SISTEMA EMBARCADO DE AQUISIÇÃO DE DADOS INTEGRADO COM A WEB

Franklin Silva de Souza

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO E ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Ramon Romankevicius Costa, D.Sc.

Prof. Liu Hsu, Docteur d'Etat

Profa. Marly Guimarães Fernandes Costa, D.Sc.

Prof. José Paulo Vilela Soares da Cunha, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2005

SOUZA, FRANKLIN SILVA DE

Sistema Embarcado de Aquisição de Dados Integrado com a Web [Rio de Janeiro] 2005

XXII, 269p., 29.7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese — Universidade Federal do Rio de Janeiro, COPPE

1. Sistema Dados Web.

I. COPPE/UFRJ II. Título (série).

Aos meus pais, José Lopes e Maria de Fátima,

e à minha família, Cátia e Ana Catharina, com amor.

Agradecimentos

Gostaria de demonstrar a minha imensa gratidão ao Prof. M. Sc. Manuel Cardoso por ter iluminado meus caminhos nos momentos mais difíceis.

Aos Profs. Dr. Cícero Costa e Dra. Marly G. Costa por terem coordenado nosso curso de Pós-Graduação.

À melhor equipe de trabalho da qual já participei, meus amigos do Nutelli, Universidade Federal do Amazonas, por sua determinação, troca de conhecimentos e paciência comigo.

Resumo da tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SISTEMA EMBARCADO DE AQUISIÇÃO DE DADOS INTEGRADO COM A WEB

Franklin Silva de Souza

Maio/2005

Orientador : Ramon Romankevicius Costa
Programa : Engenharia Elétrica

A inclusão de comunicação Ethernet TCP/IP em chão de fábrica é nova comparada aos protocolos padrões propostos pelo IEC 61158. Este trabalho apresenta o projeto e implementação de um sistema de aquisição de parâmetros de circuitos eletrônicos para produção e engenharia de qualidade e, ao mesmo tempo, usando protocolos Ethernet TCP/IP e linguagem XML disseminar os dados na Internet e enviar os mesmos para os Sistemas de Execução de Manufatura. Pela aplicação do protocolo *World Wide Web* - HTTP o usuário remoto pode monitorar o resultado e parâmetros dos circuitos eletrônicos, assim como o status dos dispositivos de teste com uso de um navegador Web. A arquitetura do sistema inclui um circuito de instrumentação e controle, unidade de comunicação e processamento (módulo microcontrolado Rabbit RCM3200 com controlador Ethernet AX8879ds) e um módulo eletromecânico. O sistema também incorpora um banco de dados local. A programação foi realizada com uso de linguagem C e um kernel preemptivo. Os resultados mostram que sistemas baseados em Ethenet TCP/IP para chão de fábrica, dependendo da aplicação, deixam de ser promessas e passam a ser empregados no mundo real.

Abstract of the Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

EMBEDDED ACQUISITION SYSTEM INTEGRATED WITH THE WEB

Franklin Silva de Souza

May/2005

Advisor : Ramon Romankevicius Costa
Department : Electric Engineering

The inclusion of Ethernet TCP/IP communication on the factory floor is innovative comparing with IEC 61158 Fieldbus standards. This work presents the design and implementation of a Web-based data parameters acquisition system of electronic boards from production and quality engineer and, at the same time, using Ethernet TCP/IP and XML language to disseminate data over the Internet and send the data to the Manufacturing Execution System. By applying the standard World Wide Web protocol - HTTP, the remote user may monitor production performance and modify control parameters with the general Internet browser. The architecture of the system includes a instrumentation and control board, processing and communication unit (Rabbit RCM3200 board microcontroller wih Ethernet AX8879ds). The system also incorporates a local data base. The programming language selected was C using a preemptive kernel. The tests results show that Ethernet TCP/IP based control system start to be real-world applications.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de abreviações

ANSI — *American National Standards Institute*

ARP — *Address Resolution Protocol*

ASCII —*American Standard Code for Information Interchange*

AS-i — *Actuator Sensor interface*

B2MML — *Business To Manufacturing Markup Language*

CAN — *Controller Area Network*

CLP — *Controlador Lógico Programável*

CRC — *Cyclic Redundancy Check*

CSMA/CD — *Carrier Sense Multiple Access with Collision Detection*

DCS — *Distributed Control System*

DDC — *Direct Digital Control*

DHCP — *Dynamic Host Configuration Host Protocol*

DIC — *Dispositivo Inteligente de Campo*

DNS — *Domain Name System*

DOM — *Document Object Model*

DSSS — *Direct Sequence Spread Spectrum*

DTD — *Document Type Definition*

E/S — Entradas e Saídas Digitais

FCS — *Field Control System*

FF — *Foundation Fieldbus*

FF HSE — FF *High Speed Ethernet*

FHSS — *Frequency Hopping Spread Spectrum*

FISCO — *Fieldbus Intrinsically Safe COncept*

FTP — *File Transfer Protocol*

GML — *Generalized Markup Language*

GPRS — *General Packet Radio Service*

GSM — *Global System for Mobile Communications*

HTML — *HyperText Markup Language*

HTTP — *Hypertext Transfer Protocol*

ICMP — *Internet Control Message Protocol*

IEC — *International Electrotechnical Commission*

IGMP — *Internet Group Management Protocol*

IHM — *Interface Homem Máquina*

IP — *Internet Protocol*

ISA — *Instrumentation, Society and Automation*

ISR — *Industrial Scientific Medical*

ISR — *Interrupt Service Routine*

JVM — *Java Virtual Machine*

LAN — *Local Area Network*

LONWorks — *Local Operating Network*

MAC — *Medium Access Control*

MBP-IS — *Manchester Coded, Bus Powered - Intrinsically Safe*

MES — *Manufacturing Execution Systems*

NUTELI — Núcleo de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e de — Informação da Universidade Federal do Amazonas

OCQ — Operação de Controle de Qualidade

OLE — *Object Linking and Embedding*

OLM — *Optical Link Modules*

OPC — OLE *for Process Control*

OTP — *One Time Programmable*

PDA — *Personal Digital Assistant*

PID — Proporcional, Integral e Derivativo

PROFIBUS — *Process field-bus*

PROFIBUS DP — *Profibus Descentralized Periphery*

PROFIBUS FMS — *Profibus Fieldbus Message Specification*

PROFIBUS PA — *Profibus Process Automation*

RAM — *Random Access Memory*

RAS — *Remote Access Service*

ROM — *Read Only Memory*

RTOS — *Real-Time Operating System*

SAX — *Simple API for XML*

SMTP — *Simple Mail Transfer Protocol*

SNVT — *Standard Network Variable Types*

SQC — *Statistics Quality Control*

SQL — *Structured Query Language*

SS — *Spread Spectrum*

STU — Sistema de Testes Unificado

TCP — *Transmission Control Protocol*

UCP — Unidade Central de Processamento

UDP — *User Data-gram Protocol*

UFAM — Universidade Federal do Amazonas

UML — *Unified Modeling Language*

WML — *Wireless Markup Language*

WWW — *World Wide Web*

XML — *eXtensible Markup Language*

XSL — *eXtensible StyleSheet Language*

XSLT — *XSL Transformations*

ZFM — Zona Franca de Manaus

Capítulo 1

Introdução

1.1 Motivação

O pólo industrial da Zona Franca de Manaus (ZFM) vem produzindo produtos eletro-eletrônicos desde a sua criação, em 1967. Com o advento da globalização, a competitividade passou a ser um dos principais indicadores para a permanência ou não de uma empresa no mercado. Requisitos como custo, produtividade e qualidade estão mais rígidos. As fábricas devem assegurar uma qualidade consistente, grande eficiência, baixo custo de produção e flexibilidade para novos produtos. Os novos produtos lançados a cada ano no mercado requerem índices de qualidade elevadíssimos, alguns chegando à menos que 5 falhas em cada mil peças produzidas, ou seja, menos que 0,5% da produção pode ser rejeitada.

O sistema atualmente utilizado para a realização de testes em muitas fábricas no distrito industrial da ZFM pode ser visualizado no esquema conforme Figura 1.1.

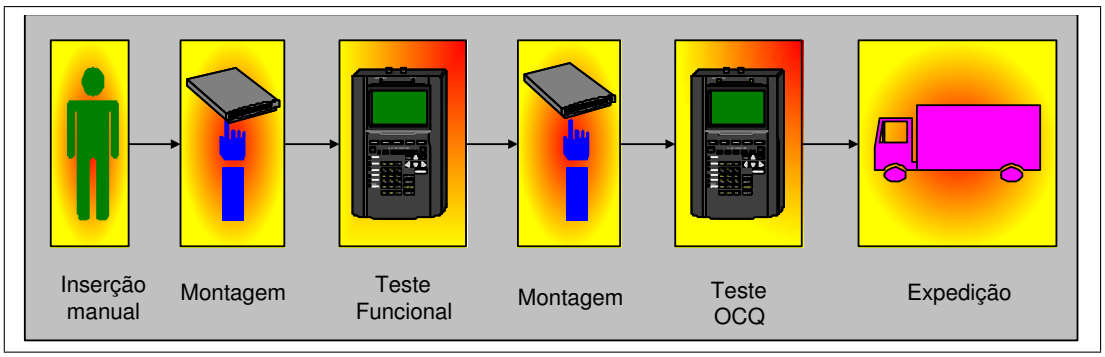


Figura 1.1: Processo de montagem típico na ZFM.

Numa primeira etapa é feita a montagem manual de determinada parte do produto, pois caso ocorram defeitos nesta fase, parte do que foi montado pode ser reaproveitado. Após o teste funcional desta primeira etapa, novos acessórios são adicionados ao produto, normalmente em relação à aparência final do mesmo. Em seguida uma nova Operação de Controle de Qualidade (OCQ) será realizada e finalmente o produto será embalado para expedição.

Desde a última década, não só as empresas da ZFM como outras empresas têm se empenhado em melhorar os procedimentos de qualidade. Filosofias e técnicas têm sido definidas para alcançar estes aperfeiçoamentos. Para atingir tais metas com sucesso, os pontos críticos do processo devem ser cuidadosamente monitorados através de análises do desempenho do processo usando métodos estatísticos de controle de qualidade (*Statistics Quality Control* (SQC)). As medidas requeridas para implementação com êxito do SQC requerem instrumentação mais sofisticada que simples circuitos de aferição e seleção tais como "GO/NO-GO" (aprovado ou não aprovado). Além da checagem dos limites, a instrumentação necessita fornecer as medidas que serão utilizadas para cálculo estatístico do SQC (HELPS & HAWKS 1999) , assim como também, fornecer o estado dos dispositivos de testes (DTs) (produzindo, parado ou em manutenção) e tempo de ciclo de produção de cada produto testado para efeitos de verificação do desempenho da linha de produção.

Tecnologias emergentes, tais como, acesso móvel nas telecomunicações, redes de comunicação a baixo custo e redução no custo do processamento e armazenamento das informações digitais têm revolucionado o modo das organizações e indivíduos pensarem sobre captura, transmissão e uso da informação. As empresas estão reconhecendo que os sistemas de aquisição de dados podem ajudar a integrar o controle do processo (chão de fábrica) com o sistema de informação gerencial. Neste momento, há uma grande oportunidade para os fabricantes terem melhor entendimento e controle de seus processos. Para otimizar a troca de informações nas empresas, sistemas de informações distribuídas, isto é, controle e medida, estão sendo rapidamente desenvolvidos. Baseado no *Fieldbus*, ou barramentos de campo, as empresas planejam ou otimizam sistemas de medidas desde suas especificações até funcionalidades. O *Fieldbus* é aceito pela indústria assim como nós aceitamos a *Internet* (TIAN 2001).

A tecnologia *Fieldbus* trás muitos outros benefícios além da redução dos custos

de fiação. Entre eles, podemos citar:

- **Funções descentralizadas.** Inclui mais controle nos instrumentos de campo, pois é possível a transmissão de múltiplas variáveis em um simple par de cabo, reduzindo o processamento nos computadores de controle.
- **Configuração e manutenção simplificada.** A disseminação de métodos de diagnósticos, inclusive manutenção pró-ativa, determinam exatamente o momento em que a planta necessitará de manutenção.
- **Gerenciamento de ativos.** Todas as informações dos dispositivos de campo podem ser vistas da sala de controle.
- **Confiabilidade.** Além dos sinais digitais serem mais robustos que os sinais analógicos, a introdução de técnicas de checagem de *frame* tornam a ocorrência de erros muito pequena.
- **Interoperabilidade.** Permite que sistemas compatíveis passem a trocar informações entre si.
- **Flexibilidade.** Adaptam-se ao tamanho, configuração ou aplicação do sistema.
- **Expansibilidade.** Facilitam expansão e re-configuração.

Tecnicamente, o *Fieldbus* deveria ser uma solução padronizada se não fossem os interesses políticos e comerciais envolvidos. Existem atualmente mais de 50 diferentes *Fieldbus* disponíveis tais como Foundation Fieldbus, Profibus, Interbus, WorldFip, Lonworks, CAN, DeviceNet, SwiftNet e P-net. Embora o *International Electrotechnical Commission* (IEC) tenha aprovado a proposta para padronização internacional dos barramentos de campo no ano de 1999, atualmente compreende oito tipos diferentes de protocolos (YABIN & NISHITANI 2003). Porém, requerimentos de interoperabilidade em um ambiente industrial integrado, o qual provavelmente consiste de *Fieldbus* heterogêneos e LANs convencionais, ainda está longe de ser alcançado, devido a sérios problemas de tecnologia, tais como incompatibilidade de interface de programação e dependência da plataforma dos equipamentos (KOULAMAS, ORPHANOS, KOUBIAS & PAPADOPOULOS 1999).

Nos sistemas de gestão corporativa, as *Local Area Networks* (LANs) na maior parte baseada em rede *Ethernet* e protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP) são usadas para interconectar diferentes sistemas de barramento de campo.

Portanto, o mapeamento entre as informações do *Fieldbus* para as LANs de gestão corporativa tem sido amplamente discutido. Nos últimos anos, com o desenvolvimento da Tecnologia de Informação (TI) o foco deixou de ser a sedimentada tecnologia de controle e supervisão e passou a ser a disponibilização das informações aos usuários. A combinação do *Fieldbus* com *Ethernet* TCP/IP é a chave para o sucesso do processo de integração. Integração *Web* é uma decisão direcionada a protocolos abertos e modelos padronizados, ou seja, tecnologia não proprietária. Contudo, é necessário tomar cuidados com esta tecnologia com respeito aos ambientes aonde ela pode ser usada. A segurança é freqüentemente o principal obstáculo (WOLLSCHLAEGER 2001).

Soluções baseadas em *Web* são disponíveis para uma ampla faixa de aplicações, por exemplo, em Interfaces Homem Máquina (IHM) e ferramentas de engenharia. A tecnologia *Web* é também importante para a chamada "integração vertical", preenchendo o vácuo entre os sistemas de chão de fábrica e os sistemas no mundo empresarial. O objetivo é levar as informações do chão de fábrica para sistemas baseados em navegadores *Web* que se encontram no nível corporativo. Temas de discussão atuais são: independência de plataforma, reduzida necessidade de pré-requisitos de *software* e *hardware*, acesso mundial, ilimitado número de usuários e tecnologia já aceita (WOLLSCHLAEGER 2001) e (WOLLSCHLAEGER, NEUMANN & BANGEMANN 2002).

Tecnologias *Web* influenciam todas as aplicações e sistemas de comunicação tanto direta como indiretamente. Diretamente significa que as tecnologias são usadas como parte integral da aplicação. Não obstante, uma influência indireta pode ser mais importante. Uma das mais promissoras tecnologias *Web* é o uso de linguagem de descrição de dados *eXtensible Markup Language* (XML) (GRAVES 2003) e (DEITEL, DEITEL & NIETO 2003) para definição de interfaces. A seguir relacionamos as vantagens no uso da linguagem XML:

- Os dados contidos em um arquivo XML podem ser transformados para outros formatos através da utilização de uma folha de estilo (XSLT). Exemplos típicos são *HyperText Markup Language* (HTML) para navegadores *Web* ou *Wireless markup Language* (WML) para navegadores de um telefone celular *Global System for Mobile Communications* (GSM).

- Soluciona o problema de integração entre diferentes protocolos proprietários industriais e diferentes sistemas em uma única interface de troca de dados, ou seja, uma estrutura de gerenciamento genérica. Deste modo, resolve os seguintes problemas:

  - Método de troca de dados entre a interface *Fieldbus* e o servidor *Web* como exemplo de integração;
  - Projeto do *front-end* nas funções de gerenciamento;
  - Gerenciamento dos componentes de *software*.

- Há programas de análise gramatical XML (*parsers*) que têm códigos abertos e disponíveis sem custo para um grande número de linguagens de programação como C, C++, Perl ou Java. Os desenvolvedores de produtos têm o esforço radicalmente reduzido na implementação destes mecanismos e funções. Como exemplo, transportar os dados de um arquivo XML para uma estrutura em linguagem C e vice-versa.

- A troca de dados entre diferentes aplicações é o primeiro passo no processo de integração *Web*. Outra tarefa mais importante é interpretar as informações dos dispositivos e sistemas. Para termos êxito em um *Fieldbus* genérico e independente há a necessidade de se padronizar a descrição XML. Padrões derivados da linguagem XML, como a *Business To Manufacturing Markup Language* (B2MML) implementam modelos de padronização para os dados dos protocolos de barramento proprietários (WOLLSCHLAEGER 2001), (WOLLSCHLAEGER et al. 2002), (WBF 2003*a*), (WBF 2003*b*) ,(BERGE 2002) e (KOULAMAS et al. 1999).

Com a aplicação das tecnologias *Web* as fábricas do distrito industrial da ZFM poderão fornecer em tempo real para qualquer parte do mundo:

- Parâmetros de projetos dos semicondutores para os desenvolvedores dos mesmos;
- Parâmetros dos lotes de semicondutores para os fabricantes dos mesmos;
- Dados das linhas de produção para as matrizes das fábricas. Estas verificarão a eficiência das mesmas e tomarão decisões de planejamento a produção mundial.

