Serviço de Pós-Graduação EESC/USP

EXEMPLAR REVISADO

Data de entrada no Serviço: 16/01/03

Ass.:

DESENVOLVIMENTO DE UM SISTEMA DE APOIO AO DIAGNÓSTICO À DISTÂNCIA PARA PROCESSOS DE RETIFICAÇÃO

Douglas Dantas Soares

Dissertação apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia Mecânica.

ORIENTADOR: Prof. Dr. João Fernando Gomes de Oliveira

São Carlos, 2002

DEDALUS - Acervo - EESC

31100037179

EESC-USP Biblioteca Campus de S. Carlos

FOLHA DE JULGAMENTO

Candidato: Bacharel **DOUGLAS DANTAS SOARES**

Dissertação defendida e julgada em 18-11-2002 perante a Comissão Julgadora:

Aprovado

Prof. Tit. **JOÃO FERNANDO GOMES DE OLIVEIRA (Orientador)**
(Escola de Engenharia de São Carlos/USP)

Aprovado

Prof. Assoc. **REGINALDO TEIXEIRA COELHO**
(Escola de Engenharia de São Carlos/USP)

Aprovado

Profa. Dra. **SOLANGE OLIVEIRA REZENDE**
(Instituto de Ciências Matemáticas e de Computação/USP)

Prof. Associado **ARTHUR JOSÉ VIEIRA PORTO**
Coordenador do Programa de Pós-Graduação
em Engenharia Mecânica

Profa. Assoc. **MARIA DO CARMO CALIJURI**
Presidente da Comissão de Pós-Graduação

Dedico este trabalho à minha mãe Inês e
ao meu pai Guilherme em nome do amor,
do carinho, da paciência, dos esforços,
da saudade e dos princípios, fatores
que foram essenciáis na minha formação.

Agradecimentos

A DEUS, pela vida e pelas oportunidades traçadas em meu caminho.

Aos meus pais, pelos ensinamentos, esforços e orientações.

Aos meus irmãos Rodolfo, Déborah, Adriana, ao meu cunhado Rildo, e ao meu sobrinho Thiago, pelo amor e carinho.

Ao professor João Fernando, pela oportunidade, orientação, amizade, e pelos diversos ensinamentos de vida e princípios proporcionados durante todo o período de elaboração deste trabalho.

Ao professor Reginaldo, pelos conselhos, amizade, e pelo ótimo humor no dia-a-dia do trabalho.

A Irene, pela amizade, carinho, pelos conselhos e pela valiosa ajuda dada durante toda essa caminhada.

Ao Ronaldo pela amizade, dedicação e pelos serviços prestados.

Ao pessoal do OPF, Alexandre, Fábio, Carlão, Eraldo, Marcelo, Bob, Aldo, Salete, Mariana, João Jorge, Roger, Leonardo, Jalon, Octaviano, Gustavo Meireles, Giuliano, Tadeu, Américo, Paulinho, Marcelinho, Daniel e toda a turma da nova fase 2002.

Agradeço ao pessoal da secretaria de pós graduação e a todos os funcionários que colaboraram para a efetivação deste trabalho.

Aos meus amigos de São José dos Campos, Gutito, Fabiano, Galvão, Luis Otávio, Jorge, Cassiano e Flávio, pela duradoura amizade, incentivo e pela descontração.

Aos músicos de São Carlos, em especial ao Danilo, Maria e família, ao Sato e ao André, pelo companheirismo, ensinamentos e amizade.

A república Santo Mé, pela amizade, fraternidade e pelos ensinamentos de vida durante toda a minha graduação em Bauru.

A TRW por proporcionar a realização de testes referentes a este trabalho.

A FAPESP pela bolsa de estudo que permitiu a dedicação integral a este trabalho.

A EESC pela oportunidade de aprofundar meus conhecimentos nesta reconhecida e competente instituição de ensino.

Sumário

Lista de Figuras 7

Lista de Tabelas 8

Lista de Siglas 8

Resumo do Projeto 10

1 - Introdução 12

1.1 - Objetivos 14

1.2 – Estrutura da Dissertação 14

2 – Conceitos Computacionais Relacionados aos Sistemas Remotos 16

2.1 - A Escolha do Sistema Operacional 16

2.2 - Conceitos de Comunicação e Distribuição 19

2.2.1 - O Modelo de Referência OSI 19

2.2.2 - O Protocolo TCP / IP 22

2.3 - Softwares de Aquisição e Análise de Dados e Sinais 25

2.3.1 - LabVIEW 25

2.3.2 - Softwares SCADA 25

2.4 - Programação Orientada a Objeto 27

2.5 - Base de Dados 31

2.5.1 - A Linguagem SQL (Strutured Query Language) 34

2.5.2 - Escolha da Base de Dados 34

2.6 - Acesso à Base de Dados via Web 35

2.6.1 - Páginas Dinâmicas na Web Implementadas em Delphi 38

2.6.2 - Aplicações Delphi via TCP/IP 40

2.7 - Padrões de Integração entre Base de Dados e Softwares 40

2.8 - Servidor HTTP 43

3 - Conceitos sobre Retificação e seu Monitoramento ____ **44**

3.1 - O Rebolo ____ *45*

3.2 - Dressagem ____ *49*

3.3 - Ciclo de Retificação ____ *51*

3.4 - Monitoramento da Retificação ____ *52*

3.5 - Sensoreamento ____ *55*

4 - Proposta de Arquitetura de um Sistema para Monitoramento Remoto ____ **62**

4.1 - Sistema de Aquisição ____ *63*

4.2 - Problemas de Retificação & Medidas de Sinais ____ *80*

4.3 - Detecção Automática de Falhas ____ *84*

4.4 - Base de Dados ____ *89*

4.5 - Sistema de Análise Remota ____ *92*

4.6 – Proposta de uma Possível Solução Integrada ____ *96*

5 - Validação do Sistema ____ **97**

6 - Conclusões ____ **102**

7 – Sugestões para Trabalhos Futuros ____ **103**

8 - Referências Bibliográficas ____ **104**

9 - Apêndices ____ **108**

1 – Breve Histórico da Internet ____ *108*

2 - Informações Adicionais sobre a Implementação de CGIs ____ *111*

3 - Informações Adicionais sobre a Linguagem PHP ____ *115*

4 - Descrição das Tabelas da Base de Dados ____ *120*

5 - Informações Adicionais sobre Integração entre PHP e MySQL ____ *124*

6.-.Informações Adicionais sobre Componentes PageProducers e DataSetTableProducers _____ *127*

7 - Códigos Utilizados na Programação da Aplicação Remota _____ *132*

LISTA DE FIGURAS

Figura 1 - Modelo de Referência OSI 20
Figura 2 - Comunicação TCP/IP 25
Figura 3 - Modelagem Oientada a Objeto de uma Retificadora 28
Figura 4 - Níveis de Hierarquia das Classes 29
Figura 5 - Elementos do Modelo de Entidades e Relacionamentos 32
Figura 6 - Atributos de um Modelo de Entidades e Relacionamentos 33
Figura 7 - Relacionamento entre Tabelas 34
Figura 8 - Esquema Genérico de uma CGI 36
Figura 9 - Funcionamento de uma CGI 37
Figura 10 - Passos para Uso do ODBC 42
Figura 11 - Dressadores Estáticos 50
Figura 12 - Dressadores Rotativos 51
Figura 13 - Ciclo de Retificação de Mergulho [1] 52
Figura 14 - Acelerômetro Piezelétrico 58
Figura 15 - Componentes de um LVDT [15] 60
Figura 16 - Princípio de Funcionamento de um Sensor de Proximidade 61
Figura 17 - Arquitetura do Sistema de Diagnóstico à Distância 63
Figura 18 - Interface do Sistema de Aquisição 64
Figura 19 - Fixação do LVDT na retificadora centerless Microma 65
Figura 20 - Fixação do Sensor de EA Próximo ao Rebolo 66
Figura 21 - Fixação do Sensor de EA Próximo ao Fenômeno de Dressagem. 66
Figura 22 - Fixação do Sensor de Proximidade para Retratar o Curso da Dressagem em Passos Duplos 67
Figura 23 - Fixação do Sensor de Proximidade para Retratar o Curso da Dressagem em Passos Únicos 67
Figura 24 - Dispositivo Fixado no Rebolo para a Captura de sua Rotação 68
Figura 25 - Sensor de Proximidade para Capturar a Rotação do Rebolo 68
Figura 26 - Sensor de Efeito Hall Fixado na Máquina 69
Figura 27 - Aquisição do Tempo de Ciclo 70
Figura 28 - Aquisição do Tempo de Retificação 71
Figura 29 - Aquisição do Sobremetal da Peça 72
Figura 30 - Aquisição da Energia Consumida e Pico de Potência na Retificação 73
Figura 31 - Aquisição do Tempo de Dressagem 75
Figura 32 - Energia Consumida pela Máquina na Dressagem 76
Figura 33 - Arquitetura de Hardware do Sistema de Aquisição 77
Figura 34 - Problemas de Dressagem 81
Figura 35 - Ciclo da Retificação 81
Figura 36 - Sinal de EA – Rebolo Colidindo na Peça com Velocidade de Aproximação 82
Figura 37 - Deslocamento de Corte Excessivo 82
Figura 38 - Sinal de EA – Peça com Sobremetal Abaixo do Mínimo Necessário 83
Figura 39 - Sinal de EA – Tempo de Spark-out Inadequado 83
Figura 40 - Sinais de Potência e Vibração – Perda de Capacidade de Corte (comportamento duro) e de Capacidade Volumétrica do Rebolo (comportamento mole). 84
Figura 41 - Rebolo Avançando Não Uniformemente 84
Figura 42 - Cálculo do Sobremetal da Peça 85

Figura 43 - MySQL Web Interface 90
Figura 44 - Sistema de Análise Remota 92
Figura 45 - Aplicação do Operador 93
Figura 46 - Análise de Ciclo na Aplicação do Consultor 94
Figura 47 - Ferramenta de Apoio à Tomada de Decisão 95
Figura 48 - Análise Geral do Teste 96
Figura 49 - Peça Utilizada no Teste em Questão 98
Figura 50 - Análise Geral das Grandezas Monitoradas 99
Figura 51 - Análise do Comportamento do Sobremetal Removido das Amostras 99
Figura 52 - Velocidade Média e Mínima do Rebolo no Decorrer do Teste 99
Figura 53 - Comparativo entre Ciclos Diferentes num Mesmo Gráfico 100
Figura 54 - Gráficos de Ciclos de Amostras Diferentes 100
Figura 55 - Distribuição do Tempo 101
Figura 56 - Modelo de um Componente WebModule 127

LISTA DE TABELAS

Tabela 1 - Afinidade Química dos Grãos Abrasivos 477
Tabela 2 - Valores da Relação ρ 888
Tabela 3 - Valores de Λw 888
Tabela 4 - Organizações de Controle da Internet 1099

LISTA DE SIGLAS

ERP - *Enterprise Resource Planning*
DBMS - Sistema de Gerenciamento de Base de Dados
S.O. - Sistema Operacional
G.U.I - *Graphics User Interface*
OSI - *Open System Interconection*
IP - *Internet Protocol*
ARP - *Adress Resolution Protocol*
ICMP - *Internet Control Message Protocol*
UDP - *User Datagram Protocol*
TCP - *Transport Control Protocol*
RIP - *Routing Internet Protocol*
SMTP - *Simple Mail Transfer Protocol*
DNS - *Domain Name Server*
LabVIEW - *Laboratory Virtual Instrument, Engineering Workbench*
VIs - *Virtual Instruments*
SCADA - *Supervisory Control and Data Acquisition*
MES – *Manufacturing Execution System*
RAD - *Rapid Application Development*
DLL - *Dynamic Link Libraries*

MER - Modelo de Entidades e Relacionamentos
SQL - *Strutured Query Language*
CGI - *Commom Gateway Interface*
ODBC - *Open Database Conectivity*
ADO - *ActiveX Data Objects*
HTTP - *Hypertext Transfer Protocol*
LDAP - *Lightweight Directory Access Protocol*
LVDT - *Linear Variable Differential Transformer*
EA - Emissão Acústica
RPM - Rotações por Minuto
PHP - *Personal Home Page Tools*
P = Potência consumida (W)
a = Profundidade de penetração do rebolo por volta (mm)
b = Largura de corte (mm)
Fn = Força Normal (N)
Ft = Força Tangencial (N)
Vs = Velocidade periférica do rebolo (m/s)
Vw = Velocidade periférica da peça (m/s)
ρ = Relação entre força tangencial e normal
heq = Espessura de corte equivalente (mm)
Kh = Constante de Hahn (proporcionalidade entre força e heq) (N/mm)
Λw = Parâmetro de remoção de material (mm^3 / N . s)

RESUMO DO PROJETO

Dentre os processos de usinagem, a retificação é um dos mais difíceis de serem controlados. Em virtude de ser um processo de acabamento final, a peça, nesta etapa, possuí alto valor agregado. A retificação apresenta altos níveis de complexidade em virtude do grande número de possíveis condições de processo. Além disso, um sistema composto por máquina, peça, ferramentas de corte, sistema de afiação e dispositivo de fixação tem a solução de problemas e a otimização do processo difíceis de serem atingidas. Por ser um processo ainda pouco dominado, a resolução de problemas, a qual exige observação contínua dos fenômenos envolvidos, fica a cargo da escassa gama de experientes operadores de máquina. Como conseqüência, máquinas e produção podem ficar paradas ou serem utilizadas com baixo desempenho, resultando em prejuízos para a empresa.

Com o crescente uso de múltiplos sensores para se realizar o monitoramento de processos, a análise de sinais têm-se mostrado eficiente na detecção de problemas e na otimização das operações de retificação. Esses sensores, integrados à ferramentas de aquisição de sinais e a uma base de dados constituem um sistema de monitoramento e diagnóstico para processos de retificação. Além disso, com o uso de tecnologias da Internet, é possível se realizar diagnósticos e monitoramentos em retificadoras à distância. O objetivo desse trabalho é o desenvolvimento de um sistema de diagnóstico e correções do processo de retificação via Internet. É importante ressaltar a necessidade de portabilidade do sistema que deve ser instalado em qualquer máquina conectada à Internet, para monitorar qualquer operação de retificação.

Palavras Chaves: Retificação, Diagnóstico, Remoto, Internet.

ABSTRACT

Among the machining processes, grinding is one of the most difficult to be controlled. Since it is a finishing process, the component, in this step, has a high aggregated value. Grinding has high levels of complexity because of the large

number of process conditions. Besides this, a system composed by machine, part, cutting and sharpener tools and a fixing device make difficult to solve problems and to optimise the process. Because grinding is a process which is not well dominated , the problem solving that requires continuous observation of the involved phenomena lies in the hands of the limited group of experts. As a result, machines and production can be stopped or used with low performance conditions. This may represents loss to the enterprises.

With the growing use of multiple sensors to realise the process monitoring, the signal analyses has shown to be an efficient way in the problem detections and in the optimization of grinding operations. These sensors, integrated with signal acquisition tools and a data base may compose a monitoring and diagnosys system for grinding processes. More than this, with the use of Internet technologies it can execute remote monitoring and diagnosys in grinding machines. The objective of this research is to develop this grinding diagnosis system based on sensor and Internet tools. It is important to emphasize the portability of the system that should be installed in any machine connected to the Internet and to monitor any grinding operation.

Keywords: Grinding, Diagnosys, Remote, Internet.

1 - INTRODUÇÃO

No panorama mundial, o mercado vem tornando-se cada vez mais complexo. Regido pela globalização, ele vem apontando a competitividade como um dos requisitos com o qual a empresa mais deve se preocupar. E alcançá-la significa que baixo custo, qualidade e alta produção devem caminhar juntos em busca de satisfazer as exigências do cliente, de forma eficiente e rápida.

Analisando a realidade das indústrias de manufatura, particularmente as que utilizam processos de usinagem, percebe-se que em meio ao crescente desenvolvimento tecnológico, o monitoramento da produção é bastante deficiente. Os dados de produção e de máquina, na maior parte dos casos, são coletados de forma manual e não retratam a realidade do chão-de-fábrica. Conseqüentemente, as informações geradas a partir desses apontamentos manuais são inseridas nos sistemas ERP (*Enterprise Resource Planning*) que geram respostas irreais para análises em um processo de tomada de decisão. Essa cascata de problemas pode provocar sérios abalos numa empresa, tais como: não atendimento aos requisitos de qualidade desejados pelo cliente, baixas taxas de produção, altos custos e definição inadequada de necessidades de melhorias, prejudicando assim o aumento da sua competitividade.

Dentre os processos de usinagem conhecidos, pode-se ressaltar a retificação como sendo um dos mais importantes da manufatura. Por se tratar de um processo de acabamento final, a peça, nesta etapa, possui um valor agregado bastante elevado. A retificação apresenta alto grau de complexidade em virtude do grande número de possíveis condições de processo, sendo ainda hoje um processo muito obscuro e difícil de ser controlado. Além disso, a complexidade de um sistema composto pela máquina, ferramenta, peça e dispositivos de fixação torna a solução de problemas difícil de ser atingida, já que a sua análise requer uma observação contínua dos fenômenos envolvidos. Somado a isso, o escasso grupo de especialistas gabaritados no assunto encontra-se disperso pelo mundo, fato que torna impraticável seu auxílio *in loco* para a análise e solução dos problemas. Como conseqüência, máquinas e produção podem ficar paradas ou serem utilizadas com baixo desempenho, resultando em prejuízos para a empresa.

Em meio às dificuldades de controle de uma operação de retificação, as exigências para se manter em carater competitivo vêm se tornando cada vez mais difíceis de serem atendidas. A complexidade do sistema faz com que o seu controle fique a cargo quase que totalmente do operador. Na necessidade de escassa mão de obra especializada, capaz de decifrar os enigmas deste processo, a análise local torna-se mais uma dificuldade para a empresa. Nesse cenário, observa-se máquinas e produções paradas a espera de soluções demoradas, uma vez que a escassez de especialistas e a distância de suas localidades tornam-se barreiras difíceis de serem quebradas.

A análise de sinais provenientes de um sistema de retificação pode facilitar o diagnóstico de problemas. Esses sinais podem ser: emissões acústicas, vibrações e potência consumida [44].

Por outro lado, a facilidade de se poder transmitir dados via Internet viabiliza um sistema de diagnóstico à distância, eliminando assim a análise local do sistema para o apontamento de possíveis soluções.

O levantamento de dados do processo e suas inserções em uma base de dados normalmente demanda grandes períodos no cotidiano do setor de controle de produção de uma empresa. Um sistema de supervisão de processos é capaz de armazenar os dados em uma base de dados, realizar análises, fazer estatísticas, disponibilizar consultas e gerar relatórios, tudo de uma forma eficaz e segura.

Tudo isso, junto às vantagens da comunicação via Internet, possibilita consultorias necessárias nas soluções de problemas e otimizações à distância.

Um sistema deste tipo pode então dispor de:

- Visualizador de sinais do processo (emissão acústica, vibrações, potência consumida pela máquina);
- Gráficos de análise e desenhos de sinais;
- Som em tempo real;
- Dados de máquina e produção;
- Imagens da máquina em tempo real.

1.1 - OBJETIVOS

Este projeto tem como objetivo principal desenvolver um sistema para o diagnóstico à distância de problemas de operações de retificação que até então é dependente de análises locais. Ainda tem-se o intuito de melhorar o monitoramento do processo por meio de um eficiente sistema de supervisão, baseado na coleta e análise de sinais de máquinas e produção através de sensores de Emissão Acústica, Vibração, Potência, entre outros. A ferramenta deve ainda ser capaz de gerar informações necessárias para tomadas de decisões incluindo a geração de relatórios amigáveis e robustos.

1.2 – ESTRUTURA DA DISSERTAÇÃO

Este documento - apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo, trata de uma proposta de desenvolvimento de um **Sistema de Diagnóstico à Distância para Processos de Retificação**.

Para melhor entendimento deste assunto, dividiu-se este documento em 8 capítulos.

- No Capítulo 1 dá-se uma introdução ao trabalho;
- No Capítulo 2 é feita uma revisão bibliográfica dos conceitos computacionais relacionados a sistema remotos;
- No Capítulo 3 aborda-se uma revisão bibliográfica dos conceitos sobre retificação e monitoramento de processos;
- No Capítulo 4 descreve-se a arquitetura proposta para o desenvolvimento do sistema de diagnóstico remoto para processos de retificação;
- No Capítulo 5 é feita a análise de alguns testes realizados com o sistema implementado;
- No Capítulo 6 descreve-se as conclusões do trabalho;
- No Capítulo 7 é apresentada uma lista de referências bibliográficas consultadas.

- Os assuntos complementares foram agrupados em 7 apêndices inseridos no Capítulo 8.

2 – CONCEITOS COMPUTACIONAIS RELACIONADOS AOS SISTEMAS REMOTOS

Num sistema em que se deseja avaliar dados e informações remotamente, objetivando-se detectar problemas, buscar soluções, observar possíveis otimizações e tomar decisões, deve-se primeiramente disponibilizar informações confiáveis, que retratem justamente a realidade do processo em observação. Esse tipo de sistema é composto basicamente por:

- *software* cliente;
- *software* servidor;
- protocolo de transmissão e interconexão remota (TCP/IP, Frame Relay, e outros);
- base de dados.

Desses itens citados, um dos mais importante, levando em consideração a confiabilidade e o gerenciamento da informação, sem dúvida é a base de dados. Portanto, todo cuidado é necessário na escolha e modelagem do Sistema de Gerenciamento de Base de Dados (DBMS) que viabilizará o sistema [14].

De um sistema de monitoramento à distância espera-se eficiência, portabilidade, escalabilidade, robustez e confiabilidade nas informações disponibilizadas. Nesse sentido, esse tipo de solução contém algumas peculiaridades, as quais devem ser atentamente analisadas [14]. A seguir serão estudados os principais elementos adotados para se efetuar monitoramentos e diagnósticos remotos.

2.1 - A ESCOLHA DO SISTEMA OPERACIONAL

Mediante às vantagens que serão apresentadas adiante, escolheu-se o sistema operacional Linux como plataforma básica de desenvolvimento do sistema proposto.

Desde 1994, o sistema operacional Linux vem se multiplicando nos computadores PCs por ser uma plataforma *freeware* com muita confiança e boa

operabilidade. O Linux nasceu como um projeto de um único homem, Linus Torvalds que naquela época estudava na universidade da Finlândia em Helsinque.

Por um bom tempo, o Linux foi um sistema operacional em desenvolvimento, distribuído em grande parte pela Internet. Foi em 1994 que a primeira versão 1.0 do Linux foi disponibilizada [7].

As principais vantagens do uso da plataforma Linux são [8]:

- É o resultado de um constante trabalho de voluntários que acreditam numa abordagem simples para desenvolvimento de *software*.
- É um sistema operacional completamente aberto. O código fonte do Sistema Operacional Linux vem no próprio CD de instalação, ou está disponibilizado em vários sites na Internet. Dessa forma, um programador mais experiente pode alterar o S.O. para usá-lo de acordo com suas próprias necessidades;
- Contém uma série de ferramentas de desenvolvimento e programação em diversas linguagens C, C++ e Perl. Além disso, uma gama de aplicativos comerciais, gráficos, financeiros, de multimídia e inúmeros sistema de gerenciamento de base de dados são disponibilizados na Web a custo zero;
- Oferece grande quantidade de informações *online*, incluindo tutoriais, manuais, FAQs e grupos de discussões sobre tópicos relacionados à parte de instalação, configuração e manuseio do S.O. para vários tipos de PC e periféricos;
- Possui grande estabilidade em ambientes multiusuários e multiprogramados, o que difere de muitos sistemas operacionais existentes no mercado.

Linux roda perfeitamente sobre os processadores Intel 80386, 80486 e Pentium, em todas as suas variações. A placa-mãe deve ser de barramento ISA ou EISA. O Linux não suporta o barramento MCA da IBM e aconselha-se o uso do barramento VESA.

São necessários no mínimo 16 Mb de RAM . Para sua instalação enxuta, 802 Mb de disco rígido são suficientes. No entanto, no desejo de se instalar todos os

aplicativos disponíveis no CD, um espaço bem maior em torno de 1 a 2,5 Gb é necessário [5].

O Ambiente Linux no Modo Texto

Ao observar o Linux em modo texto, nota-se a presença de vários programas capazes de interpretar os comando do sistema. Esses programas interpretadores são chamados de *Shell*, e fazem a interface de comunicação entre o S.O. e o usuário. Essa interface é feita através de comandos recebidos pelo teclado. Por esses comandos, o usuário pode acessar serviços e aplicativos do S.O.. É importante ressaltar que toda a interface entre o sistema operacional e o *hardware* envolvido é feita através de arquivos, alocados em diretórios padronizados. Essa arquitetura facilita o gerenciamento de eventos, a configuração de periféricos e a detecção de possíveis erros e incompatibilidades do sistema [10].

O Bash é o *shell* mais usado no Linux por possuir uma grande variedade de comandos existentes em outros *shells*. Sendo assim, a operacionalidade do sistema é maior com o seu uso.

Existem outras interfaces com o usuário mais amigáveis que o *shell* e elas podem ser carregadas do próprio *shell* a partir de um simples comando. Essas interfaces são chamadas de G.U.I (*Graphics User Interface*). Algumas delas têm aparências semelhantes à interface gráfica do Windows 98, como é o caso do XWINDOW.

O Ambiente Linux no Modo Gráfico

O sistema XWINDOW (ou X11) executa um processamento gráfico em uma arquitetura de múltiplos processos. O principal processo chamado X é o próprio X *server*. O Servidor lida com solicitações locais e solicitações via rede baseadas em TCP/IP. Em virtude desta capacidade é possível rodar um aplicativo X em uma estação de trabalho e apresentar os resultados em outra estação.

O X *server* controla o monitor, teclado e mouse e permite que aplicativos gráficos chamados X *clients* criem as janelas e as ocupem.

O Processo X *server* é o único autorizado a desenhar pontos na tela ou rastear o mouse. Os programas aplicativos X se conectam ao X *server* através de um *link* de comunicação entre processos através de *sockets* (*ports*) na rede TCP/IP [10].

O Linux em Rede

O trabalho em rede se refere a todos os aspectos relacionados à troca de mensagens e dados entre um ou mais computadores desde a parte física da rede, até a camada mais alta da aplicação [4].

O protocolo de rede é um conjunto de regras utilizadas para padronizar uma comunicação de dados em uma rede de computadores. No mundo do Linux, os principais protocolos de comunicação situados nas diferentes camadas do Modelo OSI são suportados [4]. A arquitetura OSI de redes será detalhada na próxima sessão.

2.2 - Conceitos de Comunicação e Distribuição

Um sistema de comunicação é uma combinação de *hardwares* e *softwares* que suportam troca de informações entre sistemas e processos, entre componentes de *software* e estações (nós, ou *hosts*) distribuídas e interligadas por uma rede [15].

Os nós estão interconectados por uma rede que oferece um caminho físico entre eles. Um sistema que executa as principais funções de um aplicativo e que controla comunicações é denominado servidor.

Esses conceitos citados acima são de extrema importância em um sistema remoto, que nada mais é que um sistema de comunicação tendo o TCP/IP como protocolo fundamental de comunicação.

2.2.1 - O Modelo de Referência OSI

A normalização da **ISO** estruturou a rede como um conjunto de camadas (níveis) hierárquicas, cada uma sendo construída utilizando as funções e serviços oferecidos pelas camadas inferiores. Esse modelo, chamado de OSI (*Open System Interconection*), é usado para permitir um grande fluxo de dados entre equipamentos e redes, ou seja, permite a definição de procedimentos que possibilitem a interconexão e a subseqüente troca efetiva de informações entre os equipamentos.

Essas trocas podem ser realizadas entre um ou mais computadores, *softwares* associados, servidores, periféricos, terminais, processos físicos, meio físico de transferência de informações, etc. Forma-se, então um conjunto autônomo capaz de desempenhar o processamento e/ou a transferência da informação [16]. Os computadores do tipo PC utilizam este modelo de comunicação.

O modelo OSI divide em sete tarefas diferentes o trabalho do deslocamento dos dados de um ponto para o outro de um sistema de comunicação. Essas tarefas ou camadas são organizadas de modo hierárquico, conforme é mostrado na figura 1.

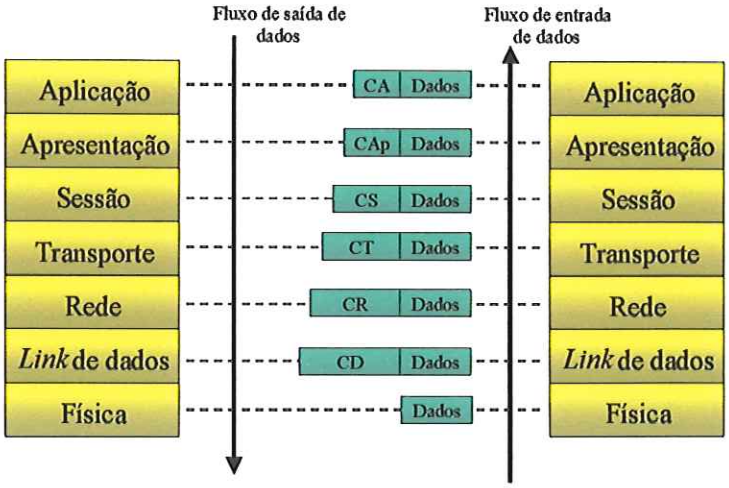

Figura 1 - Modelo de Referência OSI

Cada camada contribui para o agrupamento ou separação de um pacote. Um pacote nada mais é que um feixe discreto de *bits* que trafega pela rede de comunicação [15]. Cada pacote por sua vez, realiza as seguintes funções:

- Avisa as placas de recebimento que um pacote está a caminho;
- Informa o destino, a origem, o tipo do pacote (se é um pacote de controle, de dados ou de rede);
- Transporta os dados;
- Controla erros através dos *bits* finais de verificação de erros.

CAMADA FÍSICA

A camada física é representada por um conjunto de regras que especificam as conexões elétricas e físicas entre os equipamentos e pela sinalização. As camadas

subseqüentes se comunicam através da camada física. Os serviços que ela especifica para a camada de enlace de dados são aqueles relacionados com o conectar, manter conectado e desconectar os circuitos físicos que formam a conexão física. Ela é composta por fios de par trançado, cabos RS-232, RS-485, cabos de fibra ótica ou cabos coaxiais e transporta os sinais das camadas mais altas. O *hardware* e o *software* que operam nesse nível tratam dos tipos de conectores, da sinalização e dos esquemas de compartilhamento dos meios físicos usados na rede.

CAMADA DE ENLAÇE DE DADOS

É o primeiro nível que reúne os *bits* e trata os dados como pacotes. Ela realiza a primeira inspeção nos dados. Esta camada adiciona uma correção de erros para os pacotes que estão saindo e uma soma de verificação para os pacotes que estão chegando. Os pacotes incompletos são descartados e se a camada de Enlace conseguir determinar a origem do pacote, ela retornará um pacote de erro.

CAMADA DE REDE

Esta camada se preocupa em rotear os pacotes de modo a garantir que o pacote chegará ao dispositivo e a sub-rede correta. Ela mantém tabelas de roteamento, determina a rota mais rápida disponível e o momento de se utilizar rotas alternativas. O protocolo IPX da *Netware* e o protocolo TCP/IP operam neste nível. Essa camada é o nível de operações de serviços "sem conexão".

CAMADA DE TRANSPORTE

Este é um nível de transição, ou seja, o último nível que gerencia os pacotes de roteamento e a recuperação de erros. A camada de tranporte é responsável pelo controle de fluxo de pacotes transmitidos e pela multiplexação e difusão dos mesmos. Os pacotes recebidos da camada superior são divididos em pacotes menores, se necessário. Se os pacotes forem recebidos de maneira confiável no nível inferior, o trabalho neste nível será simples. Porém se o sistema de comunicação não puder oferecer uma transmissão segura, este nível fará a compensação realizando um trabalho mais complexo. Além do mais é nesse nível que são estabelecidas as conexões e desconexões.

CAMADA DE SESSÃO

Esta camada é responsável pelo controle de tráfego de pacotes, pelo gerenciamento de *tokens* e pela sincronização por pontos. Às vezes, para a que as mensagens sejam enviadas ou recebidas com um alto grau de segurança, é necessário que se estabeleça uma conexão formal, ou seja, transmissões "orientadas a conexões". Nesse nível assume-se que esses pacotes são seguros. Para um sistema remoto, deve-se ressaltar que a base de dados é acessada através de conexões formais e o protocolo TCP é utilizado para transmissões orientadas a conexões.

CAMADA DE APRESENTAÇÃO

O processamento nesta camada realiza qualquer conversão que possa ser exigida pela camada de aplicação para que os dados tornem-se utilizáveis. Por exemplo, os processos de compactação e descompactação, criptografia e decriptografia podem ser implementados neste nível.

CAMADA DE APLICAÇÃO

A camada de aplicação trata de assuntos de segurança e disponibilidade de recursos. Ela tende a lidar com transferência de arquivos e *jobs*, além de lidar com protocolos de terminais virtuais (TELNET).

2.2.2 - O Protocolo TCP / IP

O TCP/IP é uma família de protocolos utilizada nas comunicações entre computadores. TCP/IP é o acrônimo para *Transmission Control Protocol / Internet Protocol*. O TCP e o IP são protocolos individuais que podem ser discutidos de modo isolado, todavia não são os únicos que compõem esta família. Como o TCP/IP foi desenvolvido pelo Departamento de Defesa norte-americano essa família de protocolos é algumas vezes denominada conjunto DoD.

