d} \tag{A.4}$$

O controle por ciclo integral tem a vantagem de exigir um número menor de operações de chaveamento e apresenta baixa interferência de radiofrequência por causa do controle durante o cruzamento por zero da tensão CA, ou seja, para cargas resistivas, o chaveamento ocorre apenas na tensão zero. A taxa de variação da corrente na carga depende da frequência do sistema, que é pequena. Com isso, esse tipo de controle apresenta baixo ruído elétrico quando comparado a outros métodos.

O controle por ciclo integral não é indicado para cargas com constante de tempo pequenas. Neste caso, o mais indicado é o *controle de fase*. Nele, a chave liga a carga à fonte por um período a cada ciclo da tensão de entrada. Pode-se variar a tensão na carga com a alteração do ângulo de disparo α para cada semiciclo de um período. Se $\alpha = 0$, a tensão de saída é máxima ($v_o = v_i$). Quando $\alpha = \pi$, a tensão de saída é mínima ($v_o = 0$). Portanto, ela pode ser controlada para qualquer valor entre zero e o valor da tensão na fonte. Esse processo fornece uma saída alternada controlada por fase apropriada para aplicações como controle de iluminação e controle de velocidade para motores. Os

cálculos e formas de onda do controle por ângulo de fase não serão tratados neste trabalho, podendo ser consultadas em ASHFAQ (2000).

Apêndice B

Transdutores de Corrente por Efeito *Hall*

Descoberto em 1879 pelo físico americano Edwin Herbert *Hall* na Universidade de Johns Hopkins em Baltimore, o efeito *Hall* é um fenômeno criado por forças de *Lorentz* que atuam sobre cargas em movimento através de um campo magnético.

Quando uma fina folha de material condutor é atravessada longitunalmente por uma corrente elétrica I_c, os portadores de carga desta corrente são afetados por um fluxo magnético, B, gerando forças de *Lorentz* F_L perpendicularmente ao fluxo de corrente. A deflexão resultante das correntes faz com que portadores de carga se desloquem para as bordas opostas da folha criando uma diferença de potencial denominada tensão de *Hall* (V_H).

$$F_L = q(V \times B) \tag{B.1}$$

A Figura B.1 representa um arranjo conhecido como gerador de *Hall* em que a tensão V_H é dada por:

$$V_H = \frac{K}{d \cdot I_C \cdot B} + V_{OH} \tag{B.2}$$

onde,

K é a constante de *Hall*.

d é a espessura da folha condutora.

V_{OH} é a tensão de offset do gerador de it Hall na ausência de um campo externo.

$K/d = \cdot I_c$ é a sensibilidade do gerador de *Hall*.

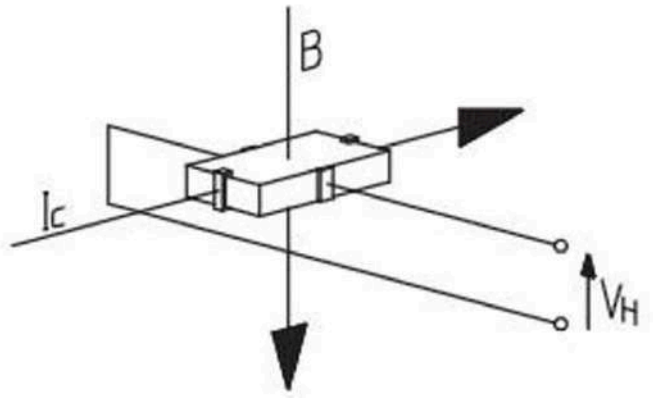

Figura B.1: Parâmetro elétricos do Efeito *Hall*.
Fonte: (MICHIGAN UNIVERSITY, 2013).

Os transdutores de corrente por efeito *Hall* são amplamente utilizados na industria, atendendo a maioria das aplicações no campo da eletrônica de potência tais como: inversores, conversores estáticos para motores CC, fontes ininterruptas de energia (*Uninterruptible Power Supplies-UPS*), fontes chaveadas (*Switched Mode Power Supplies-SMPS*) e fontes para para aplicações em soldagem.

Transdutores de Corrente por Efeito *Hall* em Laço Aberto (*Open Loop*)

A Figura B.2 ilustra o esquema de um transdutor em laço aberto. A corrente a ser medida passa pelo condutor gerando um campo magnético que se concentra em um núcleo magnético. O núcleo possui um entreferro onde um gerador de *Hall* é posicionado para medir a densidade de fluxo magnético.