Um modelo de arquitetura genérica em sistemas de controle de processo integrada com a *Web* pode ser vista na Figura 1.2. Ela consiste de três camadas. A camada mais baixa contém o sistema de comunicação de chão de fábrica (*Fieldbus*) e dispositivos de controle e automação (componentes *Fieldbus*). A camada intermediária contém o servidor *Web*, o qual armazena informações para os clientes na camada superior. É freqüentemente denominado de portal, pois mapeia os protocolos proprietários para o protocolo *Hyper Text Transfer Protocol* (HTTP) (WOLLSCHLAEGER et al. 2002) e (WOLLSCHLAEGER 2001). A camada superior, contém os sistemas de gestão corporativa *Enterprise Resource Planning* (ERP) e *Manufacturing Execution System* (MES).

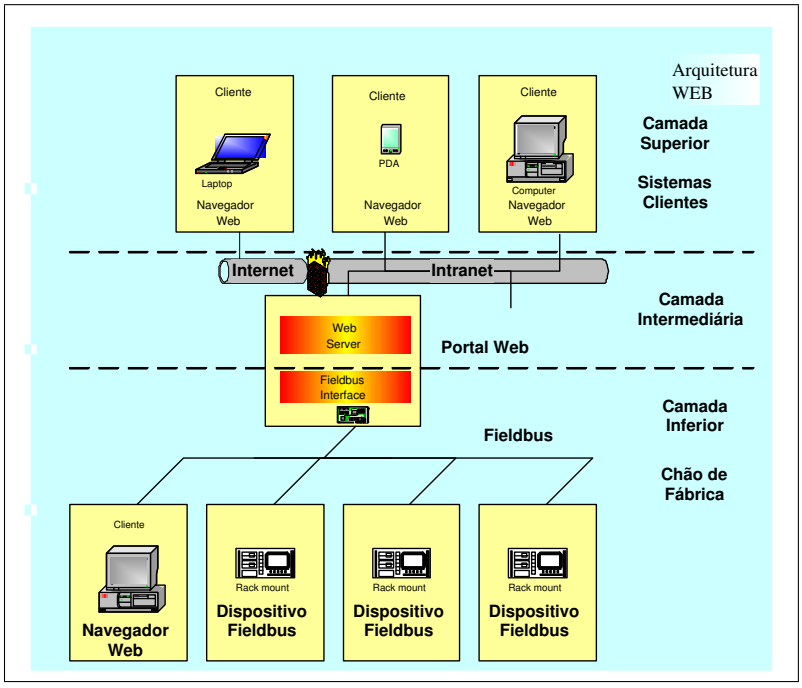


Figura 1.2: Arquitetura genérica de integração *Web*.

A arquitetura genérica de integração *Web* resolve o problema de integração entre diferentes *Fieldbus*. Porém, no caso do desenvolvimento de uma nova aplicação seria melhor, na medida do possível, utilizar as mesmas tecnologias *Web* padronizadas no nível corporativo no chão de fábrica também. Deste modo, temos uma nova arquitetura de integração *Web* modificada conforme Figura 1.3. As seguintes vantagens seriam obtidas com o uso desta arquitetura modificada:

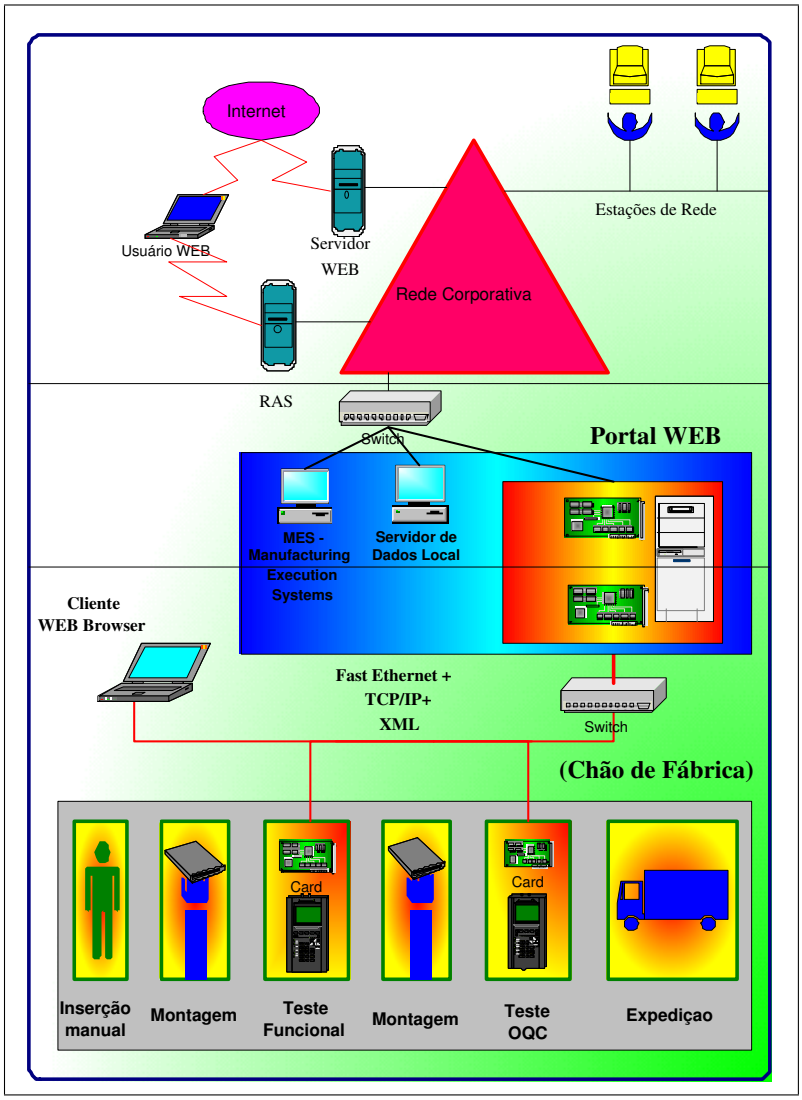


Figura 1.3: Arquitetura de integração *Web* proposta.

- Substituição do *Fieldbus* pela *Fast Ethernet*, a LAN mais utilizada no mundo;
- Eliminação da interface *Fieldbus* no Portal *Web*, tanto em relação ao *hardware* quanto ao *software*;

- Substituição dos dispositivos *Fieldbus* por dispositivos que trabalhem com as mesmas tecnologias *Web* do sistema corporativo;
- O uso de linguagens como XML nos dispositivos de campo permite a geração de arquivos de diferentes formatos de saída e a estreita relação com outras tecnologias *Web* oferece uma nova qualidade nos cenários de automação e controle (WOLLSCHLAEGER 2001);
- Padronização dos dados pelo uso de umas das linguagens derivadas da XML, como a B2MML (WBF 2003*a*) e (WBF 2003*b*);
- Interoperabilidade entre os dispositivos de campo;
- Disponibilidade de protocolos proprietários que ofereçam interface *Ethernet* para chão de fábrica;
- Obtenção de uma conectividade sem precedentes. A integração *Web* é uma decisão direcionada a protocolos abertos e modelos padronizados, ou seja, tecnologia não proprietária.
- Com dados provenientes dos Dispositivos Inteligentes de Campo (DIC) o *Manufacturing Execution Systems* (MES) poderá executar funções como: centralização dos dados de produção, acompanhamento da produção, controle de paralisações e disponibilização de dados do controle de qualidade (PADRÃO Jr, FILHO & LANNA 2002);
- Visualização dos dados de produção em qualquer computador que possua uma conexão TCP/IP e um navegador *Web*;
- Atualização de programas nos dispositivos de campo usando o protocolo *File Transfer Protocol* (FTP);
- Monitoração remota.

Por outro lado, o uso desta arquitetura acarreta algumas desvantagens. Como exemplo, podemos citar:

- Novo patamar no desenvolvimento de dispositivos de campo, pois devem suportar implementação de tecnologias *Internet* e *Web*;

- *Overhead* das mensagens XML na rede *Ethernet* TCP/IP;
- Necessidade de maior capacidade de computação para escrita e leitura dos arquivos XML, tanto no Portal *Web* quanto nos DICs;
- O alcance entre nós é de no máximo 100m, há necessidade de uso de *Ethernet switches* que operem em condições extremas iguais aos dos CLP's industriais;
- Questões de segurança relativas ao acesso as informações;
- Áreas intrinsecamente seguras não podem ser beneficiadas com o uso da *Ethernet* pois esta não pode transmitir dados e energia simultaneamente no mesmo cabo.

Certos cuidados devem ser tomados no uso da arquitetura *Web* (POZZUOLI 2003). São eles:

1. Verificar se a rede *Ethernet* TCP/IP satisfaz os requisitos da aplicação com relação ao volume de dados a serem transmitidos;
2. Verificar desempenho determinístico para aplicações que exigem tempo real;
3. Garantir redundância da rede;
4. Respeitar a faixa operacional de temperatura em ambientes industriais;
5. Utilizar cabos e conectores industriais (CAT5E);
6. Verificar imunidade a interferências eletromagnéticas;
7. Avaliar as questões de segurança.

Desde que há um aumento de desempenho nos modernos microcontroladores e microprocessadores e das redes baseadas em *Ethernet*, os problemas causados pelo uso da linguagem XML devem ter uma influência cada vez menor no futuro (WOLLSCHLAEGER et al. 2002).

Os dispositivos de testes (DTs) dos produtos (normalmente placas eletrônicas) localizados no teste funcional e teste de OCQ, conforme Figura 1.1, são específicos

para cada indústria. Não há no mercado DTs disponíveis para testes de displays de celulares, televisões, etc. Há bastante tempo eles têm sido desenvolvidos pela Engenharia dos próprios fabricantes. Na maior parte são equipamentos que trabalham sem comunicação e não exteriorizam os tipos de defeitos ocorridos nos produtos durante os testes. Devido à ausência de comunicação e instrumentação de precisão, necessários à aplicação dos algoritmos SQCs, fica inviável o aproveitamento destes DTs para controle do processo produtivo. No projeto de um novo DT, deve-se ter em mente as seguintes diretivas:

- Prever capacidade de interligação dos DTs em rede (barramento de campo);
- Os circuitos eletrônicos usados na instrumentação de precisão devem satisfazer o grau de confiabilidade exigido pelo SQC;
- Prever funções de auto-diagnose para indicação de manutenção proativa;
- Prever flexibilidade para testes de produtos semelhantes;
- Prever a possibilidade de monitoração remota.

No mundo do controle de processo, quando falamos em instrumentos baseados em tecnologia digital, ou seja, com plataforma de *hardware*, sistema operacional, comunicação em rede e informações de diagnósticos, faz mais sentido referir-se aos DTs como Dispositivos Inteligentes de Campo (DICs)((PINCETI 2004)). São vantagens dos DICs:

- Integração de múltiplos sensores e atuadores;
- Integração de diferentes subsistemas;
- Flexibilidade em implementar várias estratégias de controle;
- Capacidade de executar múltiplas tarefas complexas de controle e medida;
- Capacidade de interface com outros sistemas;

- Fornecem dados para o *Manufacturing Execution Systems* (MES) executar funções como: centralização dos dados de produção, acompanhamento da produção, controle de paralisações e disponibilização de dados do controle de qualidade (PADRÃO Jr et al. 2002);

Sistemas de controle e medida modernos integram dispositivos de campo e dispositivos de controle. O *Fieldbus* é o canal de troca de dados entre DICs e dispositivos de controle. A verdadeira tecnologia digital inteligente executa tarefas não somente em dispositivos específicos, mas sim as divide entre dispositivos inteligente de campo que são conectados através do *Fieldbus*. O *Fieldbus* é muito mais que um mero *link* de comunicação. Definem arquitetura de sistemas e são um dos fatores que determinam o desempenho das fábricas. Asseguram funções e benefícios que não eram disponíveis previamente.

1.2 Objetivos

Este trabalho em automação industrial propõe desenvolver um Dispositivo Inteligente de Campo (DIC) para aquisição de dados de engenharia e produção do chão de fábrica a ser usado em Processos Produtivos nas fábricas do Distrito Industrial da ZFM com as seguintes características principais:

- Instrumentação de precisão para atender o SQC;
- Operar na arquitetura de integração *Web* modificada conforme Figura 1.3;
- Interface *Fast Ethernet*;
- Uso dos protocolos TCP/IP para conexão *Internet*;
- Uso da linguagem XML, especificamente B2MML para encapsulamento dos dados obtidos nos testes, dados de configuração e operação dos DICs;
- Banco de dados local para armazenamento de informações numa eventual falha da rede;
- Capacidades de auto-diagnóstico;

- Flexibilidade para testar produtos semelhantes;
- Monitoração remota.

A implementação da arquitetura de integração *Web* modificada permite qualquer computador visualizar os dados de produção usando uma conexão TCP/IP, ou até mesmo modificar os programas dos DICs usando o protocolo *File Transfer Protocol* (FTP). A grande diferença da arquitetura proposta para a arquitetura genérica de integração *Web* é que não necessitamos desenvolver as interfaces *Fieldbus*, tanto a nível de *hardware* quanto de *software*, simplificando o desenvolvimento do sistema como um todo, principalmente do portal *Web*. A razão é devido ao uso de protocolos abertos desde o nível o chão de fábrica até ao nível corporativo, alcançando conectividade sem precedentes.

1.3 Etapas

As seguintes etapas serão desenvolvidas para atingir os objetivos citados no item 1.2:

1. Levantamento de requisitos dos usuários;
2. Especificação de requisitos;
3. Definição da arquitetura do sistema;
4. Escolha da Unidade Central de Processamento (UCP);
5. Escolha da linguagem de programação;
6. Implementação das camadas de acordo com o modelo *Internet*;
7. Projeto do *hardware* do DIC;
8. Desenvolvimento do *software*:
    - Diagramas de caso de uso;
    - Diagramas de seqüência;

- Máquina de estado;

9. Integração do sistema e testes;
10. Análise dos benefícios alcançados oriundos das informações dos DICs.

1.4 Metodologia

Para realização deste trabalho a seguinte metodologia foi utilizada:

1. Visita a várias indústrias do complexo da ZFM para conhecimento dos seus processos produtivos;
2. Estudo da comunicação digital no mundo empresarial, pois é objetivo principal deste trabalho a implementação do sistema de aquisição de dados visando principalmente a integração vertical (chão de fábrica para o mundo corporativo), utilizando tecnologias de amplo conhecimento e baixo preço. Tecnologias de amplo conhecimento implementam uma rápida curva de aprendizado e baixos preços aumentam o poder de competitividade das empresas. Primeiramente serão estudados tipos, topologias, etc. de redes do mundo corporativo e protocolos *Ethernet* e TCP/IP. Componentes *Ethernet*, como *hubs* e *switches* também são objeto de estudo, pois são os elos de ligação entre os nós de uma rede *Ethernet* (SCRIMGER, LASALLE, PARIHAR & GUPTA 2002).
3. Estudo da comunicação digital no chão de fábrica (BERGE 2002), pois é neste ambiente que serão coletados nossos dados. Verificar na literatura a existência de muitos protocolos de comunicação.
4. Realização de pesquisas em artigos sobre protocolos de comunicação (HELPS & HAWKS 1999).
5. Levantamento das principais características dos protocolos de comunicação industriais nas páginas Web das organizações responsáveis pelos mesmos.
6. Pesquisa em revistas técnicas para a obtenção de exemplos de implementações de sistemas de automação que utilizem banco de dados, e a verificação da ne-

cessidade de utilização de mais de um protocolo de comunicação para atender todos os requisitos de produção.

7. Verificação do problema de interoperabilidade na eventual utilização de vários protocolos proprietários e a possibilidade de usar protocolos encapsulados com *Ethernet* TCP/IP e *Control and Information Protocol* (CIP) (BROOKS 2001), (MUSKINJA, TOVORNIK & TERBUC 2003) e (FELSER & SAUTER 2002).

8. Estudo do desenvolvimento de um DIC para um Sistema de Aquisição de Dados baseado no conceito de integração *Web* (BROOKS 2001) e (SCHNEIDER 2003).

9. Pesquisa de um microcontrolador, microprocessador, módulo processado, Controlador Lógico Programável (CLP) ou PC Industrial que atenda aos requisitos de comunicação *Ethernet*, desempenho e custo.

10. Avaliação de linguagens de programação que possam ser implementadas no *hardware* escolhido (HOLZNER 2002).

11. Planejamento da infraestrutura de automação em relação a documentação de cabos, dispositivos, conexão de dispositivos, roteadores e *switches* (MORAES 2004) e (BERGE 2002).

1.5 Organização desta tese

O trabalho está organizado em mais nove capítulos. Um breve resumo é apresentado a seguir.

O Capítulo 2 revisa a comunicação digital e redes (BERGE 2002), (SCRIMGER et al. 2002) e (MORAES 2004). São descritos tipos de topologias, meios de transmissão e outros tópicos utilizados nos barramentos de campo. As redes tornaram possível coletar mais informações na planta e disseminá-las em toda a empresa. As redes tornaram-se o elo de comunicação entre os sistemas de computação e dispositivos de entrada e saída de dados. As redes mudaram completamente a organização das fábricas.

O Capítulo 3 descreve a tecnologia *Fieldbus*, ou seja,sistemas de comunicação industriais conectados por meios físicos, como cabos de cobre, fibras ópticas ou *wireless*, com transmissão serial para acoplar dispositivos de campo distribuídos (sensores, atuadores, drivers, transdutores, etc.) para um controle central ou sistema de gerenciamento, formando um sistema de automação. O *Fieldbus* é o sistema de comunicação mais utilizado em automação e controle. Neste Capítulo apresenta-se:

- Os principais requisitos que os *Fieldbus* devem satisfazer para atender as necessidades de um ambiente industrial (YABIN & NISHITANI 2003);
- A necessidade de padronização a fim de que sejam evitadas as famosas "ilhas de automação" (FELSER & SAUTER 2002) e (WOOD 2000);
- Os conceitos de interoperabilidade, tipos de arquitetura e modelo OSI da ISO (MUSKINJA et al. 2003) a fim de que sejam evitados problemas de interconexão entre dispositivos de diferentes fabricantes;
- O modelo de estrutura organizacional denominado "pirâmide organizacional" implementada pelos *Fieldbus* (PROFIBUS 2002) assim como um novo modelo denominado "rede corporativa";
- As diversas áreas de aplicação da automação a fim de facilitar a escolha do melhor barramento específico (BERGE 2002);
- A evolução das arquiteturas, desde o "Direct Digital Control" (DDC) até a arquitetura "Field Control System" (FCS), com objetivo de verificar as melhorias oriundas do processamento distribuído e a redução da vulnerabilidade a falhas (BERGE 2002);
- As funções implementadas nos DICs herdadas dos barramentos de campo e do desenvolvimento da micro-eletrônica, como a autodiagnose que tornam os sistemas mais robustos (MULLER, EPPLE, WOLLSCHLAEGER & DIEDRICH 2003).