O modelo TCP/IP tem quatro camadas: a camada de aplicação, a camada de transporte, a camada de Internet e a camada de rede. É importante ressaltar que algumas camadas do modelo TCP/IP tem o mesmo nome das camadas do modelo OSI, mas com funções diferentes [51] .

A família de protocolos TCP/IP inclui protocolos como o IP (*Internet Protocol*), ARP (*Adress Resolution Protocol*), ICMP (*Internet Control Message Protocol*), UDP (*User Datagram Protocol*), TCP (*Transport Control Protocol*), RIP (*Routing Internet Protocol*), Telnet, SMTP (*Simple Mail Transfer Protocol*), DNS (*Domain Name Server*) e muitos outros [15].

Esta família de protocolos é a base de comunicação em um sistema remoto, onde tem-se comunicações orientadas a conexão e "sem conexão" (lógica). Para que o administrador do sistema possa configurar o servidor de qualquer posição geográfica, por exemplo, basta se conectar a Internet e utilizar o protocolo de terminal virtual, o TELNET, para se conectar e administrar a máquina remotamente, desde que tenha permissões suficientes para isto [49].

No TCP/IP todos os protocolos são transmitidos por uma inter-reder IP e encapsulados em pacotes IP. O pacote IP é roteável, ou seja, dois nós em uma rede que estejam se comunicando com o IP não precisam estar conectados no mesmo fio físico para trocar pacotes.

O endereço IP é necessário para qualquer troca de informações pela Internet. Cada equipamento conectado à Internet deve ter um endereço IP único. Este endereço pode ser atribuído basicamente de duas formas:

Fixamente: cada equipamento tem sempre o mesmo número IP atribuído durante a configuração. É normalmente o caso de universidades e grandes empresas americanas que se conectaram cedo à Internet e pegaram uma grande quantidade de endereços IP.

Dinamicamente: o computador recebe um endereço IP "emprestado" enquanto estiver conectado. Assim que ele se desconectar, um outro computador pode usar o endereço. É o método normalmente usado em acesso via linha discada a um provedor.

Os endereços IP são compostos por quatro bytes (32 bits) e cada byte é gravado como um número decimal separado por pontos. Como cada parte representa um byte, seu valor tem de estar entre 0 e 255 [15].

Na dificuldade de se decorar números grandes como os endereços IP, foi criada uma espécie de "lista telefônica" que atribui sinônimos em forma de caracteres a endereços IP. Por exemplo, www.ramosdainformatica.com.br é um "sinônimo" de

IP. Esta tabela de nomes usa o protocolo "DNS". A base de dados do DNS é mantida pelos órgãos gestores da Internet (no Brasil é a FAPESP).

Os endereços IP apresentam dois componentes: um componente de rede e o outro componente de nó. Para um endereçamento adequado, os endereços IP são divididos em classes. As três classes de redes mais importantes são A, B e C. Os endereços IPs são divididos nessas três classe de acordo com o primeiro Byte:

CLASSE A – 0 a 127

CLASSE B – 128 a 191

CLASSE C – 192 a 233

Para estabelecer uma conexão TCP/IP é preciso identificar as extremidades dessa conexão tanto no processo cliente como no processo servidor. Essas extremidades são denominadas *sockets*.

Sockets são os pontos terminais (extremidades) de uma conexão. São mecanismos de comunicação do sistema operacional que estabelecem uma conexão de pacotes bidirecional (*full-duplex*, o tráfego pode ser feito em ambas as direções ao mesmo tempo) entre endereços. Os endereços são formados pelo nome do *host* (ou endereço IP) e pela porta de comunicação (representada por um número de 16 bits). Os *Sockets* podem ser utilizados por várias conexões ao mesmo tempo [46].

Observa-se na figura 2 uma comunicação cliente / servidor. A aplicação cliente em execução na máquina hostC controla a porta portC e outra aplicação servidora em execução na máquina hostS monitora a porta portS do hostS. A aplicação cliente utiliza a porta portC da máquina hostC para enviar requisições de serviços e para receber retornos a suas solicitações. A aplicação servidora monitora constantemente a porta portS da máquina hostS aguardando a chegada de solicitações de serviço. Quando alguma solicitação é recebida, a aplicação servidora executa o serviço e utiliza a conexão para enviar o retorno com os resultados do serviço [47].

Uma vez que a conexão entre cliente e servidor tenha sido estabelecida pela criação dos correspondentes *Sockets*, os dados da aplicação podem fluir através dos *streams* (pequena quantidade de dados) a ela associados.

O processo servidor na arquitetura TCP deve estar preparado para responder a solicitações de conexões por parte dos clientes, permanecendo em estado de espera (*listening*) entre solicitações.

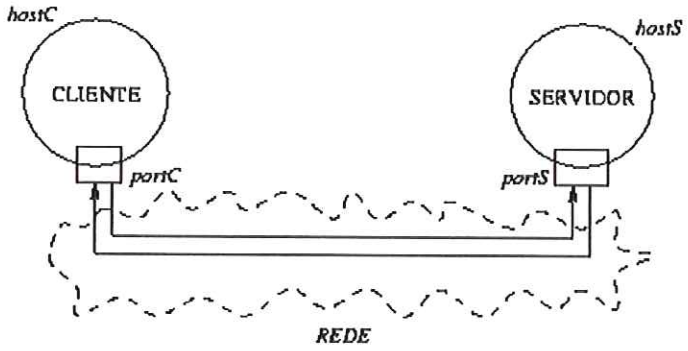

Figura 2 - Comunicação TCP/IP

2.3 - SOFTWARES DE AQUISIÇÃO E ANÁLISE DE DADOS E SINAIS

A aquisição e análises de sinais provenientes de sensores em máquinas de usinagem devem ser feitas através de um *software* de aquisição. Seguem alguns conceitos e produtos mais atuais em relação a este assunto.

2.3.1 - LabVIEW

O LabVIEW (*Laboratory Virtual Instrument, Engineering Workbench*) da National Instruments é um ambiente de desenvolvimento baseado em programação gráfica de linguagem G. Oferece ferramentas gráficas e boa interface com o usuário na criação de VIs (*Virtual Instruments*) [22]. A aquisição dos sinais é feita através de uma placa de aquisição em que os sensores são conectados. As interfaces e cartões de aquisição são controlados pelo *software*. O sinal adquirido pode ser tratado e apresentado de diversas formas, transformado em arquivo do tipo *bitmap* ou arquivado na área de memória de um computador ou ainda em uma base de dados [23].

2.3.2 - Softwares SCADA

Um *software* SCADA (*Supervisory Control and Data Acquisition*) é composto por um conjunto de ferramentas avançadas usadas para o monitoramento

de processos de chão-de-fábrica. Usando um ou mais computadores conectados a uma rede de equipamentos de campo, o *software* SCADA é capaz de fornecer uma imagem instantânea do processo monitorado, fornecendo uma grande quantidade de informações, tais como tempo de ciclo, tempo de máquina parada, ritmo da produção etc., bem como realizar o controle do processo. A partir daí, surgiu a idéia de sala de controle, um ambiente de onde o funcionário tem o controle total do chão-de-fábrica à distância.

O *software* SCADA deve captar os dados operacionais vindos do chão-de-fábrica fornecidos, por exemplo, por um sistema de aquisição de dados, e transformá-los em informações tão logo quanto o evento ocorre no chão-de-fábrica, de modo a permitir análises e respostas em tempo real. Mas, como são fornecidos milhares de dados do chão-de-fábrica por minuto, o acompanhamento atento das mudanças e a correta tomada de decisões dependem de algumas funcionalidades do *software*, tais como alarmes, relatórios e gráficos de tendências [24].

Os alarmes são usados para indicar rapidamente ao usuário uma condição específica, que pode variar desde indicações apenas informativas até indicações emergenciais. Em todos os casos, o *software* SCADA precisa apresentar a indicação em um tempo suficientemente pequeno de modo que o usuário possa analisar o problema e tomar uma decisão o mais rápido possível. O *software* é usado para organizar logicamente os dados e informações esperadas, efetuar cálculos e comparações e criar alarmes que mostrem condições de mudança da situação atual de um equipamento específico no chão-de-fábrica [24].

Os relatórios são usados para fornecer informações do *software* SCADA. As informações fornecidas pelos relatórios variam desde o cálculo do nível mais alto de informações até o mais detalhado nível de informação para a análise de aspectos bem específicos do processo de negócio. Essas informações são, na maioria das vezes, analisadas em conjunto com informações provenientes de bases de dados, possibilitando uma melhor imagem do processo em questão [24].

Os gráficos de tendência são usados para estudar as características operacionais recentes para então entender as mudanças presentes e futuras que serão feitas no ambiente de chão-de-fábrica. Esta informação é usada para avaliar a

resposta do chão-de-fábrica a novas condições operacionais. Os gráficos de tendência permitem ao usuário enxergar o futuro através da visualização do passado [24].

Além das funcionalidades descritas anteriormente, o *software* SCADA possui ainda [25]:

- Aquisição de dados: aquisição de dados das estações remotas de I/O via *drivers* de comunicação;
- Controle: controle automático em malha fechada ou controle via atuação do operador;
- *Displays* de processo: representações gráficas dos equipamentos de campo atualizadas automaticamente para retratar uma situação atual;
- *Standby*: sistemas duplicados em espera com a capacidade de tomar o controle no caso de uma falta do sistema principal;
- Interfaces com outros sistemas: transferência de dados para sistemas corporativos;
- Segurança: controle de acesso por usuário aos vários componentes do sistema;
- Gerenciamento da rede: monitoramento das condições de comunicação da rede;
- Gerenciamento de base de dados: Comunicação com base de dados via ODBC.

2.4 - Programação Orientada a Objeto

O objetivo da Orientação a Objetos é representar, em *software*, objetos que se pressupõe existir no mundo real, utilizando para isso uma característica não conseguida nos modelos tradicionais: a incorporação de 'ações' e 'dados', por meio de um objeto. Além disso, a orientação a objetos está baseada na elaboração de partes independentes, que unidas formam um todo. Se uma dessas 'partes' for danificada, ela poderá ser reparada, sem causar dano ao 'todo'. Essas partes são objetos e o todo é o sistema [51].

Objetos são entidades que encapsulam informações de estado ou dados e têm um conjunto de operações associadas que manipulam esses dados. Um objeto é

caracterizado pelos seus **atributos**, variáveis ou propriedades. Os atributos podem ser fixos quando assumem um único valor, ou variáveis, quando podem mudar de valor. Neste último caso, os atributos são responsáveis pela mudança de estado de um objeto. A mudança de estado de um objeto ocorre pela provocação de um estímulo externo, denominado evento.

Denomina-se **operação** a função atribuída aos objetos. Á implementação de uma operação da-se o nome de **método**. Os métodos são os responsáveis pelas ações do objeto, e os atributos pelo conhecimento do mesmo. O objeto então deve executar seus métodos modificando seus atributos a fim de demonstrar os efeitos de suas ações.

Na figura 3 ilustra-se o conceito de um objeto para uma retificadora.

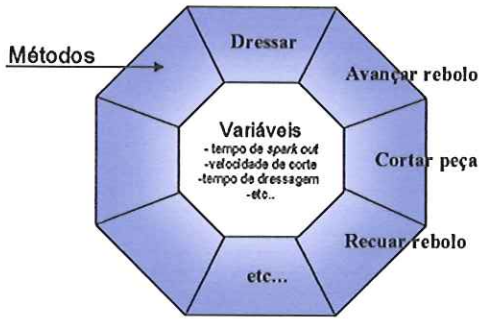

Figura 3 - Modelagem Orientada a Objeto de uma Retificadora

Classe é o nome dado ao tipo do objeto. É uma estrutura de dados que define métodos e variáveis a serem incluídos em um tipo particular de objeto. Os objetos pertencentes a uma classe são chamados de **instâncias**. Estes contém somente valores locais para seus atributos pois caracterizam um objeto único do tipo de uma determinada classe. Por exemplo, em uma classe de retificadoras *centerless* pode-se ter uma instância Microma, e outra ZEMA, identificando cada uma das retificadoras do tipo *Centerless*.

Portanto, um objeto é uma instância de uma classe em particular. Seus métodos e variáveis são definidos na classe, e seus valores são definidos localmente na instância.

Percebe-se então que um conjunto de objetos pode fornecer uma série de efeitos nas interações entre eles. Para indicar essas interações, surgiu o conceito de **mensagens**. Uma mensagem é simplesmente o nome de um objeto, seguido pelo

nome de um método. Caso o método necessite de informações adcionais para determinar suas ações, a mensagem incluirá essas informações como um conjunto de **parâmetros**. Àquele objeto que inicia a mensagem, da-se o nome de **remetente**, e o que recebe é chamado de **recebedor** [49].

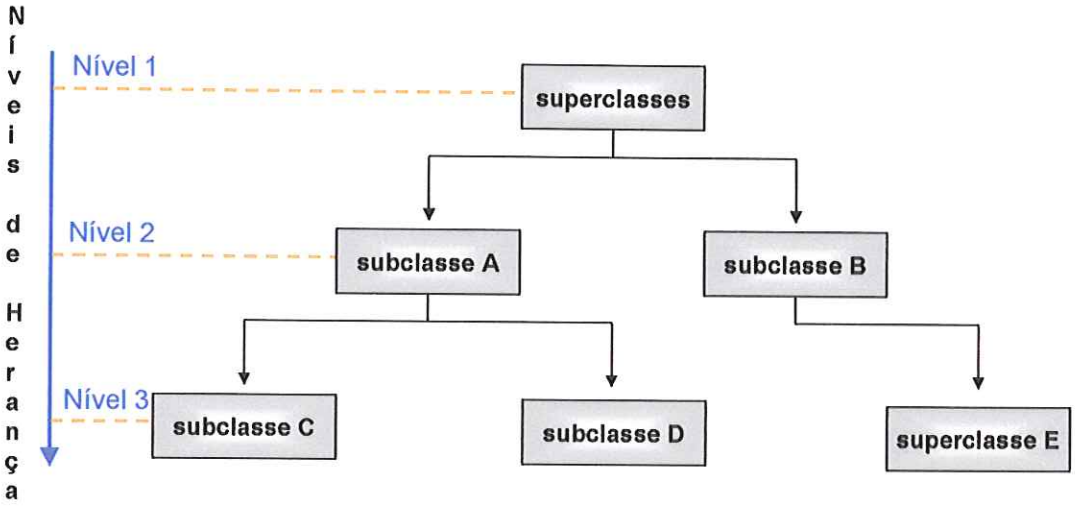

Figura 4 - Níveis de Hierarquia das Classes

Um dos mecanismos mais importantes e úteis da POO é o de **heranças**. Hierarquizar é derivar uma classe em classes especializadas, onde essas últimas herdam características (atributos e métodos) da classe superior. É por uma herança que uma classe pode ser definida como um caso especial de uma classe mais genérica, herdando seus métodos e variáveis. Essas classes herdeiras de outras classes mais genéricas são chamadas de **subclasses** e às classes mais genêricas dá-se o nome de **superclasses.** Conforme é mostrado na figura 4, as classes podem ser aninhadas em qualquer nível de herança, sendo esta automaticamente acumulada aos nívels abaixo.

Um outro mecanismo importante neste tipo de programação é o **encapsulamento**. Encapsular é uma forma de abstrair, ocultar detalhes de uma estrutura complexa, de forma a facilitar a análise de um problema [51]. Esta é uma ferramenta interna do objeto "escondida" do usuário. Uma pessoa por exemplo pode abrir uma janela girando a tranca sem precisar saber o que há dentro dela.

As principais linguagens de programação tais como *Object Pascal* e *C++* são totalmente voltadas para este tipo de programação. Muitos adeptos desta tecnologia

foram atraidos principalmente pelo pouco uso de código que o projeto carrega no programa fonte, ao contrário das linguagens mais antigas como o *Clipper*. O resultado desta "limpeza" no código torna a manutenção do projeto mais simples.

Existem algumas ferramentas de grande interatividade com usuário, que torna a programação de aplicativos mais fácil com o extenso leque de recursos oferecidos para desenvolvimento. Uma dessas ferramentas é denominada Delphi, um ambiente de programação orientada a objeto desenvolvido pela Borland.

O Delphi, na sua versão 6.0, é uma ferramenta RAD (*Rapid Application Development*) para ambiente Windows, sendo perfeita para o desenvolvimento de aplicações que usam bases de dados. Delphi é visual, orientado a objeto e baseado em componentes [21].

Visual: o programador constrói as janelas e diálogos de suas aplicações dinamicamente, minimizando a digitação de código;

Orientado a Objeto: Delphi traz todas as características de uma linguagem orientada a objeto;

Componentes: Delphi fornece dezenas de componentes com os quais o programador pode construir suas aplicações. E, por ser orientado a objeto, permite a criação de novos componentes a partir de componentes antigos.

A linguagem de programação utilizada pelo Delphi é o **Object Pascal.** Esta linguagem Orientada a Objetos possui características de programação não só visuais como também de escrita para os programadores que já conhecem técnicas de estruturas de programação, como o *C*, *Basic*, *Pascal* ou x*BASE* entre outras linguagens. A linguagem força a execução de programas em passos lógicos, possibilitando maior facilidade no desenvolvimento de ambientes *Windows*. Além disso, ela possibilita a criação e reutilização de objetos e bibliotecas dinâmicas (DLL - *Dynamic Link Libraries*) [49].

O Object Pascal abrange todo o conceito da orientação a objetos incluindo encapsulamento e herança. Algumas extensões foram incluídas para facilitar o uso tais como conceitos de propriedades particulares e públicas, tipos de informações em modo *run-time*, manuseamento de exceções, e referências de classes. O resultado de toda esta junção faz com que esta linguagem suporte as facilidades de um baixo nível de programação, tais como:

- Controle e acesso das subclasses do *Windows* (API);
- Passar por cima das mensagens de *loop* do *Windows*;
- Mensagens semelhantes as do *Windows*;
- Código puro da linguagem *Assembler*.

Pelas recursos oferecidos com a Programação Orientada a Objetos, decidiu-se utilizar o Delphi como ferramenta base de implementação do projeto.

2.5 - BASE DE DADOS

Um sistema de base de dados é basicamente um sistema de manutenção de registros por computador, ou seja, é um depósito de um conjunto de arquivos de dados computadorizados. Seu objetivo global é manter as informações e torná-las disponíveis quando solicitadas. Esse tipo de sistema envolve quatro componentes principais: dados, *hardware*, *software* e usuário [52].

Os sistemas de base de dados oferece diversos recursos aos usuários, possibilitando-lhe a realização de várias operações dentre as quais destacam-se:

- Adição de novos (vazios) arquivos à base de dados;
- Inserção de novos dados nos arquivos existentes;
- Recuperação de dados dos arquivos existentes;
- Atualização de dados nos arquivos existentes.

As vantagens da utilização de base de dados para o gerenciamento de informações em relação aos métodos tradicionais, baseados em papéis são evidentes:

- *Compactação*: não há necessidade de arquivos de papéis volumosos;
- *Rapidêz*: a máquina recupera e modifica os dados com maior rapidêz comparado a um ser humano;
- *Menos trabalho braçal*: elemina a maior parte do tedioso trabalho braçal de arquivamento;
- *Fluxo corrente*: disponibiliza informações certas e atualizadas a qualquer momento;

A maioria dos sistemas desenvolvidos nos últimos anos é relacional. Um sistema relacional é aquele, no qual os dados são percebidos pelos usuários através

de tabelas. Os sistemas não-relacionais apresentam os dados através de outras estruturas de dados como árvores e listas [52].

O Sistema de Gerenciamneto de Base de Dados (DBMS) é o *software* que manipula todos os acessos à base de dados. Com a utilização de um DBMS pede-se as informações ao gerenciador de base de dados e elas são devolvidas pelo mesmo. O acesso a informações em sistemas de processamento de dados que não utilizam DBMSs, é feito pelo acesso seqüencial a um ou mais arquivos. Cabe ao desenvolvedor criar mecanismos de recuperação da informação.

O processamento de dados com e sem o uso de DBMSs pode ser comparado a uma compra em uma loja de departamentos e uma compra em uma loja de autopeças, que normalmente funcionam por processos diferentes. No primeiro caso, o cliente dirige-se à loja, procura por todas as seções, encontra o produto desejado e efetua a compra. No segundo, o cliente pede ao balconista o item desejado e este lhe entrega o pedido. No caso da compra em loja de departamentos, o trabalho é todo do cliente, sendo este responsável inclusive pelas especificações necessárias (fazer a escolha certa). Já na loja de autopeças, o balconista assume toda a responsabilidade pela entrega da mercadoria desejada [17].

Com o intuito de melhor representar as informações de uma base de dados e também de garantir a sua integridade referencial, criou-se um modelo chamado Modelo de Entidades e Relacionamentos (MER). O MER foi introduzido por Chen e serve para a representação das estruturas de informações [53]. Este modelo é composto por 3 elementos, mostrados na figura 5:

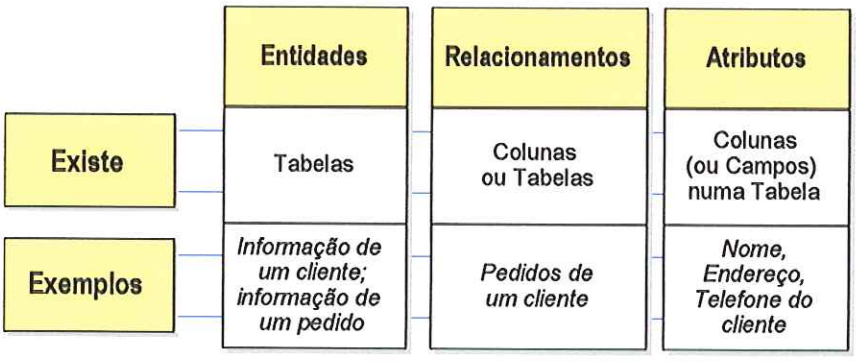

Figura 5 - Elementos do Modelo de Entidades e Relacionamentos

Uma entidade é uma representação abstrata de um "objeto" do mundo real. Assim, uma entidade pode ser a representação de um ser, de um fato, de uma coisa,

de um organismo social, etc [53]. Elas são representadas por tabelas. Exemplos: tabela de clientes; tabela de pedidos de clientes.

As entidades podem ser relacionadas entre si pelos relacionamentos. Por exemplo: relacionamento entre a entidade de clientes e a entidade de pedidos ("clientes fazem pedidos").

Ao conjunto de valores de uma entidade, da-se o nome de atributos. Atributos são as características das entidades e são representados pelas colunas das tabelas. Como na figura 6, por exemplo: nome, endereço do cliente.

Clientes				
identificador	*nome*	*endereço*	*telefone*	.....
1001	João	.....	5554444	~~~
1002	Alberto	.....	4687999	~~~
1003	Franciso	.....	NULL	~~~
1004	Maria	.....	5678900	
1005	Sônia	.....	0988855	~~~
1006	Roberto	.....	NULL	~~~

Figura 6 - Atributos de um Modelo de Entidades e Relacionamentos

Uma das colunas de uma tabela é uma *primary key* (chave primária). Isso indica para o gerenciador de base de dados que uma coluna (ou um conjunto de colunas) deve ter um valor único para identificar a linha inteira. O gerenciador faz então o controle para que não entrem duas linhas com o mesmo valor na coluna que é *primary key*. Esse campo quando pertence também a outra tabela é chamado de chave estrangeira (*foreign key*). Isso garante o que é denominado de integridade referencial, ou seja, não pode haver inconsistência nas linhas que estão associadas nas tabelas.

Na figura 7 é mostrado o relacionamento entre tabelas utilizando-se chaves primárias (*PK*) e estrangeiras (*FK*).

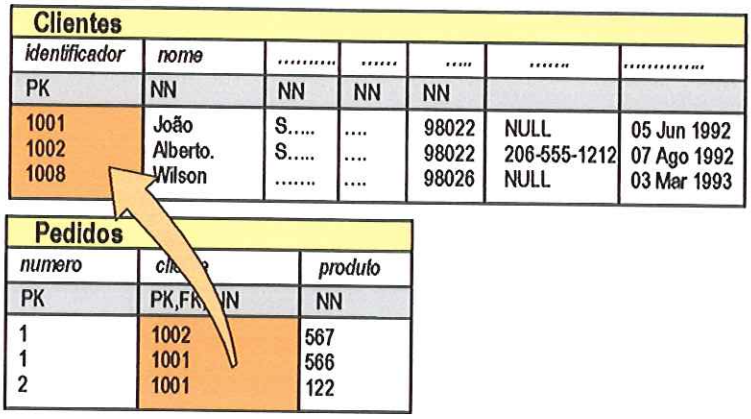

Figura 7 - Relacionamento entre Tabelas

Pedidos se relacionam aos Clientes, através do campo cliente da tabela de pedidos. Com a chave estrangeira da tabela pedidos, por exemplo, o gerenciador não permite que clientes que tenham pedidos sejam removidos da tabela clientes, nem que pedidos sejam realizados por clientes inexistentes.

2.5.1 - A Linguagem SQL (Strutured Query Language)

A linguagem SQL foi desenvolvida no início dos anos 70 pelo Dr. Edgar Codd, na época um programador da IBM. Com o tempo e com a proliferação dos DBMSs relacionais, SQL tornou-se o padrão para a manipulação de base de dados [49].

Como no exemplo do pedido de compra para o funcionário da loja de autopeças, cada comando SQL é um pedido de busca ou alteração de dados para o gerenciador da base de dados. Quem vai executar propriamente o comando é o gerenciador.

2.5.2 - Escolha da Base de Dados

Na escolha de uma base de dados, alguns fatores devem ser cuidadosamente considerados [9]:

- Custo;
- Recursos de base de dados;
- Suporte de *Hardware*;
- Quantidade de Dados;

- Carga de usuário;
- Suporte de rede;
- Interfaces disponíveis;
- Suporte de terceiros;
- Nível de administração do servidor;
- Manuais de usuário.

A potencialidade de uma base de dados está intimamente relacionada com o poder da plataforma junto a ela usada. As diversas plataformas Unix são indicadas para rodar base de dados que utilizam um conjunto completo de recursos. Acessos multiusuários podem ter plataformas diferentes em relação ao cliente-servidor, através da existência de um *software*-cliente [9];

2.6 - Acesso à Base de Dados via Web

A criação de páginas Web dinâmicas com acesso a base de dados vem se tornando cada vez mais importante, tanto na Internet quanto nas Intranets corporativas. Com o avanço dos *browsers* e o amadurecimento dos padrões da Internet, pode-se criar soluções sofisticadas, completamente baseadas em tecnologias da Web.

Um dos métodos mais utilizados para permitir a integração entre um servidor Web e outros programas executados no sistema é a CGI (*Commom Gateway Interface*). Trata de um pequeno programa que serve para dinamizar certas atividades na Home Page, como por exemplo ativar um formulário de cadastro, ou colher informações de visitantes de um site. Esta interface foi desenvolvida originalmente para servidores Web da plataforma UNIX (como o CERN e NCSA).

A programação CGI representa a criação de programas que recebem e traduzem dados enviados via Internet para seu servidor WWW. Este tipo de programação envolve o uso desses dados traduzidos e também a compreensão de como se pode enviar cabeçalhos válidos de resposta HTTP e tags HTML ao cliente WWW (*browser*) [54].

Com uma frequência bastante grande, os programas precisam atuar como interfaces ou elementos de transição entre outras aplicações maiores. Por isso são chamados de *gateways*.

Uma das aplicações mais utilizadas com GCI é o acesso à base de dados via Internet. Existem muitas CGIs gratuitas na rede. Os *scripts* mais complexos são vendidos em sites especializados.

Um programa CGI é de fácil identificação pois possue a extensão ".cgi" ou ".pl". Uma linguagem muito utilizada para a programação CGI é a linguagem Perl. O funcionamento de uma CGI é dividido nas seguintes etapas:

1 - Chamada de programa CGI através de *link*, botão ou pelo carregamento do programa enquanto o *browser* carrega também o documento HTML;

2 - O *browser* pede permissão ao servidor web para executar o programa CGI;

3 - O servidor HTTP verifica as configurações e acessa arquivos;

4 - O servidor verifica também se o programa CGI existe;

5 - Faz-se a execução do programa;

6 - A saída do programa é retornada ao *browser*;

7 - O *browser* apresenta a saída da CGI em formato HTML.

As informações da CGI são passadas junto à tags HTML que são interpretadas pelo *browser* como um documento qualquer. O servidor web deve então ser configurado para permitir acesso à CGI pois somente os programas de usuários autorizados pelo servidor são executados. Na figura 8 apresenta-se o esquema genérico de uma CGI [9].

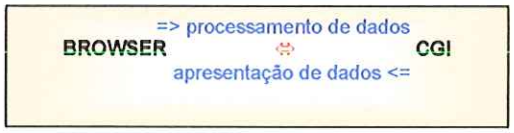

Figura 8 - Esquema Genérico de uma CGI

Na volta de informações de um programa CGI para um *browser*, a CGI deve retornar um cabeçalho que diz ao *browser* como apresentar a saída do programa.

As informações podem ser texto puro ou HTML. O funcionamento de uma CGI é mostrado na figura 9.

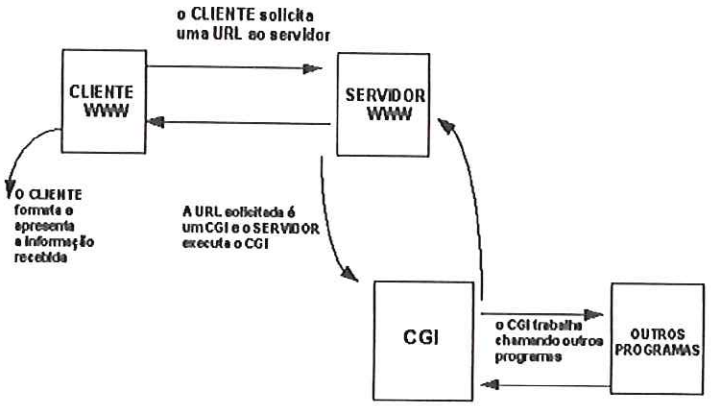

Figura 9 - Funcionamento de uma CGI

Uma outra maneira de acessar base de dados pela Internet é através da implementação de páginas dinâmicas [18]. O PHP é uma linguagem que permite criar sites Web dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL (endereço Web) e *links*. A diferença de PHP em relação a linguagens semelhantes a *Javascript* é que o código PHP é executado no servidor, sendo enviado para o cliente apenas código HTML puro. Desta maneira é possível interagir com bases de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial [19].

O que diferencia PHP de um *script* CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o *script* CGI gere todo o código HTML, ou leia de um outro arquivo.

Basicamente, qualquer coisa que possa ser feita por algum programa CGI pode ser feita também com PHP, como coletar dados de um formulário a uma base de dados, gerar páginas dinamicamente ou enviar e receber *cookies*.

PHP também tem como uma das características mais importantes o suporte a um grande número de bases de dados, como dBase, Interbase, mSQL, MySQL, Oracle, Sybase, PostgreSQL e vários outros. Construir uma página baseada em uma base de dados torna-se uma tarefa extremamente simples com PHP.

Além disso, esta linguagem tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível a abertura de *sockets* (portas de conexões TCP/IP) para interagir com outros protocolos [19].

A Base de Dados mais utilizada em conjunto à linguagem PHP é o MySQL. Ele pode ser encontrado no site *www.mysql.com.* [20].

Para o PHP interagir com uma base de dados SQL, independente de qual base esteja utilizando, existem três comandos básicos que devem ser utilizados: um que faz a conexão com o servidor de base de dados, um que seleciona a base de dados a ser utilizada e um que executa uma "query" SQL. As execuções dessas ações são descritas no Apêndice 3 encontrado ao final deste documento.

2.6.1 - Páginas Dinâmicas na Web Implementadas em Delphi

Utilizando a ferramenta Delphi, a partir de sua versão 3.0 é possível criar aplicativos web através da tecnologia *Web Broker*. Esta tecnologia compreende um conjunto de componentes e classes especializados para aplicativos Web, seguindo os padrões CGI, WinCGI, NSAPI e ISAPI. A vantagem de se utilizar esta tecnologia é que pode-se aproveitar toda a infra-estrutura gráfica do Delphi. Sendo assim, não é necessário o uso de outras linguagens CGI para se implementar páginas dinâmicas.

Os elementos oferecidos pela tecnologia *Web Broker* necessários a uma implementação Web em Delphi são [51]:

WebModule: é um descendente do componente *DataModule*, dotado de recursos adicionais para sua integração com servidores Web, como por exemplo o Apache. Neste componente pode-se adicionar quaisquer componentes não-visuais do Delphi;

Action Items: são interfaces de integração criadas dentro do *WebModules*. O código associado a cada uma delas gera o código HTML retornado pelo servidor.

PageProducers: são componentes que geram páginas HTML a partir de tabelas de base de dados, consultas SQL e arquivos HTML. Os componentes produtores de páginas mais utilizados são: *PageProducer* e *DataSetTable Producer*, os quais são detalhados a seguir.

PageProducer

Este componente é baseado em uma *String* ou arquivo com código HTML denominados templates HTML. As templates HTML incluem tags transparentes que apresentam o seguinte formato:

<#Nome param1=valor1 param2=valor2 param3=valor3 ...>

O "#" no início da tag indica que ela será ignorada pelo *browser* . Durante a geração do HTML pelo componente *PageProducer*, as tags são substituídas por comandos HTML comuns, definidos no evento OnHTML Tag.