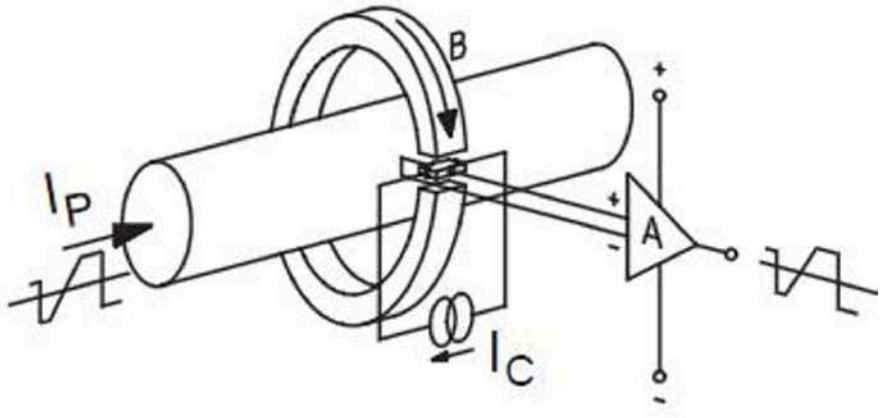

Figura B.2: Transdutor de corrente em laço aberto.
Fonte: (MICHIGAN UNIVERSITY, 2013).

Dentro da região linear da curva de magnetização do material usado para o circuito

magnético, a densidade de fluxo, B é proporcional à corrente do primário I_P e a tensão de *Hall* V_H é proporcional à densidade de fluxo. Portanto, a saída do gerador *Hall* é proporcional à corrente no primário mais a tensão de *offset* V_{OH}. O sinal medido é então compensado para remover a componente de *offset* e os efeitos da temperatura. Em seguida ele é amplificado para fornecer ao usuário a saída desejada. A Figura B.3 mostra a curva de magnetização do transdutor em laço aberto.

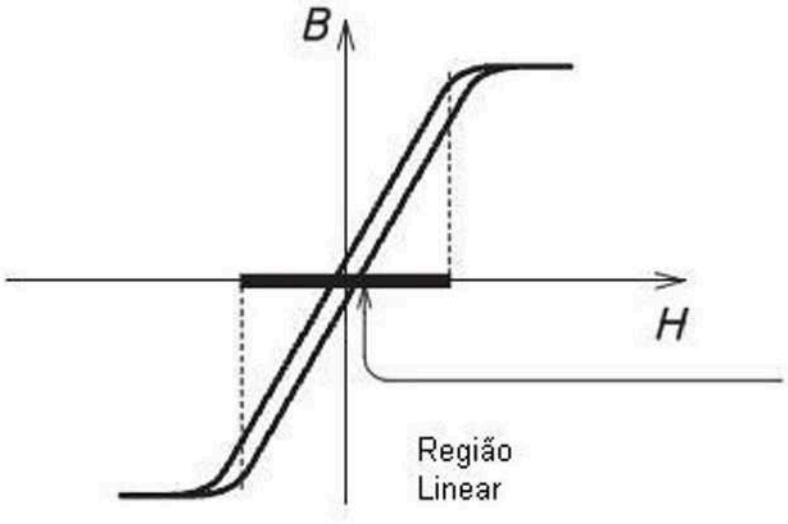

Figura B.3: Curva do transdutor de corrente em laço aberto.
Fonte: (MICHIGAN UNIVERSITY, 2013).

O transdutor em laço aberto mede tanto corrente CC quanto CA ao mesmo tempo que assegura isolamento galvânico. Suas principais vantagens são: leveza, tamanho pequeno, baixo custo e baixo consumo de energia. Eles são especialmente vantajosos em aplicações para medições acima de $300A$. Comparados com outras tecnologias suas limitações são: largura de banda e tempo de resposta moderadas e maior variação de ganho com a temperatura.