O Capítulo 4 apresenta um resumo das principais características de alguns protocolos proprietários e há um quadro comparativo a fim de auxiliar uma escolha entre

eles. O conhecimento das características dos protocolos definirá se determinado protocolo atende às especificações de projeto, como por exemplo, taxa de transmissão de rede, números de nós que podem ser conectados, compatibilidade com outros protocolos, segurança intrínseca, capacidade de redundância, interligação com *Ethernet*, etc.

O Capítulo 5 trata dos servidores *Web*. O uso de conceitos baseados na *Web* em automação industrial está crescendo rapidamente. Este Capítulo mostra motivações, conceitos, pré-requisitos e detalhes de implementação de uma integração *Web* (WOLLSCHLAEGER 2001) e (WOLLSCHLAEGER et al. 2002). O *Fieldbus* foi uma revolução nos sistemas de automação, mas trouxe muitos problemas em relação à padronização, e até hoje, mesmo com o uso da *Ethernet*, o formato das mensagens continua proprietário. As arquiteturas abertas estão sendo inevitavelmente implementadas no chão de fábrica.

O Capítulo 6 apresenta conceitos e características de CPUs que podem ser implementados no chão de fábrica, desde os pequenos microcontroladores, passando-se pelos CLPs até chegar aos PCs industriais (SOUZA & MATA 2004). A escolha do *hardware* adequado deve levar em consideração o custo, tamanho, área de atuação e poder de processamento.

O Capítulo 7 aborda a escolha das linguagens de programação. Veremos que nem sempre a escolha da linguagem de programação está baseada em critérios técnicos (NAIDITCH 1998). As características das linguagens são estudadas brevemente (MAURER 2002). Após análise não técnica duas linguagens foram selecionadas: C e Java. Após estudo comparativo (MOREIRA, MIDKIFF & GUPTA 1998) verifica-se que Java seria a linguagem ideal, mas apresenta problema de performance. A linguagem C, por ser mais rápida, foi escolhida por testar mais aparelhos no mesmo intervalo de tempo. Mas isto não significa que Java não possa ser usado em projetos futuros (HARDIN 2000) após serem resolvidos alguns problemas de desempenho (HIGUERA-TOELDANO & ISSARNY 2000) e (BOLLELLA & GOSLING 2000). É fornecida uma introdução a linguagem XML, pois a mesma facilita o processo de integração *Web* (DEITEL et al. 2003).

O Capítulo 8 estuda os critérios (OLSON 2000) para seleção de um banco de

dados para sistemas embarcados (GRAVES 2003). Segundo (OLSON 2000), embora os sistemas embarcados compartilhem muitas características com computadores pessoais eles apresentam limitações. Os requisitos das aplicações que eles executam necessitam de um cuidadoso processo de seleção e implementação sob medida. Em nosso estudo de caso, as linhas de produção devem continuar seu trabalho, mesmo com eventual queda da rede de comunicação dos DICs para o servidor corporativo de banco de dados. Então existe a necessidade de armazenar estas informações por algumas horas nos DICs.

O Capítulo 9 apresenta um estudo de caso: desenvolvimento de um sistema de testes para displays de cristal líquido de celulares a ser utilizado na Empresa Philips MDS. Primeiramente são levantados os requisitos dos usuários através de uma análise textual. Com estes requisitos definimos a especificação de requisitos. Em seguida escolhemos a arquitetura do sistema para posterior escolha de *hardware* e *software*. Apresentamos o protótipo do sistema. Finalizando, comentamos os benefícios alcançados com a implantação do sistema.

O Capítulo 10 apresenta conclusões deste trabalho e recomendações para trabalhos futuros.

Capítulo 2

Redes de comunicação digital

2.1 Introdução

Os primeiros controladores de processo eram pneumáticos. Em seguida foram criados os sistemas de controle predominantemente analógicos, simples dispositivos com sinais formatados de acordo com a arquitetura com uso mínimo de unidades centrais de processamento (UCPs). As redes foram introduzidas na automação industrial nos anos 70 e primeiramente usadas no Controle Digital Direto (DDC) entre os sistemas de computação e dispositivos de entradas e saídas (E/S). Mais tarde, foram usadas em Sistemas de Controle Distribuído (DCS) e sistemas de lógica programável para conectar controladores e Interfaces Homem Máquina (IHMs). Contudo, a comunicação digital em pequenos dispositivos como transmissores no chão de fábrica só foi efetivamente utilizada a partir dos anos 80 e barramentos de campo só foram amplamente aceitos nos anos 90 (BERGE 2002) (MORAES 2004).

No outro extremo, no nível corporativo, as corporações conectam todo o ambiente de produção do chão de fábrica das filiais à matriz através da *Internet*. Coordenação de produção e vendas devem trabalhar juntas para estruturar a tecnologia de informação (aplicação da tecnologia no processamento de informações). As redes de comunicação possibilitaram coletar mais informações dos transmissores e receptores, ou seja, mais informações da planta e disseminá-las na empresa.

A união de redes de transmissão de dados em banda larga com baixo custo, instrumentação de precisão com uso de interface *Ethernet* e programa de reconhecimento

de padrões tornam viável economicamente realizar o monitoramento de equipamentos e estruturas a longa distância, aumentando a confiabilidade das fábricas (www.pasa.com.br).

Dispositivos inteligentes de campo devem trabalhar em conjunto com as redes para aumentar a confiabilidade das fábricas. As redes de comunicação tornaram-se essenciais para a automação e mudaram completamente a organização das empresas.

Neste capítulo, o item 2.2 descreve as vantagens das redes de comunicação digital sobre os tradicionais sistemas 4-20mA utilizados no chão de fábrica, o item 2.3 descreve as mídias de transmissão, ou seja, os canais usados para a realização da comunicação de dados. A seção 2.4 apresenta as várias classificações das redes em função do fluxo de sinais, do tipo de transmissão, controle de acesso ao meio físico e relacionamentos de comunicação.

2.2 Vantagens da comunicação digital

Muitas redes de comunicação são analógicas, como rádio e televisão, mas a tendência definitiva é torná-las redes de comunicação digital. As redes usadas na automação também são digitais, isto é, os dados são transmitidos na seqüencia de "1s" e "0s". Com a comunicação digital é possível transferir dados não somente dos controladores para os dispositivos de campo ou vice-versa, mas entre os dispositivos de campo também, é o verdadeiro controle distribuído.

Há uma série de vantagens no uso das redes de comunicação digital sobre os tradicionais sistemas analógico, como por exemplo o laço de corrente de 4 a 20mA utilizado no chão de fábrica. São elas:

- **Redução da quantidade de cabos**. Uma das grandes vantagens das redes de comunicação digital sobre os tradicionais sistemas analógicos 4-20 mA é a capacidade de conectar muitos dispositivos de campo em um único par de cabos (Figura 2.1) .Isto reduz a quantidade de cabos instalados, especialmente para sistemas de controle e instrumentação que envolvem grandes distâncias e muitos dispositivos (FF 1996).

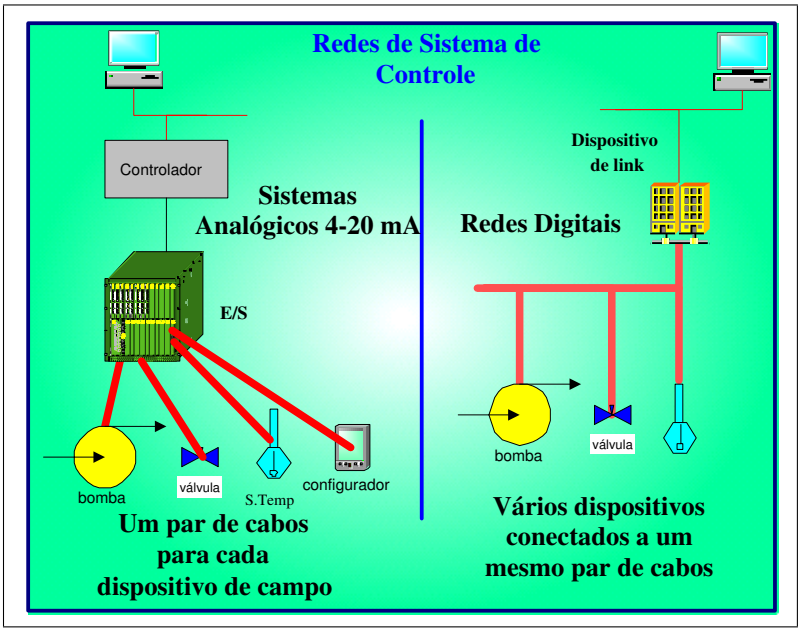


Figura 2.1: Redes - redução da quantidade de cabos.

- **Mais Informações**. A maior vantagem da comunicação digital é a grande quantidade de informação que pode ser transmitida em um simples par de cabos (Figura 2.2). Em vez de uma única variável por par de cabos, centenas ou milhares de informações podem ser transmitidas em um único par de cabos na rede. Isto faz possível extrair muito mais informações de um dispositivo de campo do que nos sistemas analógicos. Por exemplo, antes da comunicação digital ser introduzida era impossível transmitir remotamente outras informações além de simples variáveis contínuas de controle ou discretas de entrada e saída. Sensores e atuadores estavam ligados diretamente aos controladores usando um par de cabo dedicado e transmitindo não mais que uma simples variável de processo ou manipulação. O sinal analógico transitava somente em uma direção, do sensor para o controlador ou do controlador para o atuador. Controladores, indicadores, registradores e processos estavam localizados muito próximos uns do outros. Ajuste de sintonia e controle tinham que ser feitos localmente. O advento da comunicação digital possibilitou os controladores serem colocados mais distantes do processo. Todas as informações de centenas de loop e pontos de monitoração poderiam ser transmitido ao console do operador em uma simples rede. Não somente processo de E/S ou variáveis poderiam ser transmitidas mas também operações de informações

como set-points, alarmes e sintonia. A comunicação digital habilita o processamento distribuído, diagnóstico, configuração e outras funcionalidades que não poderiam ser adicionadas, inicialmente nos controladores mas também nos instrumentos de campo.

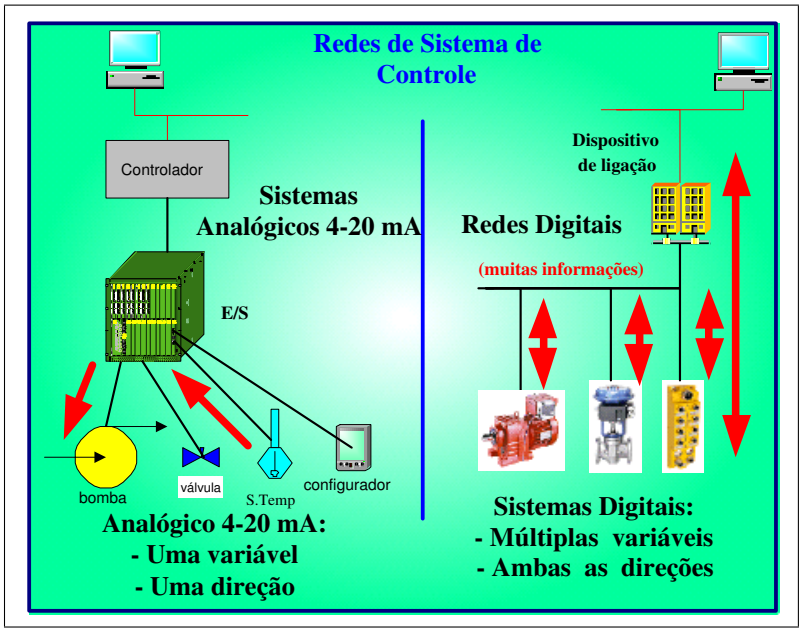


Figura 2.2: Redes - mais informações.

- **Robustez**. Nos sistemas analógicos é usual codificar a informação (por exemplo, o valor lido em um instrumento) por um nível de corrente, entre 4 e 20 mA. O problema é que um sinal degenerado (atenuado, distorcido ou com ruídos), ou seja, com erro, ainda é um sinal válido. A comunicação digital tem a vantagem de ser robusta pois apenas dois estados são válidos (nível lógico "0" ou nível lógico "1"), e muito menos suscetível a perturbações. E o mais importante, é que os protocolos (conjunto de regras que a comunicação de dados segue a fim de complementar várias transações da rede ((HALLBERG 2003),p.64)) possuem mecanismos que tornam a possibilidade de corrupção dos dados muito pequena. Esses mecanismos são:

  – **Formato de Pacotes**. Um pacote é qualquer coleção de dados enviados por meio de uma rede, e o termo é normalmente usado para referir-se a unidade de dados enviados em qualquer camada do modelo

de rede *International Standards Organization/Open Systems Interconnection* (ISO/OSI). Os pacotes podem ter tamanho fixo ou variável. Os pacotes possuem cabeçalho, dados e *bytes* de verificação. O cabeçalho possui informações para identificar os pacotes, como endereço de origem e destino. O endereço de origem refere-se ao endereço do nó onde o pacote foi originado. O endereço de destino contém o endereço do nó que processará as informações encapsuladas no pacote. Os dados contém as informações definidas pelo usuário que precisam ser transmitidas de um nó para outro na rede. Os *bytes* de verificação podem ser conhecidos como *checksum* ou *Cyclic Redundancy Check* (CRC). Quando o nó destinatário recebe um pacote, ele gera uma simples soma de verificação (*checksum*) ou uma soma de verificação polinomial (CRC). Ele então cruza as verificações dos dados com a soma de verificação recebida. Se as duas somas não corresponderem, assume-se que o quadro está corrompido e ele é subseqüentemente descartado. O nó destinatário pode ou não requerer a transmissão dele novamente.

- **Confirmação de Recepção de Dados**. A confirmação é a indicação de sucesso na recepção de dados pelo nó destinatário. Quando o nó de origem recebe uma confirmação, ele descarta os pacotes de dados retidos para possíveis retransmissões.
- **Tratamento de Erros**. Um protocolo deve ser capaz de detectar erros, reportá-los e corrigí-los. A detecção de erros é executada através do número de seqüência, do tamanho e *bytes* de verificação. O número de seqüência é utilizado para detectar dados perdidos, a entrega fora de ordem e a proteção contra dados duplicados. O campo relativo ao tamanho dos dados é encarregado de verificar a entrega completa dos dados. Os *bytes* de verificação protegem contra a modificação de dados dentro dos nós de roteamento.

2.3 Mídias de transmissão

Para que as transmissões sejam realizadas são necessárias as mídias de transmissão. Mídias de transmissão são os canais usados para a realização da comunicação de dados.

As principais mídias de transmissões utilizadas são:

- **Cabos**

  Existem diversos tipos de cabos. Os mais comuns são os pares trançados não-blindados (*Unshielded Twisted Par* (UTP)) e os coaxiais, sendo o UTP o mais usado, principalmente em aplicações comerciais. Outros tipos comuns de cabeamento são os pares trançados blindados (*Shielded Twisted Par* (STP)) e os cabos de fibra óptica. As características elétricas dos cabos, tais como resistência e capacitância, são os fatores que podem determinar o comprimento e o número máximo de dispositivos conectados em uma rede. Mas as regras para quantidade de dispositivos e comprimento não são absolutas e muitas são baseadas nas condições de pior caso, por exemplo, áreas de segurança intrínseca impõem limites nos comprimentos dos cabos, normalmente para menos. São tipos de cabos:

  - **Cabo de par trançado**: São constituídos de dois ou mais pares de condutores isolados dentro de um invólucro de vinil ou teflon. Em cada par, os condutores são entrelaçados a fim de diminuir os efeitos de interferência eletromagnética. São de dois tipos:

    * **Cabo UTP**: Possui pares de fio não blindados, portanto sujeito a ruídos, mas possuem baixo custo;
    * **Cabo STP**: Possui pares de fio blindados. Apresenta excelente imunidade, mas são rígidos.

    Os tipos de cabo de par trançado recomendados são listados na Tabela 2.1.

    A *Electronics Industry Association* (EIA) classificou os cabos de rede de par trançado de acordo com sua capacidade de transporte de tráfego,

Tabela 2.1: Tipos de cabo coaxiais

Par	Blindado	Trançado	Bitola	Dist. Máx.	Tipo
Simples	Sim	Sim	AWG 18	1.900m	A
Multi	Sim	Sim	AWG 22	1.900m	B
Multi	Não	Sim	AWG 26	400m	C
Multi	Sim	Não	AWG 16	200m	D

por exemplo, categoria 3 (Cat-3) está limitada à velocidade até 10Mbps, Cat-5 à velocidade entre 100Mbps a 1Gps.

- **Cabo coaxial**: É formado por um condutor central que pode ser um único fio de cobre maciço ou um conjunto de fios torcidos. Um invólucro plástico envolve o condutor central que, por sua vez, é envolvido por uma folha metálica e depois por uma blindagem de fio trançado. Um invólucro externo protege o cabo. Este tipo de cabo é normalmente utilizado em redes com topologia de barramento;
- **Cabo de fibra óptica** : Um cabo de fibra óptica não transporta elétrons como os cabos elétricos e sim sinais através da modulação da luz. A principal vantagem é a isolação elétrica dos equipamentos e proteção da integridade dos dados da interferência eletromagnética (EMI). A fibra óptica necessita de cuidados especiais de instalação e manuseio. A distância máxima de transmissão depende da potência do transmissor e sensibilidade do receptor e outros fatores como atenuação, conectores e emenda dos cabos. São utilizadas em redes que necessitam transpor longas distâncias. As fibras ópticas podem ser classificadas de acordo com o diâmetro do núcleo da fibra:
  - **Fibra multímodo** : Diâmetro do núcleo na faixa de $62,5\,\mu m$ a $125\,\mu m$. Alcance do sinal em até $2\,km$ e usa LED como fonte de luz;
  - **Fibras monomodo** : Diâmetro do núcleo na faixa de $10\,\mu m$ a $125\,\mu m$. Alcance do sinal até $40\,km$ e usa laser de alta intensidade luminosa como fonte de luz.