As principais propriedades do componente *PageProducer* são descritas a seguir:

HTMLDoc: é uma propriedade do tipo *TString* que contém comandos HTML e tags transparentes que são processadas pelo componente.

HTMLFile: indica um arquivo HTML com tags transparentes que é processado pelo componente.

Content: este método lê o código HTML do HTMLDoc ou HTMLFile, e gera o cógigo HTML final com os tags transparentes substituídos.

O componente *PageProducer* apresenta um único evento denominado OnHTMLTag. Neste evento é definido o código usado para substituir as tags transparentes lidas. O evento é chamado uma vez para cada tag lida.

DataSetTableProducer

Este componente gera tabelas HTML a partir de base de dados ou consultas SQL de maneira muito simples. A tabela HTML gerada é obtida através do método *Content* que retorna uma *String* com o código HTML gerado [51]

As principais propriedades deste componente são:

Columns: é a propriedade que define as características físicas da tabela como a cor da tabela, bordas, número de colunas, entre outros;

DataSet: associa o componente a uma tabela ou consulta definida em um componente DataSet;

Footer: indica o código HTML exibido no corpo da tabela. Este código pode conter tags transparentes para serem processadas por *PageProducers*;

Headers: contém o código HTML exibido antes do corpo da tabela HTML;

MaxRows: representa o número máximo de linhas geradas para a tabela HTML com o intuito de evitar que as tabelas imensas, com milhares de registros sejam geradas automaticamente.

2.6.2 - Aplicações Delphi via TCP/IP

Conforme visto anteriormente, o protocolo TCP/IP é a base da comunicação via Internet. Ele pode ser utilizado para permitir que dois ou mais programas troquem informações entre si. Os programas podem estar localizados na mesma máquina ou em máquinas remotas conectadas à Internet [51].

As vantagens da troca de dados via TCP/IP são:

- facilidade de uso;
- independência de localização do programa (mesma máquina ou máquinas diferentes);
- rapidez e eficiência.

Toda a comunicação via TCP/IP é feita através de "portas" de comunicação. Uma porta é um numero de 16 bits que identifica um destino dentro do servidor. Ou seja, para estabelecer comunicação entre dois computadores precisa-se da porta, além do endereço IP. Algumas portas tem protocolos normalmente associados a elas, como: FTP "21", Telnet "23", SMTP (Envio de correio) "25", POP3 (Recepção de correio) "125", HTTP (Web) "80", NNTP (News) "119". Ao criar um programa para troca de informações via TCP/IP, deve-se escolher uma porta que não esteja sendo usada para se evitar conflitos [47].

O Delphi, a partir das versões 3 e 4 vem com dois componentes destinados a implementar conexões TCP/IP: *TClientSocket* e *TServerSocket*. O *TServerSocket* implementa o lado "servidor" de uma conexão TCP/IP. Para usa-lo, deve-se ajustar a porta de comunicação do serviço e chamar o método *Open*. O componente então fica "escutando" a porta por uma conexão. Depois de efetuada a conexão, o evento *OnClientRead* é disparado quando existirem dados a serem lidos.

Com a implementação de conexões TCP/IP em Delphi é possível de se acessar base de dados remotamente via Internet.

2.7 - Padrões de Integração entre Base de Dados e Softwares

Da necessidade de se integrar base de dados a aplicações diversas, algumas camadas de integração foram criadas.

Uma das camadas comumente utilizadas é a camada ODBC (*Open Database Conectivity*). A ODBC é um protocolo que permite que um programa possa acessar diferentes bases de dados sem conhecer sua estrutura interna. Pode-se dizer que é similar a um SQL interno manejado por um *driver* que se encarrega de traduzir os comandos do programa para alguma coisa que o DBMS possa entender .

A interface ODBC foi criada para tentar resolver problemas que surgem na implementaçaõ de múltiplas APIs para DBMSs. Em um sistema de base de dados distribuído, o desenvolvedor de aplicações precisa conhecer diferentes bibliotecas de aplicações, o que complica o desenvolvimento e a manutenção do código pois a cada nova versão de um DBMS, deve-se alterar as funções implementadas [49].

Com o ODBC, deve-se aprender somente os detalhes de uma única biblioteca de chamada ao DBMS, que é justamente a API ODBC.

A sigla ODBC quer dizer "Conexão Aberta com a Base de Dados" (*Open Database Conectivity*). Desenvolvido pela Microsoft, este protocolo disponibiliza o acesso à Base de Dados através da comunicação destes com aplicativos para Windows. A utilização da ODBC é amarrada ao ambiente WINDOWS, não sendo portável para outros ambientes. Os passos para o uso da ODBC são exemplificados na figura 10.

Nela, um programa fonte escrito em linguagem C++ é compilado. É gerado um código objeto, o qual é linkado a um programa executável. Este programa executável, quando há uma instrução de acesso à Base de Dados, se comunica com a ODBC. A ODBC se comunica com o SQLNET do cliente. Este, por sua vez, se comunica através de protocolos de comunicação via rede com o SQLNET do servidor. O servidor então se comunica com a Base de Dados Oracle.

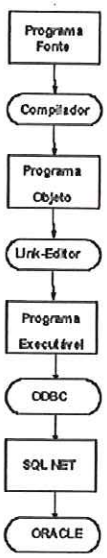

Figura 10 - Passos para Uso do ODBC

Com o ascensão das aplicações de acesso a base de dados via Web, percebeu-se a deficiência desta camada na integração com aplicações Internet. Foi criada então a camada OLE DB que possui *drivers* nativos para acesso a base de dados e é altamente integrado à Internet [45].

A camada ADO (*ActiveX Data Objects*) é um conjunto de classes de alto nível para acesso à camada OLE DB. O Delphi integrava-se com base de dados através de uma camada intermediária de comunicação proprietária denominada BDE. Mesmo com a utilização da camada ODBC, era necessário intermediar a integração utilizando a camada BDE. Com a ascensão da Internet, viu-se a necessidade de se estender o padrão de acesso, pois muitos provedores não suportavam a camada BDE. A partir da versão 5 do Delphi, permitiu-se também o acesso à Base de Dados via ADO, o qual está sendo testado no sistema de diagnóstico à distância para processos de retificação desenvolvido. A integração é bem mais simples que a da camada ODBC, de boa eficiência, qualidade e praticidade [55].

2.8 - SERVIDOR HTTP

O servidor HTTP é projetado para atender a um propósito simples: responder aos pedidos de clientes de documentos disponíveis no site Web que está sendo manipulado pelo *software*, usando o protocolo Hypertext Transfer [9].

O servidor de páginas mais utilizado é o Apache, que nasceu do trabalho realizado para concertar o NCSA HTTPD, um dos servidores Web originais do sistema Linux, corrigindo problemas e incluindo funcionalidades [7].

Como um servidor não comercial, o Apache oferece os melhores recursos para administradores do sistema Linux, e ainda a sua própria API que pode ser usada como uma alternativa à CGI. Além disso, a API pode ser usada para produzir módulos de *plug-in* que servem para propósitos citados abaixo [9]:

- Sistemas de autenticação alternativos, incluindo a autenticação de servidores NIS ou de base de dados LDAP (*Lightweight Directory Access Protocol*);
- Módulos projetados para melhor desempenho da CGI tradicional. Por exemplo, o módulo Perl permite que *scripts* Perl sejam executados em um único processo e compilados apenas na primeira execução. Isso faz com que a CGI baseada em Perl processe quase que com a mesma rapidez que os programas CGI compilados e alguns aplicativos Web baseados em API.

3 - CONCEITOS SOBRE RETIFICAÇÃO E SEU MONITORAMENTO

A usinagem por abrasão é um processo de remoção de material que envolve a interação entre grãos abrasivos e a ferramenta que avança sobre a peça em condições elevadas de velocidade. É considerada a forma mais antiga entre os processos básicos da usinagem.

A remoção de material por abrasão tem duas importantes características. A primeira é que cada aresta de corte é muito pequena e a outra é que muitas das arestas podem cortar simultâneamente. Quando máquinas adequadas são utilizadas, é possível de se obter apertadas tolerâncias de rugosidade na peça [42].

A busca por qualidade em industrias metal-mecânicas está freqüentemente relacionada à retificação, o processo de usinagem por abrasão mais comum no ambiente manufatureiro. Trata-se de um processo de acabamento final da peça que possibilita o alcance de estreitas tolerâncias na usinagem. Neste processo, os grãos abrasivos são fixados junto a ferramenta através de um ligante ou liga. É importante afirmar que o desempenho de uma ferramenta de retifição é muito afetado pelo material do ligante e pelo arranjamento espacial dos grãos abrasivos na ferramenta. A esse arranjamento espacial é dado o nome de estrutura.

Com a escassa gama de profissionais gabaritados a dominar suas peculiares exigências, a retificação finda por muitas vezes deixar que a resolução de problemas e o desempenho do processo fiquem a cargo da experiência e habilidade do operador [1].

Genericamente pode-se dividir os problemas de retificação em:

- a falta de dados consistentes que possibilitem a especificação de uma nova operação;
- a inconsistência de uma operação já especificada que passa a não fornecer as qualidades desejadas;
- a falta de dados e experiência para a aplicação de novas tecnologias.

A inconsistência no desempenho do processo de retificação é causada por vários fatores. Por exemplo, as mudanças nas características topográficas de um

rebolo pelo desgaste das arestas de corte podem provocar grandes variações no resultado final da retificação. Dai a importância da afiação regular da ferramenta.

Dressagem é o nome dado ao processo de afiação do rebolo. Nela é usada uma ferramenta denominada dressador para se usinar a superfície abrasiva do rebolo. Esta operação é também complexa e suas condições de execução podem provocar mudanças nos parâmetros de saída da retificação.

3.1 - O REBOLO

Rebolo é a ferramenta abrasiva utilizada na retificação. Trata-se de um corpo geralmente cilíndrico. Nele, a liga reúne os inúmeros grãos abrasivos, que em contato com a peça realizam a usinagem. Assim, cada grão abrasivo retira uma quantidade minúscula de material da peça [39]. Pode-se desta maneira obter pequenas tolerâncias com a retificação, característica importante em se tratando do acabamento final da peça.

Como já dito, o material da liga aglomerante é um importante fator a ser considerado na seleção de um rebolo. Ele determina a resistência do rebolo, bem como a máxima velocidade de operação da ferramenta. É responsável também pelo comportamento elástico ou deflexão dos grãos abrasivos durante o processo [42]. Os quatro tipos de ligantes mais comuns, atualmente utilizados pelos fabricantes de rebolos, são: resinóides, vitrificados, metálicos e eletrodepositados.[43]

Rebolos com ligante resinóide são fabricados pela mistura de quantidades medidas de resina fenólica ou poliamida e agentes de preenchimento com o peso e tamanho apropriado do abrasivo. Estes rebolos removem material rapidamente mas têm características de agregamento e forma limitados.

Rebolos eletrodepositados tem uma camada única de grãos abrasivos que são ligados a um núcleo metálico, por um banho de níquel eletrodepositado. A matriz de níquel eletrodepositado promove uma excelente retenção do abrasivo.

Rebolos com ligante vitrificado, também conhecidos como rebolos com ligante de cerâmica, têm maior força de agregação do que os resinóides e também permitem ao fabricante variar o volume de saída do cavaco, pela alteração da porosidade e da estrutura do rebolo.

Outra variável a considerar num rebolo é a disposição espacial dos grãos abrasivos sobre a liga. Se estes se posicionam muito juntos um do outro, significa que o rebolo possui mais arestas cortantes por unidade de área. Já estruturas (arranjos dos grãos abrasivos no rebolo) mais abertas apresentam maiores cavidades para cavacos [40].

A topografia ou forma física do rebolo é dada não somente pela sua especificação, mas também pelo tamanho e dureza dos grãos abrasivos, pela disposição espacial dos mesmos e também pelas condições de dressagem [41].

A aplicabilidade prática dos resultados das pesquisas que enfocam o rebolo e suas características Físico-Químicas tem sido muito pequena. A escolha do rebolo ideal para o binômio "material – operação de retificação" é normalmente feita por tentativa. A eficácia desta prática depende em demasia de experiência pessoal e ainda pode gastar quantidades inadequadas de tempo [1].

Para efeito de classificação e com o intuito de facilitar a escolha do rebolo a ser usado para usinar determinada peça, foram criadas 2 classes de grãos abrasivos, cada uma contendo duas subclasses:

Grãos Convencionais:

Al_2O_3 (Óxido de Alumínio)

- Branco
- Marrom
- Rubino ou Vermelho
- Monocristalino
- SeadedGel – SG (Policristalino)
- Targa

SiC (Carbureto de Silício)

- Verde
- Preto

Grãos Superabrasivos

CBN (Nitreto de Boro Cúbico)

Para diversos tipos de geometria e recobrimento.

Diamante

Para diversas categorias de integridade, de recobrimento, geométricas e de grau de purezas.

A escolha do grão abrasivo mais adequado para um determinado material a ser retificado é feita através dos seguintes critérios: afinidade química, material de peça obra e propriedades mecânicas [2].

1 - Afinidade Química

É a afinidade química que diversos materiais tem com os principais abrasivos. Veja a Tabela 1.

Abrasivo	**Estabilidade térmica até: °C**	**Afinidade química com**	**Adequado para**
Óxido de Alumínio	2000	Óxidos, Vidros, Cerâmicas, Rocha	Aços de todas as qualidades
Nitreto de Boro Cúbico (CBN)	1370		
Carbureto de Silício	1300	Materiais que assimilam carbono. Ex.: aço baixo carbono	Aços com carbono satura, ferro fundido, óxidos, (vidro, cerâmica, etc)
Diamante	900		

Tabela 1 - Afinidade Química dos Grãos Abrasivos [1]

2 - Propriedades Mecânicas

Resistência à abrasão, friabilidade, grau de afiação dos abrasivos.

Friabilidade (Capacidade de Quebra)

SiC (verde)
SiC (preto)
A_2O_3 Branco / SG / Targa / Monocristalino
A_2O_3 Vermelho
A_2O_3 Marrom
CBN
Diamante

Grau de afiação do abrasivo

SG / Targa
CBN / Diamante
SiC (verde)
SiC (preto)
A_2O_3 Branco
Monocristalino

Resistência do material à abrasão

Diamante
CBN
SiC (verde)
SiC (preto)
A_2O_3 Vermelho
Monocristalino
A_2O_3 Branco / SG / Targa
A_2O_3 Marrom

3 - Material de peça / obra

Há algumas recomendações práticas para a combinação entre o grão abrasivo da ferramenta e o material da peça a ser usinada. Algumas delas são:

Al_2O_3

Aços macios até ligados, retificação a seco.

CBN

Aços rápidos, aços temperados (até 60 HRC), retificação a alta velocidade.

SiC

Aços inoxidáveis (com fluido), ferro fundido cinzento, materiais não metálicos.

Diamante

Materiais cerâmicos, vidro, fofo cinzento, metais duros e ligas metálicas aplicadas por pulverização.

3.2 - DRESSAGEM

Conhecida também como dressamento, diamantação ou retificação do rebolo, a dressagem é o fenômeno onde o rebolo tem sua face de trabalho usinada com o uso de uma ferramenta (dressador).

Esta operação tem 3 principais finalidades: obtenção da concentricidade da face de trabalho com o eixo de rotação, o perfilamento da face de trabalho para operações de forma e arrancamento dos grãos abrasivos gastos, melhorando assim a agressividade da face de trabalho. Esta última é muito importante, pois a agressividade do rebolo determina a capacidade de remoção de material. Além do mais um rebolo com grãos abrasivos gastos exigem mais potência consumida da máquina de retificação [40].

Nota-se então a grande influência que as condições de dressagem podem provocar no desempenho das operações de retificação. Por se tratar de uma condição inicial do processo, a dressagem do rebolo deve ser bem conhecida e dominada pelo pesquisador de retificação.

A topografia do rebolo tem grande influência na performance do processo de retificação. Vários fatores podem provocar mudanças topográficas durante o tempo de vida do rebolo. Depois de um certo período de uso, o rebolo deve ser dressado introduzindo novos grãos para o processo de corte e reestabelecendo a forma original da ferramenta abrasiva.

A dressagem é necessária para reestabelecer a superfície do rebolo, quando os grãos perdem suas arestas de corte ou quebram, ou mesmo quando cavacos começam a preencher os espaços entre os grãos abrasivos, em estruturas mais abertas. Este processo deve ser realizado quando houver rejeição de uma peça causada principalmente pelo aumento da rugosidade. Determinar o exato momento de se realizar uma dressagem ainda é um problema sem soluções precisas. Geralmente, o operador decide a hora da dressagem do rebolo baseando-se em sua experiência pessoal [35].

A dressagem de um rebolo pode ser feita de diversas maneiras, utilizando vários tipos de dressadores. Estes podem ser divididos de acordo com o princípio de atuação cinemática. Os dois grupos principais conhecidos são: dressadores estáticos

(figura 11) e dressadores rotativos (figura 12). Os dressadores estáticos não se movimentam na direção tangencial do rebolo durante a afiação, enquanto os rotativos, além do movimento de translação, rodam também quando em contato com o rebolo. Algumas operações têm como objetivo apenas o aumento da agressividade da superfície de corte do rebolo (avivamento). O jato abrasivo é um exemplo deste tipo de operação, onde não há correção de forma nem a concentricidade da superfície de corte ativa [1].

Os dressadores estáticos se dividem em dois tipos básicos de ferramentas: as de ponta única e as conglomeradas. As de ponta única são constituídas de um corpo, ou suporte, onde é fixado um diamante de ponta única. Nas conglomeradas diversos tipos de diamantes são posicionados em uma liga metálica e, de acordo com a configuração diamante/liga, tem-se alguns tipos característicos (por exemplo: bastão dressador, fliese, etc.).

Figura 11 - Dressadores Estáticos

As ferramentas rotativas existentes são basicamente o disco dressador e o rolo dressador. O disco ou rebolo dressador é um disco de liga metálica impregnado com diamante e tem funcionamento semelhante a de um rebolo. É uma ferramenta muito utilizada no perfilamento de rebolos para operações de forma. Já o rolo dressador é uma ferramenta mais larga que o disco e trabalha de duas maneiras distintas: por corte ou por esmagamento.

As variáveis de processo mais abordadas nas operações de dressagem são a profundidade de corte a_d e o passo de dressagem s_d. Pesquisas mostram um acréscimo da agressividade com o aumento do passo e da profundidade de dressagem.

Figura 12 - Dressadores Rotativos

3.3 - CICLO DE RETIFICAÇÃO

Denomina-se ciclo de retificação o conjunto de etapas que ocorrem em uma operação desde a colocação da peça na máquina até a saída desta dentro de especificações predeterminadas [1]. O ciclo é subdividido em fases onde em cada uma delas há um tempo para a sua realização. A soma de todos os tempos das fases resulta no tempo de ciclo, parâmetro de extrema importância na produção de grandes e médios lotes.

Um diagrama de ciclo pode representar qualquer tipo de operação de retificação. Tal diagrama possui no eixo das abscissas o tempo e nas ordenadas a posição Z em que estaria o rebolo, se não houvesse deformação, além da posição real da superfície de contato peça/rebolo [23].

Na figura 13 é mostrado um ciclo de retificação de mergulho.

A primeira fase (tempo T1) corresponde à aproximação do rebolo na peça e, portanto não há corte. Na segunda (T2), ocorre o contato rebolo/peça e um aumento da potência consumida pelo motor e da emissão acústica gerada pelo contato. Na terceira (T3), a profundidade de corte é igual ao avanço por volta do rebolo. O tempo T4 corresponde à fase de *Spark out*, ou centelhamento, aonde se chega à dimensão

final da peça. Nas fases 5 e 6, ocorrem o afastamento do rebolo (T5) e a troca de peça com possível dressagem (T6) [1].

Com a otimização do ciclo de retificação é possível de se alcançar qualidades predeterminadas como: qualidade de forma, qualidade dimensional, rugosidade superficial, integridade superficial (sem queima, trincas, mantendo a dureza), etc [39].

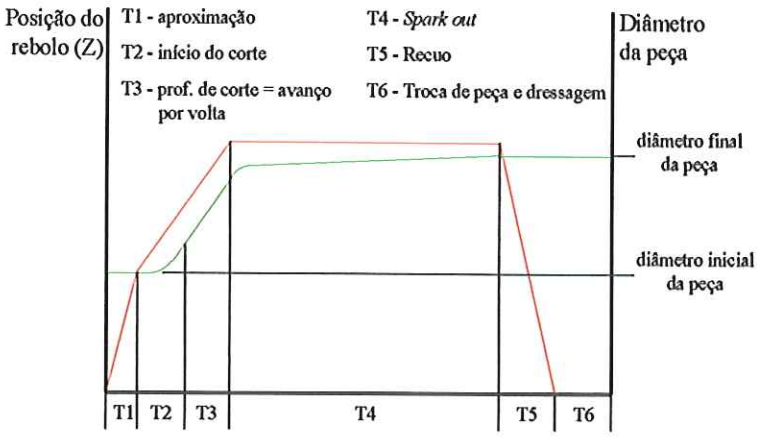

Figura 13 - Ciclo de Retificação de Mergulho [1]

3.4 - MONITORAMENTO DA RETIFICAÇÃO

Com o intuito de monitorar o processo de retificação algumas grandezas do mesmo são analisadas para que delas possa-se extrair resultados benéficos ao controle e a otimização do processo. Algumas destas grandezas e suas importâncias são discutidas a seguir. Tais considerações foram elaboradas com base na literatura [3], a partir de discussões com profissionais da área e também através de visitas a empresas.

Tempo e motivo de parada: Uma das tarefas mais difíceis em uma indústria é o controle preciso dos motivos pelos quais as máquinas param e quanto tempo elas permanecem nesta condição. Para o gerenciamento dos motivos de parada normalmente é utilizado o sistema tradicional nas empresas, onde o operador no fim do turno indica, de forma manual, com base em sua memória, o tempo parado relativo aos vários motivos discriminados por códigos em uma ficha. Tal

procedimento não oferece dados confiáveis para análises posteriores e para o processo de tomada decisão, os quais acabam por traçar caminhos irreais dentro de uma empresa.

Tempo de ciclo e capacidade da linha: É o tempo de uma operação desde a colocação da peça na máquina até a sua saída dentro de especificações de qualidade predeterminadas [1]. Como foi visto, o ciclo é subdividido em fases onde em cada uma delas há um tempo para a sua realização. A soma de todos os tempos de cada fase resulta no tempo de ciclo, grandeza importante na produção de grandes e médios lotes, bem como na otimização do desempenho da máquina. Nesse sentido, é de extrema relevância apontar qual é a capacidade máxima da máquina em processo, ou seja, qual é o seu tempo de ciclo mínimo na produção de determinada ordem.

Tempo de aproximação: O tempo de aproximação, que também compõe o tempo de ciclo é o tempo em que o rebolo se aproxima da peça, não havendo assim corte nesta fase. É uma informação muito importante em se tratando de otimização do ciclo total de retificação. A redução do tempo de aproximação da ferramenta pode significar uma grande redução no ciclo total da peça e, conseqüentemente, redução nos prazos, custos e um aumento na capacidade da máquina [3].

Tempo de corte: Um dos componentes do tempo de ciclo é o tempo em que o rebolo efetivamente está em contato com a peça, ou seja, o tempo em que a peça está sendo usinada. Essa grandeza é uma dos componentes de maior importância dentro do tempo de ciclo por ser o tempo de usinagem em si.

Tempo de *spark out*: *Spark out* ou centelhamento é a etapa do processo de retificação onde o rebolo, ao final da etapa de corte para com o objetivo de eliminar a diferença dimensional causada pela deformação elástica do sistema que ocorre durante a usinagem. O tempo de *spark out* é de grande importância, pois se muito longo, pode significar grande perda de tempo de ciclo e, se muito curto, ocasionar erros de forma na peça.

Tempo de afastamento: É o tempo depois da operação de corte e *spark out* onde o rebolo se afasta até a posição inicial. É aí que possivelmente inicia-se a troca de peça. Por também se tratar de um tempo improdutivo, o tempo de afastamento também é apontado como ponto forte na otimização do ciclo.

Sobremental da peça: É a quantidade de material retirada da peça numa operação de retificação. Analisando esta variável é possível comparar o sobremetal retirado da peça (real) com o desejado e avaliar discrepâncias do processo.

Peças por dressagem: É a quantidade de peças produzidas entre duas dressagens. Esse parâmetro serve para se determinar o tempo de vida de um rebolo e a capacidade de produção da máquina.

Energia: É a integral da curva de potência consumida pela máquina num determinado ciclo. Ela auxilia na análise do rebolo em relação a sua capacidade de corte em relação à velocidade de avanço.

Pico de potência: É o ponto mais alto atingido na potência consumida pela máquina. Este indicador determina os picos de trabalho exigidos da máquina os quais podem indicar falhas na programação do sistema de avanço.

Produção: É a contagem total de peças produzidas por uma determinada máquina. Ela serve de parâmetro para calcular, o ritmo e a taxa de produção real da máquina em questão.

Refugo: É a quantidade de peças produzidas fora das especificações. Um refugo muito alto pode significar um *setup* inadequado das condições de processo da máquina.

Velocidade do Rebolo: É a velocidade periférica do rebolo. Com ela, é possível avaliar o desempenho da máquina durante o processo. Em algumas situações o rebolo perde muita velocidade durante o corte devido à falta de potência no sistema. Isso pode levar a um desgaste prematuro no rebolo.

Tempo de dressagem: Como visto anteriormente, o passo e a profundidade de dressagem influem diretamente na qualidade da peça. Alem disso, o tempo gasto no processo de dressagem é de grande relevância para análises em relação ao tempo de produção total da máquina. Adicionalmente ainda seria possível inferir sobre a qualidade da peça usinada pelo rebolo em determinada operação de dressagem.

Profundidade de Dressagem: É o quanto um dressador penetra em um rebolo durante o processo de dressagem. Esse parâmetro,como já visto, está intimamente ligado ao desempenho final da retificação.

Número de dressagens: Junto ao item discutido acima, o número de dressagens durante a produção em determinada máquina deve ser avaliado para que o tempo de produção possa ser otimizado com maior exatidão.

Gargalos: São os parâmetros ou grandezas responsáveis pela lentidão de uma produção. Os gargalos podem ser identificados pelos tempos de ciclo de operação, tempo de dressagem ou mesmo tempo de máquina parada, entre outros.

Custo de ferramental: A partir de informações do valor do ferramental (rebolo, dressador e rolo diamantado), dos parâmetros de dressagem (profundidade de dressagem) e das informações obtidas em tempo real (número de peças por dressagem) pode-se calcular com exatidão qual o custo do ferramental por peça.

Tempo produtivo e improdutivo da máquina: É o tempo total em que a máquina permanece produzindo ou sem produzir peças respectivamente. Essa grandeza traça o andamento produtivo da máquina, possibilitando um maior monitoramento e otimização da produção, do processo e do pessoal operador.

3.5 - SENSOREAMENTO

O uso de sensores para a realização do monitoramento das máquinas e processos tem se tornado cada vez mais comum nas indústrias que buscam um aumento da qualidade e produtividade, uma vez que eles fornecem informações mais precisas e confiáveis. Essas informações, tratadas de forma a fornecer subsídios necessários para o diagnóstico preciso dos problemas das máquinas e processos e análises reais da produtividade e status da produção, por exemplo, fazem dos sensores um importante alicerce na construção de um sistema de informação da manufatura. [23].

A utilização de múltiplos sensores (combinação de diferentes sensores) está aumentando conforme o custo do sensoreamento tem se reduzido. Praticamente todas as empresas que oferecem sistemas de monitoramento têm pelo menos dois tipos de sensores no mercado. Eles combinam, por exemplo, medição de potência com informações dos sinais de emissão acústica. Isto pode trazer vantagens, pois aproveita as diferentes características de cada um dos sensores, aumentando a faixa de aplicação e confiabilidade do sistema [26].

Os sensores para o monitoramento do processo devem atingir os seguintes requisitos [27]:

- a medição deve ocorrer o mais próximo possível da máquina;
- não deve haver redução na rigidez estática e dinâmica da ferramenta;
- não deve haver restrição de espaço útil de usinagem e dos parâmetros de corte;
- devem ser livres de desgaste e de manutenção, ser facilmente trocados e possuir um baixo custo;
- devem ser resistentes à sujeira, cavacos e interferências mecânicas, eletromagnéticas e térmicas;
- devem funcionar independentemente da ferramenta e da peça;
- devem adequar-se às características metrológicas;
- devem transmitir o sinal de forma confiável.

Para detectar problemas e otimizar processos de retificação são utilizados variados tipos de sensores, dos quais alguns deles são citados a seguir:

Emissão acústica: são ondas de tensão produzidas por movimento súbito em materiais. As fontes clássicas das emissões acústicas são os processos de deformação tais como o crescimento de uma trinca e a deformação plástica. O movimento súbito na fonte produz uma onda de tensão, o qual irradia para dentro da estrutura e excita um transdutor piezelétrico sensitivo. Quando a tensão no material é elevada, várias destas emissões são geradas. Os sinais de um ou mais sensores são amplificados e medidos para produzir dados para exibição e interpretação [28]. O princípio físico desta técnica se resume na utilização do sinal acústico gerado durante o processo de usinagem que é propagado pelo material. A variação da intensidade de energia deste sinal é captada por um sensor piezelétrico e posteriormente correlacionada com o processo em questão. A emissão acústica, por captar sinais de alta freqüência e livre dos ruídos advindos de fontes que não interessam especificamente ao processo, é um método bastante sensível e, portanto, adequado ao monitoramento do processo de retificação, que possui baixas taxas de remoção de cavaco. A emissão acústica precisa de um meio físico de propagação, sendo que na retificação, os mais utilizados são os componentes metálicos das máquinas, que estão em contato com a região de

corte, como o contra ponto, o mancal, o flange ou o eixo do rebolo (o sensor pode ser fixado nestes locais). O sensor pode ser fixado ainda no tubo do fluido de corte, utilizando este como meio físico para propagação do sinal.

Uma grande vantagem da tecnologia da emissão acústica utilizada para o monitoramento de máquinas e processos de usinagem é o fato de que a larga faixa de freqüência do sinal não é afetada pela baixa freqüência de vibração da máquina. Além disso, o fato de os sinais de EA serem gerados diretamente na zona de deformação do processo de corte, fornece, no caso do processo de retificação, informações seguras para poder investigar os processos de *spark out*, desgaste do rebolo e dressagem [29].

Os sensores de EA são de fácil instalação, tanto em máquinas novas quanto em máquinas com vários anos de uso. Por serem pouco intrusivos, estão sendo usados com muita freqüência em máquinas de usinagem, principalmente para a detecção de quebra da ferramenta. A análise do sinal de EA em conjunto com aquele vindo de outro sensor, como o de potência, aumenta a confiabilidade da informação a respeito da máquina ou do processo, como por exemplo, a detecção de quebra da ferramenta [30], e permite uma tomada de decisão ou uma ação de controle mais rápida e confiável.

Potência (corrente): um sensor de potência mede a potência consumida pelo motor da máquina. Sua medição baseia-se na medição da corrente e da tensão do motor. A corrente consumida pelo motor é medida usando-se sensores de efeito Hall. O efeito é baseado na interação entre o movimento de condutores elétricos e um campo magnético externo. Nos metais, estes condutores são os elétrons. Quando um elétron se move através de um campo magnético, sobre o mesmo age uma força lateral que empurra o elétron para o lado, tornando-o mais negativo que o lado oposto. Assim, o campo magnético e a corrente elétrica produzem a chamada diferença de potencial Hall transversal, cujo sinal e amplitude dependem da intensidade do campo magnético e da corrente elétrica.

Os sensores de efeito Hall constituem-se numa forma de medição indireta das forças de corte através da medição da potência consumida pelo motor. A vantagem desta medição indireta das forças de corte reside no posicionamento do sensor no

motor da máquina, ou seja, isolado do ambiente industrial que, via de regra, é bastante agressivo.

Por ser de fácil instalação, os sensores de potência são usados em larga escala no monitoramento de máquinas de usinagem [30].

Devido a algumas desvantagens inerentes do processo, esses tipos de sensores são comumente usados para detectar colisão, quebra da ferramenta e desgaste se instalados adequadamente [27].

Vibração: oscilações devido às forças de corte produzem vibrações na estrutura das máquinas, que variam de acordo com o desgaste da ferramenta. Elas podem ser medidas utilizando-se acelerômetros piezelétricos [27]. Eles baseiam-se no efeito piezelétrico dos cristais de quartzo ou de cerâmica para gerarem um sinal elétrico proporcional à aceleração aplicada.

O efeito piezelétrico produz um acúmulo oposto de partículas de carga no cristal, ou seja, quando uma força é aplicada ao cristal de quartzo, o alinhamento dos ions positivos e negativos é alterado, resultando em um acúmulo dos ions de carga em superfícies opostas (positivos de um lado e negativos de outro). Esses ions são então conduzidos por um eletrodo e, em última instância, são condicionados por um transistor microeletrônico [31].

Em um acelerômetro, a tensão nos cristais ocorre como o resultado da imposição de uma força da massa sísmica sobre o cristal (figura 14). Dentro da faixa de freqüência especificada, essa estrutura obedece a primeira Lei de Newton: F = m.a. Por isso, a quantidade total de carga acumulada é proporcional à força aplicada que, por sua vez, é proporcional à aceleração [31].