Transdutores de Corrente por Efeito *Hall* em Laço Fechado (*Closed Loop*)

Enquanto o transdutor de corrente em *loop* aberto amplifica a tensão do gerador *Hall* para fornecer uma tensão de saída, o transdutor em laço fechado usa a tensão do gerador *Hall* para criar uma compensação de corrente no enrolamento secundário. A corrente no secundário, I_S cria um fluxo igual em amplitude mas na direção oposta à criada pela corrente do primário. A operação do gerador *Hall* na condição de fluxo zero elimina a variação do ganho com a temperatura. Além disso, nessa configuração, em frequências mais altas, o enrolamento secundário se comportará como um transformador de corrente aumentando significativamente a largura de banda e reduzindo o tempo de resposta do transdutor. A Figura B.4 mostra o esquema de um transdutor em laço fechado.

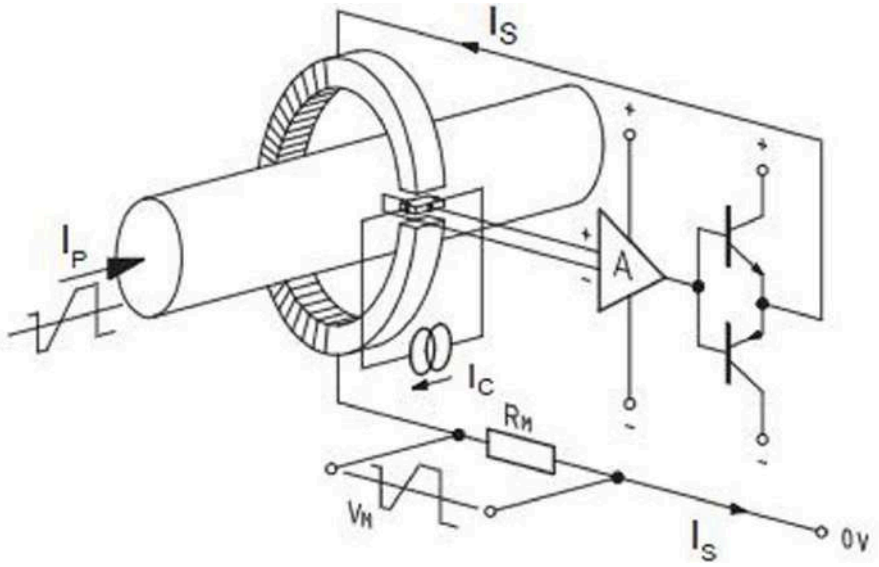

Figura B.4: Curva do transdutor de corrente em laço aberto.
Fonte: (MICHIGAN UNIVERSITY, 2013).

Quando o fluxo magnético é completamente compensado, e portanto zero, o potencial magnético dos dois enrolamentos são idênticos. Assim, $N_P \cdot I_P = N_S \cdot I_S$. Consequentemente, a corrente no secundário é exatamente a imagem da corrente no primário. Inserindo uma resistência de medição R_M em série com o enrolamento secundário, é criada uma tensão de saída que é a imagem da corrente medida.

Os transdutores de corrente em laço fechado também medem tanto corrente CC quanto CA e enquanto confere ao circuito isolamento galvânico. As vantagens desse dispositivo incluem excelente precisão e linearidade, baixa variação de ganho, largura de banda aumentada e tempo de resposta rápido. Suas principais limitações são: alto consumo de corrente na fonte secundária, maior dimensão e maior custo. Mais informações sobre transdutores de corrente por efeito *Hall* em MICHIGAN UNIVERSITY (2013).

Referências

ASHFAQ, A. *Eletrônica de Potência*. 2ª.ed. São Paulo, SP: Pearson Prentice Hall, 2000.

COSTA, . C. *Desenvolvimento de um Laboratório Virtual para Espectroscopia Mössbauer*: labvem. 2005. Programa de pós-graduação em engenharia elétrica — Universidade Federal de Minas Gerais. Escola de Engenharia, Belo Horizonte, MG.

DOEBELIN, E. O. *Measurement Systems, Aplication and Design*. 4ª.ed. New York, NY: McGRAW-HILL, 1990.

FRANCO, A. E. O. *Controle automático de um processo térmico com múltiplas entradas e múltiplas saídas utilizando técnicas de controle moderno*. CEFET-MG, Faculdade de Engenharia Mecatrônica.

GRIMALD, D.; RAPUANO, S.; LAOPOULOS, T. State of Art of the Distributed Measurement System for Industrial and Educational Purposes. *IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, [S.l.], p.289–294, September 2005.

GUSTAVISSON, I. *et al*. The VISIR project an Open Source Software Initiative for Distributed Online Laboretories. *Editorial Especial Engenharia Remota e Instrumentação Virtual - REV* 2007, [S.l.], December 2007.