Os diversos tipos de rede *Ethernet* exigem cabos diferentes, por exemplo,

o padrão “100Base-T”, significa que o número “100” indica a taxa de dados (em Mbps) que o padrão transporta. A palavra “Base” significa que a rede é uma rede de conexão banda-base (*baseband* - um sinal por vez) e não uma rede de banda larga (*broadband*). A letra “T” significa que o cabo usado é de par trançado. Mais informações podem ser encontradas em (HALLBERG 2003)

- ***Wireless***: Utilizam o ar como meio de transmissão. As redes locais sem fio são chamadas de *wireless* LAN e são muito eficazes em locais que precisam de uma rápida troca no *layout*. No nosso estudo de caso, Capítulo 9, é sugerida uma proposta para substituir os cabos STP por rede *wireless*. São exemplos:
  - **Infravermelho;**
  - **Laser;**
  - **Radiodifusão;**
  - **Sistemas de Microondas.**

2.4 Classificação das redes

Nesta seção são apresentadas as várias classificações das redes em função do fluxo de sinais, do tipo de transmissão, controle de acesso ao meio físico e relacionamentos de comunicação.

2.4.1 Fluxo dos sinais

As redes em função do fluxo dos sinais podem ser classificadas em:

- *Simplex*. Quando a comunicação ocorre em um só sentido;
- *Half duplex*. Quando a comunicação ocorre nos dois sentidos, mas não simultaneamente; e
- *Full duplex*. Quando a comunicação ocorre em ambos os sentidos no canal simultaneamente.

2.4.2 Tipos de transmissão de sinal

Existem dois tipos de transmissão de sinal:

- **Transmissão assíncrona**: A transmissão ocorre em intervalos de tempo não predeterminados. O dado é transmitido sem um sinal de sincronismo. Na transmissão, o byte de dados enviado é acompanhado de *bits* de controle: *start bit*, *parity bit* e *stop bit* . A transmissão dos dados é inicializada pelo *start bit* com nível lógico "0" indicando ao destinatário que os dados reais estão sendo enviados. Os outros próximos oito *bits* (um byte) são os dados reais. Sucede o bit de paridade que assume o nível lógico "0" ou "1" a fim de que o número total de *bits* no nível lógico "1" no dado real transmitido e paridade sejam sempre ímpar. E finalmente o *stop bit* com nível lógico "1" indicando a finalização da transmissão. A principal vantagem da transmissão assíncrona é o baixo custo dos circuitos eletrônicos, em compensação possui baixa eficiência devido à necessidade de se enviar muitos *bits* de controle.

- **Transmissão síncrona**: Os blocos de informação são transmitidos apenas no momento determinado pelo *clock* de sincronismo, não havendo necessidade de *start bit* e *stop bit*. Uma das vantagens é que o *overhead* (custo de transmitir ou processar *bits* extras) é muito pequeno. Portanto, transmite mais dados reais que a transmissão assíncrona no mesmo intervalo de tempo. Em compensação os dispositivos são de alto custo.

2.4.3 Os padrões de controle de acesso ao meio físico

As redes utilizam regras específicas conhecidas como padrões de controle de acesso ao meio físico (*Media Access Control* (MAC)) para controlar o momento em que um dispositivo pode transmitir o pacote de dados.

Topologias diferentes suportam padrões diferentes de controle de acesso ao meio físico. Por exemplo, a topologia barramento suporta "disputa", a topologia estrela suporta *polling* e a topologia anel suporta a "passagem de fichas".

Os padrões de controle de acesso ao meio físico estão relacionadas com um proto-

colo na camada 2 (enlace) do modelo OSI, especificamente com a subcamada MAC, a qual realiza o controle de acesso ao meio. Esses protocolos foram definidos pelo IEEE, em 1980, na norma 802.x .

A norma 802.x definiu os seguintes padrões de *frame* (unidade de dados transmitidas na camada enlace) para redes de comunicação digital:

- 802.3 (*Carrier Sense Multiple Access with Collision Detection*(CSMA/CD));
- 802.4 (*token bus*);
- 802.5 (*token ring*);
- 802.11 (*wireless*).

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

O padrão *Ethernet* utiliza o método de acesso por disputa. O acesso a mídia é permitido com base no sistema "o primeiro a chegar é o primeiro a ser servido", o que significa que cada dispositivo de rede tem que competir para acessar o meio de transmissão. Quando um nó deseja transmitir ele verifica a existência ou não de portadora no barramento. Se não houver portadora, ou seja, nenhum nó transmitindo, ele passa a transmitir. Enquanto uma estação está transmitindo as outras aguardam o barramento ficar livre para transmissão. O problema ocorre quando duas estações verificam ao mesmo tempo a ausência de portadora e passam a transmitir, então ocorre a colisão. Para resolver o problema de acesso ao barramento foram criados os protocolos conhecidos como CSMA. Exemplo de protocolo CSMA é o CSMA/CD. O CSMA/CD ao verificar a colisão aguarda um tempo aleatório para enviar o pacote novamente.

Token Ring

O padrão *Token Ring* utiliza o método de acesso de passagem de fichas para controlar um acesso do nó ao meio físico de transmissão. Para acessar o meio de transmissão, o nó deve estar de posse de um quadro especial que circula pelo anel denominado de *token*. O processo de transmissão é simples. Quando um nó deseja enviar um

frame, ele aguarda o *token* chegar. Quando o *token* chega, o nó verifica se ele está livre ou transportando dados. Se o *token* estiver livre, o nó seta o status do *token* como ocupado e envia junto com ele a sua mensagem e o endereço do nó destino. Quando um nó recebe o *token* e verifica que a mensagem enviada é para ele, lê a mensagem e muda o status do *token* para lido. Quando a mensagem retornar ao nó que a enviou, o nó retira a mensagem do *token* e seta o status do *token* como livre, liberando outros nós para transmitir. Somente o nó que possui o *token* é que pode ter acesso ao barramento, deste modo, não há possibilidade de colisões.

Token Bus

O MAC 802.4 utiliza um *token* único que dá ao seu detentor o direito de transmitir, embora a topologia seja do tipo barramento, em termos lógicos forma-se um anel onde cada nó sabe qual é o nó anterior e o nó seguinte. A implementação mais divulgada é o ArcNet a 2,5 Mbps, em especial nos EUA. O *Manufacturing Automation Protocol* (MAP) é uma implementação *token bus* bastante usada na indústria.

Polling

Polling é um método de controle de acesso ao meio que designa dois dispositivos. São eles:

- **Mestre** (*Master*). É um elemento da rede que toma a iniciativa de enviar mensagens para outro elemento da rede. É conhecido como estação ativa;
- **Escravo** (*Slave*). É um elemento da rede que não tem iniciativa de comunicação numa rede, e só respondem as mensagens solicitadas pelo mestre. É denominado de estação passiva.

O dispositivo mestre faz uma requisição seqüencial (*polling*) a cada escravo de uma maneira predeterminada para os escravos responderem a estas requisições. Somente o escravo que está sendo sondado pode responder as requisições. Os dispositivos escravos não podem comunicar-se entre si. O método de *polling* talvez cause atraso em alguns aplicativos à medida que outros dispositivos são colocados na rede. O

método de *polling* permite uma utilização completa da capacidade do meio físico de transmissão eliminando completamente a probabilidade de uma colisão. As redes que utilizam *polling* são mais apropriadas para aplicações em tempo real (determinismo), como em equipamentos de automação.

Em relação ao número de mestres as redes podem ser multi-mestre e mono-mestres. A Figura 2.3 apresenta estes tipos de rede. Na rede multi-mestre o mestre que possui o *token* é habilitado a enviar e receber mensagens desde o primeiro escravo até o último escravo. Este mestre passa o *token* para o próximo mestre, o qual faz o *polling* de todos os seus escravos. O número de nós determina o tempo de ciclo.

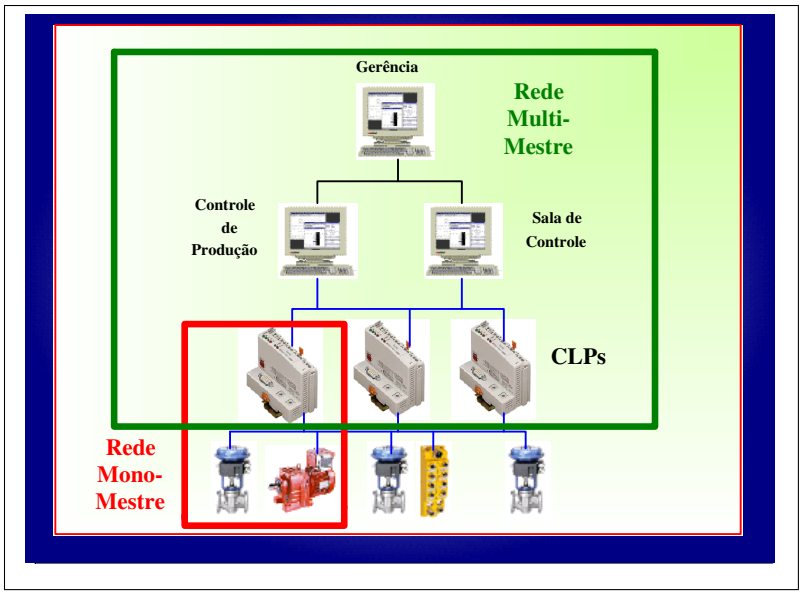


Figura 2.3: Rede mono-mestre e multi-mestre.

Redes *wireless*

É um padrão de *frame* de comunicação digital cuja mídia não é palpável. Combina conectividade com mobilidade dos dados. Deve ser usada em ambientes que necessitam mudanças rápidas de lay-out, ou aonde não é possível instalar os cabos tradicionais (MORAES 2004).

Em relação às redes tradicionais apresentam mobilidade, rápida instalação, escalabilidade, etc. Mas deve-se estar atento as questões de segurança, interferência na faixa de operação, desempenho e conectividade com a rede local existente.

Há basicamente três tipos de redes sem fio: infravermelho, laser e radiofreqüência (WiFi e *Bluetooth*). Descreveremos sinteticamente as de radiofreqüência, pois podem representar um próximo passo na expansão do nosso estudo de caso.

Os sistemas de radiofreqüência utilizam microondas para transmissão de sinal através do ar. Geralmente usam faixas de freqüência de 900 MHz, 2,4 GHz e 5 GHz padronizadas pelo *Industrial Scientific Medical* (ISM). Fatores, como, freqüência, potência de transmissão, antenas, tipo de construção e sinais refletidos podem afetar a propagação do sinal.

A maior parte das redes *wireless* LAN utilizam técnicas de espalhamento espectral (*spread spectrum* (SS)) as quais trabalham com baixa relação sinal/ruído. Dentre essas técnicas tem-se a *Frequency Hopping SS* (FHSS) e *Direct Sequence SS* (DSSS). A técnica FHSS usa uma portadora de banda estreita que muda a freqüência, acompanhando uma seqüência tanto conhecida pelo transmissor quanto pelo receptor. Para um receptor não conhecido, o modo de operação FHSS aparece como um ruído de curta duração. A norma IEEE 802.11 padroniza a velocidade de 2 Mbps para o FHSS.

São características do modo DSSS:

- Geração de um bit redundante para cada um transmitido. Isto permite recuperar dados originais sem necessidade de uma nova transmissão;
- Padronizado pelo IEEE 802.11 com velocidades de até 11Mbps.

As redes *wireless* LAN são parecidas com a rede *Ethernet*. Cuidados com o número de usuários, volume de dados trafegados e taxa de erro do rádio afetam diretamente o desempenho da mesma.

Na instalação de uma LAN os seguintes elementos devem ser pesquisados:

- Placa de rede *wireless* (conectados a um computador, notebook ou placas microcontroladas/microprocessadas);
- *Access point*. É responsável por gerenciar as conexões entre os usuários e a rede. É também o ponto de acesso a rede com fio;

- Antenas. Podem ser do tipo direcional e omnidirecional.

Diferente das topologias de redes com cabo, as redes *wireless* podem ter topologia estruturada e topologia *ad hoc*. Na topologia estruturada tem-se o uso de um *access point* (concentrador). Na topologia *ad hoc*, como não existe um ponto central de controle, os serviços são gerenciados e oferecidos pelos participantes.

Para maiores detalhes de padronização verifique:

- IEEE 802.11. Protocolo de acesso ao meio (MAC) da camada física;
- IEEE 802.11b. Trabalha com *Spread/Spectrum Direct Sequence*;
- IEEE 802.11a. Trabalha com OFDM com velocidades até 54 Mbits/s (50m);
- IEEE 802.11 g. Trabalha com OFDM, mas atingem o mesmo alcance do IEEE 802.11 b (100m);
- IEEE 802.11 e. Melhoramento da camada MAC;
- IEEE 802.11 f. Permite access point interoperarem entre si;
- IEEE 802.11 i. Padrões de segurança mais robustos.

2.4.4 Relacionamentos de rede

O termo relacionamento de rede se refere como um nó utiliza os recursos de outro nó pela rede.

Cliente/Servidor

Um relacionamento de rede cliente/servidor (Figura 2.4) é aquele em que existe distinção entre os nós que disponibilizam os recursos de rede (servidores) e aqueles que usam os recursos (clientes ou estações de trabalho). O relacionamento cliente/servidor é usado, por exemplo, em leitura e escrita acíclicas de parâmetros em dispositivo de campo, transferência de arquivos de configuração e outras atividades do *host*. Em uma comunicação cliente/servidor as requisições são enviadas

baseada por prioridade. Uma nova requisição só é transmitida quando o cliente reconhecer que o servidor aceitou a requisição. O relacionamento cliente/servidor é de um para um, significando que o valor é enviado somente para um destino. São exemplos, protocolo HART e PROFIBUS.

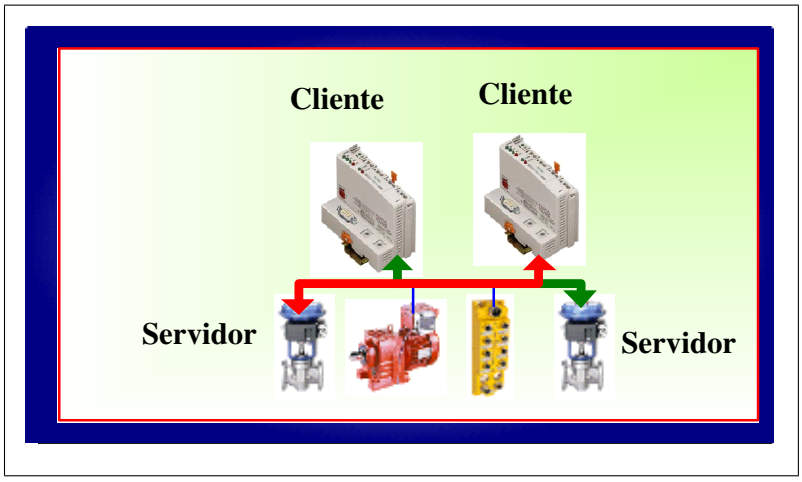


Figura 2.4: Relacionamento cliente/servidor.

Produtor/Consumidor

No relacionamento produtor/consumidor (*Publisher/Subscriber*) (Figura 2.5) o produtor transmite um pacote de dados para todos os consumidores. É ideal para comunicação cíclica. É muito eficiente, pois um valor é transmitido de um dispositivo para outros em uma simples comunicação. Protocolo como o FOUNDATION Fieldbus utilizam este serviço de comunicação para transferência de dados na malha.

Exceção

O relacionamento de exceção (*Report by Exception*) é usado para transmissão acíclica de alarmes e notificação de eventos para o *host*. É ideal para ambientes aonde os operadores desejam somente receber alarmes do processo ou evento de falhas quando eles ocorrerem, caso contrário o sistema mantém-se em silêncio. A comunicação de exceção é baseada em fila, significando que as requisições são enviadas baseada por prioridade. Uma nova requisição só é transmitida quando a requisição anterior foi reconhecida. Na comunicação de exceção o valor gerado é enviado para muitos

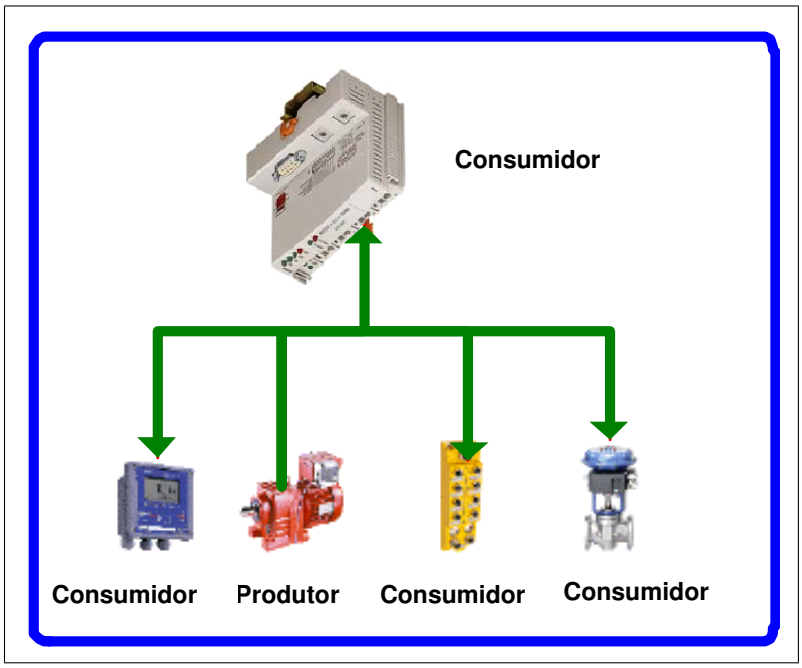


Figura 2.5: Relacionamento produtor/consumidor.

consumidores simultaneamente.