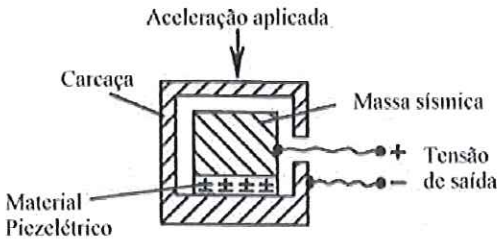

Figura 14 - Acelerômetro Piezelétrico

Deslocamento: Um LVDT, do inglês *Linear Variable Differential Transformer*, é basicamente um transformador com ponto médio (também designado diferencial). A principal diferença reside no fato de o núcleo magnético ser móvel e se encontrar fixo ao objeto cujo deslocamento se pretende medir. Neste sensor, a variação da posição do núcleo altera os coeficientes de indução mútua entre os enrolamentos primário e secundário, tendo como conseqüência a alteração da diferença entre as forças electro-motrizes induzidas nos dois enrolamentos secundários. Este transdutor caracteriza-se por uma relativa linearidade entre a diferença de potencial medida na saída e o deslocamento operado sobre o núcleo magnético. Esta classe de transdutores, com algumas variantes, é utilizada na medição do deslocamento, da velocidade e da aceleração de objetos[32].

Na figura 15 mostra-se os componentes de um LVDT típico. Sua estrutura interna consiste de uma bobina primária centrada entre duas bobinas secundárias espaçadas simetricamente da primeira. Elas são enroladas em um polímero com formato oco em anel, o qual é reforçado com um vidro termicamente estável. As bobinas são encapsuladas contra umidade e então protegidas em um cilindro de aço. Esta montagem em anel é a parte fixa do sensor de posição [33].

O elemento móvel é feito de um material permeável magneticamente chamado de eixo central. Ele se movimenta axialmente dentro do furo da bobina e é acoplado mecanicamente ao objeto que se deseja medir o deslocamento. O furo da bobina é grande o suficiente para assegurar o espaçamento radial entre o furo e o eixo central, sem nenhum contato físico entre este e as bobinas.

Em operação, a bobina primária do LVDT é energizada por uma corrente alternada de amplitude e freqüência conhecidas (excitação primária). O sinal de saída é a tensão diferencial alternada entre as duas bobinas secundárias, a qual varia com a posição axial do eixo central. Geralmente, a tensão de saída AC (baixa) é convertida em uma tensão DC (alta) ou corrente, uma vez que são mais fáceis e convenientes de serem usados.

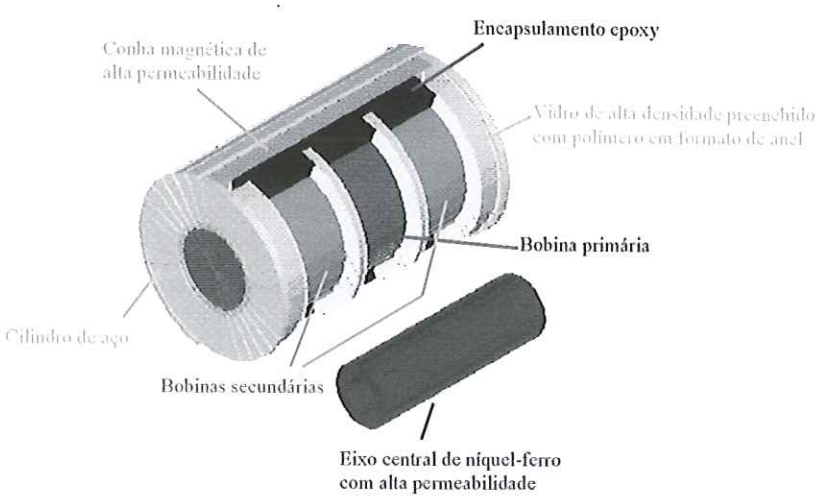

Figura 15 - Componentes de um LVDT [15]

A maioria dos benefícios e funções significativas dos sensores LVDT deriva de seus princípios físicos fundamentais de operação ou dos materiais ou técnicas usadas na sua construção. Entre algumas funcionalidades, podemos citar [33]:

Alta resolução: como o LVDT trabalha baseado no princípio de acoplamento eletromagnético em uma estrutura sem atrito por fricção, ele é capaz de medir mudanças infinitesimais do eixo central. Esta resolução é limitada apenas pelo condicionador de sinal e pelo painel de saída. Além disso, o sensor possui alta repetibilidade.

Vida mecânica ilimitada: por não ter contato entre o eixo central e a estrutura das bobinas, não existe desgaste das partes que compõem o LVDT. Este fato é muito importante em se tratando de aplicações de alta confiabilidade, tais como aeronaves, satélites, veículos espaciais e instalações nucleares, além de sistemas de controle de processos industriais e automação de fábrica.

Sensibilidade unidimensional: o LVDT responde ao movimento do eixo central ao longo do eixo da bobina, mas é geralmente insensível a deslocamentos perpendiculares ou radiais ao eixo central. Assim, ele funciona sem efeitos indesejáveis em aplicações que envolvem desalinhamento ou movimentos flutuantes das partes e em casos onde o eixo central não trabalha em uma linha precisa.

Eixo central e bobinas separáveis: devido à interação entre o eixo central e a bobina ser magnética, a estrutura da bobina pode ser isolada do eixo central através da inserção de um tubo não magnético entre eles. Assim, um fluido pressurizado

pode ser colocado dentro do tubo, dentro do qual o eixo central está livre para mover-se, enquanto que a estrutura das bobinas está sem pressurização. Esta função é muito utilizada em realimentação de posição de cilindros em válvulas hidráulicas.

Grande robustez: os materiais e as técnicas de construção usados na montagem do LVDT resultam em um sensor durável e robusto em várias condições ambientais;

Repetibilidade do zero: a localização do zero intrínseco do LVDT é extremamente estável e com boa repetibilidade, mesmo com um intervalo de variação da temperatura de operação tão grande;

Rápida resposta dinâmica: a falta de fricção na operação permite uma resposta muito rápida às mudanças de posição do eixo central. Ela só é limitada pelos efeitos da inércia da massa do eixo central;

Saída absoluta: o LVDT é um equipamento de saída absoluta, ou seja, no caso de queda de energia, os dados de posição mandados pelo sensor não serão perdidos. Quando o sistema de medição for ligado novamente, o valor de saída do LVDT será o mesmo de antes da queda de energia.

Proximidade: sensores de proximidade indutivos tubular são dispositivos totalmente eletrônicos utilizados para chavear circuitos elétricos ou eletrônicos de baixa potência. A detecção ocorre sem que haja contato físico entre o acionador e o sensor, permitindo um elevado número de atuações. São geralmente aplicados na detecção de aproximação de peças, componentes, elementos de máquinas, etc, em substituição às chaves de fim de curso ou micro-interruptores, pois sua durabilidade, precisão, confiabilidade e velocidade de resposta compatibilizam-se com os atuais circuitos de comando e controle [34].

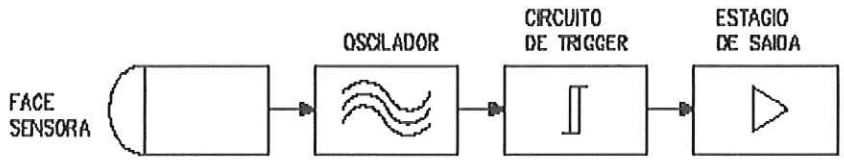

Figura 16 - Princípio de Funcionamento de um Sensor de Proximidade

Na figura 16 ilustra-se o princípio de funcionamento de um sensor de proximidade.Um oscilador de alta freqüência gera um campo eletromagnético que emerge da face sensora do sensor de proximidade indutivo. A introdução de objetos metálicos neste campo resulta na diminuição da amplitude do oscilador, provocando uma mudança de estado em seu circuito de comutação, através de um discriminador Schmidt *Trigger*. Nestas circunstâncias a saída normalmente aberta (NA) torna-se fechada e a saída normalmente fechada (NF) torna-se aberta [34].

As vantagens da utilização de sensores de proximidade são:

- Operam sem contato físico, permitindo um elevado número de atuações;
- Expectativa de vida ilimitada;
- Imunidade à vibrações;
- Alta resistência mecânica e a agentes químicos;
- Totalmente vedados, sendo encapsulados em resina epóxi (classe de proteção elevada);
- Alta freqüência de Comutação.

4 - PROPOSTA DE ARQUITETURA DE UM SISTEMA PARA MONITORAMENTO REMOTO

O primeiro passo dado para o desenvolvimento do Sistema de Diagnóstico à Distância para Processos de Retificação foi fazer uma revisão bibliográfica a qual englobasse os possíveis parâmetros envolvidos na elaboração do sistema.

Posteriormente, algumas arquiteturas foram traçadas, com o intuito de esboçar a solução do sistema. Esta etapa possibilitou a formulação do projeto lógico e o início de sua implementação prática.

Após a revisão bibliográfica, elaborou-se uma arquitetura que realmente atendesse aos requisitos de um sistema de diagnóstico à distância, de onde se possa extrair informações que solucionem problemas e/ou otimizem processos de retificação. A arquitetura proposta é apresentada na figura 17 e será detalhada nas seções a seguir.

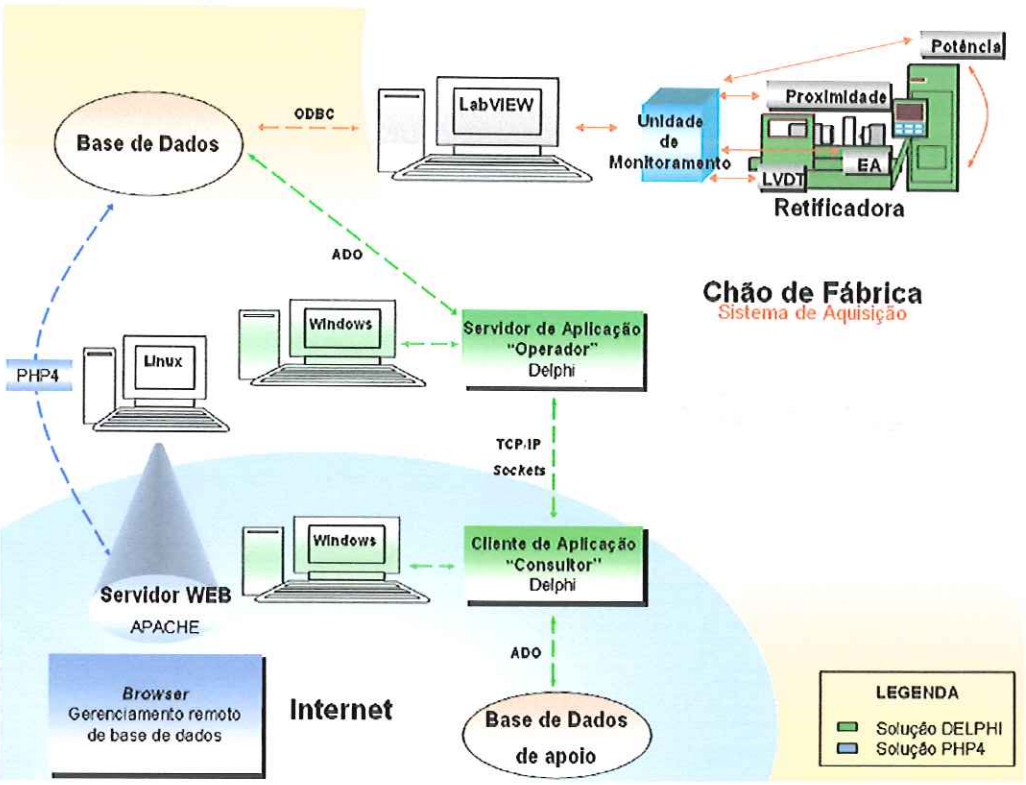

Figura 17 - Arquitetura do Sistema de Diagnóstico à Distância

4.1 - SISTEMA DE AQUISIÇÃO

Nesta parte da arquitetura será detalhado todo o sistema de aquisição de sinais desenvolvido. Incluir-se-ão os *hardwares* usados, as grandezas monitoradas, o sensoreamento envolvido, o tratamento de sinais, a interface de aquisição e análise de sinais, e as configurações do sistema.

O *software* do sistema foi desenvolvido com a ferramenta LabVIEW 5.1. Sua lógica foi baseada em *triggers* preconfigurados que garantem que o sistema seja flexível e se adapte a qualquer situação. São feitas basicamente coletas de 3 eventos:

- Eventos relacionados com cada peça produzida (máximo de potência, tempo de ciclo, tempo de corte, etc.);
- Ciclo completo de produção traçados em gráficos (potência, EA, deslocamento);
- Dados de dressagem.

A Interface principal do sistema de aquisição é mostrada na figura 18.

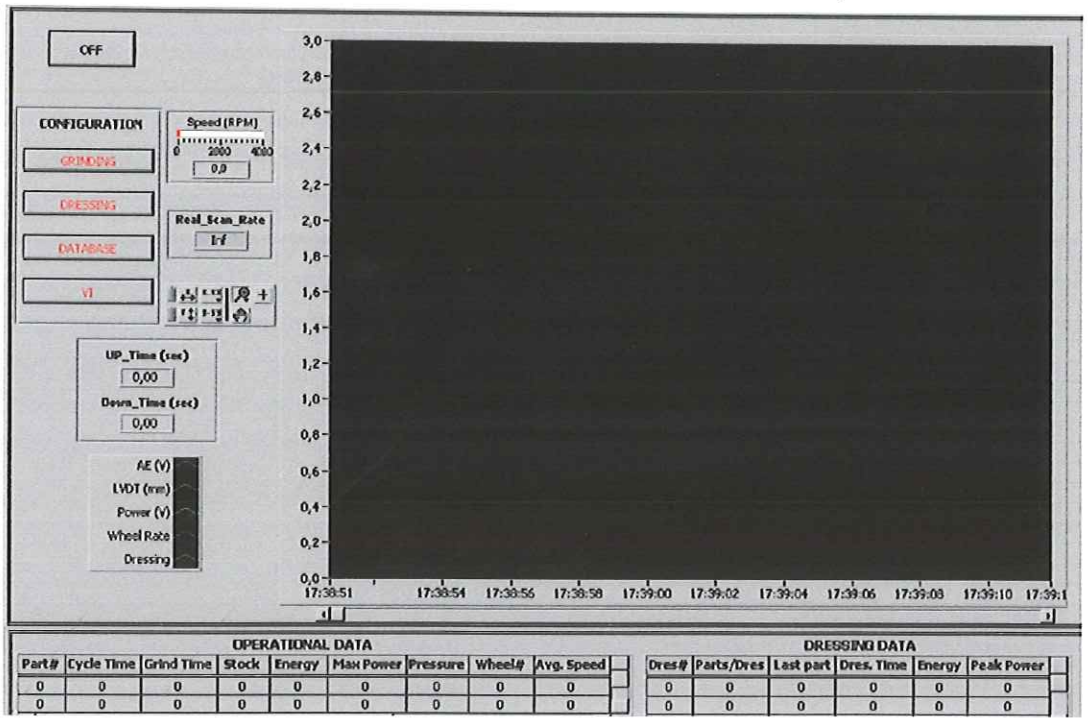

Figura 18 - Interface do Sistema de Aquisição

Grandezas Monitoradas e o Sensoreamento Envolvido

Com base na pesquisa bibliográfica realizada, o sistema de aquisição abordou algumas grandezas envolvidas no processo de retificação. Essas informações são extraídas da máquina com o uso de múltiplos sensores fixados em sua estrutura. As informações podem ser divididas em: dados operacionais, dados de dressagem, dados de produção e dados gerais. Estes últimos são informações dadas pelo usuário através de um formulário de entrada de dados. Os outros três grupos de grandeza e a maneira como cada uma das grandezas são extraídas do processo serão detalhadas a seguir. É importante ressaltar que para cada tipo de máquina e/ou para cada operação, a disposição dos sensores é feita de forma extremamente peculiar. Esta particularidade é satisfeita levando-se em consideração os sensores a serem usados, as informações que se deseja extrair e todo o ambiente que envolve o processo. Para explicar os fatores envolvidos no sistema de aquisição de dados foi usada uma retificadora *centerless* marca Micromа.

Para realizar o monitoramento da retificadora *centerless* Microma foram utilizados os seguintes sensores:

- 2 sensores de emissão acústica (EA);
- 1 sensor de efeito Hall;
- 3 sensores de proximidade;
- 1 sensor de deslocamento (LVDT) com o curso de 1polegada.

O LVDT tem o papel de fazer a aquisição do curso do rebolo. Dependendo do tipo de processo pode ser usado mais que um sensor. O LVDT é fixado de tal maneira que possa acompanhar o deslocamento do rebolo, como é mostrado na figura 19.

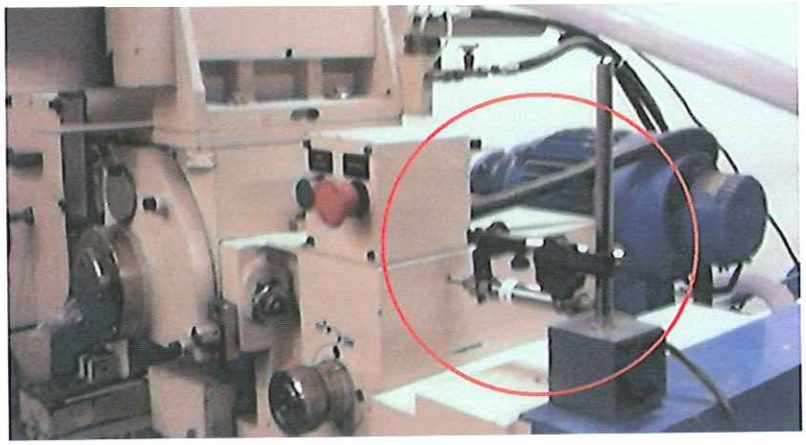

Figura 19 - Fixação do LVDT na Retificadora *Centerless* Microma

Um dos sensores de EA é fixado o mais próximo possível do rebolo para captar a emissão acústica do processo. A fixação deste sensor é observada na figura 20.

O segundo sensor de EA é fixado o mais próximo possível da dressagem. Ele tem a tarefa de fazer a aquisição da emissão acústica do fenômeno, conforme mostra-se na figura 21.

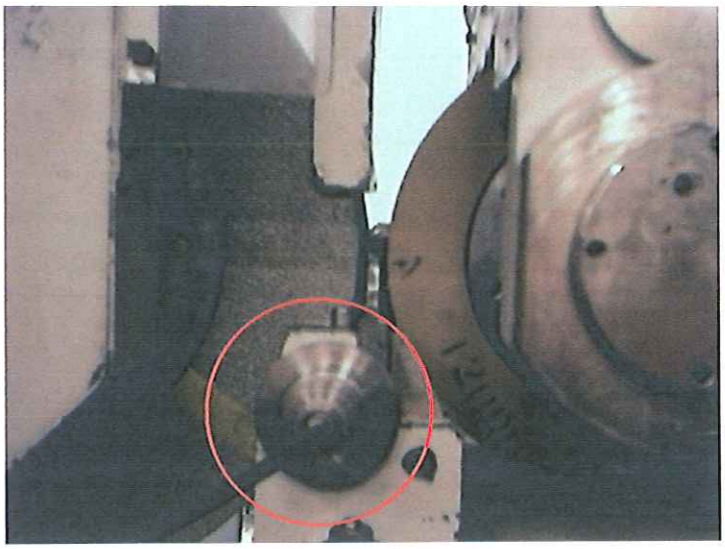

Figura 20 - Fixação do Sensor de EA Próximo ao Rebolo

Para fazer a aquisição do curso da dressagem é usado o sensor de proximidade. Deve-se, no entanto observar como é feita a dressagem. Este processo pode ser feito em passes únicos de sentidos inversos, ou em passes duplos. Se a dressagem termina a onde começou, um sensor de proximidade apenas é suficiente (dressagem em passos duplos). Este então será capaz de pegar o início e o fim do processo, conforme é mostrado na figura 22.

Figura 21 - Fixação do Sensor de EA Próximo ao Fenômeno de Dressagem.

Figura 22 - Fixação do Sensor de Proximidade para Retratar o Curso da Dressagem em Passos Duplos

Caso contrário, se a dressagem é feita em passes únicos e inversos, deve-se utilizar dois sensores de proximidade: um para pegar o início e o outro para o fim do processo. Esta situação é demonstrada na figura 23.

Figura 23 - Fixação do Sensor de Proximidade para Retratar o Curso da Dressagem em Passos Únicos

Um outro sensor de proximidade é fixado próximo ao rebolo, em paralelo ao eixo de rotação do abrasivo, de maneira que possa pegar a rotação do rebolo. Para

isso, um dispositivo foi colocado no rebolo de forma que, toda vez que este der uma volta, o sensor seja capaz de capta-lo. Na figura 24 mostra-se o dispositivo fixado no rebolo. A fixação do sensor de proximidade na máquina é mostrada na figura 25.

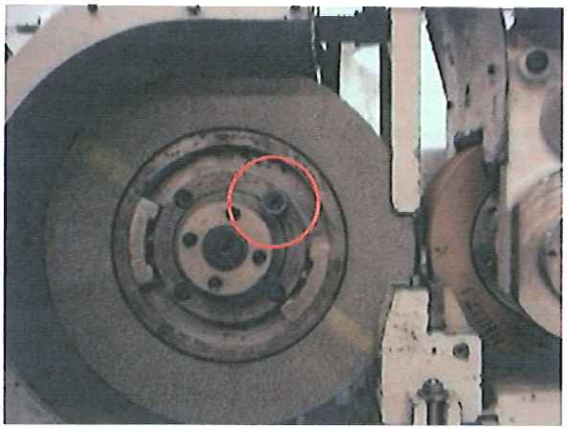

Figura 24 - Dispositivo Fixado no Rebolo para a Captura de sua Rotação

Figura 25 - Sensor de Proximidade para Capturar a Rotação do Rebolo

Um sensor de efeito Hall é envolvido pelo cabo do motor da máquina a fim de fazer a aquisição da corrente consumida. Quanto mais voltas o cabo do motor der no sensor, maior a amplitude do sinal adquirido. No entanto deve-se tomar cuidado, pois o número de voltas do cabo no sensor influencia diretamente na tensão fornecida pelo mesmo. A tensão fornecida pelo sensor foi calibrada em relação à corrente elétrica (cabo do motor).

A instalação do sensor de efeito Hall na máquina é mostrada na figura 26. Fazendo-se a multiplicação da corrente proveniente do sensor pela tensão da máquina tem-se o sinal de potência.

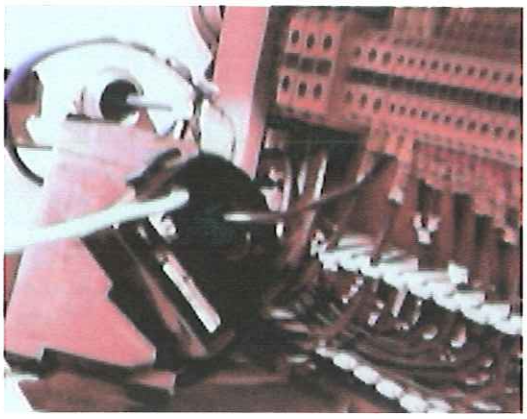

Figura 26 - Sensor de Efeito Hall Fixado na Máquina

Dados de Operação

São os dados do processo de retificação em si. As variáveis do processo monitoradas pelo sistema de aquisição implementado são detalhadas a seguir.

Tempo de Ciclo

O tempo de ciclo é calculado através do LVDT o qual faz a aquisição de todo o curso do rebolo, ou seja, o avanço, o corte, *spark out* e afastamento do rebolo. A presença de peça é garantida pelo sinal de EA e de potência, respectivamente coletados pelos sensores de EA próximo ao rebolo (figura 20) e pelo sensor de efeito Hall instalado no painel da máquina. Na figura 27 retrata-se essa situação em um ciclo normal de retificação.

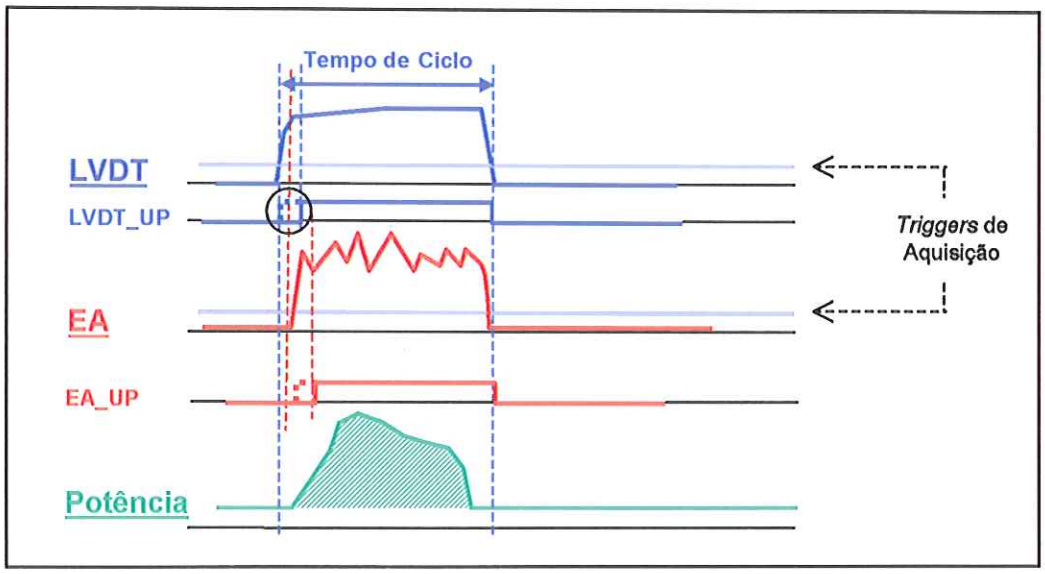

Figura 27 - Aquisição do Tempo de Ciclo

O início de ciclo é tomado quando a subida do sinal de LVDT ultrapassa um limite. Este *trigger* é preconfigurado no sistema de aquisição. Mediante a vulnerabilidade dos sinais, os quais podem sofrer picos inesperados provenientes do processo e do ambiente externo, o sistema espera um tempo mínimo, também ajustável pelo sistema, ao detectar que o sinal em questão ultrapassou o *trigger* de aquisição. Este tempo mínimo garante com precisão o início do ciclo, ou que o sinal ultrapassou o *trigger* de aquisição por um outro motivo (por exemplo, um esbarrão no LVDT). O tempo mínimo é indicado na figura 27 pelo círculo preto.

O fim de ciclo é dado quando o sinal do LVDT ao cair, ultrapassa o mesmo *trigger* de aquisição setado para o sinal. Tomando-se o tempo entre o ponto de início de ciclo e fim de ciclo tem-se o *tempo de ciclo*.

Tempo de Retificação (ou de corte)

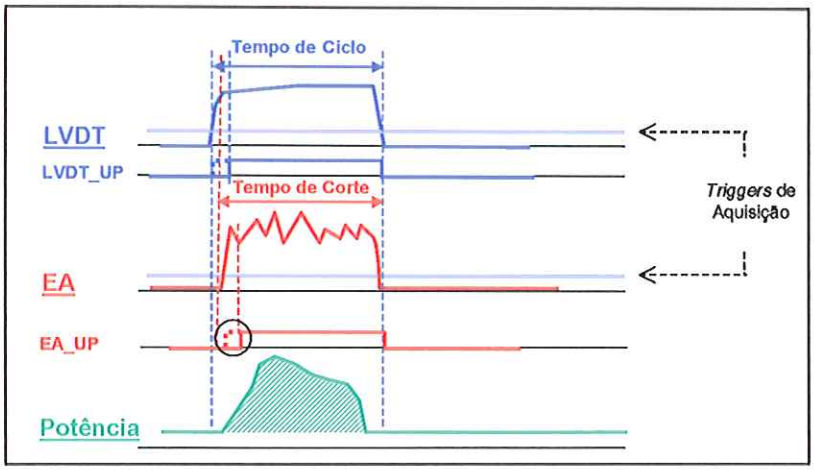

Figura 28 - Aquisição do Tempo de Retificação

Na figura 28 é mostrado o esquema de aquisição do *tempo de retificação* (ou tempo de corte). No sistema de aquisição implementado, o *tempo de corte* é tomado com o auxílio dos sensores de EA (fixado perto do rebolo) e de deslocamento.

A contagem do tempo é iniciada quando o sinal de EA em questão ultrapassa um limite de aquisição preconfigurado. Neste caso também há um tempo mínimo (indicado pelo círculo preto na figura 28) que garante que o corte realmente foi iniciado, ou seja, que o sinal não ultrapassou o *trigger* de aquisição por um outro motivo (por exemplo, ruído, vibração da régua de apoio, colocação da peça).

O fim da contagem do *tempo de corte* é dado pelo mesmo ponto que indica o fim de ciclo, ou seja, quando o sinal do LVDT cai e ultrapassa o mesmo *trigger* de aquisição ajustado para o sinal de deslocamento do rebolo.

Número da Peça

Uma vez tendo os tempos de ciclo e de retificação calculados é possível executar a contagem de peças retificadas. Cada peça usinada recebe então um número de identificação para facilitar análises e consultas posteriores.

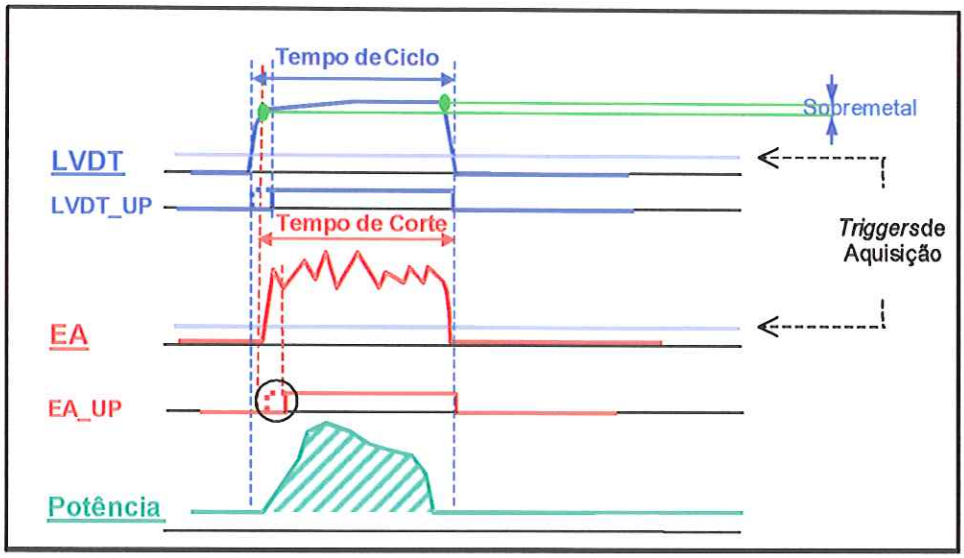

Figura 29 - Aquisição do Sobremetal da Peça

Sobremental da Peça

O *sobremetal* da peça, ou seja, o quanto de material foi retirado da peça na sua retificação é calculado fazendo-se a diferença entre o ponto mais alto atingido pelo deslocamento do rebolo (LVDT) e o ponto onde se iniciou a retificação da peça (o início do *tempo de retificação* dado pelo sensor de EA próximo ao rebolo). Na figura 29 é mostrada essa situação.

Máximo (pico) da potência consumida

Na figura 30 é possível se observar o *pico da potência* consumida pela máquina na retificação. Essa grandeza é dada pela diferença entre o ponto mais alto atingido pela potência e o nível inicial da potência, automaticamente coletado pelo sistema quando a máquina está ligada em estado improdutivo.

Energia consumida pela máquina na retificação

A *energia consumida* é dada fazendo-se a integral da área do gráfico de potência, delimitada pelo nível inicial da potência, automaticamente coletado pelo sistema quando a máquina está ligada e improdutiva.

O sensor de efeito Hall fornece a corrente consumida da máquina. Esta, multiplicada pela tensão e corrigida pelo fator de potência do motor fornece a sua

potência. A potência integrada no tempo fornece a *energia consumida*. A aquisição da *energia consumida* na retificação é mostrada na figura 30.

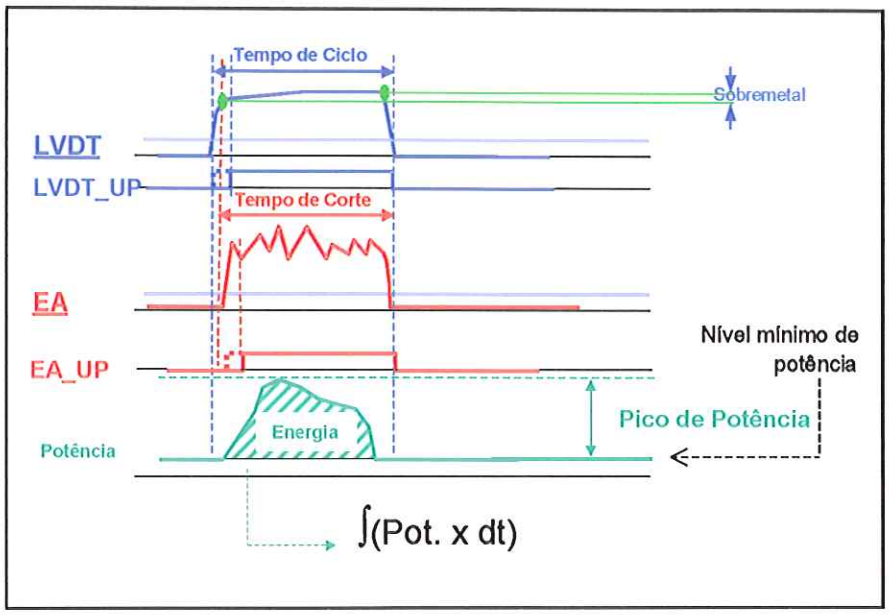

Figura 30 - Aquisição da Energia Consumida e Pico de Potência na Retificação

Número do Rebolo

Uma outra variável abordada pelo sistema é a numeração de rebolos usados durante o monitoramento. Todas as vezes que há troca de rebolo, esta pode ser monitorada através de um formulário. Sendo assim cada rebolo recebe um número de identificação o qual facilitará análises e consultas posteriores.