HAWARD, J. *et al*. iLab: a scalable architecture for sharing online experiments. , [S.l.], October 2004.

ISSA, F. G. S. *Estudo Comparativo entre Banco de Dados Relacionais e Banco de Dados NoSQL na Utilização por Aplicações de Business Intelligence*. Faculdade de Tecnologia de São José dos Campos.

KOSTULSKI, T.; MURRAY, S. The National Engineering Laboratory Survey Labshare Project. , [S.l.], December 2010.

KRISHNAMURTHY, B.; REXFORD, J. *Redes para Web*. 1ª.ed. Boston, MA: Campus, 2001.

LERRO, F. *et al*. A remote lab like a didactic resource in the teaching of the physics of eletronic devices. 11^th *International Conference on Interactive Computer aided Learning - ICL* 2008, [S.l.], September 2008.

LOPES, V. J. F. *Instrumentação virtual aplicada ao ensino experimental de engenharia elétrica*. 2007. Programa de pós-graduação em engenharia elétrica — Escola Politécnica, Universiade de São Paulo, São Paulo, SP.

LOWE, D. *et al*. LabShare: towards a national approach to laboratory sharing. 20^th *Australasian Association for Engineering Education Conference - AAEE* 2009, [S.l.], December 2009.

MACHADO, F. N. R. *Banco de Dados - Projeto e Implementação*. 1ª.ed. São Paulo, SP: Érica, 2008.

MEIRA, R. *Banco de Dados*. Ilheus, BA: Instituto Federal da Bahia, 2011.

MICHIGAN UNIVERSITY. *Isolated Current and Voltage Transducers*. Disponível em `http://www.lem.com/images/stories/files/Products/` Acessado em: 13 Jun 2013.

NATIONAL INSTRUMENTS. *LabVIEW Intermediate I*: successful development practices course manual. Austin, Texas: National Instruments Corporation, 2008.

NATIONAL INSTRUMENTS. Instrumentação Virtual. , [S.l.], Nevember 2009.

NATIONAL INSTRUMENTS. Controladores Programáveis para Automação - o futuro para controle industrial. , [S.l.], October 2012.

NATIONAL INSTRUMENTS. *Instrumentação Virtual*. Disponível em `http://www.ni.com/white-paper/4752/pt` Acessado em: 15 Mai 2013.

NGOLO, M. A. F. *Arquitetura Orientada a Serviços REST para Laboratórios Remotos*. 2009. Programa de pós-graduação em engenharia electrotécnica e de computadores — Universidade Nova Lisboa, Faculdade de Ciências e Tecnologia, Departamento de Engenharia Eletrotécnica, Lisboa, Distrito de Lisboa.

NOVUS AUTOMATION. *Relé de Estado Sólido-SSR*. Disponível em `http://www.novus.com.br/downloads/Arquivos/5000015` Acessado em: 26 Jun 2013.

OLIVEIRA, E. de. *Prototipagem Rápida de Sistemas Mecatrônicos Baseada em Instrumentação Virtual*. 2008. Programa de pós-graduação em engenharia mecânica — Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica, Campinas, SP.

SILBERSCHATZ, A.; KORTH, F. H.; SUDARSHAN, S. *Sistema de Banco de Dados*. 5ª.ed. Rio de Janeiro, RJ: Elsevier, 2006.

SILVA, J. B. da *et al*. Uso de Dispositivos Móveis para Acesso a Experimentos Remotos na Educação Básica. *VAEP RITA*, [S.l.], v.1, n.2, p.129–134, June 2013.

SILVA, J. V. V. *Modelagem Matemática a parâmetros distribuídos e Controle em Malha Fechada de um Sistema de Aquecimento de Ar*. CEFET-MG, Faculdade de Engenharia Mecatrônica.

SIMEÃO, J. D. *Controle de Sistemas com Atraso nos Estados*: uma abordagem convexa. 2009. Programa de pós-graduação em engenharia mecânica — Centro Federal de Educação Tecnológica de Minas Gerais, Divinópolis, MG.

TAKAY, O. K.; ITALIANO, I. C.; FERREIRA, J. E. *Introdução a Banco de Dados*. São Paulo, SP: DCC-IME-USP, 2005.

ZUBIA, J. G. *et al*. Integración del laboratorio remoto Weblab Deusto en Moodle. *International Journal of Online Engineering* 2009, [S.l.], May 2009.