Mestre/Escravo

Neste relacionamento, o mestre que pode ser uma estação de trabalho ou CLP, envia requisições de escrita ou leitura para outro dispositivo de campo denominado de escravo. O escravo que foi endereçado responde a requisição. Este relacionamento está presente no protocolo MODBUS. Outro exemplo deste relacionamento é um mestre CLP lendo o valor de processo de um transmissor escravo e após executar o algoritimo de controle escreve o valor calculado num posicionador escravo.

2.5 Padrões de rede de comunicação

A padronização sempre foi um dos maiores problemas em todas as indústrias e não poderia ser diferente no mercado de redes. No início dos anos 80, houve um grande crescimento na área de redes, porém atrás do crescimento existia um grande problema que era a quantidade de padrões existentes, ou seja, cada fabricante possuía soluções próprias, ou seja, um padrão proprietário, que obrigava o cliente a ado-

tar soluções fechadas de um único fabricante, visto que as soluções de diferentes fabricantes não inter-operavam (MORAES 2004).

Em razão dessa grande dificuldade, os maiores fabricantes e representantes da indústria se reuniram em uma comissão especial da *International Standardization Organization* (ISO) e após alguns meses de estudo começou a ser criado o conhecido Modelo de Referência *Open Systems Interconnection* (OSI) (PROFIBUS 2002) (MUSKINJA et al. 2003). Por ter sido definido como um padrão em que sistemas de diferentes fabricantes pudessem interoperar, esse modelo foi um dos primeiros padrões a levar o nome de "sistema aberto".

O modelo OSI é composto de sete camadas: física, enlace, rede, transporte, sessão, apresentação e aplicação. A figura 2.6 apresenta as camadas do modelo OSI.

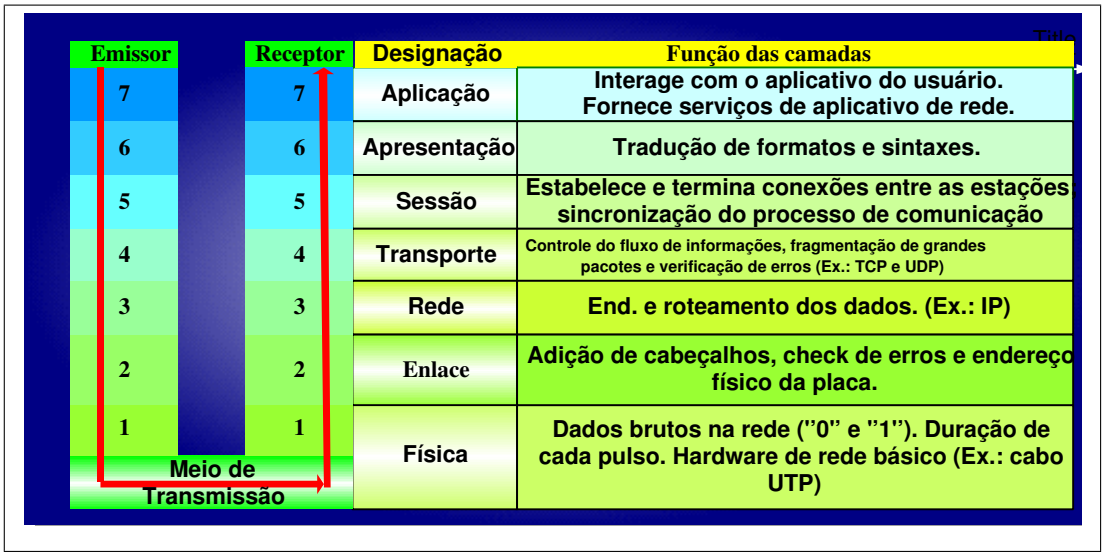


Figura 2.6: Modelo de referência OSI da ISO.

O funcionamento da hierarquia em camadas é relativamente simples. A camada superior faz uso dos serviços da camada diretamente inferior e presta serviços à camada diretamente superior. Por exemplo, a camada enlace faz uso dos serviços da camada física para enviar os sinais no meio de transmissão e presta serviços à camada rede para disponibilizar o enlace fim a fim. Quando um dado é transmitido, cada uma das camadas recebe os dados da camada superior, acrescenta as informações necessárias dessa camada e envia para a camada inferior. Quando o dado é recebido do outro lado, ocorre o procedimento contrário.

Se um sistema de comunicação não requerer alguma função específica, a correspondente camada pode ser retirada. A maioria das redes *Fieldbus* precisa somente de três camadas: física, enlace e aplicação. A camada Usuário não faz parte do modelo OSI. Exemplos de *Fieldbus* são os protocolos Interbus e Profibus.

Camada física

Essa camada específica todo cabeamento, os sinais elétricos e luminosos a serem trocados no meio, as pinagens e os conectores da rede. Ela é ainda responsável pela modulação dos *bits* "Os" e "1s" em sinais elétricos ou ópticos a serem transportados pelo meio físico. A camada física determina ainda características mecânicas das placas de redes e dos dispositivos. Em resumo, a camada física define como um cabo de rede é anexado á "placa de rede" e como os dados devem ser formatados para transmissão.

Camada enlace

A camada enlace tem como responsabilidade garantir de forma correta e confiável a comunicação em uma conexão física. Ela é responsável por montar os elementos de dados transportados (quadros). Os quadros nesta camada mais conhecidos são o *Ethernet* e o *Token Ring*.

A camada enlace é ainda subdividida em duas sub-camadas:

- ***Logic Link Control* (LLC)**: A sub-camada de Controle de Enlace Lógico é responsável por inserir um cabeçalho e um *trailer*. Toda camada adiciona as informações do cabeçalho ao pacote de dados, entretanto a camada enlace (por meio da sub-camada de controle de Enlace lógico) adiciona um *trailer* ao pacote de dados. O trailer é um teste de redundância cíclica (*Cyclical Redundancy Check* (CRC)), ou seja, um cálculo de paridade do pacote de dados. Quando o cliente receptor recebe o pacote de dados, um CRC é realizado e o resultado é comparado com o CRC do remetente. Se os resultados forem correspondente, os dados são considerados válidos e são passados à próxima camada. Se os dois resultados não forem correspondentes, os dados serão considerados inválidos e

são descartados.

- ***Media Access Control* (MAC)** : A sub-camada MAC coloca o endereço físico da placa de rede no cabeçalho que é adicionado ao pacote de dados. Um endereço MAC é um número hexadecimal único de 12 algarismos que está em todas as placas de rede. Um exemplo de um endereço MAC é: 00-80-C7-4D-B8-26. Os protocolos MAC mais conhecidos são: 802.3 e 802.5 *Token Ring*.

É bom notar que as duas subcamadas trabalham de forma independente. A camada MAC é dependente do meio e a LLC não, portanto existe uma camada MAC específica para o *Ethernet* e outra para o *Token Ring*.

Camada rede

A camada rede é responsável pelo endereçamento e roteamento da rede. A camada de rede define os diferentes pacotes de protocolos, como o *Internet Protocol* (IP) e o *Internet Protocol Exchange* (IPX). O roteamento da informação contida em cada pacotes de protocolos incluem a origem e o roteamento da informação. O roteamento da informação contida em cada pacote informa à rede qual o destino do pacote e, paralelamente, comunica ao computador receptor a origem do mesmo.

O *Internet Protocol* (IP) é utilizado para pacotes de endereçamento e especificará o endereço de origem (remetente) e endereço de destino (receptor) do pacote de dados. O endereço especificado é um endereço de 32 *bits* único, conhecido como endereço de TCP/IP.

O IP também fragmenta os pacotes e identifica cada um com um único ID. Uma vez que o pacote é recebido na camada rede do cliente receptor, o *Internet Protocol* irá remontar os pacotes fragmentados e passar os dados à camada de Transporte.

O roteamento é realizado na camada rede para determinar o melhor caminho ou rota do destino. Protocolos comuns de roteamento que operam na camada rede incluem o *Routing Information Protocol* (RIP), *Open Shortest Path First* (OSPF) e *Border Gateway Protocol* (BGP).

É fácil referir-se à camada rede como o "policial do tráfego" da rede. Ela especifica o envio e o recebimento dos endereços IP e determina a melhor rota para o destino. Quando os endereços de IP são especificados, um endereço físico tem de ser determinado.

A camada rede se torna mais importante quando a conexão de rede passa por um ou mais roteadores, que são os dispositivos de *hardware* que examinam cada pacote e, através de seus endereços de origem e de destino, os encaminham para seus devidos destinos. Através de uma rede complexa como a *Internet*, um pacote pode percorrer 10 ou mais roteadores até chegar ao seu destino final. Em uma rede LAN, um pacote pode percorrer somente um ou mais roteadores até chegar ao seu destino.

Os protocolos na camada rede podem transportar quaisquer variações de protocolos das camadas inferiores. No contexto real, um pacote IP pode ser enviado por uma rede *Ethernet*, *Token Ring* ou até mesmo por um cabo serial conectado a dois computadores.

Camada transporte

A quarta camada, a camada transporte, é responsável por verificar erros e o controle de fluxo de dados. Nessa camada, dois protocolos são utilizados para transmissão de dados: O *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP).

Nessa camada, um nível adicional de conexão é fornecido se o TCP for utilizado como protocolo de transporte. Esse nível adicional de conexão é o resultado de um *handshake* de três vias e garante a entrega do pacote de dados por meio do reconhecimento de pacotes. O *handshake* de três vias é um conjunto de saudações utilizados para determinar que tanto o remetente como o receptor estejam prontos para a transferência de dados.

O controle de fluxo fornecido pela camada transporte é um resultado do tamanho da janela do TCP/IP. O tamanho da janela especifica a quantidade de dados que um remetente enviará para o receptor sem receber um pacote de reconhecimento. Um tamanho típico de janela é 4096 *bytes*. A camada transporte é responsável por

dividir os grandes pacotes de dados em pacotes menores, normalmente de 1500 *bytes*, porém esse valor pode ser alterado. Com o tamanho típico da janela de 4096 *bytes*, pode haver um total de quatro pacotes não reconhecidos na rede. Em geral, quando um cliente receptor recebe um pacote, um pacote de reconhecimento será enviado ao remetente. Uma vez que o remetente recebe este pacote, os pacotes de dados adicionais podem ser enviados ao receptor. No caso de alguns pacotes de dados não serem reconhecidos, poderá ocorrer uma retransmissão, mas isso depende do protocolo que é utilizado. A principal diferença entre os dois protocolos da camada transporte, TCP e UDP, é o pacote de reconhecimento.

Camada sessão

Uma das principais funções da camada sessão é sincronizar o diálogo, ou seja, recepção e transmissão. Essa camada ainda tem a capacidade de recuperar conexões de transporte sem perder a conexão de sessão. Ela faz isso estabelecendo uma conexão virtual, com base no nome de usuário, nome do computador ou credenciais de rede do cliente.

A camada sessão gerencia essa conexão virtual emitindo pontos de verificação nos dados que ela recebe. Um ponto de verificação informa ao aplicativo quais dados foram recebidos. Se ocorrer uma falha de conexão, a camada sessão avaliará os pontos de verificação e começará a transferência a partir do último ponto de verificação. Suponha, por exemplo que o computador 1 esteja transferindo 10 MB de dados para o computador 2 e a conexão cai no ponto dos 8 MB. Em vez de retransmitir todos os dados, a camada de sessão procurará o último ponto de verificação começará a retransmissão no local do ponto de verificação (nesse caso, 8MB). Pelo fato de a camada sessão gerenciar a comunicação, a transferência dos dados é resumida em vez de re-inicializada. Pelo fato de a conexão utilizada na camada sessão ser uma conexão virtual, ela não garante a entrega do pacote.

Camada apresentação

A função da camada apresentação é traduzir formatos e sintaxes, para que possam ser compreendidos pelos dois subsistemas que estão se comunicando.

Camada aplicação

A camada mais alta (camada 7) no modelo OSI é a camada aplicativo. Essa camada é responsável por interagir com o aplicativo do usuário. Ela aceita os dados dos *softwares* aplicativos a partir do aplicativo de *software* e fornece o serviço de aplicativo de rede que é responsável pela solicitação do usuário. Alguns exemplos de transformação de dados na camada de aplicativo incluem o seguinte:

- Quando um usuário envia um e-mail, a camada de aplicativo fornecerá acesso ao serviço do *Simple Mail Transfer Protocol* (SMTP);
- Uma transferência de arquivo pode ser realizada utilizando o *File Transfer Protocol* (FTP). O serviço de FTP é uma responsabilidade da camada aplicativo;
- Solicitar um site da web como www.uol.com no seu navegador colocará uma solicitação na camada de aplicativo para conversão do nome por meio do *Domain Name System* (DNS) e também uma solicitação de protocolo para o *Hyper Text Transfer Protocol* (HTTP).

Os aplicativos existem na camada aplicação, mas são transparentes para o usuário. A camada aplicação é a única camada que interage com o aplicativo de *software* do usuário.

(SANTORI & ZECH 1996) ilustram as funções das camadas *Fieldbus* fazendo um comparativo com a entrega de um serviço de correio, conforme Figura 2.7. O processo inicia-se com a criação de uma mensagem na camada do usuário com respeito a temperatura de um reator. Abaixo da camada do usuário, a mensagem é escrita num pedaço de papel em uma linguagem compreensível para um futuro receptor (camada de aplicação).Deste ponto em diante o papel é o que importa, e não a mensagem. O papel é colocado em um envelope endereçado. O foco de atenção é o envelope, e não mais o papel ou a mensagem (camada de enlace). O envelope é colocado numa caixa postal e um aviso é colocado para indicar que a caixa postal não está vazia (camada física). Em seguida, o caminhão dos correios recolhe o envelope (mídia de transporte).

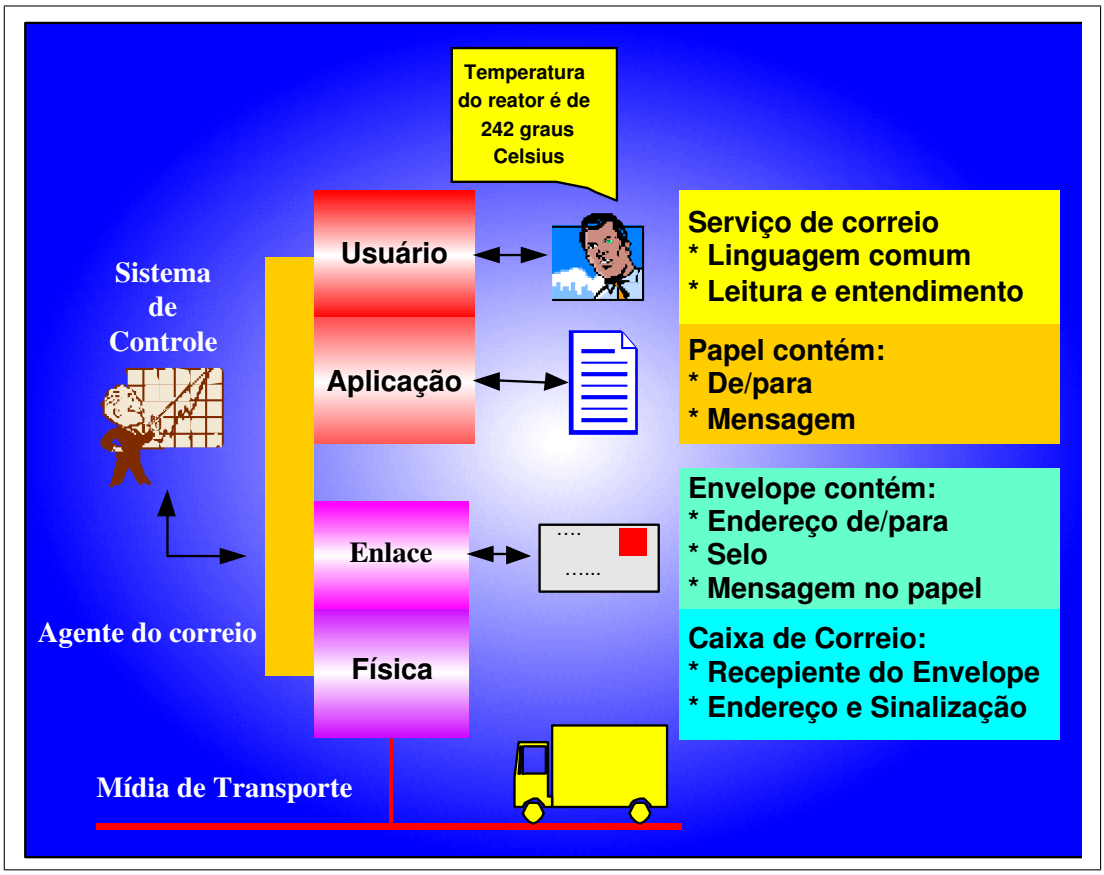


Figura 2.7: Analogia entre *Fieldbus* e serviços de correio.

Neste ponto, não importa que caminho o envelope tomará até chegar a seu destino. É responsabilidade do correio entregar o envelope. Finalmente o envelope chega a pessoa destinada, só e somente só quando a pessoa lê e entende a mensagem é que a comunicação ocorreu. Note que a mensagem pode conter informações para o destinatário ou pode requerer informações dele.

Como no serviço de correios, os protocolo *Fieldbus* e serviços são simples mecanismos para assegurar uma comunicação confiável entre duas entidades, neste caso, dois dispositivos de campo ou um dispositivo de campo e um sistema de controle. Além do cabeamento, o mais importante elemento em um sistema *Fieldbus* é uma linguagem em comum.

2.6 Componentes de uma rede

Os dispositivos de uma rede são os responsáveis pela transferência de dados entre os nós que constituem uma rede. Cada dispositivo possui diversas propriedades e utilidades.

O principal teste de qualquer projeto de rede é a capacidade de direcionar o tráfego de rede de um nó a outro. É necessário conectar os vários dispositivos de rede em uma configuração que permita à rede transmitir sinais entre os dispositivos da forma mais eficiente possível, levando em consideração o tipo de rede e as diferentes necessidades de conectividade da mesma. Entre os dispositivos básicos utilizados para conexão, podemos incluir:

- **Placa adaptadora de rede ou *Network Interface Card* (NIC)**: Placa ou módulo microcontrolado que tem a função de serializar os dados para o meio de transporte. Opera tanto na camada física como na camada enlace do modelo OSI. A placa de rede utiliza o endereço MAC da camada enlace de dados, que é único para cada placa e utiliza a topologia da camada física. Esse endereço é formado por um identificador do fabricante, seguido por um sufixo de identificação da placa.