Média da Velocidade do Rebolo

A velocidade do rebolo é coletada com o auxílio do sensor de proximidade mostrado na figura 25. O sensor passa do estado lógico 0 para 1 toda vez que o dispositivo fixado no rebolo dá uma volta. A cada período de tempo pré-configurado no sistema, verifica-se quantas voltas o rebolo completou naquele espaço de tempo. Tento o número de voltas (rotações) em um período de tempo conhecido é possível calcular a velocidade do rebolo em RPM (rotações por minuto). A cada ciclo é calculada a média aritmética da velocidade do rebolo.

Velocidade Mínima do Rebolo

O sistema de aquisição também armazena a velocidade mínima do rebolo a cada ciclo. Esta grandeza é importante para se dimensionar a queda de velocidade do rebolo durante o ciclo.

Velocidade do Rebolo em Tempo Real

O gráfico da velocidade do rebolo é visualizado em tempo real durante todo o teste monitorado para facilitar análises junto aos outros gráficos traçados pelo sistema.

Dados de Dressagem

São monitoradas as seguintes variáveis do processo de dressagem:

Número da Dressagem

Um contador é incrementado a cada dressagem realizada no rebolo. Este dado é fundamental para análises e consultas posteriores onde será avaliado o desempenho do processo durante o monitoramento.

Peças por Dressagem

Entre as dressagens realizadas, é armazenada a quantidade de peças retificadas. A grandeza *peças por dressagem* é um indicador de desempenho de produtividade e vida do rebolo.

Última peça antes da dressagem

Ao se realizar uma dressagem, o número identificador da peça anteriormente retificada é armazenado indicando a *última peça antes da dressagem.* Este indicador é de extrema importância na avaliação da curva de qualidade das peças usinadas entre dressagens.

Tempo de dressagem

Com o auxílio de sensores de proximidade (um ou dois sensores conforme a necessidade discutida anteriormente) fixados nas extremidades do curso de dressagem é possível extrair o *tempo de dressagem.* Na figura 31 mostra-se os dois casos abordados pelo sistema.

O sensor de proximidade, no caso de uma dressagem com o uso de apenas um sensor, se apresenta a principio com seu estado lógico em 1. No momento em que se inicia a dressagem e a mesa do dressador se afasta desencostando do sensor, este passa a ter o estado lógico em 0. Inicia-se aí a contagem do *tempo de dressagem*. Ao final do curso, a mesa do dressador encosta novamente no sensor, o qual passa novamente para o estado lógico 1 e a contagem do *tempo de dressagem* se encerra.

No segundo caso, onde a dressagem é feita em passos únicos, dois sensores de proximidade são usados. Cada sensor é colocado em uma das extremidades do curso de dressagem, como é mostrado na figura 31. A principio, o estado lógico do sensor 1 se apresenta em 1 e o do sensor 2 em 0, pois este último não está em contato com a mesa do dressador. No momento em que se inicia a dressagem e a mesa do dressador se afasta, desencostando-se do sensor 1, este passa a ter o estado lógico em 0. Inicia-se então a contagem do *tempo de dressagem*. Neste período o sensor 2 também se encontra em 0. Ao final do fenômeno, a mesa do dressador encosta no sensor 2, o qual passa para o estado lógico 1. É aí que a contagem do *tempo de dressagem* termina. O fenômeno, ocorrendo no sentido inverso é monitorado da mesma maneira. Em todos os casos, a dressagem é confirmada pelo sinal de potência e de EA gerados pelo processo.

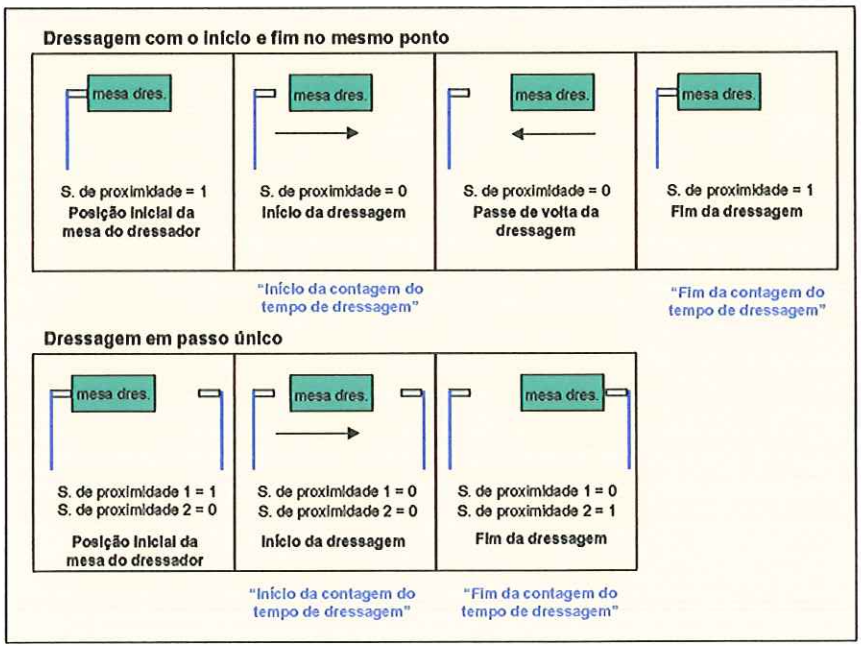

Figura 31 - Aquisição do Tempo de Dressagem

Energia consumida na dressagem

A *energia consumida* pela máquina na dressagem é coletada baseando-se nos mesmos princípios de aquisição dessa grandeza na retificação. É calculada a integral da área do gráfico de potência delimitada pelo seu nível mínimo, o qual é automaticamente coletado pelo sistema quando a máquina está ligada improdutivamente. Na figura 32 exibe-se a aquisição da *energia consumida* pela máquina na dressagem.

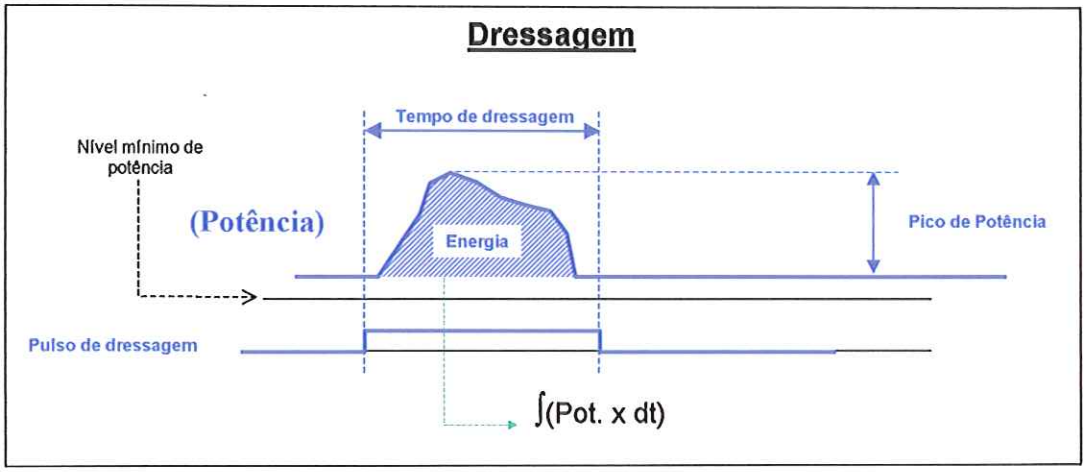

Figura 32 - Energia Consumida pela Máquina na Dressagem

Máximo (Pico) da Potência consumida na dressagem

Como na retificação, o *pico de potência* consumida é dado pela diferença entre o ponto mais alto atingido pela potência e o nível inicial da potência, conforme é mostrado na figura 32.

Dados de produção

Tempo produtivo de máquina (Up Time) e Tempo improdutivo de máquina (Down Time)

O *tempo produtivo de máquina* se inicia no instante em que se inicia o ciclo da primeira peça. Um tempo de tolerância é previamente setado no sistema. Se a máquina ultrapassa esse limite de tempo sem produzir, o *tempo improdutivo de máquina* automaticamente passa a ser contado. Caso a máquina volte a produzir, a

contagem retorna novamente para o *tempo de máquina em produção*. E assim se sucede até que o sistema de aquisição seja desligado.

O *Hardware* Utilizado no Sistema de Aquisição

Para o desenvolvimento do Sistema de Aquisição foram utilizados os seguintes *hardwares*:

- 1 unidade de monitoramento de emissão acústica Sensis;
- 1 Placa de Aquisição PCI – MIO-16E-4 da National Instruments;
- PC Pentium III 450 MHZ com 128 MB RAM;
- 2 sensores de emissão acústica (EA);
- 1 sensor de efeito Hall;
- 3 sensores de proximidade;
- 1 sensor de deslocamento (LVDT) .

Na figura 33 mostra-se como é feita a integração desses *hardwares*.

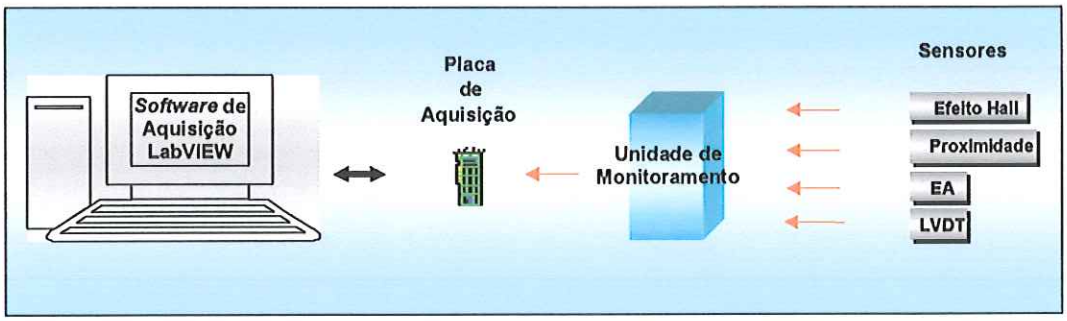

Figura 33 - Arquitetura de *Hardware* do Sistema de Aquisição

Os sensores fixados na máquina são conectados na unidade de monitoramento. Esta é responsável por realizar o tratamento dos sinais dos sensores. Esta unidade de monitoramento é capaz de memorizar diferentes configurações para os seus canais de entrada. Assim é possível tratar o sinal de LVDT de diferente maneira do de EA, em virtude da diferença de comportamento entre os dois sinais. Essa funcionalidade é de extrema importância tendo em vista que o comportamento dos sinais coletados na retificação e na dressagem também difere de um fenômeno para outro. Portanto os sinais devem ser tratados de maneira peculiar em cada processo. A mudança de uma configuração também é feita automaticamente. Um cabo paralelo interliga a unidade de monitoramento à placa de aquisição encaixada

em um dos *slots* PCI, na *Mother Board* do PC. É esta placa da National Instruments que está integrada com o LabVIEW e é configurada via *software* pelo mesmo.

O Tratamento de Sinais

A unidade de monitoramento oferece alguns parâmetros para o tratamento dos sinais. Estes parâmetros são:

***Ganho de sinal*:** é a amplificação do mesmo. Este parâmetro leva em consideração os filtros de ruídos aplicados ao sinal;

***Redução de ruído*:** executa a subtração dos ruídos no sinal de entrada;

***Ganho de entrada*:** aumenta ou diminui o sinal de entrada puro, com todos os ruídos provenientes do ambiente de aquisição;

***Filtro de passa alta*:** este parâmetro estabelece um limite inferior de freqüência para os sinais coletados;

Filtro de passa baixa: é o limite superior de freqüência para os sinais coletados. Como os sinais usados para o monitoramento de processos de retificação apresentam altas freqüências, este parâmetro normalmente não é ajustado.

***Constante de tempo*:** O valor da constante de tempo representa o tempo de integração do sinal em milésimos de segundos. Ele interfere diretamente na sensibilidade do sinal, uma das características mais importantes da EA.

Configurações do Sistema de Aquisição

O Sistema de Aquisição apresenta quatro seções para a sua configuração. São elas:

- Retificação;
- Dressagem;
- Base de Dados;
- VI.

Configuração da Retificação

Essa seção permite que sejam configurados parâmetros de:

EA

- *Trigger* de EA;
- Mínimo tempo de EA (mseg).

LVDT

- *Trigger* de deslocamento;
- Mínimo tempo de deslocamento (mseg);
- Tempo Limite de máquina sem produção (seg).

Configuração da Dressagem

É nessa seção onde se configura o modo de dressagem a ser monitorado:

- *Single Stroque* (2 sensores de proximidade);
- *Double Stroque* (1 sensor de proximidade).

Configuração da Base de Dados

Nesta seção é feita configuração de quantas em quantas peças os gráficos de ciclo (EA, deslocamento, Potência) devem ser armazenados na base de dados. Da mesma maneira, se em toda dressagem o seu gráfico deve ser armazenado, ou de quantas em quantas dressagens deve-se armazena-lo. É possível ainda desabilitar o armazenamento dos gráficos de dressagem, caso se queira monitorar apenas o processo de retificação.

Configuração da VI

A placa de aquisição inserida dentro do PC é configurada nesta seção. Nela também é efetuada a calibração dos sensores utilizados no monitoramento. Configura-se:

- o *Scan Rate* ou velocidade de aquisição;
- a calibração dos sensores de: EA, LVDT, Potência;
- a calibração do dressador.

4.2 - PROBLEMAS DE RETIFICAÇÃO & MEDIDAS DE SINAIS

Consultando as bibliografias mais tradicionais sobre automação da retificação [35], [36], [37] e [38], pôde-se entender a gama de problemas básicos referentes a estes processos e as suas possíveis soluções a partir do sistema proposto.

"Por ser um dos processos de acabamento mais utilizados pelas indústrias para a obtenção de peças metálicas com melhor qualidade superficial e tolerâncias mais apertadas, a retificação tem sido estudada intensivamente, descobrindo-se, nestes estudos, diversos problemas que afetam a operação" [38]. As soluções para tais problemas normalmente são detectadas por inspeções visuais ou por verificações dimensionais da peça. Isso implica na parada do processo para a realização da correção, o que resulta em perda de tempo e de produção. Ao mesmo tempo, análises visuais estão sujeitas a altos índices de discrepância, uma vez que dependem da experiência do operador.

A seguir serão apresentados alguns dos principais problemas que ocorrem em operações de retificação em ambiente industrial, cujo monitoramento e soluções são normalmente difíceis de serem controlados.

Problemas de Dressagem

Quando as peças retificadas começam a apresentar problemas de acabamento, desvios de forma ou dimensional, normalmente realiza-se a dressagem do rebolo, devolvendo-lhe o perfil original e restabelecendo as características de afiação. [23]

Muitas vezes essa dressagem é feita de forma irregular, seja por problemas de *setup* de máquina, seja por erros excessivos no rebolo. De qualquer forma, o processo de dressagem acaba por piorar o acabamento final da peça.

Através do uso de um único sensor de emissão acústica (EA) fixado em pontos da máquina onde possam captar os sinais gerados por esse fenômeno, pode-se detectar falhas na dressagem do rebolo no decorrer do próprio processo [27].

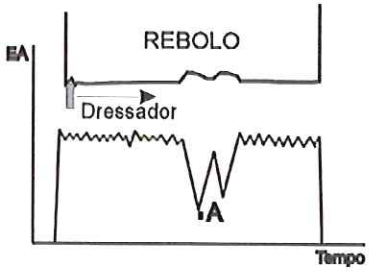

Figura 34 - Problemas de Dressagem

Analisando o gráfico da figura 34, pode-se identificar em A, através do desenho de um sinal de EA, um problema de dressagem. O sinal acústico gerado pelo contato do dressador com o rebolo deve ter amplitude significativa e constante. Se o sinal descer abaixo de um limite mínimo de EA, como observado no ponto A, indica que a dressagem pode não ter corrigido completamente o rebolo. Nesse caso, será necessário efetuar passos adicionais de dressagem até que o sinal de EA fique constante.

Problemas no Ciclo de Trabalho

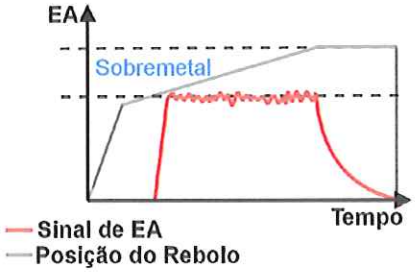

Figura 35 - Ciclo da Retificação

O ciclo mostrado na figura 35 é típico de uma Retificação Cilíndrica de Mergulho. Neste ciclo, o rebolo avança com velocidade máxima até uma distância de segurança entre o rebolo e a peça. A partir de então esta velocidade é reduzida à velocidade de avanço de corte. No contato do rebolo com a peça, observa-se um aumento considerável no sinal de EA, que se mantém constante até que o rebolo pare de avançar e entre em processo de *Spark out*. Durante esse processo, observa-se que o sinal de EA decresce moderadamente até o rebolo desencostar da peça e o sinal desaparecer por completo. Partindo desta última análise, pode-se identificar os seguintes problemas de conseqüências e soluções conhecidas. Salienta-se que os

problemas apresentados a seguir são muito comuns tanto em equipamentos convencionais como nos de comando numérico. Isso ocorre porque boa parte das variáveis escolhidas de forma empírica e seu ajuste preciso depende de uma análise visual dos sinais, o que é proposto no presente trabalho.

a) Problema: Rebolo colide com a peça no início do ciclo com velocidade de aproximação (figura 36);

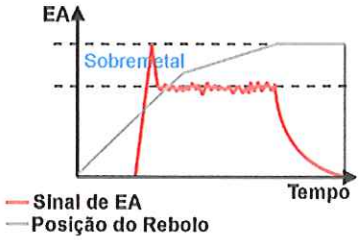

Figura 36 - Sinal de EA – Rebolo Colidindo na Peça com Velocidade de Aproximação

Conseqüência: Deformação geométrica na superfície do rebolo, redução na vida do rebolo, má qualidade na peça.

Solução: Corrigir a posição de comutação de velocidade de aproximação para velocidade de avanço de corte.

b) Problema: Deslocamento de corte excessivo em relação ao valor do sobremetal (figura 37);

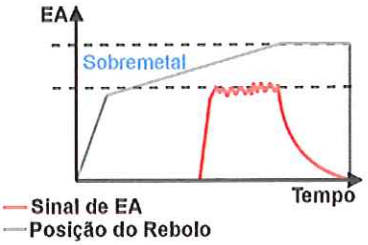

Figura 37 - Deslocamento de Corte Excessivo

Conseqüência: Desperdício de tempo;

Solução: Corrigir a posição de comutação de velocidade de aproximação para velocidade de avanço;

c) Problema: Peça com sobremetal abaixo do mínimo necessário (figura 38);

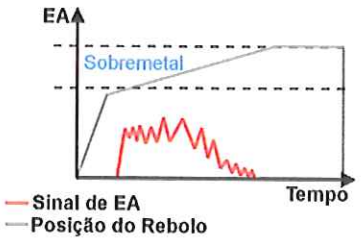

Figura 38 - Sinal de EA – Peça com Sobremetal Abaixo do Mínimo Necessário

Conseqüência: Erro de forma da peça, manchas da operação anterior, erros dimensionais e outros;

Solução: Verificar controle dimensional da peça na operação anterior;

d) Problema: Tempo de *Spark out* inadequado (figura 39);

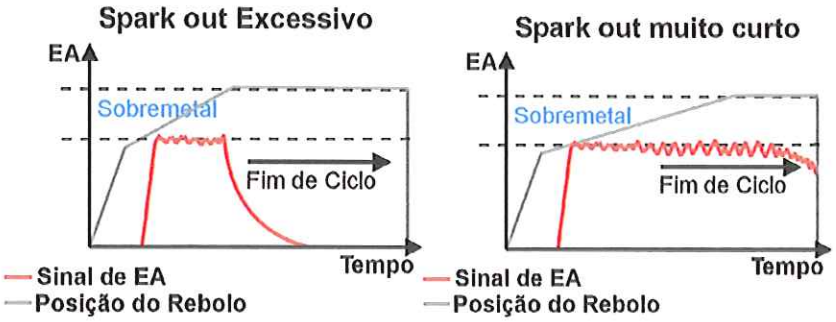

Figura 39 - Sinal de EA – Tempo de *Spark-out* Inadequado

Conseqüência: desperdício de tempo (no caso de tempo de *Spark out* maior que o ideal), erro de forma (em se tratando de tempo de *Spark out* menor que o ideal), variação dimensional;

Solução: corrigir tempo de *Spark out* no processo para o ideal.

Analisando agora os sinais de potência junto aos sinais de vibração no decorrer de vários ciclos, conforme é mostrado na figura 40. Pode-se identificar a perda de capacidade de corte do rebolo (A) ou perda volumétrica do rebolo (B).

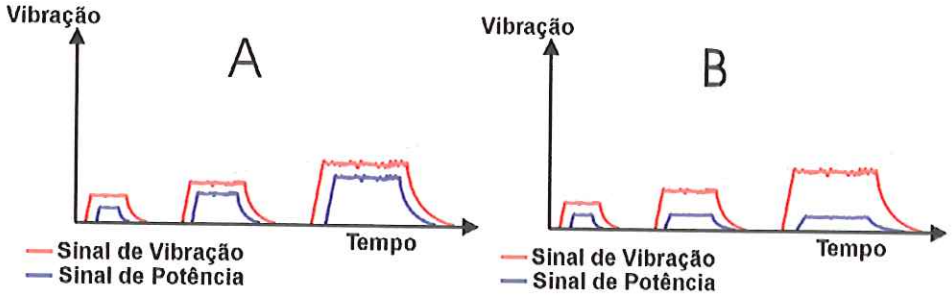

Figura 40 - Sinais de Potência e Vibração – Perda de Capacidade de Corte (comportamento duro) e de Capacidade Volumétrica do Rebolo (comportamento mole).

Na figura 41 pode-se observar um problema característico de máquinas hidráulicas. Nota-se pelo sinal de EA que o rebolo avança com velocidade muito variável (em solavancos). Pode ser problema no sistema hidráulico (como ar no sistema, guias ou fixação da peça).

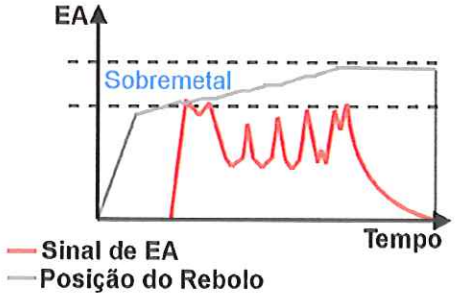

Figura 41 - Rebolo Avançando Não Uniformemente

Além dos problemas descritos acima, pode-se identificar várias outras deficiências no processo de retificação através de análises de sinais de sensores. Por exemplo, pode-se citar as deformações estruturais térmicas da máquina e desgastes do rebolo (EA, temperatura), descentralização do rebolo (EA), entre outros.

4.3 - Detecção Automática de Falhas

Decidiu-se implementar no sistema de aquisição desenvolvido rotinas para detecção automática de falhas no processo de retificação monitorado pela aplicação. As rotinas desenvolvidas são apresentadas a seguir.

Detecção Automática de Sobremetal Inadequado

Um problema muito comum é a variação do sobremetal intermediário do processo (aquele advindo de operações anteriores como o forjamento, o torneamento ou uma retificação em desbaste). Como a qualidade do produto está associada às dimensões finais da peça, não há tanto rigor em relação ao sobremetal intermediário. No entanto, a ocorrência deste problema pode gerar danos, tanto no rebolo como na peça usinada. É importante então detectar em um lote de fabricação a ocorrência do não cumprimento das tolerâncias intermediárias.

O sistema de monitoramento desenvolvido proporciona o cálculo do *sobremetal* da peça, ou seja, o quanto de material foi retirado da peça na sua retificação. Faz-se a diferença entre o ponto mais alto atingido pelo deslocamento do rebolo (medido pelo sensor LVDT) e o ponto onde se iniciou a retificação da peça (o início do *tempo de retificação* dado pelo sensor de EA próximo ao rebolo). Na figura 42 é demonstrada essa situação.

A solução adotada foi a criação de um parâmetro de entrada no sistema que indique o sobremetal intermediário máximo (*Smax*) tolerado. Este parâmetro preestabelecido é inserido no sistema pelo usuário através do teclado do microcomputador e comparado com o sobremetal calculado automaticamente pelo sistema. Caso *Smax* seja menor do que o valor calculado, um alarme é mostrado indicando a ocorrência da falha. Esta ocorrência também é registrada na base de dados, podendo posteriormente ser consultada e avaliada.

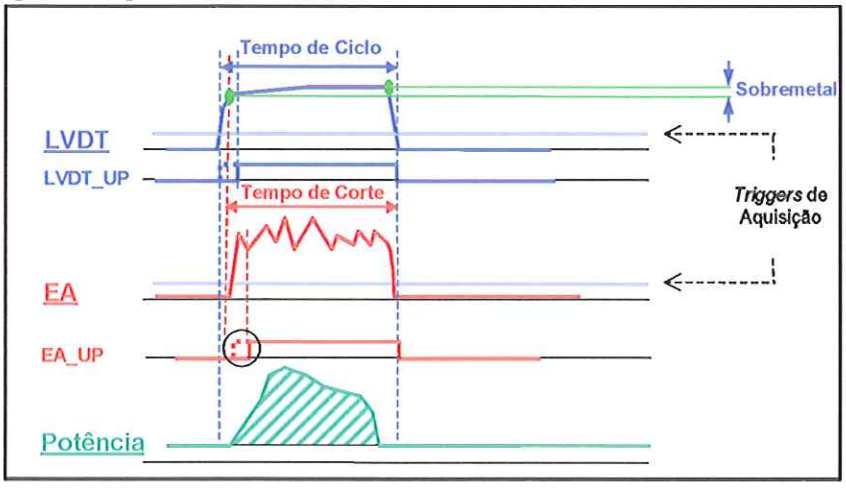

Figura 42 - Cálculo do Sobremetal da Peça

Detecção Automática de Aumentos Excessivos da Potência Consumida pela Máquina

As forças de corte influenciam na qualidade dimensional e geométrica do produto acabado, na rugosidade, no desgaste do rebolo e também no tempo de retificação. Nesse sentido é importante que estas forças sejam previstas, objetivando a otimização e mesmo a automação do processo [1].

Um outro sério e obscuro problema do processo de retificação é quando a potência consumida pela máquina ultrapassa o limite máximo suportado pelo motor da mesma. O processo de retificação é regrado para se ter o consumo de uma dada quantidade de energia por peça. A curva de potência tem, portanto, que ter um formato preestabelecido, ou seja, o pico de potência não deve ultrapaçar o limite máximo suportado pelo motor (potência nominal da máquina). Quando a potência consumida é maior do que a suportada pela máquina, o rebolo perde velocidade e com isso, certa quantidade de grãos abrasivos supersolicitados se solta, deixando a superfície de trabalho saliente. Isso implica em redução do tempo de vida do rebolo e má qualidade das peças usinadas.

Tendo-se o valor da potência nominal é possível então compará-lo com o pico de potência detectado pelo sistema de monitoramento. Nesse sentido, foi implementado no sistema um procedimento para a detecção de aumentos de potência consumida pela máquina não suportado pela mesma. O valor da potência nominal descrito no motor ou manual da máquina é um parâmetro de entrada do sistema, inserido pelo usuário através do teclado do computador. Após o término de cada ciclo, o sistema compara o pico de potência calculado com 80% do valor da potência nominal, inserido pelo usuário. Se o pico de potência for maior do que essa porcentagem da potêncial nominal, o sistema acusa a ocorrência de um problema a qual é registrada na base de dados.

O sistema ainda proporciona um auxílio ao diagnóstico da causa e à solução do problema. Quando é notificada a ocorrência do problema o sistema questiona ao usuário se a afiação do rebolo está inadequada. Caso afirmativo, o problema "afiação inadequada do rebolo" é armazenado na base de dados junto à notificação da ocorrência. Se o usuário responder a esta questão de forma negativa, o sistema pergunta ao usuário se o avanço do rebolo está fora dos padrões (como por exemplo, problemas no sistema hidráulico da máquina, que avança em solavancos). Este tipo

de problema pode ser confirmado avaliando-se o sinal de EA e LVDT no módulo de análise de sinais completos de ciclos, também gravados periodicamente na base de dados). Desta maneira, é possível auxiliar o usuário na resolução de problemas de aumento inadequado de potência consumida pela máquina durante o processo.

A potência consumida também pode ser calculada pelo sistema com o intuíto de se obter um valor teórico da mesma. Este cálculo é baseado na teoria do processo de retificação e é descrito abaixo:

Sabe-se que:

P = Potência consumida (W)

a = Profundidade de penetração do rebolo por volta (mm)

b = Largura de corte (mm)

Fn = Força Normal (N)

Ft = Força Tangencial (N)

Vs = Velocidade periférica do rebolo (m/s)

Vw = Velocidade periférica da peça (m/s)

ρ = Relação entre força tangencial e normal

heq = Espessura de corte equivalente (mm)

Kh = Constante de Hahn (proporcionalidade entre *força* e *heq*) (N/mm)

Λw = Parâmetro de remoção de material (mm^3 / N . s)

Segundo a referência [2],

$P = Ft \cdot Vs$

$\rho = Ft / Fn \quad \Leftrightarrow \quad Ft = Fn \cdot \rho$

$Fn = Kf \cdot heq$

$Kh = Vs / \Lambda w$

$heq = (a \cdot Vw) / Vs$

Substituindo tem-se:

$Ft \cdot Vs = (Vs / \Lambda w) \cdot ((a \cdot Vw) / Vs) \cdot b \cdot \rho$

então

"Expressão Final": $P = (a \cdot Vw \cdot b \cdot \rho) / \Lambda w$

Através da Expressão Final demonstrada, é possível calcular um valor teórico para a potência consumida. Os valores de ρ e Λw podem ser extraídos das tabelas 2 e 3 respectivamente. No sistema, o usuário entra com os valores de cada variável da Expressão Final. A Potência consumida teórica então é calculada. Quando o pico de potência coletado pelo sistema for maior do que 110% da potência teórica, o sistema acusa a ocorrência de um problema a qual é registrada na base de dados. Há também um auxílio ao diagnóstico da causa e à solução da mesma maneira descrita acima na detecçào do problema utilizando a potência nominal

Material da peça	ρ
1045	0,48
GCr15	0,36
W18 Cr 4V	0,27

Tabela 2 - Valores da Relação ρ

Material da peça	Vs (m/s)	Λw
T – 15	30	0,1
M – 2	30	0,1
René 41	30	0,15
Iconel X	30	0,18
M – 4	60	0,12
M – 4	20	0,06
M – 50	55	0,28
M – 50	18	0,12
René 80	60	0,12
8620 Rc 60	60	0,36

Tabela 3 - Valores de Λw

4.4 - BASE DE DADOS

Inicialmente optou-se por utilizar o sistema de base de dados MySQL em plataforma Linux.

O MySQL é um servidor de base de dados famoso por sua rapidez e facilidade de instalação e configuração. É uma base de dados *freeware* em modo texto, que a cada dia vem ganhando maior popularidade entre desenvolvedores, principalmente de aplicações integradas a Web [11]. Muitos desenvolvedores em Microsoft Access, SQLServer, ou mesmo Oracle tem migrado para o MySQL [12]. Um exemplo típico é a NASA que no início de 2001 migrou seus servidores de base de dados, antigamente implementados em Oracle, para o MySQL [13]. Algumas das razões por esta preferência são [12]:

- *software* a custo zero;
- causa menos sobrecarga na rede;
- é altamente otimizável;
- possibilita fácil *back up* das aplicações;
- provê interfaces flexíveis para vários formatos de dados;
- é fácil de aprender e rápido de implementar;
- oferece alto nível de suporte.

Além dessas vantagens, o MySQL pode rodar em ambientes Linux e Windows, oferece ferramentas de desenvolvimento em interface gráfica, e de gerenciamento à distância (via *browser*).

Uma aplicação PHP4 foi utilizada neste projeto na implementação da administração remota da base de dados MySQL. Ela implica em acessar uma base de dados remotamente via páginas dinâmicas, no caso PHP4, através de um *browser*. A solução PHP4 necessita de um servidor HTTP. No sistema desenvolvido foi utilizado o Servidor Web Apache, o qual demonstrou-se rápido ao servir as páginas e também de fácil configuração. A linguagem PHP4 é simples e fácil de se implementar, uma vez que os *scripts* PHP4 são escritos junto ao código HTML.

Através de um *script* PHP4 pode-se fazer toda a implementação, configuração e gerenciamento de uma base de dados MySQL e dos dados nela armazenado. E o

mais importante é que tudo isso pode ser operado via *browser*, ou seja, fica disponibilizado na Internet [19].