- **Repetidor**: É um dispositivo que prolonga a distância que uma determinada rede percorre. Ele recebe um sinal de um lado, amplifica-o e o envia para sua outra extremidade. Freqüentemente, eles são encontrados em redes *Ethernet* fina ou par trançado de *Token Ring*. Eles são usados para conectar apena o mesmo tipo de mídia.Os repetidores funcionam na camada física do modelo de rede OSI da ISO. Eles não possuem inteligência para entender os sinais que estão transmitindo.

- ***Hub*:** É usado para conectar os nós de rede aos *backbones* ou em redes mais simples, somente os nós. Os nós são conectados aos *hubs* por meio de uma disposição física em estrela tanto em redes de topologia lógica em estrela quanto em redes de topologia lógica em anel. Os *hubs* possuem duas propriedades importantes. São elas:

– Amplificam, regeneram e repetem sinais elétricos provenientes de uma porta para todas as outras portas. Embora os *hubs* estejam ligados em estrela, na verdade, eles funcionam eletricamente (logicamente) de forma parecida com o segmento de uma topologia de barramento. Todas as portas de um *hub* fazem parte de um mesmo domínio de colisão, ou seja, quando um nó está usando a rede, outro nó não pode acessá-la. Contudo, há um risco de dois nós acessarem a rede ao mesmo tempo, causando uma colisão da comunicação. Neste caso, eles tentarão retransmitir uma fração de segundo mais tarde, mas não ao mesmo tempo. Devido à possibilidade de um atraso aleatório ser introduzido neste tipo de rede (exemplo: rede *Ethernet*) elas são consideradas não determinísticas. Conectar todos os *hubs* juntos de alguma forma resulta em um grande domínio de colisão, abrangendo todos os *hubs*. Para resolver este problema seria ideal usar uma configuração na qual todos os *hubs* separados são conectados a um *switch* que mantém cada *hub* em seu próprio domínio de colisão.
– Particionamento automático. O *hub* consegue isolar qualquer nó defeituoso de outros nós. Por exemplo, tais particionamentos ocorrem se for detectado um curto no cabo ou se a porta do *hub* estiver recebendo pacotes em excesso congestionando a rede. O particionamento automático impede que uma conexão defeituosa cause problemas a todas as outras conexões.

Os *hubs* estão se tornando muito mais sofisticados. Muitas vezes, eles possuem vários atributos embutidos, inclusive os seguintes:

– Auto-sensor de diversas velocidades de conexão. Por exemplo, são comuns os *hubs Ethernet* que conseguem detectar e executar automaticamente cada nó a velocidade de 10Mbps ou 100 Mbps;
– *Uplinks* de alta velocidade que conectam o *hub* ao *backbone*. Normalmente, eles funcionam em uma velocidade 10 vezes maior que à velocidade básica do *hub*;
– Funções embutidas de ponte e roteamento, o que torna desnecessário o uso de dispositivos adicionais para executar ponte e roteamento.

Utilizar *hubs* torna a arquitetura do sistema extremamente escalonável, mas apresentam problemas de confiabilidade devido ao fato de serem alimentados separadamente do barramento e utilizarem complexos circuitos digitais. Em redes com velocidade de 10Mbit/s deve-se usar no máximo 4 *hubs* em série, perfazendo um alcance de 500 metros. A Classe I (lenta) permite somente um *hub* para 100 Mbit/s e a Classe II (rápida) permite dois *hubs*.

- **Ponte**: É a a versão mais inteligentes de um repetidor. As pontes conseguem conectar dois segmentos, mas elas possuem inteligência para passar o tráfego de um segmento para outro somente quando o tráfego for destinado a outro segmento. Existem algumas pontes disponíveis que conseguem transpor diferentes mídias de rede: da *Ethernet* Fina coaxial a *Token Ring* de par trançado, mas que utilizem o mesmo protocolo. As pontes funcionam na camada enlace (Camada 2 do modelo de rede ISO/OSI). Elas examinam o endereço *Media Access Control* (MAC) de cada pacote que recebem, a fim de determinar se os mesmos deverão ser encaminhados a outra rede. As pontes contém informações de endereço de todas as partes de sua rede, tanto através de uma tabela de roteamento estática que você programa, como por um sistema dinâmico de aprendizagem em árvore que descobre automaticamente todos os dispositivos de rede e endereços da rede. As pontes devem ser utilizadas em redes de pequeno porte ou em situações que se deseja impedir que o tráfego em um segmento fosse desnecessariamente transmitido para outro segmento. Roteadores e *switches* oferecem soluções melhores.

- **Roteador**: Um roteador é mais inteligente que uma ponte. Eles funcionam na camada rede do modelo OSI da ISO e são bem mais inteligentes que as pontes com relação ao envio de pacotes recebidos aos seus destinos. Os roteadores analisam o pacote recebido e definem o melhor caminho que o pacote vai seguir até alcançar o destino. Este é o motivo pelo qual os pacotes chegam fora de ordem no destino.

- ***Switch*:** O *switch* é parecido com uma ponte, exceto por que ele possui muitas portas e normalmente é parecido com um *hub*. Pode-se pensar em um *switch* como uma ponte com diversas portas. Eles podem alternar rapidamente as conexões de uma porta para outra. Como os *switches* formam conexões entre

portas uma a uma, todas as portas de um *switch* não fazem parte de um único domínio de colisões. Neste sentido, o *switch* atua como uma espécie de super ponte. Os *switches* são freqüentemente usados para conectar uma série de *hubs* a um *backbone* mais veloz. Um *switch* utiliza uma *Management Information Base* (MIB) que é uma base de dados do status das portas, taxas de colisão e tabela de endereços MAC para ajudar a segmentar a rede. Os segmentos criados com um *switch* são denominados de redes locais virtuais. Os *frames* (quadros) são copiados somente para a porta destino, criando um domínio de colisão distinto. O *switch* distribui a largura de banda para cada porta. Um *switch* pode praticamente eliminar as colisões de rede que utilizam uma abordagem 100% *switch*. Não há limites para o número de *switches* que podem ser cascateados. Os *switches* podem fazer adaptação do *Ethernet* para o *Fast Ethernet* e Gigabit *Ethernet* e podem possuir placas de diferentes tecnologias como FDDI, ATM e *Token Ring*. Mais informações sobre *switches*, tais como *switches wire speed*, *blocking*, *non blocking*, *store and forward*, *Cut-Thoughe routing Switches* podem ser obtidas em (MORAES 2004). Pode-se aumentar a confiabilidade da rede implementando-se redundância de *switches* (IEE 802.1.2) (KOULAMAS et al. 1999), contudo, os *switches* devem suportar algoritmos que previnem a formação de loops desabilitando uma das pontes deste loop (*Rapid Spanning Tree* - IEEE 802.1.d). Ocorrendo uma falha, a ponte desabilitada, seria automaticamente habilitada. A diferença de preço entre um *switch* de 24 portas e um *hub* de 24 portas agora é relativamente pequena.

- ***Gateways.*** *Gateways* são interfaces específicas de aplicativos que vinculam todas as sete camadas de um Modelo ISO/OSI quando elas são diferentes em qualquer um dos níveis ou em todos eles. Um gateway de rede é utilizado para traduzir protocolos e também pode ser utilizado para traduzir endereços de um protocolo para outro.

2.7 Redes quanto à disposição geográfica

As redes podem ser classificadas quanto à disposição geográfica e normalmente estão em uma das duas categorias desta classificação: *Local Area Network* (LANs)(redes locais) e *Wide Area Networks* (WANs)(redes remotas).

Redes Locais

Quando você divide qualquer tarefa em partes menores, essa tarefa freqüentemente torna-se mais fácil de gerenciar. A mesma situação ocorre se uma grande rede é dividida em segmentos, os administradores de rede podem gerenciar a rede mais facilmente. Uma rede local pode ter vários segmentos, todos conectados por um dispositivo de rede denominado de roteador. Um roteador é responsável por conectar segmentos de uma rede (SCRIMGER et al. 2002).

Quando segmentos diferentes de uma rede forem conectados por meio de uma mídia permanente de conexão, o resultado é uma rede local. Uma rede local não tem qualquer conexão que conte com linhas de dial-up ou linhas dedicadas. Todos os cabos de uma rede local fazem parte da rede e não têm um sinal a não ser aquele do roteador ou dos clientes.

Principais características:

- Redes de propriedade privada;
- Até 300 nós;
- Distância entre os nós de até 2km;
- Atraso na transmissão menor que $10\,ms$;
- Topologias: estrela, anel e barramento;
- Taxas de transmissão de até 1 Gbit/s.

Redes Remotas

As redes remotas existem em quase todos os ambientes de rede. Quase todas as conexões da *Internet* são feitas por meio de uma conexão de rede remota. Uma

conexão de rede remota é uma mídia de conexão que não é parte da rede local, o que significa que serviços externos de um fornecedor são requeridos para a comunicação. Freqüentemente, o tipo de conexão é obtida de uma operadora local de telefonia.

Uma das derivativas das redes remotas são as *Metropolitan Area Networks* (MANs). São características desta rede:

- Até 1000 pontos;
- Distância entre os nós de até 100km;
- Meios de transmissão: cabos ópticos ou coaxiais;
- Altas taxas de transmissão.

Quando as distâncias são maiores temos as *Wide Area Network (WAN)*. São características desta rede:

- Distância entre cidades, estados ou países;
- Atrasos maiores;
- Meios de transmissão: satélites, linhas telefônicas e links de microondas.

2.8 Topologias de rede

Topologia de rede refere-se à estrutura geométrica da rede de comunicação. As diferentes topologias possuem custos variados (tanto para instalar quanto para manter), diferentes níveis de desempenho e confiabilidade. A topologia está ligada à camada física e a subcamada *Media Access Control* (MAC) da camada de enlace.

2.8.1 Topologia barramento

Na topologia de barramento apresentada na Figura 2.8, chamada *Common Bus Multipoint Topology*, os nós estão conectados fisicamente a um cabo principal denominado de *backbone*, através de conectores tipo T e possuem terminadores nas

pontas (exemplo: resistores de 50 ohms). Os terminadores possuem duas funções, a primeira é prevenir a reflexão do sinal e subseqüente erros de comunicação. A segunda função do terminador é converter a mudança de corrente produzida pelo dispositivo de transmissão em uma mudança de tensão que é percebida por todos os dispositivos da rede. O canal é compartilhado por todos os nós, podendo o controle ser centralizado ou distribuído. Os nós possuem somente uma porta de comunicação. Os dados se propagam ao longo do barramento em todas as direções. A rede de barramento mais predominante nos dias de hoje é a denominada *Ethernet* 10Base-2 (HALLBERG 2003).

Vantagem: Custo baixo. Usam menos cabos que as topologias estrela e anel e, consequentemente, exigem menos material e menos mão-de-obra.

Desvantagens: Uma falha nos cabos, conectores ou terminadores fará com que toda a rede falhe. E, mais importante, o defeito poderá demorar a ser identificado pois todos os cabos, conectores e terminadores deverão ser testados até encontrar aquele que está causando problema.

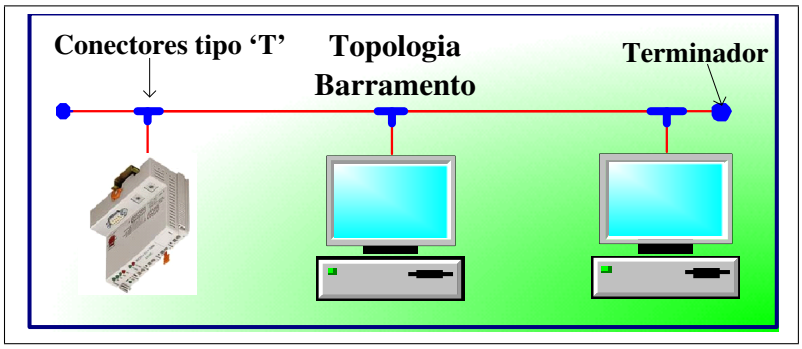


Figura 2.8: Topologia barramento.

2.8.2 Topologia estrela

Nesta topologia, os nós estão conectados diretamente a um concentrador (*hub* ou *switch*) que gerencia a comunicação entre os nós, conforme Figura 2.9. Cada nó tem uma conexão ponto a ponto com o concentrador. O acesso do nó à rede é feito pelo protocolo CSMA/CD. Como exemplo, as redes de topologia em estrela podem ser usadas nos padrões *Ethernet* 10Base-T ou 100Base-T (HALLBERG 2003).

Vantagens: São mais confiáveis que as topologias de barramento. Caso ocorra alguma falha na conexão de rede somente esta conexão será afetada. Além disso, uma vez que cada cabo é colocado diretamente do concentrador para o nó fica extremamente fácil localizar o defeito.

Desvantagens: O custo de instalação é bem maior que as redes de topologia barramento, pois requer mais cabos, mão-de-obra e aquisição de concentradores. Toda a rede fica inoperante se o concentrador falhar completamente.

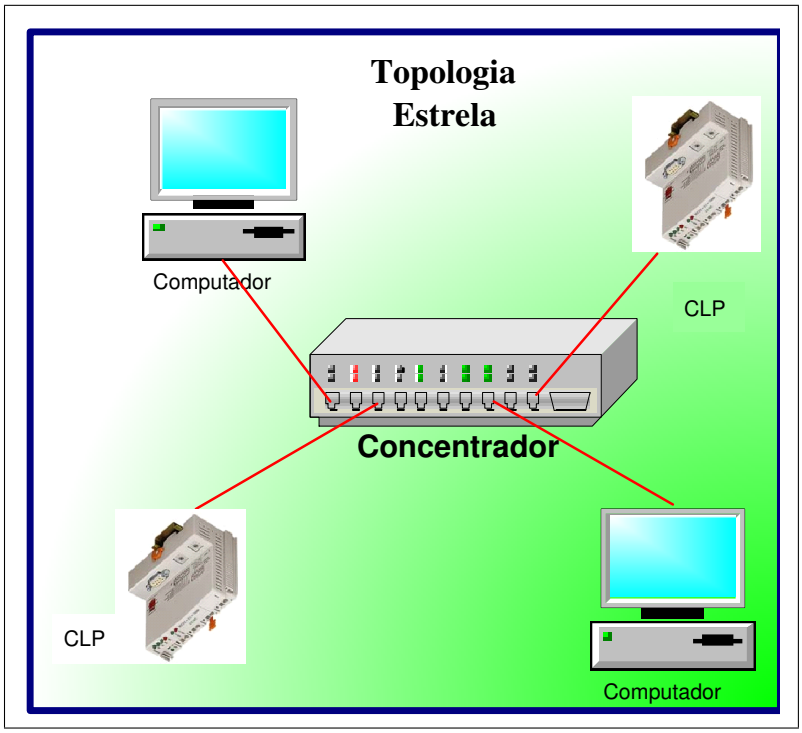


Figura 2.9: Topologia estrela.

2.8.3 Topologia em anel

A topologia em anel, na verdade, não é uma disposição física de um cabo de rede, como se imagina. Ao contrário, os anéis são uma disposição lógica, conforme Figura 2.10. Fisicamente, os cabos são conectados em estrela, com cada nó conectado por meio de seu próprio cabo ao *Multi-station Acess Unit* (MAU). "Físico" define a planta da rede conectada e o *layout* real do cabo ou mídia."Lógico" define como os nós acessam a mídia (fio e cabo) e se comunicam em um meio físico. Cada nó recebe toda a informação que está sendo trafegada, regenera-a e a envia ao nó seguinte,

deste modo a deterioração do sinal é baixa. A transmissão só pode ocorrer quando a estação possui a autorização ou *token*. Não há colisões, diferentemente da topologia barramento, ou seja, as redes de topologia em anel baseiam-se no padrão de controle de acesso ao meio físico *Token Ring* (HALLBERG 2003). A topologia em anel é bastante confiável, mas apresenta atrasos à medida que a rede vai se expandindo. Grandes ambientes de rede utilizam um design de anel tolerante a falha. O anel secundário permitirá que a rede continue em operação caso o anel principal pare. O protocolo *FOUNDATION Fieldbus H1* usa topologia anel.

Vantagens: Não há colisões, portanto, operam de forma previsível (determinismo).

Desvantagens: Se não for baseada no transporte bi-direcional, se um nó parar, toda a rede pára. O custo dos equipamentos é superior ao da topologia barramento. A adição ou remoção de um nó pára a rede.

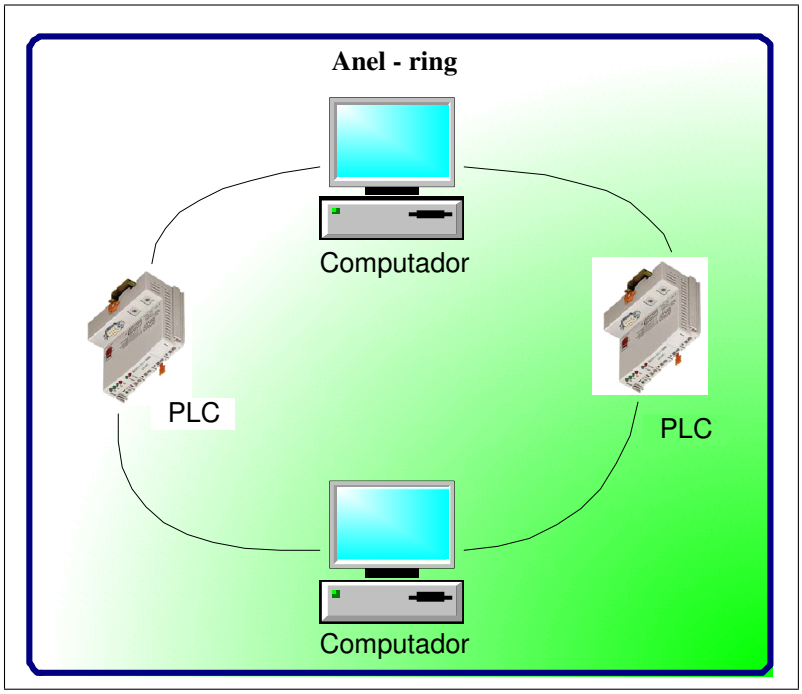


Figura 2.10: Topologia anel.

2.8.4 Topologia em grafo

Em uma topologia em grafo todos os nós estão conectados uns aos outros, conforme a Figura 2.11. Geralmente, esta topologia é usada em redes muito pequenas por causa dos requisitos de *hardware*. O *overhead* para uma topologia em grafo é mínimo. É possível ter vários componentes falhos nesta topologia por causa da redundância em cada cliente.

Vantagem: muito confiável.

Desvantagem: cara de implementar pois necessita de muitas conexões físicas e *hardware* adicional.

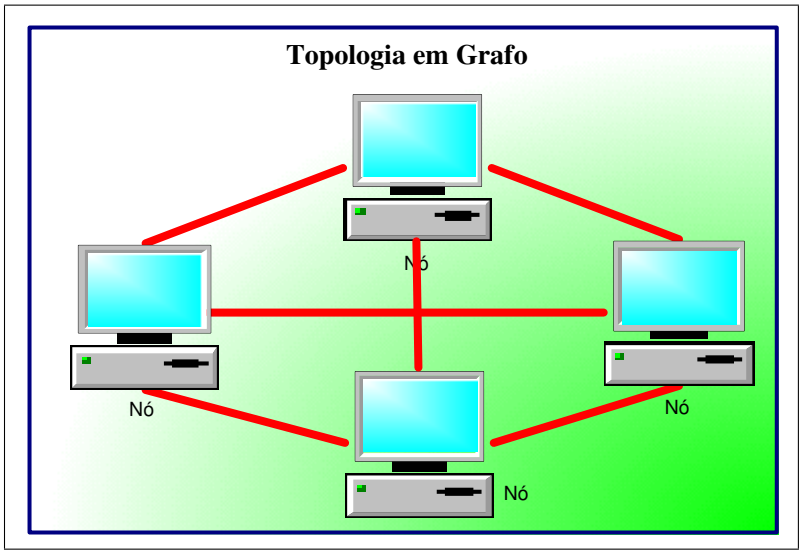


Figura 2.11: Topologia grafo.

2.8.5 Topologia híbrida

É raro um ambiente que utilize apenas uma única topologia. Freqüentemente, as necessidades de uma empresa requerem o uso de várias topologias diferentes, ou seja, topologia híbrida. A topologia híbrida tornas as redes muito mais robustas. Na Figura 2.12 temos o exemplo de uma topologia híbrida. Os nós estão ligados ao *switches* usando topologia estrela. Os *switches* estão ligados em topologia anel. Caso ocorra falha em um dos *switches* ou cabos na topologia em anel, os nós ainda continuarão se comunicando.

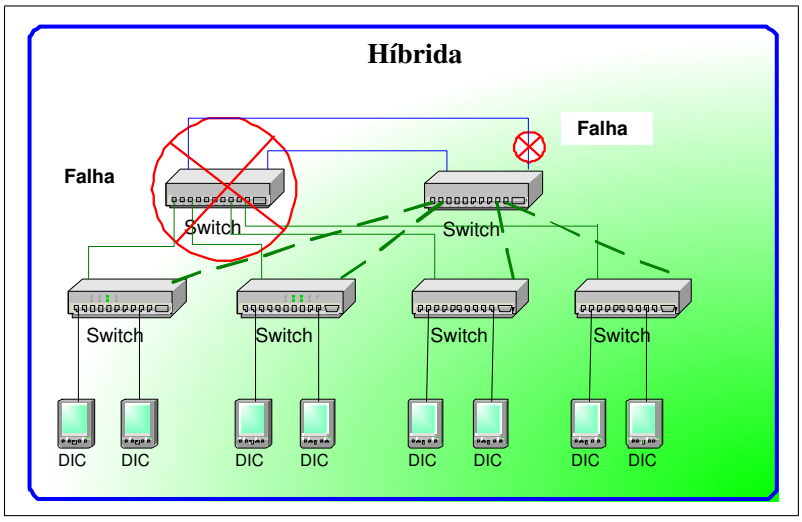


Figura 2.12: Topologia híbrida.

Vantagem: muito confiável, pois permite redundância de equipamentos e cabos.

Desvantagem: sua implementação é onerosa, pois necessita de muitos equipamentos e cabos adicionais.

Capítulo 3

Tecnologia *Fieldbus*

3.1 Introdução

O *Fieldbus* começou a ser desenvolvido a partir de 1986, na Europa, com o objetivo de substituir a fiação paralela utilizada na transmissão de sinal dos instrumentos tradicionais analógicos com laços de corrente 4–20 mA, híbridos e digitais (Figura 3.1) para os controladores de processo por um único barramento serial digital (Figura 3.2) que conectasse todos os sensores, atuadores e equipamentos de controle e que permitisse a comunicação entre sistemas de diferentes fabricantes. Em outras palavras, passou-se de arquiteturas *Direct Digital Control* (DDC) e *Programmable Logic Controller* (PLC) para arquiteturas *Digital Control System* (DCS) e *Field Control System* (FCS) (Obs: estas figuras são explicadas com maiores detalhes no item 3.5). A história das redes de controle de processo é a história do padrão *Fieldbus* IEC 61158 e IEC 61784.

Conceituar *Fieldbus* não é uma tarefa simples. A seguir temos várias definições em pesquisas realizadas em diversos artigos.

(SANTORI & ZECH 1996) primeiramente conceituam *Fieldbus* como uma rede de comunicação totalmente digital para conexão da instrumentação de campo com os controladores de processo. A seguir definem como um protocolo de comunicação totalmente digital, de alto desempenho que substitui os sistemas tradicional com laço de corrente 4-20 mA, híbrido 4-20mA/digital e os provenientes da arquitetura DDC.

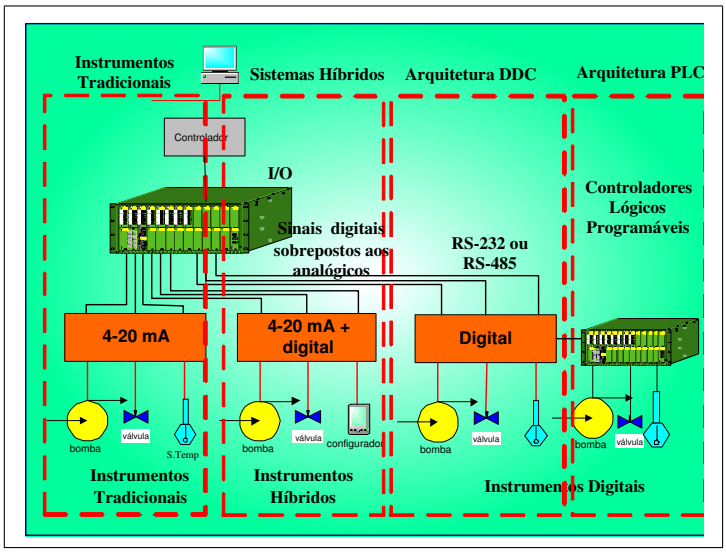


Figura 3.1: Sistemas tradicionais, híbridos e digitais DDC

(RAJA, VIJAYANANDA & DECOTIGNIE 1993) definem *Fieldbus* como uma rede local em tempo real colocada no mais baixo nível hierárquico da automação industrial.

(BENITO, FUERTES, KAHORAHO & ARZOZ 1999) retratam *Fieldbus* como redes digitais que proporcionam uma troca de dados serial eficiente entre um ou mais controladores e dispositivos de campo. O principal requisito é ser em tempo real.

(MANTOVANI 1998) define *Fieldbus* ou barramentos de campo como sistemas de troca de dados serial, constituem atualmente tecnologia de ponta para as mais diversas áreas de automação. De maneira simples, essa tecnologia se baseia na conexão dos elementos que compõe um sistema de automação através de um cabeamento comum, formando uma rede de dispositivos, que podem ser acessados individualmente, utilizando mensagens padronizadas por um protocolo.

O PROFIBUS Trade Organization (PTO) (PROFIBUS 2002) conceitua *Fieldbus* como sistemas de comunicação industrial que usam mídias, tais como, cabos de cobre, fibra óptica ou wireless, com transmissão de bit serial para ligação de dispositivos distribuídos em campo (sensores, atuadores e transdutores) para um controle central ou sistema de gerenciamento.

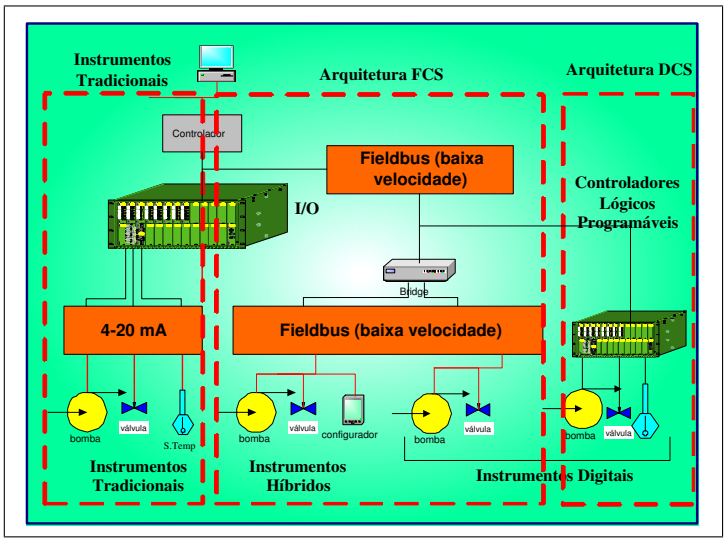


Figura 3.2: Exemplo de sistema baseado em *Fieldbus*

(WOOD 2000) conceitua o *Fieldbus* como um sistema de sensores, atuadores e controladores interconectados.

(SCOTT & BUCHANA 2000) definem *Fieldbus* como uma rede local especial que são dedicadas para aquisição de dados e controle de sensores e atuadores.

(FELSER & SAUTER 2002) referenciam ao *Fieldbus* como sistemas, (YABIN & NISHITANI 2003) e (BERGE 2002) como tecnologia, (PINCETI 2004) como um *link* de comunicação e (GOH & R. 2002) como uma tecnologia em controle e instrumentação.

O que é um *Fieldbus*? Um protocolo? Um sistema? Uma tecnologia? Simplificando, vamos nos referir ao *Fieldbus* como uma tecnologia.

O *Fieldbus* é uma das primeiras tecnologias a verdadeiramente implementar o controle distribuído (SCOTT & BUCHANA 2000). É a possibilidade de fazer o controle no local da aquisição e atuação dos processos, ou seja, no próprio sensor e atuador (TIAN 2001). Cada *Fieldbus* possui um sistema próprio, segundo regras bem definidas, estabelecidas pelo protocolo, para organização e montagem das diversas partes que compõem a mensagem além de critérios para acesso ao meio físico, viabilizando a transmissão em um barramento único.

A transmissão dos dados pode ser feita em etapas no *Fieldbus*, se necessário, já que

uma única estação não pode ocupar demasiadamente o barramento, que é compartilhado com todas as outras estações. Em cada etapa, uma dada estação que tenha dados a transmitir e esteja apta a fazê-lo, segundo os critérios de acesso ao meio estabelecido pelo protocolo, constrói e envia pelo barramento um quadro (*frame*) que contém, além dos dados propriamente ditos, diversos tipos de informações e acessórios necessários à transmissão. Estas informações incluem sincronismo, marcadores, identificação de origem e destino da mensagem, verificadores de integridade, etc.

Um dos primeiros protocolos foi o PROFIBUS-DP, usado para interligação dos CLPs da Siemens, baseado no padrão elétrico RS-485. O *Fieldbus* LonWorks é usado para automação predial predominantemente. O protocolo CAN foi concebido para automação automotiva, com técnica de controle a prova de colisões. Do CAN surgiram DeviceNet da Allen Bradley, SDS da Honeywell e CANOpen. Para automação de sistemas discretos e pequenos criou-se o protocolo *Actuator Sensor Interface* (AS-i), que transmite palavras de 4 bits. Para área de manufatura foi criado o Interbus-S, tanto para dispositivos discretos quanto analógicos (FRANCO 2001).

O item 3.2 descreve a motivação para o desenvolvimento do *Fieldbus*, ou seja, quais as necessidades da indústria que proporcionaram o surgimento desta tecnologia. O item 3.3 relata "quando" o *Fieldbus* surgiu, ou seja , a "História do *Fieldbus*". O item 3.4 relaciona as vantagens obtidas com o uso de padrões. Os itens 3.5, 3.6 e 3.7 descrevem as arquiteturas de sistema de controle e o item 3.8 apresenta dois modelos de estrutura organizacional provenientes das arquiteturas de sistemas de controle. O item 3.9 classifica as áreas de automação e indica os protocolos mais usados por cada área. O item 3.10 esclarece algumas diferenças existentes entre as redes de nível de campo e as redes do nível *host*. O item 3.11 contém exemplos de utilização de protocolos fiedlbus por diversas indústrias. A seção 3.12 mostra que as funções de gerenciamento de ativo têm ganho importância quando comparadas as funções de controle e finalmente o item 3.13 apresenta as novas perspectivas dos *Fieldbus*.

3.2 Requisitos industriais para implantação de um sistema *Fieldbus*

A tecnologia *Fieldbus* surgiu para atender os diversos requisitos dos controles de processos industriais, os quais não são encontrados em ambientes comerciais, ou mesmo, no nível corporativo das empresas. (YABIN & NISHITANI 2003) descrevem os principais requisitos que um *Fieldbus* deve ter para satisfazer as necessidades da indústria de controle e processos.

3.2.1 Requisitos de comunicação de dados

A comunicação entre os dispositivos em um ambiente industrial pode ser classificada de acordo com o tamanho e requisitos de tempo das informações. São elas:

- **Dados urgentes**: É determinística e o momento da transmissão é "aleatório";
- **Dados cíclicos**: É determinística e o momento da transmissão é conhecido;
- **Operações de dados sofisticadas**: Não é determinística, mas precisa ser rápida. A comunicação em sofisticadas operações de dados envolvem o uso de muitos dados e vários protocolos, como por exemplo, transmitir um novo programa de controle para um dispositivo de campo pela *Internet*, requer o uso de vários protocolos como FTP, TCP e IP. Não haverá nenhum problema se o tempo de transmissão for 20 segundos ou 22 segundos, ou seja, não há necessidade de determinismo, mas deve ser suficientemente veloz para que o dispositivo entre o mais rápido em operação novamente.

3.2.2 Requisitos do ambiente

O ambiente no qual o *Fieldbus* opera é diferente das LANs usualmente instaladas nos prédios comerciais. Por exemplo, existência de ruídos, atmosfera explosiva e distâncias entre nós de rede em centenas de metros ou mais. Devido a essas distâncias enormes, o custo dos cabos deve ser levado em conta. Devido às razões expostas, a camada física deve atender os requisitos de alta confiabilidade, transmissão em grandes distâncias e intrinsecamente segura (instrumentação a prova de explosões).

3.2.3 Requisitos de interoperabilidade

Interoperabilidade é a capacidade que os dispositivos possuem de troca de informações entre eles, mesmo que sejam fornecidos por fabricantes diferentes. Isto habilita uma sólida integração de dispositivos de diferentes fabricantes.

3.3 História do *Fieldbus*

Este capítulo relata sinteticamente a origem dos *Fieldbus*. Para maiores detalhes os trabalhos de (FELSER & SAUTER 2002), (WOOD 2000) e (BERGE 2002) devem ser consultados.

Quando as primeiras redes de comunicação digital surgiram, no início da década dos anos 80s, cada fabricante de equipamento para automação e controle desenvolveu o seu próprio protocolo independentemente dos outros fabricantes. Em um curto período de tempo, muitos protocolos proprietários diferentes foram introduzidos no mercado. Os equipamentos de cada fabricante poderiam somente trabalhar com equipamentos do mesmo fabricante. Além disso, a documentação de operação destes protocolos não era amplamente disponível e a tecnologia era geralmente protegida por patentes.

Esta situação resultou em muitas desvantagens. Uma das desvantagens era que nenhum fabricante tinha todos os equipamentos necessários que uma indústria de controle e automação necessitava. A seleção dos equipamentos de cada fabricante era muito limitada, então era necessário encontrar equipamentos em outros fabricantes. Além disso, um fabricante nem sempre tinha os melhores equipamentos para atender todas as áreas de uma indústria, então era desejável comprar equipamentos de fabricantes que fossem especialistas em determinada área de controle e processo. Devido ao fato que equipamentos de diferentes fabricantes tinham protocolos incompatíveis a indústria encontrava-se com poucas opções para expansão: escolher o equipamento preferido não obstante a pobre interoperabilidade com o resto do conjunto ou escolher um dispositivo não tão bom e obter melhor interoperabilidade. Contudo, na maior parte dos casos, não era possível ligar os equipamentos através de uma rede de comunicação digital, resultando nas famosas "ilhas de automação".

Um cenário era bastante comum, uma arquitetura PLC deveria ser conectado a uma arquitetura DCS, mas a integração digital do sistema era impossível desde que os equipamentos de cada arquitetura estabeleciam comunicação usando diferentes protocolos. Um *gateway* poderia ser desenvolvido caso os fabricantes permitissem o licenciamento e fornecessem a documentação do equipamento, mas a custas de um grande ônus de tempo e dinheiro. Um terceiro fabricante normalmente desenvolvia o *gateway* e quando os problemas de comunicação surgiam os dois fabricantes dos equipamentos culpavam-se entre si. Para complicar ainda mais a questão, um *gateway* era requerido para cada combinação de *hardware* e *software*, produzindo situações inimagináveis para os fornecedores de *gateways*. Muitas vezes a comunicação não era possível, e para passar os dados entre os sub-sistemas era necessário voltar a usar os tradicionais sinais analógicos com laço de corrente 4-20 mA.