Estas são funcionalidades oferecidas pelo *script* MySQL *Web Interface* - Versão 0.6, que pode ser encontrado facilmente na Web. O autor deste *script* é SooMin Kim. A interface configurada para administrar a base do sistema de diagnóstico à distância para processos de retificação é mostrada na figura 43.

Para que o MySQL Web funcione no sistema, é necessário ter o servidor HTTP Apache configurado para reconhecer a extensão .php4. O MySQL *Web Interface* gera um *software* em HTML bem interativo, de fácil manuseio e seguro, onde o acesso ao DBMS MySQL é controlado por senha. Através deste *script* é possível configurar todo o MySQL, criar base de dados, tabelas, checar suas estruturas e avaliar os dados de uma tabela.

O servidor HTTP Apache foi configurado para suportar a linguagem de páginas dinâmicas PHP4 e o sistema de gerenciamento remoto de base de dados foi implementado. O sistema demostrou grande eficiência e aplicabilidade.

Figura 43 - MySQL *Web Interface*

A próxima etapa foi então fazer com que gráficos coletados pelo sistema de aquisição fossem plotados na Internet. A revisão bibliográfica feita apontou a ferramenta Delphi como uma das mais propícias para se implementar este tipo de

funcionalidade. Através dos recursos de seus componentes gráficos, o usuário teria maior interatividade no manuseio e análise dos gráficos plotados.

Pela facilidade de integração tanto do LabVIEW como do Delphi com bases de dados do amibiente Windows, decidiu-se implementar uma base de dados no DBMS Microsoft Access 2000 com o intúito de testar um produto já bem difundido no mercado. O Microsoft Access é um DBSM que acompanha o pacote Microsoft Office. Ele oferece grande iteratividade com o usuário tanto no desenvolvimento como na administração de base de dados. Esta base de dados foi integrada com a aplicação remota desenvolvida em Delphi através da interface de integração ADO. Sua integração com o sistema de Aquisição implementado em LabVIEW foi feita através do protocolo ODBC.

Para aproveitar a ferramenta de gerenciamento remoto da base de dados MySQL em plataforma Linux, integrou-se este sistema com a base de dados Access 2000. Desta maneira, a base de dados Access alimentada pelo sistema de aquisição é espelhada em uma base de dados MySQL, podendo assim ser analisada remotamente.

A integração entre as bases de dados MySQL e Access 2000 foi feita através do protocolo MyODBC. O MyODBC é o protocolo ODBC para servidores de base de dados MySQL produzido pela Data-Consulting na Suécia. O MyODBC permite a conexão de qualquer aplicação com suporte ao ODBC a uma base de dados MySQL, mesmo que em plataforma Linux, seja uma base local ou remota. Neste último caso o MyODBC utiliza do protocolo TCP/IP para acessar servidores remotos. Além do mais a conexão com a base de dados é extremamente segura, uma vez que o MyODBC exige que a aplicação cliente esteja cadastrada como usuário do MySQL, sendo necessários *login* do usuário e o *password* do mesmo para se efetuar a conexão. A camada de integração ADO se demonstrou eficiente na integração do Sistema de Análises Remotas desenvolvido em Delphi com a base de dados Access 2000.

A base de dados relacional foi implementada através da linguagem SQL e a descrição de suas tabelas é detalhada no Apêndice 4. O vínculo entre as tabelas criadas é feito por uma chave primária identificadora do teste (monitoramento) realizado: Test_Code.

4.5 - SISTEMA DE ANÁLISE REMOTA

O desenvolvimento de aplicativos com ricas interfaces gráficas e de fácil programação, a qual orientada a objeto e baseada em componentes foi de grande incentivo para se formular uma solução desenvolvida em Delphi, linguagem criada pela Borland. Esta solução é apresentada na figura 44.

O sistema de consultoria remota para processos de retificação foi desenvolvido com a ferramenta Delphi. Ele fornece ferramentas de acesso e análises de informações (dados, gráficos), necessárias para se realizar o diagnóstico de problemas e possíveis otimizações de processos de retificação, tudo isso remotamente. Este sistema é detalhado a seguir.

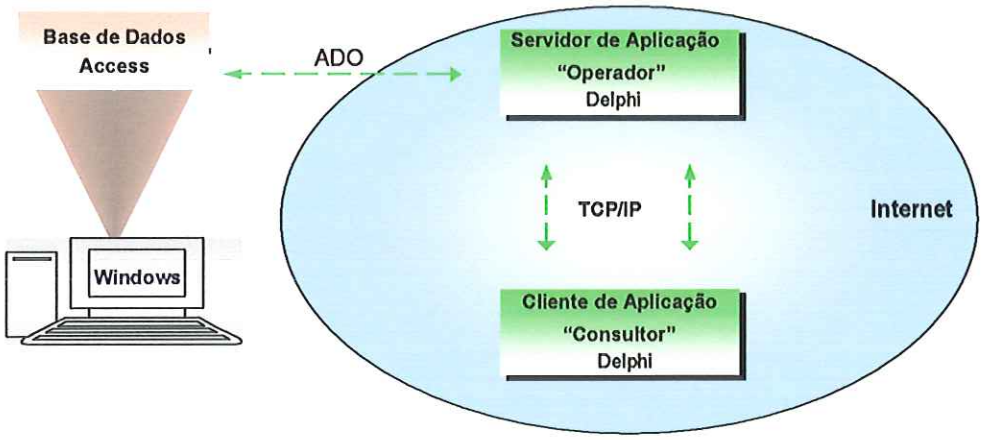

Figura 44 - Sistema de Análise Remota

A integração entre *Aplicação Servidora* e a base de dados é feita via ADO pelas facilidades do ambiente Delphi em lidar com esta camada de integração. A fácil geração de gráficos de análise e a quantidade de recursos gráficos e de programação levaram a um estudo mais detalhado desta solução.

Ela implica no desenvolvimento de uma aplicação servidora integrada a uma base de dados via ADO que se conecte com uma aplicação cliente através de *sockets* TCP/IP. Desta maneira o sistema possibilita o compartilhamento de dados e informações provenientes da base de dados, ou seja, do monitoramento do processo.

Ele é composto por duas aplicações: uma cliente (aplicação do consultor) e outra servidora (aplicação do operador).

A aplicação servidora fica localizada no microcomputador junto ao sistema de aquisição de dados. Ela é responsável por enviar à aplicação do consultor dados e gráficos de testes de monitoramento do processo de retificação realizados. Esta aplicação é mostrada na figura 45.

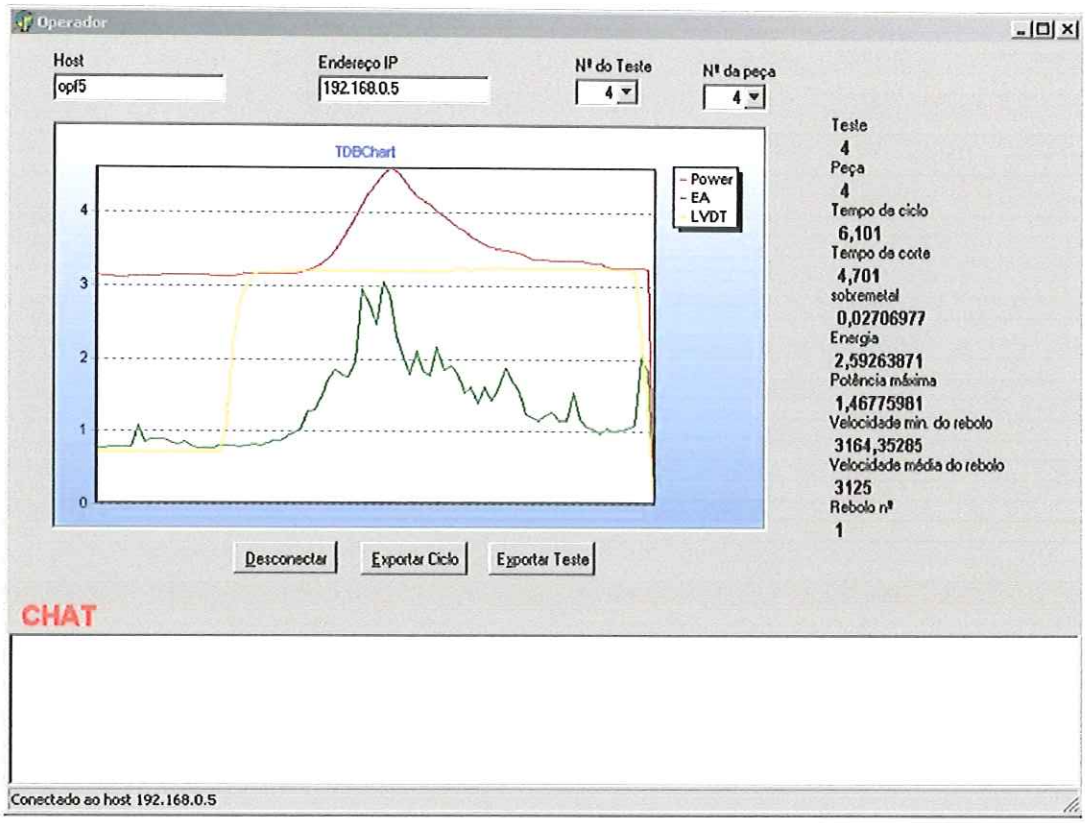

Figura 45 - Aplicação do Operador

Na aplicação do operador especifica-se o nome do computador a se comunicar (*host)* e/ou o endereço IP da mesma. Feito isso, a conexão é concretizada ao apertar o botão Conectar / Desconectar. Na figura 45 nota-se que a aplicação do operador já está conectada com a aplicação do consultor, conforme é mencionado na barra de status localizada na parte inferior da tela. Para maior integração entre operador e consultor durante a realização da consultoria, criou-se uma ferramenta de troca de mensagens instantâneas (*Chat*). Nesta aplicação ainda pode-se escolher o teste e peça a serem exportados à aplicação do consultor.

A aplicação do consultor pode estar localizada em qualquer computador conectado a Internet. Esta aplicação está integrada via ADO a uma base de dados Access 2000 de apoio. Esta base de dados recebe os dados enviados pela aplicação do operador e os disponibiliza aos componetes gráficos da aplicação. Na plicação do consultor é possível de se analisar gráficos e dados de ciclos conforme é mostrado na figura 46.

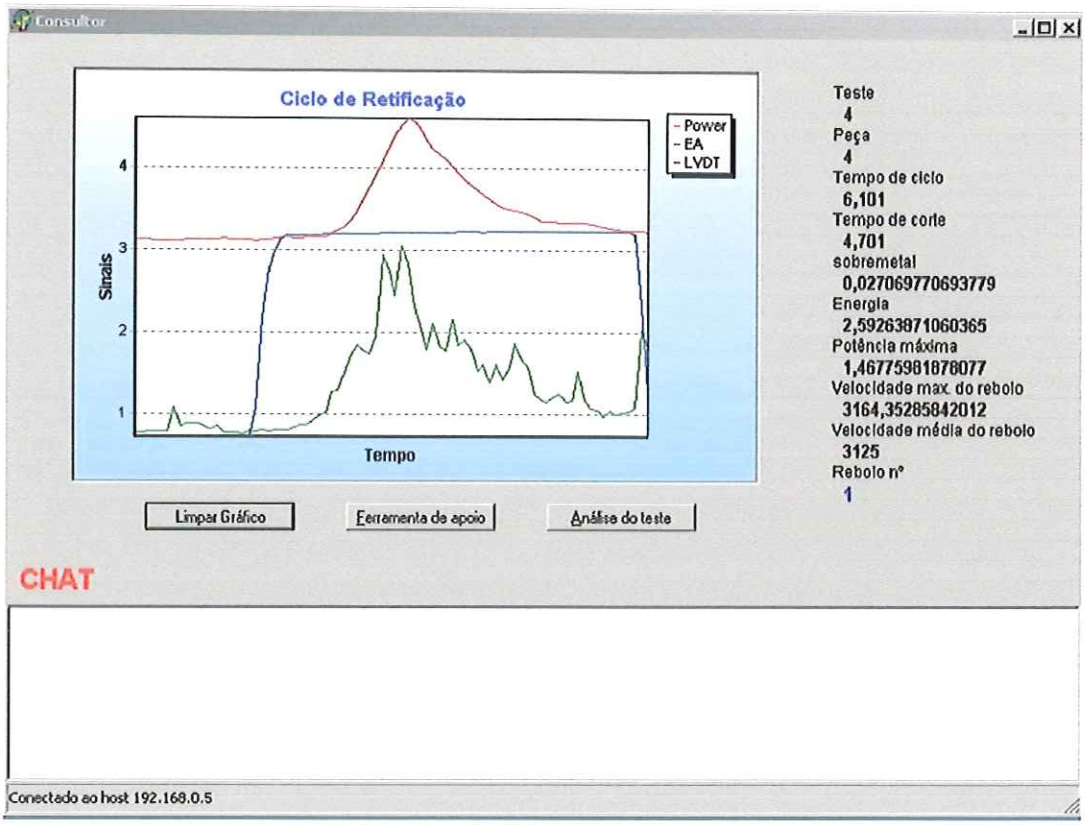

Figura 46 - Análise de Ciclo na Aplicação do Consultor

Nesta aplicação, além da ferramenta *chat* desenvolvida, outras funcionalidades foram implementadas para assegurar a robustez e usabilidade do sistema. Através de um botão é possível limpar o gráfico mandado pela aplicação do operador. Uma ferramenta de apoio à tomada de decisão também foi desenvolvida e é mostrada na figura 47.

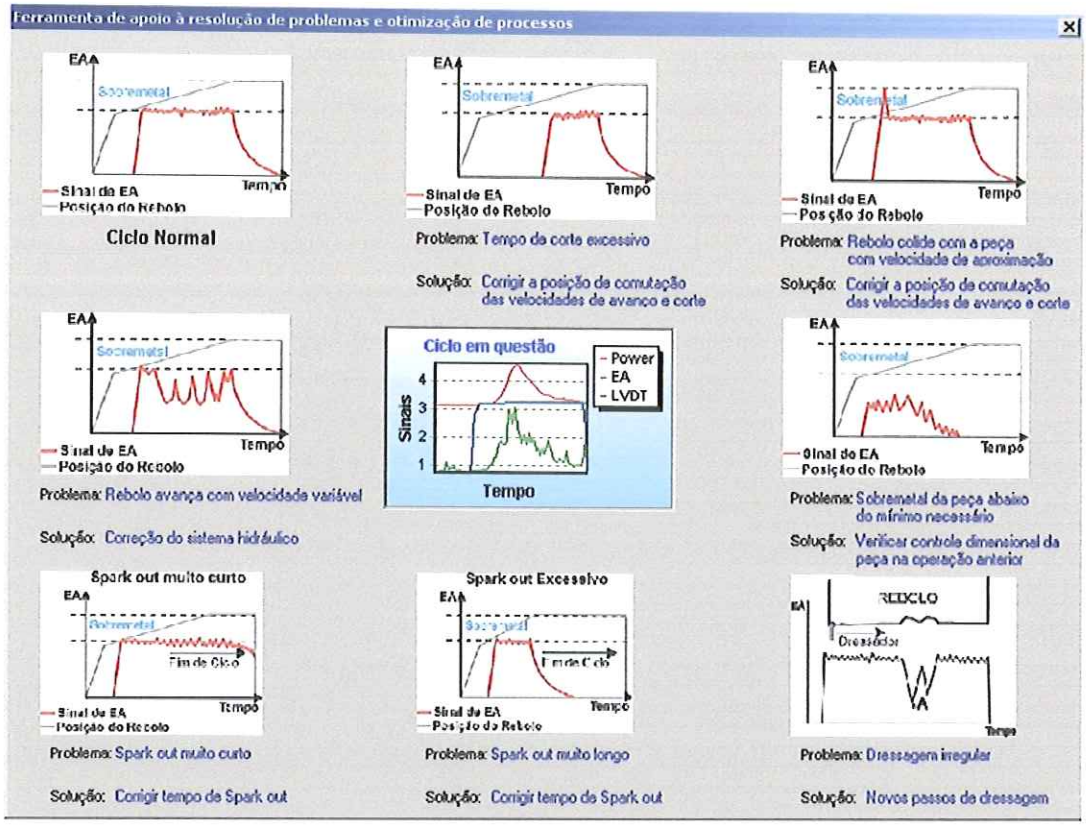

Figura 47 - Ferramenta de Apoio à Tomada de Decisão

Esta ferramenta proporciona uma comparação visual entre gráficos indicadores de problemas característicos do processo e o gráfico do ciclo em análise. Esta ferramenta aponta além dos problemas, possíveis soluções a serem aplicadas.

O consultor ainda conta com uma ferramenta de análise do teste em geral. Através dela, pode-se avaliar o comportamento das grandezas do processo durante a produção do lote monitorado. Esta ferramenta é mostrada na figura 48.

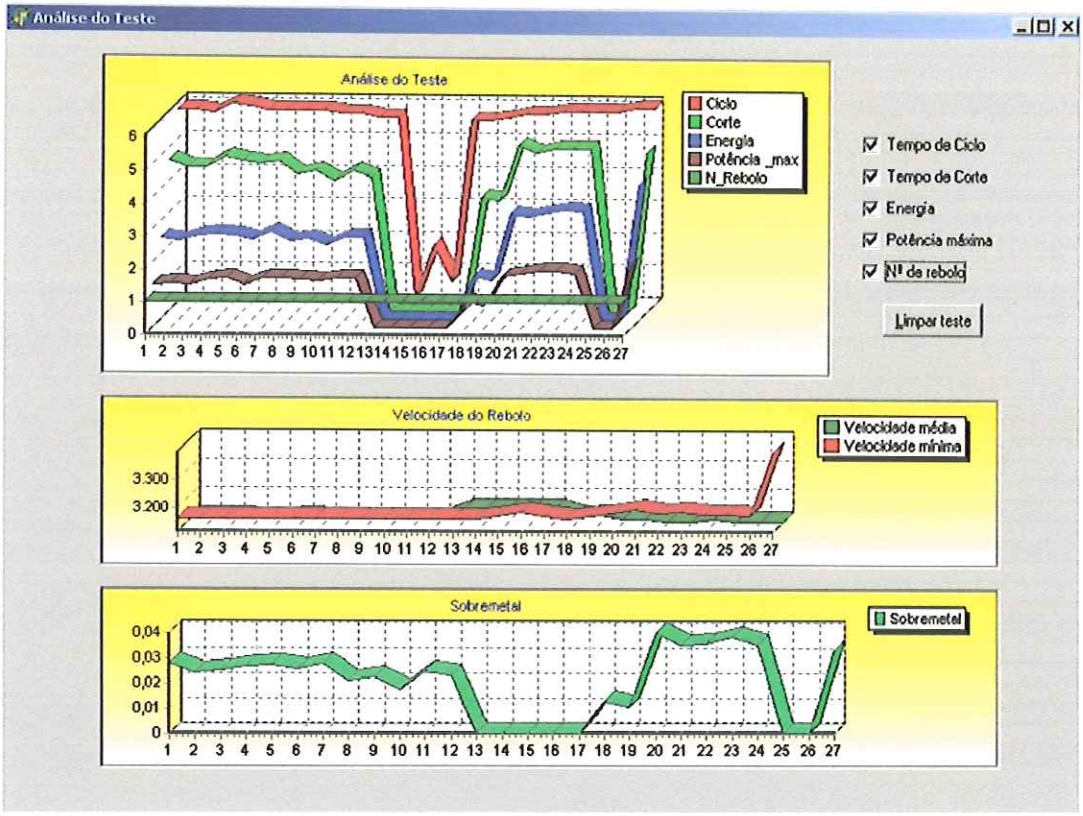

Figura 48 - Análise Geral do Teste

4.6 – PROPOSTA DE UMA POSSÍVEL SOLUÇÃO INTEGRADA

Para a implementação do Sistema de Diagnóstico Remoto pode-se utilizar uma integração entre as soluções implementadas em PHP e em Delphi, o que optou-se por identifica-la como **Solução Integrada.** Esta solução pode ser implementada em Delphi, utilizando-se CGI, PHP4, HTML e TCP/IP (aplicação Cliente /Servidor).

Com os componentes *PageProducers* do Delphi, via CGI, é possível gerar através de um servidor Web (Apache) páginas HTML contendo dados de tabelas ou consultas de uma base de dados remota. A linguagem PHP4 pode auxiliar no tratamento de dados e na integração Web com a base de dados.

De uma maneira mais simples, via TCP/IP em Delphi é possível a comunicação entre uma aplicação cliente remota (aplicação do consultor) e uma aplicação servidora conectada a base de dados (aplicação do operador). A criação e

análise dos gráficos dos sinais coletados neste tipo de aplicação é bem mais iterativa, pois nela pode-se utilizar todos os recursos visuais do Delphi.

5 - VALIDAÇÃO DO SISTEMA

Para maior compreensão e possível refinamento do sistema de aquisição e de consultoria remota implementados, foram feitos alguns ensaios de aquisição e análises dos dados coletados dos sistemas desenvolvidos. Os dados obtidos são coletados pelo sistema de sensoreamento e armazenados na base de dados por meio do *software* de aquisição implementado em LabVIEW. Os gráficos apresentados nos testes são gerados automaticamente a partir de consultas na base de dados, as quais são feitas de maneira remota através de aplicações TCP/IP cliente/servidor implementadas em Delphi. Tanto os gráficos como os dados de testes são transferidos através de *sockets* TCP/IP. O código de implementação é mostrado no Apêndice 7. Com este módulo de análise, é possível buscar na base de dados gráficos específicos, e compará-los, em busca de soluções de problemas de retificação e possíveis otimizações do processo.

O teste em demonstração foi feito em uma máquina da FIM (Fábrica Integrada Modelo) no NUMA (Núcleo de Manufatura Avançada), laboratório da USP – São Carlos. Os dados gerais do teste são:

- Máquina: *centerless* MICROMA
- Material: W45
- Dureza do material: 48
- Rebolo: A54.5N5V1
- Número de amostras: 27 peças
- UpTime: 431,914 segundos
- Down Time: 2576,483 segundos
- Peça utilizada: válvula de motor automotivo (figura 49)

Figura 49 - Peça Utilizada no Teste em Questão

Na figura 50 é feita uma análise global das grandezas monitoradas no teste realizado no decorrer de todas suas amostras. Pelos gráficos desta figura nota-se a estabilidade das grandezas monitoradas. As quedas dos valores das grandezas entre as peças 13 e 17 são devidas à ocorrência de ajustes realizados no sistema de aquisição e no processo durante o monitoramento das amostras.

Os valores estáveis da potência consumida indicam que o rebolo não está perdendo capacidade de corte, mas sim soltando grãos abrasivos ao longo do processo. Isso implica a provável necessidade de aumento na dureza de rebolo para se buscar uma otimização de processo (normalmente deve haver um acréscimo de potência da ordem de 30% no decorrer da produção de um lote).

O tempo de ciclo também é constante durante o monitoramento do lote em análise. Isso indica que a cinemática da máquina é estável, apesar de ser um equipamento com controle pneumático, que estaria sujeito a variações com as oscilações de pressão na rede. O sobremetal retirado não sofre variações consideradas, justificando a boa cinemática do processo conforme é mostrado na figura 51.

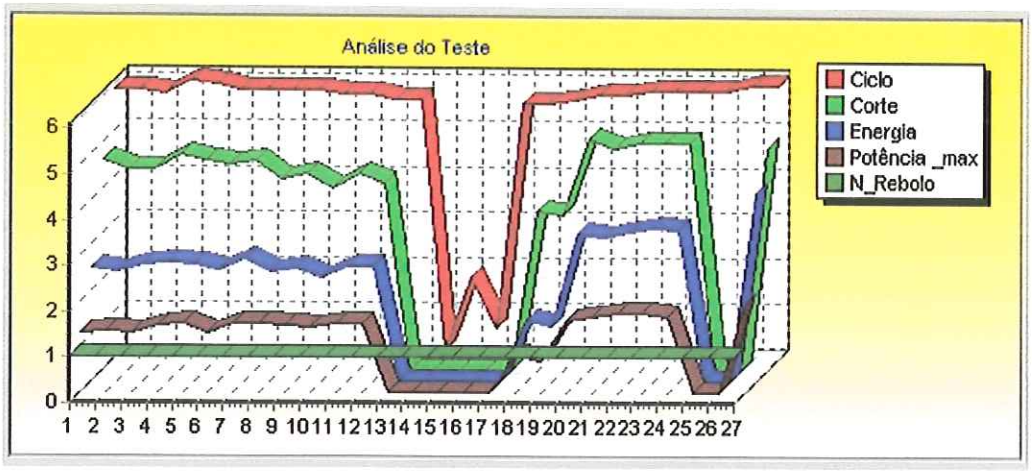

Figura 50 - Análise Geral das Grandezas Monitoradas

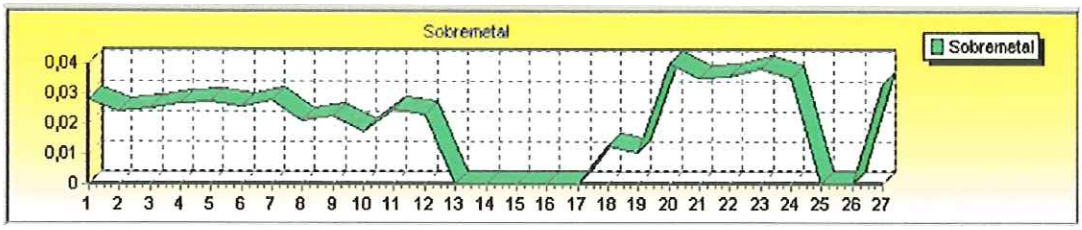

Figura 51 - Análise do Comportamento do Sobremetal Removido das Amostras

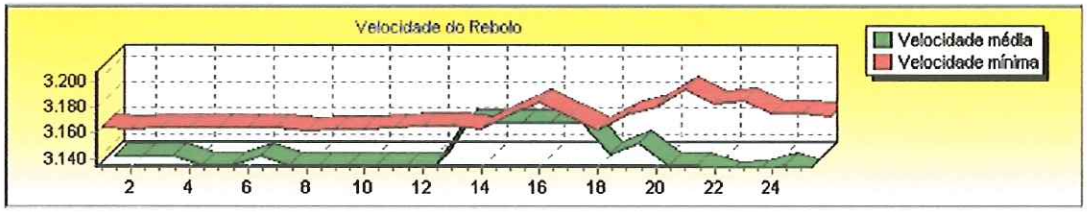

Figura 52 - Velocidade Média e Mínima do Rebolo no Decorrer do Teste

Observa-se na figura 52 que as velocidades médias do rebolo sofrem poucas variações ao decorrer do teste. As variações de velocidade não ultrapassam a ordem de 1,58%, o que corresponde a uma variação de velocidade desprezível para o processo.

Nas figuras 53 e 54 são feitos comparativos entre dois ciclos de amostras diferentes.

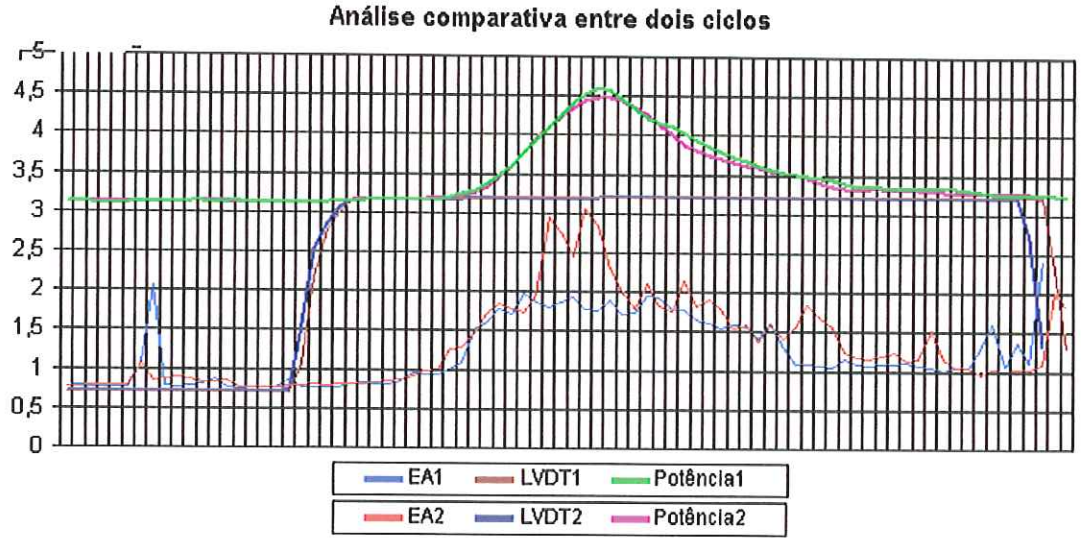

Figura 53 - Comparativo entre Ciclos Diferentes num Mesmo Gráfico

Avalia-se pouca diferença de sobremetal entre as curvas das duas peças. As curvas dos sinais de EA apresentam formas mais distintas, o que implicam somente em variações locais do processo.

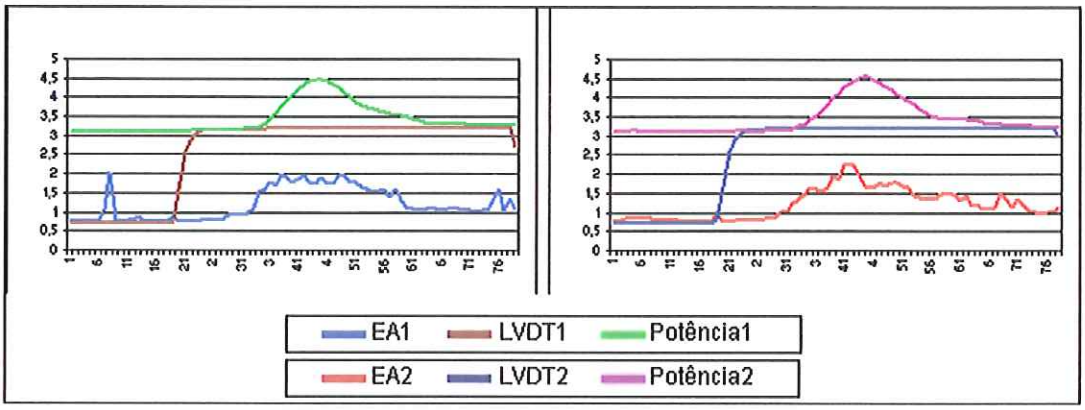

Figura 54 - Gráficos de Ciclos de Amostras Diferentes

Através do monitoramento das grandezas de *Down Time*, *Up Time*, e tempo de dressagem é possível avaliar a distribuição do tempo durante o monitoramento da produção de um lote de peças. No gráfico da figura 55 é mostrada uma ferramenta de apoio à supervisão da produção. Percebe-se que, durante o teste, a máquina ficou parada (improdutiva) 74% do tempo de produção total. Isso pode identificar paradas desnecessárias na produção, o que aponta possíveis otimizações no processo.

EESC-USP
Biblioteca
Campus de S. Carlos

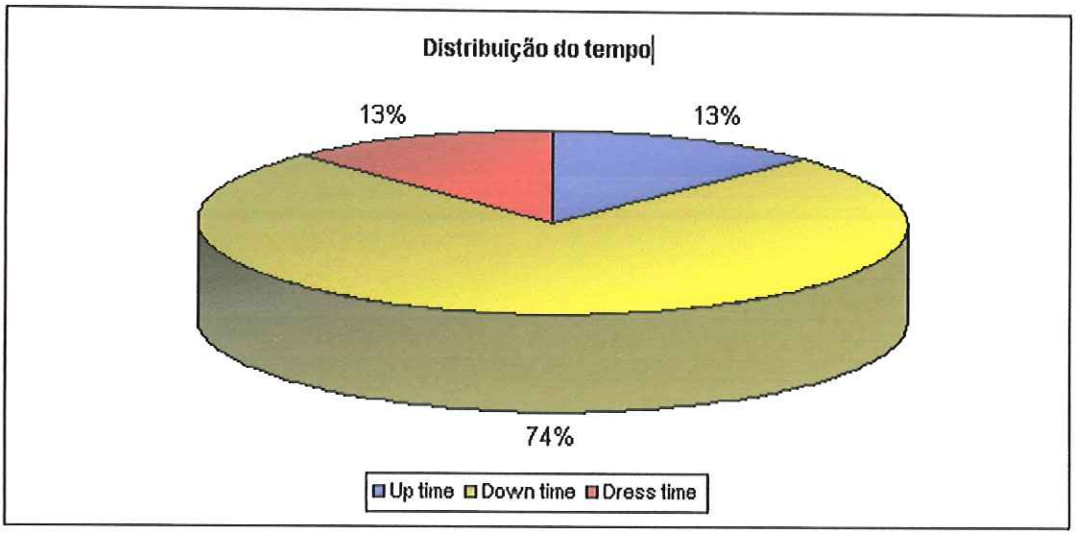

Figura 55 - Distribuição do Tempo

Através deste módulo de análise pode-se extrair gráficos da base de dados alimentada pelo sistema de aquisição, realizar análises de problemas e avaliar possíveis otimizações no processo.

Após a realização de testes em ambientes industriais (como a TRW Automotives, Krupp e Tecumseh) com o sistema de aquisição desenvolvido como parte do sistema de diagnóstico à distância para processos de retificação, notou-se que o sinal de EA coletado do processo de retificação muitas vezes é prejudicado pelos ruidos advindos da própria máquina e do ambiente externo que a envolve. Os abalos causados pelos ruídos dificultam a configuração da Unidade de Monitoramento responsável pelo tratamento dos sinais e, portanto, diminuem a credibilidade das grandezas monitoradas e iniciadas com base na emissão acústica (como o Tempo de Corte).

Neste panorama, decidiu-se incluir a opção de se escolher entre o sinal de EA e o de potência consumida pela máquina como *trigger* a ser utiizado no monitoramento. O sinal de potência, embora menos sensível do que o de EA, é mais confiável. Isto se deve ao fato de sua coleta ser feita fora do ambiente externo industrial, uma vez que o sensor de Efeito Hall utilizado na aquisição é colocado dentro do painel da máquina, envolvendo o cabo do motor (conforme visto anteriormente).

Após esta implementação, avaliou-se que a confiabilidade da grandeza monitorada permanesce alta com o *trigger* de potência configurado. Com isso, o sistema abriu possibilidades de ajustes para os diferentes ambientes de monitoramento.

Essencialmente, todos os resultados experimentais obtidos pelo sistema de aquisição puderam ser transmitidos para a aplicação remota com um tempo de tranferência da ordem de 2,5 segundos. Isso é o suficiente para se realizar o diagnóstico de processos de retificação à distância de maneira remota. Os resultados retratam a realidade do processo. O sistema oferece informações confiáveis tanto na aquisição como na transmissão de dados ao usuário final.

Os sensores são fáceis de instalar e o *set up* do sistema dura em torno de 40 minutos. É importante que o sinal seja tratado pela unidade de monitoramento de maneira adequada ao bom funcionamento do sistema.

6 - CONCLUSÕES

Ao final deste estudo torna-se clara a viabilidade de um Sistema de Diagnóstico e Monitoramento à Distância para processos de Retificação. Por meio deste trabalho concluiu-se que:

- É possível detectar problemas de retificação através da análise de sinais;
- Os ensaios de análises de sinais do processo realizados enrijeceram a habilidade de se detectar problemas de retificação;
- Por meio do sistema de aquisição baseado em sensores é possível coletar informações confiáveis, as quais realmente retratam a realidade do processo de retificação. Essas informações armazenadas em uma base de dados podem ser posteriormente analisadas através de consultas e *softwares* de análises remotas;
- Os dados transmitidos via Internet são confiáveis e condizem com as informações coletadas do processo monitorado;
- As taxas de transferência de informações entre as aplicações remotas são suficientes para realizar o diagnóstico do processo remotamente;

- O tempo gasto para a configuração do sistema permite que o mesmo seja utilizado no dia-a-dia do ambiente industrial;
- Através das informações descritas neste estudo fica clara a importância da sistematização de informações em um sistema de monitoramento e diagnóstico para processos de retificação baseado em análises de sinais;
- O uso de múltiplos sensores pode fornecer informações muitas vezes ocultas mesmo aos operadores mais experientes, sendo uma grande vantagem para a detecção e resolução de problemas;
- Com o Sistema de Aquisição e de Análise Remota desenvolvidos e com a integração destes com a base de dados, a qual é administrada e acessada remotamente via Internet, torna-se claro que o desenvolvimento do projeto em questão obteve êxito;
- Foi possível desenvolver o sistema de diagnóstico e monitoramento remoto para processos de retificação.

Analisando os resultados e as conclusões apresentados neste documento percebe-se que o projeto alcançou os objetivos propostos.

7 – SUGESTÕES PARA TRABALHOS FUTUROS

Com o estudo realizado para se desenvolver um sistema de apoio ao diagnóstico a distância para processos de retificação, algumas oportunidades de pesquisas futuras são sugeridas:

- Possível solução integrada utilizando PHP, CGI, Delphi, TCP/IP, HTML;
- Integração do sistema desenvolvido com sistemas baseados em base de dados (como *softwares* SCADA, MES, ERPs);
- Utilização de recursos de vídeoconferência para disponibilizar a imagem e som do processo de retificação, além de melhorar a interatividade do sistema;
- Uso de aplicações LabVIEW para Web desenvolvidas em Java Class.

8 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] OLIVEIRA, J. F. G. (1989). Tópicos avançados sobre o processo de retificação. *Apostila - Escola de Engenharia de São Carlos. Universidade de São Paulo.* São Carlos.

[2] LEICHTER, S. (1975). A escolha de abrasivos favoráveis para a fabricação. *Tradução da firma Robert Boch.* Campinas.

[3] MELLO, R.G.; SOARES, D.D.; MEIRELES, G.; OLIVEIRA, J.F.G.; BIFFI, M.(2000). Sistema Supervisão para Processos de Usinagem: estudo de caso. *1°COBEF Congresso Brasileiro de Engenharia de Fabricação*. Curitiba. PR. 10 p.

[4] ANUNCIAÇÃO, H. (1998). Linux para Redes Brasileiras, Érica, São Paulo.

[5] WATTER, L. H.; SZTOLTZ, L. (2000) Guia do Usuário, Conectiva Linux.

[6] GUPTA, M.; PARIHAR, M.; LASALLE, P.; SCRINGER, R. (2002). *TCP/IP – A Bíblia.* São Paulo, Campus.

[7] BARKAKATI, N.(1997) *Linux Secrets.* 2. ed. IDG Books, Califórnia, EUA.

[8] VOLKERDING, P. F.; JOHNSON, E.; REICHARD, K. (1998). *Programando para Linux.* São Paulo, Makron Books.

[9] ROWE, J. (1997). *Construindo Servidores de Banco de Dados Internet com CGI.* São Paulo, Makron Books.

[10] DANESH, A.(1999). *Dominando o Linux - "A Bíblia".* São Paulo, Makron Books.

[11] ZAWODNY, J. D. (2001). MySQL Performance Tuning. *Linux Magazine,* jun, 10 p.

[12] HARKINS, S. S.; REID, M. W. P.(2001) Many Web developers prefer MySQL. *Linux Magazine*, p. 5-113, jul, 5 p.

[13] TRIMBLE, P. S. (2000).Open minds on open source. *Federal Computer Week.* Dez, 12 p.

[14] FELDMANN, K.; GÖHRINGER, J. (1999). Multimedia System for Remote Diagnosis of Complex Placement Machines, *International Journal of Advanced Manufacturing Technology*, 10 p.

[15] TANENBAUM, A. S.(1997) Redes de computadores. Rio de Janeiro, Campus, p.12-302.

[16] FREEMAN, R. L.(1995) Practical data communication, USA, Wiley.

[17] Microsoft. (1992). www.microsoft.com.(3/08/2001).

[18] BARRETO, M. V. S. (2000). Aplicações Web com PHP, Aracajú, jun, 23 p.

[19] BARRETO, M. V. S. (2000). Tutorial da linguagem PHP, Aracajú, set, 96 p.

[20](1998). MySQL Reference Manual for version 3.23.3. www.mysql.com. (17/06/2001).

[21] CANTÚ, M.(2001). Dominando o Delphi 6– "A Bíblia". Makron Books.

[22] National Instruments. (1998). www.ni.com. (12/09/2001).

[23] SOARES, D. D.; OLIVEIRA, J. F. G. (2001). Diagnóstico de processos de retificação através da análise de sinais. COBEM. nov, 10 p.

[24] GERMAIN, G. L.; CLARKE, T. W. (1999). A.R.T. of supervisory control and data acquisition. *International Pipeline Conference.* Calgary, p. 895-902.

[25] TEK SOFT CONSULTING (1999). www.iinet.net.au. (23/07/2000).

[26] KRISHNAMURTHY, R.; (1996). Sensor-fusion for identification of process performance in deep hole drilling. *International Manufacturing Engineering Conference.* University of Connecticut, London, p. 454-456.

[27] BYRNE, G. et al. (1995). Tool condition monitoring (TCM) – the status of research and industrial application. *Annals of the CIRP.* p. 1-27.

[28] POLLOCK, A. A. (1989). *Acoustic emission inspection, American Society for Metal, Metals handbook.* Ohio,USA, Metals Park.. v.17, p.278-294.

[29] BLUM, T.; DORNFELD, D. A.(1990). Grinding process feedback using acoustic emission. *International Grinding Conference and Exposition.* Detroit, 10 p.

[30] JEMIELNIAK, K. (1999). Commercial tool condition monitoring systems. *The International Journal of Advanced Manufacturing Technology.* v. 15, p. 711-721.

[31] (1999). PCB – Technical support. Introduction to Piezoelectric Accelerometers. www.pcb.com. (24/06/2000).

[32] KONIG, W.; KLUMPEN, T. (1993). Monitoring and sensor concepts for higher process reliability. *The International Grinding Conference and Exposition.* Cincinnati, 10 p.

[33] (1999). Macro Sensors Corpotare. www.macrosensors.com/primer.htm. (12/03/2000).

[34] Atos automação industrial LTDA (2000). Sensores de proximidade indutivos, Manual. jan.

[35] HASSUI, A.; DINIZ, A. E.; OLIVEIRA, J.F.G.; JR, J. F.; GOMES, J.J.F. (1998). Experimental evaluation on grinding wheel wear through vibration and acoustic emission. *Wear.* p. 271.

[36] ROWE, W. B.; YAN, L. (1994). Applications of Artificial Intelligence in Grinding. *Annals of the CIRP*, v.43, 13 p.

[37] HUNDT, W. et al. (1994). An approach to monitoring of the grinding process using acoustic emission (AE) technique. *Annals of the CIRP.* v.43, n° 1, p. 295-298.

[38] KLUFT, W. (1994). A monitoração dos processos reduz custos e desperdícios. *Máquinas e Metais.* p.60-80, out.

[39] KING, R. I.; HAHN, R. S.(1995) *Handbook of modern grinding technology.* New York, Chapman and Hall.

[40] OLIVEIRA, J. F. G.; BIANCHI, E. C; SOUZA, G. F. (1998). O desempenho de rebolos pode ser controlado pela dressagem.

[41] SAKAKURA, M.; INASAKI, I. (1992). A Neural Network Approach to the Decision-Making Process for Grinding Operation. Keio University. Yokoama/Japan. jan.

[42] DEGARMO, E. P.; BLACK, J. T.; KOHSER, R. A.(1998). Materials and Processes in Manufacturing. Editora Prentice Hall, ed. 8.

[43] BIANCHI, E. C.; MONICI, R.D.; NEEVES, R. C.; SILVA, E. J. ; AGUIAR, P. R. (2001). Analysis of behavior between grinding wheels with different bonds and cutting conditions. *COBEM.* nov.

[44] BYRNE, G.; DORNFELD, D. A.; KETTELER, G.; KÖNIG, W; TETI, R. (1995). Tool condition monitoring (TCM) – The Status of Research and Industrial Application.. *Annals of the CIRP.* v.44, 10 p.

[45] (2000). www.clubedelphi.com.br. (23/04/2002).

[46] ALVES, L. (1994). *Comunicação de dados.* São Paulo, Makron Books.

[47] COMER, D. E. (1998). *Interligação em redes com TCP/IP.* Campus, São Paulo.

[48] OLIVEIRA, J.F.G. (1988). *Análise da Ação do Microefeito de Dressagem de Rebolos no Desempenho do Processo de Retificação.* Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

[49] JUNIOR, J. H. T.; SUAVÉ, J. P.; MOURA, J. A. B.; TEIXEIRA, S. Q. R. (1999). *Redes de Computadores – Serviços, Administração e Segurança..* São Paulo, Makron Books.

[50] FELIPE, Jr. J., OLIVEIRA, J. F. G. (1998). A retificação inteligente é mais confiável e menos dependente do operador. *Máquinas e metais*, n.386, p.80-98, mar.

[51] OLIVEIRA, W. J. (2001), *Programação para Internet com Aplicações em Delphi*, Visual books, p.5-194, nov.

[52] DATE, C. J. (2000), *Introdução a Sistemas de Banco de Dados*, Campus, Rio de Janeiro, p.5-86.

[53] SETZER, V. W. (1991), *Banco de Dados – Conecitos, Modelos, Gerenciadores, Projeto Lógico, Progeto Físico*, Edgard Blucher, São Paulo, p.12-68.

[54] HERRMANN, E. (1997), *Aprenda em 1 semana – Programação CGI com Perl5*, Campus, Rio de Janeiro, p.5-87.

[55] SWAN, T. (1999), *Delphi 4 – Bíblia do programador*, Berkeley, São Paulo, p.4-76.

9 - APÊNDICES

1 – BREVE HISTÓRICO DA INTERNET

Conhecida também como rede mundial, a Internet é uma rede de abrangência em todo o mundo, baseada na arquitetura TCP/IP. O TCP/IP surgiu com a criação de uma rede patrocinada pelo Departamento de Defesa dos Estados Unidos. O intuito era fazer a comunicação de órgãos do governo e universidades, em caso de guerras ou catástrofes que afetassem os meios de comunicação daquele país. Surgiu então a ARPANET, uma rede que permaneceria intacta caso um dos servidores perdesse a conexão. Esta rede necessitava de um modelo de protocolos que assegurasse tal funcionalidade esperada, mostrando-se confiável, flexível e de fácil implementação. É então desenvolvida a arquitetura TCP/IP, que se torna um padrão de fato. A ARPANET cresceu e tornou-se a rede mundial de computadores – Internet. A utilização (e facilidades) do padrão TCP/IP utilizado pelos fabricantes de outras redes, teve a finalidade de conectar toda a Internet. A normalização do TCP/IP chegou após a sua utilização em massa [47].

A Internet cresceu, abrangendo centenas de redes individuais localizadas nos Estados Unidos e na Europa. A utilização de protocolos TCP/IP e o crescimento da Internet não se limitaram a projetos financiados pelo governo. Grandes companhias voltadas para o setor de computadores conectaram-se à Internet, bem como muitas outras organizações de grande porte.

Uma expansão acelerada trouxe problemas de escala não previstos no projeto original e motivou os pesquisadores a encontrar técnicas para gerenciar numerosos recursos distribuídos. Novos protocolos foram então desenvolvidos e um sistema de atribuição de nome foi colocado em vigor através da Internet para permitir que qualquer usuário automaticamente determinasse o nome de uma máquina remota. Conhecido como Domain Name System (DNS), o mecanismo conta com máquinas denominadas servidoras de nome para responder a consultas sobre nomes. Nenhuma máquina contém todo a base de dados de nomes de domínios. Em vez de uma

máquina, os dados são distribuídos por um conjunto de máquinas que utilizam protocolos TCP/IP para se comunicarem entre si quando estiverem respondendo a uma consulta. Tão logo a Internet tornou-se popular e os usuários passaram a buscar informações através de serviços como a World Wide Web [15].

A Internet não é controlada por nenhum órgão governamental ou comercial, mas sim por organizações voluntárias que controlam os usuários e os artigos publicados na Internet. Eis algumas organizações:

IAB	A IAB (Internet Advisory Board) é constituída de várias organizações e seu objetivo principal é coordenar a organização geral da Internet.
InterNIC	A InterNIC (Internet Network Information Center) foi criado pela NSF para distribuir endereços IP.
IRTF	O IRTF (Internet Research Task Force) é um dos comitês que constituem a IAB. Ele é responsável por várias atividades de pesquisa, como o desenvolvimento de protocolos.
RFC	RFC (Request for Comments) são documentos técnicos relacionados aos protocolos da Internet. Alguns deles contém padrões para os protocolos, outros são recentemente desenvolvidos, podendo obter sucesso e se tornarem padrões. Esses documentos formam a documentação da Internet.
FNC	FNC (Federal Networking Council) é um comitê que exerce a parte informativa da Internet. A FNC realiza o intermédio entre a IAB e as instituições governamentais, além de prestar suporte a agências no uso da Internet.
IETF	IETF (Internet Engeneering Task Force) é um subcomitê da IAB que realiza a manutenção de problemas construtivos e também a implementação de novas tecnologias.

Tabela 4 - Organizações de Controle da Internet

A Internet é considerada por muitos como um dos mais importantes e revolucionários desenvolvimentos da história da humanidade. Criar, gerenciar e distribuir informações em larga escala, no âmbito mundial, algo que somente uma

grande organização poderia fazer usando os meios de comunicação convencionais. Essa perspectiva abre um enorme mercado para profissionais e empresas interessadas em oferecer serviços de informação específicos.

2 - Informações Adicionais sobre a Implementação de CGIs

As informações enviadas por um programa CGI podem ser passadas por:

- Passagem de parâmetro na linha de comando;
- Passagem de variáveis ambiente para programas CGI;
- Passagem de dados por meio de entrada padrão;
- Uso de informação de caminhos extras;

Existem diversas linguagens para se programar uma CGI. AS mais comuns e suas respectivas aplicações são:

- *Script* de *Shell* Unix (usada para executar comandos de sistema);
- Linguagem C (ideal quando a velocidade de execução e a segurança de código fonte forem importantes);
- Perl (para implementar aplicações que necessite de rico conjunto de recursos);

Para se recuperar informações de uma base de dados e disponibiliza-la na Internet via CGI deve-se seguir os seguintes passos:

1 - Usuário chama programa de gateway usando CGI;

2 - O *browser* reúne as informações introduzidas pelo usuário para enviar ao programa CGI;

3 - O *browser* entra em contato com o servidor HTTP a onde reside o programa CGI pedindo para que o programa seja encontrado e que se passe as informações a ele;

4 - O servidor HTTP verifica permissão de acesso;

5 - O programa gateway é executado;

6 - O gateway processa informações em um formato entendido pelo B.D;

7 - O gateway usa módulos de B.D para passar a consulta para a interface de B.D;

8 - A interface analisa a precisão da consulta;

9 - Em caso de erro, mensagem de erro é passada a programa de gateway e enviada ao servidor HTTP que o envia ao *browser*, onde o processo é encerrado;

10 - Se não houver erros, a interface de B.D. envia a consulta ao B.D;

11 - O B.D executa a consulta e a passa de volta ao programa de gateway, por meio da interface;

12 - O programa de gateway formata o resultado e o envia, através do CGI, para o servidor retransmitir ao *browser* Web;

13 - O *browser* apresenta então os resultados;

Alguns *softwares* são necessários para se integrar uma base de dados à Web. Estes requisitos são:

- S.O.;
- Servidor Web;
- Servidor de Base de Dados;
- Linguagem de programação CGI;
- Interface de Base de Dados da Web;

As CGI's apresentam uma gama de possíveis aplicação. Alguns exemplos do uso de CGIs são:

- Atualização Automática de notícias;
- Cartões Virtuais;
- Chat;
- Classificados;
- Consulta em Base de Dados;
- Contador de Acesso;
- Contagem Regressiva;
- Formulários de auto-resposta;
- Grupo de discussão;
- Livros de visitas;
- Proteção de acesso por senha;
- Publicidade rotativa;
- Sistema de busca;
- Sistemas de compras on-line;
- Sistema de pesquisa de opinião;
- Testes de avaliação;
- Texto randômico;

Para se formatar a saída de uma CGI deve-se, primeiramente entender as partes que a compõe. A saída de uma CGI tem basicamente 2 partes:

- cabeçalho de saída ;
- dados de saída.

O cabeçalho não é visível ao usuário e serve para passar informações sobre saída, tanto ao servidor quanto ao cliente Web. Já os dados de saída compõem a parte que realmente será enviada ao cliente Web. O conteúdo da saída deve estar correlacionado com o tipo de conteúdo informado no cabeçalho.

É importante observar que os programas CGIs devem ter as permissões de leitura e execução atribuídas a todos os usuários, e a permissão de escrita apenas para o proprietário do arquivo. Caso contrário o servidor Web não será capaz de executar ou ler o arquivo CGI.

A chamada do arquivo .cgi não pode ser feita localmente pelo cliente Web. Neste caso, o servidor não enxergaria a chamada. Portanto ao se fazer a referência a uma CGI dentro de um documento HTML, deve-se informar a URL absoluta do arquivo CGI (Ex: http://penta.ufrgs.br/edu/*forms*/tutorial.cgi).

Pode-se rodar CGIs achados na Internet. Para isso, deve-se seguir algumas instruções:

- O *script* deve ser compatível com UNIX, mas os feitos para o Windows NT não têm essa necessidade;
- Alguns deles exigem um "*author permission*", o que significa que terá que pagar para usá-lo;
- Se o *script* não estiver compilado, deve-se ter o compilador correto para o mesmo. Se ele estiver em Perl, não é necessária uma compilação;
- O *script* não pode consumir demasiadamente recursos da CPU;
- Após verificar a possibilidade de se executar a CGI no sistema, deve-se mudar alguns diretórios especificados no *script* como, por exemplo, onde fica o diretório do Perl.

Um dos grandes problemas das implementações CGIs é a questão da segurança. Um programa CGI é visto como brecha em potencial na segurança de sistemas. Portanto, há precauções básicas a se tomar, as quais são listadas a seguir:

- Não escrever programas CGI que permitam aos usuários passarem comandos diretamente ao sistema operacional a ser executado;
- Colocar um arquivo index .html em todos os diretórios disponíveis para servidores HTTP. Isso evita que possa se ter acesso ao conteúdo de seus arquivos, fazer *downloads* e estudar pontos fracos de programas;
- As senhas de arquivos .htaccess devem ser transferidas pela rede no formato unencoded;
- Deve-se verificar com freqüência serviços do servidor para examinar a possível tentativa de acessos suspeitos (Verificando os arquivos *erro_log* e *access_log* regularmente);
- Não deve-se alimentar dados fornecidos pelo usuário a um comando *shell* sem antes verificar esses dados cuidadosamente.

3 - INFORMAÇÕES ADICIONAIS SOBRE A LINGUAGEM PHP

A linguagem PHP foi concebida durante o outono de 1994 por Rasmus Lerdorf. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua *home page* apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como "**P**ersonal **H**ome **P**age Tools" (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava algumas *macros* e alguns utilitários que rodavam "por trás" das *home-pages*: um livro de visitas, um contador e algumas outras coisas.

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de **PHP/FI**, o "FI" veio de um outro pacote escrito por Rasmus que interpretava dados de formulários HTML (***Form*** **I**nterpreter). Ele combinou os *scripts* do pacote *Personal Home Page Tools* com o FI e adicionou suporte a mSQL, nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

Estima-se que em 1996 PHP/FI estava sendo usado por cerca de 15.000 *sites* pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do PHP. Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por **Zeev Suraski** e **Andi Gutmans**, e esse novo interpretador foi a base para a versão 3.

O lançamento do PHP4, ocorrido em 22/05/2000, trouxe muitas novidades aos programadores de PHP. Uma das principais foi o suporte a sessões, bastante útil pra identificar o cliente que solicitou determinada informação. Além das mudanças referentes à sintaxe e novos recursos de programação, o PHP4 trouxe como novidade um otimizador chamado Zend, que permite a execução muito mais rápida de *scripts* PHP. A empresa que produz o Zend promete o lançamento de um compilador de PHP. Códigos compilados serão executados mais rapidamente, além de proteger o código-fonte da aplicação [18].

Enviando Dados para o Servidor HTTP

Programar para a web pode ser considerado como um procedimento que consiste em receber os dados do usuário, processá-los e enviar a resposta dinamicamente. Uma vez enviada a resposta, é encerrado o contato entre o servidor e o cliente. Portanto o primeiro passo a ser tomado é receber os dados enviados pelo *browser* ao servidor.

O protocolo HTTP provê dois principais métodos para enviar informações para o servidor HTTP, além da URL referente ao arquivo solicitado. Esses métodos são o POST e o GET.

O protocolo HTTP/1.0 também especifica o método HEAD, utilizado apenas para transmitir informações do *header*, além dos métodos PUT e DELETE.

A especificação do protocolo HTTP/0.9 (a primeira implementação do HTTP) possuía a definição do método GET, utilizado pelo *browser* para solicitar um documento específico. Por exemplo: a seguinte requisição HTTP retornaria o documento "index.html", localizado no diretório do servidor chamado "teste" [19]:

GET /teste/index.html CRLF

Deve-se notar que a requisição GET inicia-se com a palavra GET, inclui o documento solicitado e encerra-se com a combinação dos caracteres *carriage return* e *line feed* (CRLF). Para um melhor entendimento, pode-se fazer uma requisição GET conectando-se diretamente em algum servidor Web através de um programa de TELNET (geralmente o servidor HTTP utiliza a porta 80). A resposta será o código da página solicitada.

telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
GET /index.php3
(... página solicitada ...)
Connection closed by foreign host.

Obviamente a diferença do *browser* é que ele trata as informações recebidas e exibe a página já formatada. Através do método GET também é possível passar parâmetros da requisição ao servidor, que pode tratar esses valores e até alterar a resposta a depender deles, como no exemplo abaixo:

```
telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
GET /index.php3?id=0024horas&tipo=Taxi
(... página solicitada ...)
Connection closed by foreign host.
```

No exemplo, são passados dois parâmetros: id e tipo. Esses parâmetros estão no formato conhecido por URLencode. Apesar de ser possível passar parâmetros utilizando o método GET, e com isso gerar páginas dinamicamente, este método tem pelo menos dois problemas que em determinadas circunstâncias podem ser considerados sérios. O primeiro é que o GET permite uma quantidade de dados passados limitada a 1024 caracteres, o que pode gerar perda de informações em certos casos. O segundo é que pelo fato de que as informações fazem parte da URL, todos os dados podem ser vistos pelo usuário. Isso pode ser extremamente perigoso quando informações sigilosas estão envolvidas (senha, por exemplo).

A versão 1.0 do protocolo HTTP trouxe boas inovações. Uma delas foi a criação de *headers* nas mensagens de requisição e de resposta. Os *headers* são informações trocadas entre o navegador e o servidor de maneira transparente ao usuário, e podem conter dados sobre o tipo e a versão do navegador, a página de onde partiu a requisição, os tipos de arquivos aceitos como resposta, e uma série de outras informações.

Assim foi possível definir um outro método de requisição de arquivos, que resolveu os principais problemas do método GET. Surge então o método POST.

Através da utilização de *headers* é possível enviar os parâmetros da URL solicitada sem expor esses dados ao usuário, e também sem haver um limite de tamanho. Uma conexão ao servidor HTTP utilizando o método POST seria algo semelhante ao que segue:

```
telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
POST /index.php3
Accept */*
Content-type: application/x-www-form-urlencoded
Content-length:22

id=0024horas&tipo=Taxi

(... página solicitada ...)
Connection closed by foreign host.
```

Deve-se, no entanto, observar os *headers* enviados ao servidor: a linha "*Accept*" informa os tipos de dados que podem ser enviados como resposta (no caso, todos). A linha "*Content-type*" informa o tipo de dado que está sendo enviado (*urlencoded*). O terceiro *header* é o mais importante, pois informa o tamanho do corpo da mensagem, que contém os parâmetros. Após todos os *headers* há um salto de linha e então é iniciado o corpo da mensagem, no formato *urlencoded*.

Certamente, o usuário não deve se preocupar com os *headers*, em codificar os dados ou em calcular o tamanho do corpo da mensagem. O *browser* faz isso de maneira transparente.

Por um outro lado, o método GET pode ser utilizado através da digitação de um endereço no local apropriado do navegador ou através de um *hiperlink*, ou seja, uma referência de uma página a outra. Nesses casos, é preciso converter os dados para o formato *urlencode*. A terceira maneira de utilizar o GET é através de

formulários HTML. Deste modo, o usuário não precisa se preocupar com a codificação dos dados. A utilização de formulários HTML é a única maneira possível de submeter dados pelo método POST.

4 - DESCRIÇÃO DAS TABELAS DA BASE DE DADOS

Na base de dados foram criados três grupos de tabelas.

Tabelas Gerais: armazenam as informações genéricas sobre o teste;

- General_Data: dados gerais do teste.

Coluna	Tipo de dados	Descrição
Test_code	Inteiro auto incremental	Identificador do teste
Client_code	Inteiro	Código do cliente
Machine_code	Número	Código da máquina
Operator	Texto	Nome do operador
Material_spec	Texto	Especificação do material da peça
Material_hardness	Texto	Dureza do material
Abrasive_spec	Texto	Especificação do Abrasivo
Coolant_spec	Texto	Especificação do fluído de corte
Coolant_Type	Texto	Tipo do fluído de corte
Coolant_Conc	Texto	Concentração do fluído de corte
Objective	Texto	Objetivo do teste
Obs	Texto	Observações sobre o teste
Dresser_type	Texto	Tipo do dressador
Dresser_Width_diam	Real	Diâmetro / Largura do dressador
Dresser_rpm	Real	Velocidade do dressador
Wheel_mounting	Data	Data da troca do rebolo
Dressing_Deph	Real	Profundidade de dressagem
Traverse_Rate	Real	Velocidade dressagem
Lead	Real	Passo de dressagem
Overlap	Real	Grau de recobrimento
Speed_Ratio	Real	Raio do disco dressador
Date	Data	Data de início do teste
Part_Spec	Texto	Especificação da peça

Down_Time	Real	Tempo improdutivo da máquina
Up_Time	Real	Tempo produtivo da máquina

- Operatio_Data: dados da operação de retificação do teste

Coluna	Tipo de dados	Descrição
Test_code	Inteiro	Identificador do teste
Cod_Operation	Inteiro	Código da operação
Wheel_rpm	Real	Velocidade do rebolo
Work_rpm	Real	Velocidade do rebolo de arraste
Work_diam	Real	Diâmetro do rebolo de arraste
Wheel_speed	Real	Velocidade de avanço do rebolo
Work_speed	Real	Velocidade de avanço do rebolo de arraste
Equivalent_diam	Real	Diâmetro equivalente
Deph_cut	Real	Profundidade de corte
Table_speed	Real	Velocidade do avanço
Part_length	Real	Comprimento da peça
Part_widh	Real	Largura da peça
Cycle_det	Texto	Detalhes do Ciclo

Tabela de dados:

- Part: para todo ciclo realizado são calculados dados sobre a operação

Coluna	Tipo de dados	Descrição
Test_code	Inteiro	Identificador do teste
Part_code	Inteiro	Número de identificação da peça
Cycle_time	Real	Tempo de ciclo
Cut_time	Real	Tempo de retificação
Stock_part	Real	Sobremetal da peça
Energy	Real	Energia da retificação
Power_max	Real	Pico de potência na retificação
Final_part_position	Real	Posição final da ferramenta no ciclo
Wheel_speed_Min	Real	Velocidade mínima do rebolo

Wheel_Speed_Avg	Real	Media da velocidade do rebolo
Wheel_Index	Integer	Número identificador do rebolo
Date	Data	Data da aquisição
Time	Time	Horário da aquisição

- Dressing: para toda dressagem realizada são armazenados dados sobre o fenômeno.

Coluna	Tipo de dados	Descrição
Test_code	Inteiro	Identificador do teste
Dressing_code	Inteiro	Código da dressagem
Dressing_code	Real	Tempo de dressagem
Parts_dressing	Inteiro	Peças por dressagem
Last_part_code	Inteiro	Código da peça anterior à dressagem
Compensation	Real	Compensação do rebolo
Part_cost	Real	Custo por peça

- Graphic_Matrix: armazena os dados completos sobre o ciclo de retificação (gráfico dos sinais coletados).

Coluna	Tipo de dados	Descrição
Test_code	Inteiro	Identificador do teste
Part_Code	Inteiro	Identificador da peça
Sample_code	Inteiro	Identificador do gráfico armazenado
AE	Real	Sinal de EA
Displacement	Real	Sinal de deslocamento
Power	Real	Sinal de potência
Wheel_Speed	Real	Sinal de variação da velocidade do rebolo

- Graph_Matrix_Dress: armazena os dados completos sobre a dressagem (gráfico dos sinais coletados).

Coluna	Tipo de dados	Descrição

Test_code	Inteiro	Identificador do teste
Dressing_Code	Inteiro	Identificador da dressagem
Sample_code	Inteiro	Identificador do gráfico armazenado
Power	Real	Sinal de potência

Tabelas auxiliares: tabelas que servem apenas para auxiliar a programação e modelagem da base de dados.

- Client: armazena dados do cliente

Coluna	Tipo de dados	Descrição
Client_Code	Inteiro auto incremental	Código do cliente
Client	Texto	Nome do cliente
Address	Texto	Endereço do cliente
Phone	Texto	Telefone do cliente
CGC	Número	CGC do cliente
Contact	Texto	Nome do contato

- Machine: armazena dados da máquina

Coluna	Tipo de dados	Descrição
Machine_code	Inteiro auto incremental	Código da máquina
Clien_code	Inteiro	Código do cliente
Machine	Texto	Nome da máquina

- Operation: armazena os diferentes tipos de operação

Coluna	Tipo de dados	Descrição
Cód_operation	Inteiro auto incremental	Código da operação
Operation	Texto	Nome da operação

5 - INFORMAÇÕES ADICIONAIS SOBRE INTEGRAÇÃO ENTRE PHP E MYSQL

A conexão com o servidor de base de dados MySQL em PHP é feita através do comando *mysql_connect*, que tem a seguinte sintaxe:

```
int mysql_connect(string /* endereço do servidor */ , string /* nome do usuário */ ,
    string /*senha*/);
```

Os parâmetros são: o endereço do servidor como sendo *host*, o nome do usuário e a senha para a conexão. A função retorna um valor inteiro, que é o identificador da conexão estabelecida. O valor deve ser armazenado numa variável para ser utilizado posteriormente. No exemplo abaixo, tem-se como servidor de base de dados a mesma máquina que roda o servidor http (*localhost*), como *login* do usuário "root" e senha "teste":

```
$conexao = mysql_connect("localhost", "root", "teste");
```

Assim, se a conexão for bem sucedida (ou seja, existir um servidor no endereço especificado que possua o usuário com a senha fornecida), o identificador da conexão fica armazenado na variável $conexão. Os comandos para outras bases de dados são diferentes, mas usam padrões parecidos.

Uma vez conectado, é preciso então selecionar a base de dados existente no servidor com a qual deseja-se trabalhar. Isso é feito através da função *int mysql_select_db*, que possui a seguinte sintaxe [20]:

```
int mysql_select_db(string /*nome do database */, int /*conexao*/ );
```

O valor de retorno é 0 se o comando falhar, e 1 em caso de sucesso. O nome da base de dados a selecionar é o primeiro parâmetro fornecido, seguido pelo identificador da conexão. Se este for omitido, o interpretador PHP tentará utilizar a última conexão estabelecida. Recomenda-se sempre explicitar esse valor, para

facilitar a legibilidade do código. No nosso a seguir, a base de dados a ser selecionada possui o nome "ged":

```
$ok = mysql_select_db("ged", $conexao);
```

Após a execução desse comando qualquer consulta executada para aquela conexão utilizará a base de dados selecionada.

Essa variável deve ter um tratamento de retorno. Se for 0, ela estará informando que não foi possível estabelecer uma conexão.

Depois de estabelecida a conexão e selecionada a base de dados a ser utilizada, quase toda a interação com o servidor MySQL pode ser feita através de consultas escritas em ANSI SQL (Padrão Structured Query Language), com o comando mysql_query, que utiliza a seguinte sintaxe:

```
int mysql_query(string consulta, int [conexao] );
```

O valor de retorno é 0 se falhar ou 1 em caso de sucesso. Obter sucesso significa que a consulta está sintaticamente correta e foi executada no servidor. Nenhuma informação sobre o resultado é retornada deste comando, ou até mesmo se o resultado é o esperado. No caso da consulta ser um comando SELECT, o valor de retorno é um valor interno que identifica o resultado. Este poderá ser tratado com a função *mysql_result*() e outras. A string *query* não deve conter ponto-e-vírgula no final do comando, e o identificador da conexão é opcional. Abaixo é criada uma tabela como exemplo:

```
$cria = "CREATE TABLE exemplo (codigo INT AUTO_INCREMENT PRIMARY
    KEY, nome VARCHAR(40), email VARCHAR(50))";
mysql_query($cria, $conexao);
```

Ou

```
$insere = "INSERT INTO exemplo (nome,email) VALUES
```

```
("Fulano","phpnet@phpnet.com.br");
    mysql_query($insere, $insere);
```

Se o comando retornar uma consulta, será necessário guarda-la em uma variável como vetor. Ou mesmo que não retorne uma consulta, mas se for necessário saber se realmente o comando foi executado, pode-se usar uma variável para testes.

```
$consulta = "SELECT nome, email FROM exemplo WHERE nome = 'Teste'";
$resultado = mysql_query($consulta, $conexao);

    printf("Nome: ", mysql_result($resultado,0,"nome"), "<br>\n");
printf("e-mail: ", mysql_result($resultado,0,"email"),"<br>");
```

6.-.INFORMAÇÕES ADICIONAIS SOBRE COMPONENTES PAGEPRODUCERS E DATASETTABLEPRODUCERS

Esta sessão detalhará os procedimentos de programação utilizados para se gerar páginas dinâmicas em Delphi.

Inicia-se um aplicativo Web acrescentando todos os componentes não-visuais necessários em um componente WebModules. Para isso deve-se fazer o seguinte procedimento:

Cria-se um novo aplicativo Web (use File | New > Web Serve Application);

Na próxima tela, escolhe-se o tipo *Cgi Stand-alone Executable* (um WebModule será criado);

Adiciona-se ao WebModule criado os seguintes componentes: Database, Session, Table (paleta Data Access), PageProducer e DataSetTableProducer (paleta Internet) conforme é mostrado na figura 56.

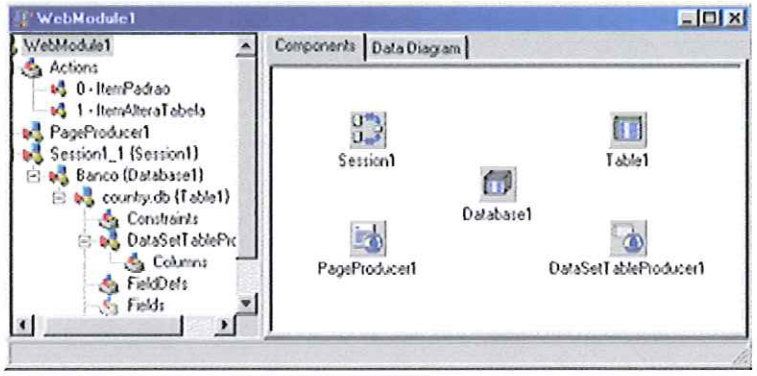

Figura 56 - Modelo de um Componente WebModule

Para o componente Database altera-se as propriedades *DatabaseName* para "NomedaBase", AliasName para "AliasMySQL" e LoginPrompt para "False";

Para o componente Session altera-se a propriedade *AutoSessionName* para "True". Este componente garante que não haja problema de concorrência no acesso à base.

Para o componente Table, altera-se a propriedade *DatabaseName* para "Base", *TableName* para "Pecas" e *Active* para "True";

Conecta-se o componente DataSetTableProducer ao Table, alterando a sua propriedade *DataSet*.

Cria-se um formulário HTML;

Criam-se *ActionsItens* dentro do WebModules (clicando duas vezes no componente WebModules);

Os códigos do formulário em HTML são inseridos na porpriedade HTMLDoc do PageProducer.;

Da-se um duplo clique no componente DataSetTableProducer;

Edita-se a propriedade *Columns*, afim de formatar a tabela a ser gerada;

Adiciona-se o evento *OnFormatCell* do componente DataSetTableProducer. Este evento recebe vários parâmetros, tais como: **CellRow e CellColumn:** o número de linha e da coluna da célula; **BgColor:** a cor de fundo da célula; **Align:** o alinhamento da célula; **CustomAttrs:** atributos personalizados de formatação; **CellData:** o conteúdo da célula convertido para String.

Nele, adciona-se códigos de formatação avançada da tabela. O código abaixo exemplifica a programação deste evento desenvolvida pelo bolsista;

```
// Este evento é chamado quando uma celula é gerada na tabela HTML
procedure TWebModule1.DataSetTableProducer1FormatCell(Sender: TObject;
CellRow, CellColumn: integer; var BgColor: THTMLBgColor; var Align:
THTMLAlign; var VAlign: THTMLVAlign;varCustomAttrs,  CellData: String);
begin
  if CellRow = 0 then
  begin
     BgColor:= 'Navy';
     Align:= haLeft;
     CellData:='<font color="white">' + CellData + '</font>';
  end
  // Alterar cor das linhas pares para cinza claro
 else if CellRow mod 2 = 0 then
     BgColor:='Silver';
end;
```

O código acima faz com que a primeira linha apareça com uma cor diferente e que o restante das linhas tenham cores alternadas;

Adiciona-se códigos ao evento OnHTMLTag no PageProducer para realizar a substituição dos tags transparentes;

```
// Este evento é chamado quando um tag transparente é encontrado
procedure TWebModule1.PageProducer1HTMLTag(Sender: TObject; Tag: TTag;
const TagString: String; TagParams: TStrings; var ReplaceText: String);
var
  i, NumTabelas: integer;
  ListaTabelas: TStringList;
begin
  // Substituir a tag <#ListaTabelas>
  if TagString = 'ListaTabelas' then
  begin
     // Ler os nomes e colocá-los em uma lista
     ListaTabelas := TStringList.Create;
     Session1.GetTableNames(Database1.AliasName, '*.*', True, False,
ListaTabelas);
     NumTabelas := ListaTabelas.Count;
     // Gerar a lista de tabelas
     ReplaceText:= '<select name="NomeTabela" size=' + IntToStr(NumTabelas) +
'>';
    For i := 0 to NumTabelas -1 do
     begin
       //Gerar item selecionado para a tabela atual
       if TabelaAtual = ListaTabelas[i] then
          ReplaceText:= ReplaceText + '<option selected>' + ListaTabelas[i]
       else
          ReplaceText:= ReplaceText + '<option>' + ListaTabelas[i];
    end;
       ReplaceText:= ReplaceText + '</select>';
```

```
        ListaTabelas.Free;
    end
    // Substituir a tag <#DadosTabela>
    else if TagString = 'DadosTabela' then
    begin
      // Usar o DataSetTableProducer para gerar uma tabela html a partir da tabela
atual
      Table1.open;
      // Ajustar o número de linhas da tabela html para que todos os registros sejam
gerados
      DataSetTableProducer1.MaxRows := Table1.RecordCount;
      // Fazer a substituição da tag transparente
      ReplaceText:= '<h3> Tabela: ' + TabelaAtual + '</h3>' +
DataSetTableProducer1.Content;
      Table1.close;
    end;
end;
```

Define-se então o código para eventos OnAction;

A página completa é gerada chamando o método *Content* do PageProducer. O código fica como segue:

```
      // Evento chamado quando o aplicativo é carregado
procedure TWebModule1.WebModule1ItemPadraoAction(Sender: TObject;
Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  try
    // Definir a primeira tabela que será apresentada
    TabelaAtual := 'peça' ;
    // Usar o PageProducer para preparar a página inicial
    Response.content := PageProducer1.Content;
  Except
    // Retornar uma mensagem de erro caso haja alguma exceção
    on e: Exception do
```

```
      Response.Content := 'Erro! ' + e.Message;
  end;
end;
```

O código abaixo verifica se alguma tabela foi selecionada e, caso positivo, altera a tabela atual e gera o código HTML:

```
  // Evento chamado quando o usuário entra na tabela
procedure TWebModule1.WebModule1ItemAlteraTabelaAction(Sender: TObject;
Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
Var
  TabelaSelecionada : String;
begin
  // Ler o nome da tabela selecionada
  TabelaSelecionada := Request.ContentFields.Values['NomeTabela'];
  if TabelaSelecionada <> '' then
    TabelaAtual := TabelaSelecionada;
  Table1.Close;
  Table1.TableName := TabelaAtual;
  // Usar o PageProducer para gerar o HTML depois da mudança de tabela
  Response.Content := PageProducer1.Content;
end;
```

7 - CÓDIGOS UTILIZADOS NA PROGRAMAÇÃO DA APLICAÇÃO REMOTA

Os procedimentos apresentados a seguir diz respeito a aplicação servidora do Sistema de Análise Remota.

```
unit servidor;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
 Dialogs, Buttons, StdCtrls, ExtCtrls, ScktComp, DB, ADODB, Grids,
DBGrids,
 ComCtrls, DBCtrls, TeEngine, Series, TeeProcs, Chart, DbChart;
type
 TForm1 = class(TForm)
  SpeedButton1: TSpeedButton;
  exportar: TBitBtn;
  computador: TEdit;
  DataSource1: TDataSource;
  DataSource2: TDataSource;
  DataSource3: TDataSource;
  DataSource4: TDataSource;
  DataSource5: TDataSource;
  Label1: TLabel;
  Label2: TLabel;
  listateste: TDBLookupComboBox;
  listapart: TDBLookupComboBox;
  ADOConnection1: TADOConnection;
  ADOTable1: TADOTable;
  ADOQuery1: TADOQuery;
  tableteste: TADOTable;
```

```
tablepart: TADOTable;
partquery: TADOQuery;
ClientSocket1: TClientSocket;
Label3: TLabel;
DBChart1: TDBChart;
Series1: TLineSeries;
Series2: TLineSeries;
Series3: TLineSeries;
DataSource6: TDataSource;
tableconsulta: TADOTable;
partquerypart_code: TIntegerField;
partqueryRMS_ID: TIntegerField;
partqueryTest_code: TIntegerField;
partqueryCycle_time: TFloatField;
partqueryCut_time: TFloatField;
partqueryStock_part: TFloatField;
partqueryEnergy: TFloatField;
partqueryPower_max: TFloatField;
partqueryFinal_part_position: TFloatField;
partqueryCoolant_pressure: TFloatField;
partqueryWheel_speed_Min: TFloatField;
partqueryWheel_Speed_Avg: TFloatField;
partqueryWheel_Index: TIntegerField;
partqueryaux1_value: TFloatField;
partqueryaux2_value: TFloatField;
partqueryaux3_value: TFloatField;
partqueryDate: TDateTimeField;
partqueryTime: TDateTimeField;
partqueryScan_Rate: TFloatField;
chat: TMemo;
ServerSocket1: TServerSocket;
ClientSocket2: TClientSocket;
```

```
Label4: TLabel;
StatusBar1: TStatusBar;
ClientSocket3: TClientSocket;
ADOQuery2: TADOQuery;
DataSource7: TDataSource;
ADOQuery2RMS_ID: TIntegerField;
ADOQuery2Test_code: TIntegerField;
ADOQuery2Part_code: TIntegerField;
ADOQuery2Cycle_time: TFloatField;
ADOQuery2Cut_time: TFloatField;
ADOQuery2Stock_part: TFloatField;
ADOQuery2Energy: TFloatField;
ADOQuery2Power_max: TFloatField;
ADOQuery2Final_part_position: TFloatField;
ADOQuery2Coolant_pressure: TFloatField;
ADOQuery2Wheel_speed_Min: TFloatField;
ADOQuery2Wheel_Speed_Avg: TFloatField;
ADOQuery2Wheel_Index: TIntegerField;
ADOQuery2aux1_value: TFloatField;
ADOQuery2aux2_value: TFloatField;
ADOQuery2aux3_value: TFloatField;
ADOQuery2Date: TDateTimeField;
ADOQuery2Time: TDateTimeField;
ADOQuery2Scan_Rate: TFloatField;
ADOQuery2code: TAutoIncField;
ADOQuery2teste: TIntegerField;
ADOQuery2part: TIntegerField;
Label5: TLabel;
ip: TEdit;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
```

```
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
part: TLabel;
DBText1: TDBText;
DBText2: TDBText;
DBText3: TDBText;
DBText4: TDBText;
DBText5: TDBText;
DBText6: TDBText;
DBText7: TDBText;
DBText8: TDBText;
DBText9: TDBText;
DBText10: TDBText;
teste: TBitBtn;
ADOQuery3: TADOQuery;
IntegerField1: TIntegerField;
IntegerField2: TIntegerField;
IntegerField3: TIntegerField;
FloatField1: TFloatField;
FloatField2: TFloatField;
FloatField3: TFloatField;
FloatField4: TFloatField;
FloatField5: TFloatField;
FloatField6: TFloatField;
FloatField7: TFloatField;
FloatField8: TFloatField;
FloatField9: TFloatField;
```

```
    IntegerField4: TIntegerField;
    FloatField10: TFloatField;
    FloatField11: TFloatField;
    FloatField12: TFloatField;
    DateTimeField1: TDateTimeField;
    DateTimeField2: TDateTimeField;
    FloatField13: TFloatField;
    AutoIncField1: TAutoIncField;
    IntegerField5: TIntegerField;
    IntegerField6: TIntegerField;
    ClientSocket4: TClientSocket;
    DataSource8: TDataSource;
    procedure exportarClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure listatesteExit(Sender: TObject);
    procedure listapartExit(Sender: TObject);
    procedure listatesteCloseUp(Sender: TObject);
    procedure listapartCloseUp(Sender: TObject);
    procedure chatKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ServerSocket1ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket1Connect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket1Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure testeClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;
implementation
{$R *.dfm}

{Código do botão de Conexão e Desconexão}
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
 if clientsocket1.active then clientsocket1.active:=false
 else begin
   clientsocket1.Address:=ip.text;
   clientsocket1.host:=computador.text;
   clientsocket1.active:=true;
 end;
 if clientsocket2.active then clientsocket2.active:=false
 else begin
   clientsocket2.Address:=ip.text;
   clientsocket2.host:=computador.text;
   clientsocket2.active:=true;
 end;
if clientsocket3.active then clientsocket3.active:=false
 else begin
   clientsocket3.Address:=ip.text;
   clientsocket3.host:=computador.text;
   clientsocket3.active:=true;
 end;
if clientsocket4.active then clientsocket4.active:=false
 else begin
   clientsocket4.Address:=ip.text;
   clientsocket4.host:=computador.text;
   clientsocket4.active:=true;
 end;
```

```
end;

{Código do botão que exporta os gráficos de um ciclo}
procedure TForm1.exportarClick(Sender: TObject);
var
 msg, msg2, msg3,{ msg4,} data: string;
 dados: array[1..10]of string;
begin
adoquery1.Active:=false;
adoquery1.Active:=true;
adoquery1.First;

 while not(adoquery1.Eof)do
  begin
    msg :=  adoquery1.FieldValues['Power'];
    msg2 := adoquery1.FieldValues['AE'];
    msg3 := adoquery1.FieldValues['Displacement'];
    msg := msg + '#' + msg2 +'#' + msg3 + '#' ;
    clientsocket1.Socket.SendText(msg);
    adoquery1.Next;
  end;
adoquery2.Active:=false;
adoquery2.Active:=true;
adoquery2.First;

 while not(adoquery2.Eof)do
  begin
    dados[1] :=  adoquery2.FieldValues['Test_code'];
    dados[2] := adoquery2.FieldValues['Part_code'];
    dados[3] := adoquery2.FieldValues['Cycle_time'];
    dados[4] :=  adoquery2.FieldValues['Cut_time'];
    dados[5] := adoquery2.FieldValues['stock_part'];
```

```
            dados[6] := adoquery2.FieldValues['Energy'];
            dados[7] :=  adoquery2.FieldValues['Power_max'];
            dados[8] := adoquery2.FieldValues['Wheel_speed_min'];
            dados[9] := adoquery2.FieldValues['Wheel_speed_avg'];
            dados[10] :=  adoquery2.FieldValues['Wheel_index'];
            data:='';
            data := dados[1] + '#' + dados[2] +'#' + dados[3] + '#' + dados[4] +'#' +
dados[5] +'#' + dados[6] +'#' + dados[7] +'#' + dados[8] +'#' + dados[9] +'#' +
dados[10] +'#' + '#&';
            clientsocket3.Socket.SendText(data);
            adoquery2.Next;
          end;

        end;

        {Código da lista de testes realizados}
        procedure TForm1.listatesteExit(Sender: TObject);
        begin
         tableconsulta.First;
         tableconsulta.edit;
         tableconsulta.Post;
         partquery.Active:=false;
         partquery.Active:=true;
        end;

        {Código da lista de peças de um determinado teste}
        procedure TForm1.listapartExit(Sender: TObject);
        begin
         tableconsulta.First;
         tableconsulta.Edit;
         tableconsulta.Post;
        end;
```

```
{Código da lista de testes realizados}
procedure TForm1.listatesteCloseUp(Sender: TObject);
begin
 tableconsulta.First;
 tableconsulta.edit;
 tableconsulta.Post;
 partquery.Active:=false;
 partquery.Active:=true;
end;

{Código da lista de peças de um determinado teste}
procedure TForm1.listapartCloseUp(Sender: TObject);
begin
 tableconsulta.First;
 tableconsulta.edit;
 tableconsulta.Post;
 partquery.Active:=false;
 partquery.Active:=true;
 adoquery1.Active:=false;
 adoquery1.Active:=true;
 adoquery2.Active:=false;
 adoquery2.Active:=true;
end;

{Código de implementação da ferramenta de mensagem instantânea (chat)}
procedure TForm1.chatKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
if key = VK_Return then
 clientsocket2.Socket.SendText(chat.Lines[chat.Lines.count-1])
end;
```

```
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
chat.Font.Color:= clRed ;
chat.Lines.Add('['+ clientsocket1.Address + ' sent] '+ socket.ReceiveText );
chat.Font.Color:= clBlack ;
end;

{Código de mensagens da barra de status}
procedure TForm1.ClientSocket1Connect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  statusbar1.Panels[0].Text:='Conectado ao host '+ clientsocket1.Address;
  speedbutton1.Caption:='&Desconectar';
end;

procedure TForm1.ClientSocket1Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
 statusbar1.Panels[0].Text:='Desconectado';
   speedbutton1.Caption:='&Conectar';
end;

{Código da exortação de dados de um determinado teste }
procedure TForm1.testeClick(Sender: TObject);
var
 data: string;
 dados: array[1..10]of string;
begin
 adoquery3.Active:=false;
 adoquery3.Active:=true;
 adoquery3.First;
```

```
while not(adoquery3.Eof)do
 begin
  dados[1] := adoquery3.FieldValues['Test_code'];
  dados[2] := adoquery3.FieldValues['Part_code'];
  dados[3] := adoquery3.FieldValues['Cycle_time'];
  dados[4] := adoquery3.FieldValues['Cut_time'];
  dados[5] := adoquery3.FieldValues['stock_part'];
  dados[6] := adoquery3.FieldValues['Energy'];
  dados[7] := adoquery3.FieldValues['Power_max'];
  dados[8] := adoquery3.FieldValues['Wheel_speed_min'];
  dados[9] := adoquery3.FieldValues['Wheel_speed_avg'];
  dados[10]:= adoquery3.FieldValues['Wheel_index'];
  data:='';
  data := dados[1] + '#' + dados[2] +'#' + dados[3] + '#' + dados[4] +'#' +
dados[5] +'#' + dados[6] +'#' + dados[7] +'#' + dados[8] +'#' + dados[9] +'#' +
dados[10] +'#' {+ '#&'};
  clientsocket4.Socket.SendText(data);
  adoquery3.Next;
 end;
end;
end.
```

Os procedimentos apresentados a seguir diz respeito a aplicação cliente (aplicação do consultor) do Sistema de Análise Remota.

```
Unit cliente;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
 Dialogs, ScktComp, ComCtrls, StdCtrls, DB, ADODB, ExtCtrls, TeeProcs,
```

```
  TeEngine, Chart, TeeFunci, Series, DbChart, Grids, DBGrids, OleCtrls,
  Chartfx3, DBClient, Provider, Buttons;

type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    ServerSocket1: TServerSocket;
    DataSource1: TDataSource;
    ADOConnection1: TADOConnection;
    tablepontos: TADOTable;
    DBChart1: TDBChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    ADOQuery1: TADOQuery;
    BitBtn1: TBitBtn;
    Chat: TMemo;
    Label1: TLabel;
    ClientSocket1: TClientSocket;
    ServerSocket2: TServerSocket;
    ServerSocket3: TServerSocket;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    teste: TLabel;
```

```
tciclo: TLabel;
tcorte: TLabel;
sobremetal: TLabel;
energia: TLabel;
potmax: TLabel;
velomax: TLabel;
velomedia: TLabel;
nrebolo: TLabel;
part: TLabel;
BitBtn2: TBitBtn;
ServerSocket4: TServerSocket;
DataSource2: TDataSource;
tableteste: TADOTable;
tabletesteTest_code: TIntegerField;
tabletestePart_code: TIntegerField;
tabletesteCycle_time: TFloatField;
tabletesteCut_time: TFloatField;
tabletesteStock_part: TFloatField;
tabletesteEnergy: TFloatField;
tabletestePower_max: TFloatField;
tabletesteWheel_speed_Min: TFloatField;
tabletesteWheel_Speed_Avg: TFloatField;
tabletesteWheel_Index: TIntegerField;
BitBtn3: TBitBtn;
procedure FormCreate(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ServerSocket1ClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ServerSocket1ClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
```

```
    procedure BitBtn1Click(Sender: TObject);
    procedure ChatKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ServerSocket2ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocket3ClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocket3ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure BitBtn2Click(Sender: TObject);
    procedure ServerSocket4ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure BitBtn3Click(Sender: TObject);
  private
    { Private declarations }
    sMensagem: string;
    sdados, pulmao2: string;
    pulmao:string;
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}

{Código de ativação dos sockets servidores}
procedure TForm1.FormCreate(Sender: TObject);
begin
serversocket1.Active:=true;
serversocket2.Active:=true;
serversocket3.Active:=true;
```

```
end;

{Código de desativação dos sockets servidores}
procedure TForm1.FormDestroy(Sender: TObject);
begin
serversocket1.Active:= false;
serversocket2.Active:= false;
serversocket3.Active:= false;
end;

{Código para recebimento de gráficos e dados de ciclos advindos da aplicação servidora}
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
 Socket: TCustomWinSocket);
var
 i: integer;
 ind: byte;
begin
 smensagem:='';
 smensagem := {smensagem +}Socket.ReceiveText;
 tablepontos.open;
 tablepontos.Insert;
 ind:=0;
for i:= 1 to length(smensagem) do begin
 if (smensagem[i]<>'#'){And (smensagem[i]<>'&')} then
 pulmao := pulmao + smensagem[i]
 else
 begin
 inc(ind);
 if (smensagem[i]='#') and (ind = 1) then
 begin
  if {(pulmao<>'0') and} (pulmao <> '') then
```

```
 begin
  tablepontos.FieldValues['pontos']:=pulmao;
 end;
  pulmao:='';
end
else
begin
 if (smensagem[i]='#') and (ind = 2) then
 begin

 if {(pulmao<>'0') and} (pulmao <> '') then
 begin
  tablepontos.FieldValues['EA']:=pulmao;
 end;
 pulmao:='';
end
else
begin
 if (smensagem[i]='#') and (ind = 3) then
 begin
  if {(pulmao<>'0') and} (pulmao <> '') then
  begin
   tablepontos.FieldValues['LVDT']:=pulmao;
  end;
  pulmao:='';
  if i<>length(smensagem) then
   tablepontos.insert;
  ind:=0;
 end;
end;
end;
```

```
end;
end;

{Código de mensagens na barra de status}
procedure TForm1.ServerSocket1ClientConnect(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 StatusBar1.Panels[0].Text:= 'Conectado ao host ' + socket.RemoteAddress;
 sMensagem:='';
 pulmao:='';
if clientsocket1.active then clientsocket1.active:=false
 else begin
  clientsocket1.Address:=socket.RemoteAddress;
  clientsocket1.host:= Socket.RemoteHost;
  clientsocket1.active:=true;
 end;
end;

procedure TForm1.ServerSocket1ClientDisconnect(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 StatusBar1.Panels[0].Text:= 'Desconectado';
end;

{Código do botão que limpa o gráfico da tela}
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 tablepontos.Active:=false;
 tablepontos.Active:=true;
 tablepontos.first;
 while not(tablepontos.eof) do
 begin
```

```
  {tablepontos.Edit;}
  tablepontos.Delete;
 end;
 teste.Caption:='';
 part.Caption:='';
 tciclo.Caption:='';
 tcorte.Caption:='';
 sobremetal.Caption:='';
 energia.Caption:='';
 potmax.Caption:='';
 velomax.Caption:='';
 velomedia.Caption:='';
 nrebolo.Caption:='';

 tablepontos.Active:=false;
 tablepontos.Active:=true;
end;

{Código de implementação do chat}
procedure TForm1.ChatKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
 If Key = VK_Return then
   clientsocket1.socket.SendText(chat.Lines[chat.Lines.count-1]);
end;

procedure TForm1.ServerSocket2ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
chat.Font.Color:= clRed ;
chat.Lines.Add('['+ clientsocket1.Address + ' sent] '+ socket.ReceiveText );
chat.Font.Color:= clBlack ;
```

```
end;

procedure TForm1.ServerSocket3ClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  sdados:='';
  pulmao2:='';
end;
```

{Código para recebimento de gráficos e dados de ciclos advindos da aplicação servidora}

```
procedure TForm1.ServerSocket3ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
 i: integer;
 ind: byte;
begin
  sdados:='';
  sdados := sdados +  Socket.ReceiveText;
  ind:=0;
for i:= 1 to length(sdados) do begin
 if (sdados[i]<>'#')And (sdados[i]<>'&') then
  pulmao2 := pulmao2 + sdados[i]
 else
 begin
  if (sdados[i]='#') and (ind = 0) then
  begin
   inc(ind);
    if (pulmao2<>'0') and (pulmao2 <> '') then
    begin
     teste.Caption:=pulmao2;
     pulmao2:='';
```

```
 end;
end
else
begin
 if (sdados[i]='#') and (ind = 1) then
 begin
  inc(ind);
  if (pulmao2<>'0') and (pulmao2 <> '') then
  begin
   part.Caption:=pulmao2;
   pulmao2:='';
  end;
 end
 else begin
  if (sdados[i]='#') and (ind = 2) then
  begin
   inc(ind);
   if (pulmao2<>'0') and (pulmao2 <> '') then
   begin
    tciclo.Caption:=pulmao2;
    pulmao2:='';
   end;
  end
  else
  begin
   if (sdados[i]='#') and (ind = 3) then
   begin
    inc(ind);
    if (pulmao2<>'0') and (pulmao2 <> '') then
    begin
     tcorte.Caption:=pulmao2;
     pulmao2:='';
```

```
 {if i <> length(smensagem) then
   tablepontos.insert;}
 end;
end
else begin
 if (sdados[i]='#') and (ind = 4) then
 begin
  inc(ind);
  if (pulmao2<>'0') and (pulmao2 <> '') then
  begin
   sobremetal.Caption:=pulmao2;
    pulmao2:='';
  end;
 end
 else begin
  if (sdados[i]='#') and (ind = 5) then
  begin
   inc(ind);
   if (pulmao2<>'0') and (pulmao2 <> '') then
   begin
    energia.Caption:=pulmao2;
    pulmao2:='';
   end;
  end
  else begin
   if (sdados[i]='#') and (ind = 6) then
   begin
    inc(ind);
    if (pulmao2<>'0') and (pulmao2 <> '') then
    begin
     potmax.Caption:=pulmao2;
     pulmao2:='';
```

```
 end;
end
else begin
 if (sdados[i]='#') and (ind = 7) then
 begin
  inc(ind);
  if (pulmao2<>'0') and (pulmao2 <> '') then
  begin
   velomax.Caption:=pulmao2;
   pulmao2:='';
  end;
 end
 else begin
  if (sdados[i]='#') and (ind = 8) then
  begin
   inc(ind);
   if (pulmao2<>'0') and (pulmao2 <> '') then
   begin
    velomedia.Caption:=pulmao2;
    pulmao2:='';
   end;
  end
  else begin
   if (sdados[i]='#') and (ind = 9) then
   begin
    ind:=0;
    if (pulmao2<>'0') and (pulmao2 <> '') then
    begin
     nrebolo.Caption:=pulmao2;
     pulmao2:='';
    end;
   end;
```

```
            end;
           end;
          end;
         end;
        end;
       end;
        end;
       end;
      end;
     end;
    end;

    procedure TForm1.BitBtn2Click(Sender: TObject);
    begin
     apoio.show;
    end;

    {Código para recebimento de gráficos e dados de testes advindos da aplicação
servidora}

    procedure TForm1.ServerSocket4ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    var
     i: integer;
     ind: byte;
    begin
      smensagem:='';
      smensagem := Socket.ReceiveText;
      tableteste.open;
      pulmao:='';
      tableteste.Insert;
```

```
 ind:=0;
for i:= 1 to length(smensagem) do begin
 if (smensagem[i]<>'#'){And (smensagem[i]<>'&')} then
 pulmao := pulmao + smensagem[i]
else
begin
 inc(ind);
 if (smensagem[i]='#') and (ind = 1) then
 begin
  if {(pulmao<>'0') and }(pulmao <> '') then
  begin
   tableteste.FieldValues['test_code']:=pulmao;
  end;
 pulmao:='';
 end
 else
 begin
  if (smensagem[i]='#') and (ind = 2) then
  begin
   if {(pulmao<>'0') and }(pulmao <> '') then
   begin
    tableteste.FieldValues['Part_code']:=pulmao;
   end;
   pulmao:='';
  end
  else begin
   if (smensagem[i]='#') and (ind = 3) then
   begin
    if {(pulmao<>'0') and} (pulmao <> '') then
    begin
     tableteste.FieldValues['Cycle_time']:=pulmao;
    end;
```

```
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 4) then
begin
 if {(pulmao<>'0') and} (pulmao <> '') then
 begin
  tableteste.FieldValues['Cut_time']:=pulmao;
 end;
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 5) then
begin
 if {(pulmao<>'0') and }(pulmao <> '') then
 begin
  tableteste.FieldValues['Stock_part']:=pulmao;
 end;
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 6) then
begin
 if {(pulmao<>'0') and }(pulmao <> '') then
 begin
  tableteste.FieldValues['Energy']:=pulmao;
 end;
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 7) then
begin
```

```
 if {(pulmao<>'0') and} (pulmao <> '') then
 begin
  tableteste.FieldValues['Power_max']:=pulmao;
 end;
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 8) then
begin
 if {(pulmao<>'0') and} (pulmao <> '') then
 begin
  tableteste.FieldValues['Wheel_speed_min']:=pulmao;
 end;
 pulmao:='';
end
else begin
if (smensagem[i]='#') and (ind = 9) then
begin
 if {(pulmao<>'0') and} (pulmao <> '') then
 begin
  tableteste.FieldValues['Wheel_speed_avg']:=pulmao;
 end;
 pulmao:='';
end
else
begin
 ind:=0;
 if (smensagem[i]='#') and (ind = 0) then
 begin

  if {(pulmao<>'0') and} (pulmao <> '') then
  begin
```

```
        tableteste.FieldValues['Wheel_index']:=pulmao;

        if i <> length(smensagem)+1 then
          tableteste.insert;
       end;
       pulmao:='';
      end;
     end;
      end;
      end;
      end;
      end;
     end;
    end;
   end;
    pulmao:='';
  end;
end;
end;
end;
end.
```