Havia uma grande anarquia no mercado de automação de controle e processo devido à falta de um protocolo padrão. A situação era intolerável. A primeira tentativa internacional de padronização começou em 1985, quando o *International Electromechanical Commission* (IEC) estabeleceu um grupo de trabalho para definir um barramento padrão para instrumentação de campo. A intenção inicial, refletindo o mais puro desejo dos usuários, era que um único *Fieldbus* padrão deveria ser produzido.

Ao mesmo tempo em que o comitê do IEC iniciou o trabalho de padronização, um número de *Fieldbus* proprietários eram padrões em seus países, por exemplo, o *Process Field Bus* (Profibus) na Alemanha e o *Field Instrumentation Protocol* (FIP) na França. Eles estavam no estágio inicial de desenvolvimento.

Estes padrões proprietários eram patrocinados por grandes companhias que com relutância não aceitavam que seus padrões protegidos deveriam sujeitar-se a um processo de convergência para um padrão em comum. O resultado foi o aparecimento de vários padrões, alguns recebendo a denominação de padrão oficial IEC, outros amparados por grupos de interesse, freqüentemente com fortes ligações regionais. Com o passar dos anos, o número de protocolos proprietários cresceu sempre protegidos por interesses regionais que se tornavam cada vez mais intransigentes. Uma clara ilustração deste efeito foi a reunião dos padrões nacionais da Europa (P-Net, Profibus e WorldFIP) como um padrão euroupeu pelo *Comité Européen de*

Normalisation Electrotechnique (CENELEC) em 1995. O grande problema era que os dispositivos dos três padrões não podiam comunicar-se entre si, pois os protocolos eram incompatíveis.

Em 1995, um trabalho feito pela *The Instrumentaion, Systems, and Automation Society* (ISA), fora dos domínios do IEC, definiu um novo *Fieldbus* americanizado otimizado para indústria de processos, o *Foundation Fieldbus* (FF). Os protocolos de companhias americanas, como FF, DeviceNet e ControlNet também foram incluídos mais tarde como padrões europeus pelo CENELEC.

Enquanto os europeus estavam ocupados padronizando seus *Fieldbus* nacionais e de certa forma negligenciando o que se passava no IEC, o *Fieldbus* FF preparou sua própria especificação para o IEC, num misto de camada de aplicativo do Profibus-FMS e acesso ao barramento do WorldFIP. A especificação FF naturalmente influenciou o trabalho no IEC. Quando esta minuta de padrão internacional foi proposta para votação em 1996, foi inicializada a guerra dos *Fieldbus*, pois o PROFIBUS não foi representado nesta minuta. Então, os países aonde o PROFIBUS tinha uma posição dominante proibiram a adoção do padrão por uma estreita margem. O processo de padronização degenerou para uma batalha política e econômica, conseqüentemente manchando todo o esforço de padronização.

Em 16 de Julho de 1999, o *Committee of Action* do IEC criou uma ampla minuta para acomodar todos os protocolos de barramento, o IEC 61158. O IEC preservou as estruturas da camada física, enlace e aplicação. O "famoso monstro com oito cabeças", apresentado na Figura 3.3, foi aprovado como padrão em 31 de Dezembro de 2000 (WOOD 2000) (FELSER & SAUTER 2002). Protocolos para E/S como CAN e AS-i não são partes do IEC 61158, mas serão incluídos no IEC 62026.

A padronização dos *softwares* aplicativos na camada do usuário (WOOD 2000) (SANTORI & ZECH 1996) trilha caminhos no desenvolvimento de componentes de *software* reutilizáveis, denominados de blocos de função. Uma aplicação pode ser meramente definida ligando entradas e saídas de blocos sem nenhuma linha de programação. Em termos de *software*, blocos de função são objetos com um conjunto predefinido de entradas e saídas com parâmetros de configuração. Os blocos de função podem ter uma contribuição fundamental para produtividade de

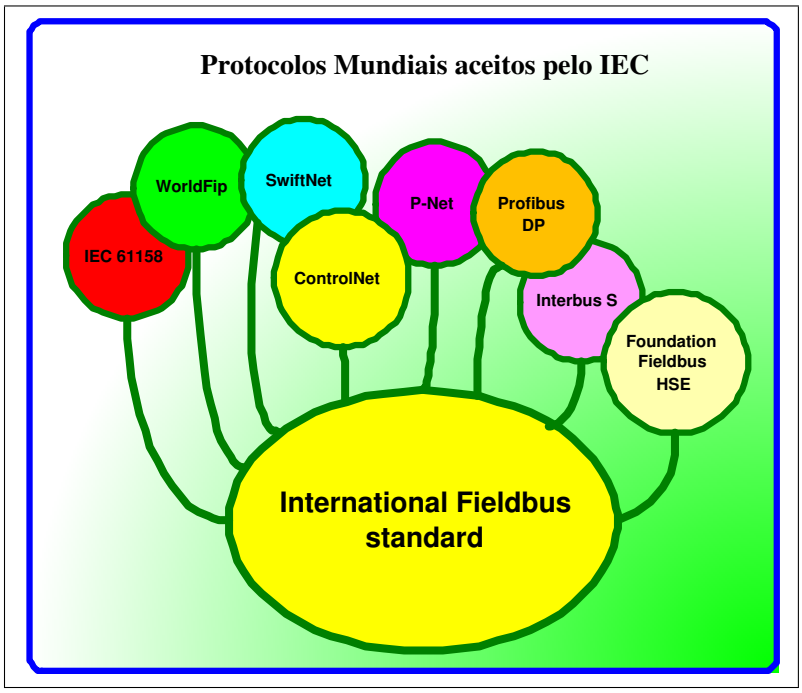


Figura 3.3: Protocolos mundiais (IEC)

software. Mais uma vez, encontra-se o mesmo problema por falta de padronização: os blocos de função diferem entre os diversos fabricantes. Uma pergunta fica no ar: "O IEC falhou espetacularmente na convergência da camada física para camada de aplicação, o que se pode esperar da camada dos usuários?" (WOOD 2000).

3.4 Vantagens dos padrões

Mesmo o IEC não tendo conseguido estabelecer um padrão único, ou seja, uma solução ótima única, não significa que não foram criadas soluções sub-ótimas, muito pelo contrário, verificou-se que determinados tipos de protocolos adaptam-se muito bem a determinadas áreas da indústria. Tanto que a coleção de *Fieldbus* especificadas no padrão IEC 61158 é inútil para qualquer implementação. É necessário um manual de uso prático mostrando qual dos protocolos pode ser melhor usado em um sistema. Estas diretrizes foram compiladas mais tarde no IEC 61784, são os denominados *profiles* (perfis) (FELSER & SAUTER 2002). O trabalho de (YABIN & NISHITANI 2003) possui uma tabela dos principais índices estáticos de *Fieldbus*, como FF(H1), Profibus-DP, Profibus-PA, FIP, DeviceNet, CAN e LON que aju-

dam a selecionar o melhor *Fieldbus* para sua aplicação. Independente do número de padrões, as seguintes vantagens foram obtidas com o uso deles (BERGE 2002) (SANTORI & ZECH 1996):

- As indústrias tornaram-se independente de um único fabricante de equipamentos de instrumentação;
- Fornecedores que utilizam o mesmo protocolo, concorrem entre si e, conseqüentemente, oferecem menores preços e melhores serviços;
- As indústrias passam a ter soluções variadas para diversas aplicações, quer seja de *software*, quer seja de *hardware*;
- O foco dos fabricantes passa a ser as inovações e não o problema de comunicação entre os protocolos.

3.5 Evolução da arquitetura de um sistema de controle

(BERGE 2002) e (SANTORI & ZECH 1996) relatam a evolução das arquiteturas de sistema de controle. O bom entendimento das arquiteturas de sistemas de controle é essencial em qualquer trabalho de automação, ainda mais, que a maior parte das arquiteturas instaladas são *Direct Digital Control* (DDC) e *Distributed Control System* (DCS). A transmissão de sinais em campo e o desenvolvimento de arquitetura de sistemas tem uma ligação muito forte. Cada desenvolvimento em transmissão de sinais tem subseqüentes incrementos no nível de descentralização e melhor acesso as informações de campo. Na era pneumática (transmissores 3-15 psi) (SCOTT & BUCHANA 2000) o controlador estava tipicamente situado no campo e lá operado localmente. Com os dispositivos analógicos ficou fácil enviar o sinal dos transmissores de campo para um controlador central e de lá atuar nas válvulas. Os dispositivos analógicos estão conectados a um sistema de Entradas e Saídas digitais (E/S) ou controladores por meio de *loops* de corrente 4–20 mA (Figura 3.1). O uso de fiação é intenso pois as conexões são ponto a ponto. Em seguida foram criados os dispositivos híbridos (4–20 mA mais digital). Um dos melhores exemplos de implementação em dispositivos híbridos é o protocolo *Highway*

Addressable Remote Transducer (HART), que superpõe um sinal de comunicação digital no sinal analógico 4–20 mA. Os dispositivos puramente digitais apareceram com interfaces abertas e criaram a arquitetura *Direct Digital Control* (DDC). A necessidade de interface de *hardware* personalizada e drivers (arquivo que contém as funções a serem integradas a um sistema operacional para controlar um determinado periférico) de *software* também personalizados é a grande desvantagem dos dispositivos que constituem a arquitetura DDC. Até mesmo quando conectados através de portas padrões, por exemplo, RS-232 ou RS-485, eles ainda requerem drivers de *software* personalizados. Segundo (BERGE 2002) em uma arquitetura inteiramente DDC toda estratégia de controle está localizada em um computador. O sistema inteiro com todos as suas malhas de controle pode deixar de operar por uma simples falha, pois todas as funções de controle estão sendo executadas em um único computador. Por esta razão, não é incomum ter controladores pneumáticos locais existindo no campo prontos para operar caso a arquitetura DDC falhe. Com este sério problema, a arquitetura centralizada proporcionou o aparecimento de arquiteturas mais descentralizadas como a *Programmable Logic Controller* (PLC) e *Distributed Control System* (DCS).

Novas arquiteturas ainda mais descentralizadas surgiram como a *Field Control System* FCS e são propostas outras como em (ODA & TORIOGE 2002). Em nosso estudo de caso optou-se pelo uso da arquitetura FCS com integração *Web* desde o chão de fábrica.

3.6 Arquitetura DCS e PLC

As arquiteturas DCS e PLC emergiram com o advento da comunicação digital, mas estas arquiteturas foram também projetadas baseadas nos transmissores de campo e posicionadores de válvula 4–20 mA. Contudo, a DCS é um grande avanço sobre a DDC devido à distribuição de muitos pequenos controladores que dividiam as tarefas, cada um tratando mais ou menos trinta malhas de controle. Isto teve um benefício imediato, pois uma falha poderia afetar somente uma parte da fábrica. Em outras palavras, um alto nível de distribuição aumenta a confiabilidade do sistema.

Um outro benefício é que a configuração pode ser melhor estruturada, pois

unidades separadas das fábricas necessitam controle e configurações separadas. As arquiteturas DCS e PLC (Figura 3.4) (BERGE 2002) (BROOKS 2001) são caracterizadas por subsistemas convencionais de E/S ou racks de E/S que são interligados aos seus respectivos controladores centralizados via rede de subsistemas. Estas arquiteturas proporcionaram o modelo de estrutura organizacional denominado "pirâmide organizacional" (item 3.8). São níveis desta arquitetura:

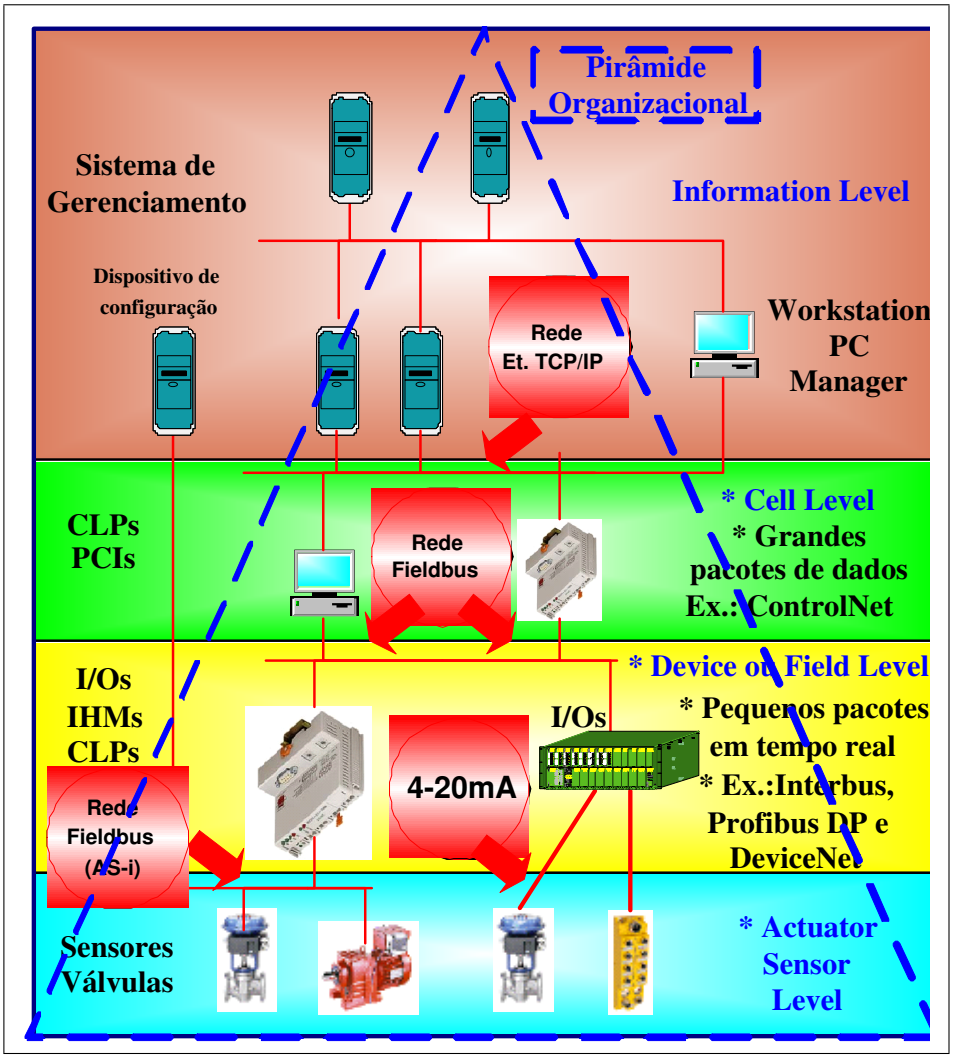


Figura 3.4: Arquitetura DCS e PLC.

- **Actuator/Sensor Level.** Segundo (BERGE 2002) os instrumentos de campo são analógicos. Segundo o *PROFIBUS Trade Organization* (PTO) (PROFIBUS 2002) os sinais dos sensores e atuadores binários são transmitidos num barramento serial atuador/sensor. Isto provê uma tecnologia mais simples e menos

onerosa aonde dados e energia são transportados no mesmo cabo. O protocolo AS-interface oferece um sistema de barramento apropriado para este campo de aplicação.

- **Field Level.** Segundo o PTO, este nível é constituído por dispositivos distribuídos como módulos de E/S, CLPs, IHMs, transdutores, unidades de drivers, dispositivos de análise, ilhas de válvulas entre outros. Estes dispositivos comunicam-se com sistemas de automação maiores em tempo real. A transmissão dos dados do processo é cíclica, enquanto interrupções adicionais, configuração de dados e dados de diagnósticos são transmitidos aciclicamente quando requeridos. Protocolo como o PROFIBUS pode ser usado neste nível. (BERGE 2002) refere-se a este nível como *RIO-level (rack of E/S)* e cita somente os módulos de E/S como dispositivos a serem usados neste nível.

- **Cell Level.** Segundo o PTO, Controladores Programáveis como CLPs e PCIs comunicam-se entre si e com sistemas de Tecnologia e Informação (TI) no mundo empresarial usando padrões como *Ethernet* TCP/IP. O protocolo PROFInet pode ser usado neste nível. Neste nível, as informações são transmitidas em grandes pacotes de dados e a velocidade de transmissão deve ser alta. (BERGE 2002) refere-se a este nível como *Control-level*

- **Plant Level.** O PTO não descreve este nível. (BERGE 2002) refere-se a este nível como sendo o mais alto da hierarquia. Estações de trabalho estão conectadas ao mundo dos negócios. O sistema de gerenciamento coleta todas as informações dos outros níveis e estabelece estratégias de controle e produção. As informações estão disponíveis por toda a rede.

Protocolos como HART e PROFIBUS usam arquitetura DCS ou PLC, seja usando Controladores Lógicos Programáveis ou lógica de *software* baseadas em PC.

Todos estes níveis de *hardware* e rede resultaram em um sistema complexo e caro. A arquitetura DCS expandiu-se e interfaces de comunicação para instrumentos inteligentes usados por protocolos proprietários tornaram-se uma opção. Isto permitiu algum grau de checagem e configuração. Mas nem todos protocolos de instrumentos inteligentes permitiam 4–20 mA e comunicação digital, e a maior parte dos dispositivos não provia interface HART devido ao fato que os fabricantes possuíam seus

protocolos proprietários. Assim, as fábricas eram inclinadas a comprar instrumentos de campo de alguns fornecedores de sistemas em detrimento dos outros.

Quando a arquitetura DCS surgiu, foi batizado como "distribuída" devido ao fato de ser menos centralizada do que a arquitetura DDC. Mas hoje, DCS é considerada centralizada. Esta arquitetura é relativamente considerável vulnerável porque uma falha pode ter conseqüências generalizadas. Devido a esta vulnerabilidade, redundância de controladores, redes de subsistemas de E/S e módulos de E/S devem ser feitos para evitar perda total de controle. Naturalmente, esta redundância envolve custos elevados.

3.7 Arquitetura FCS

O FOUNDATION Fieldbus não é somente um protocolo de comunicação, mas também uma linguagem de programação de construção de estratégias de controle (BERGE 2002). Uma das possibilidades que uma linguagem de programação padrão juntamente com o uso de poderosos recursos de comunicação é a habilidade de executar controle distribuído no campo em vez de um controle centralizado. Por exemplo, é comum um posicionador de válvula agir como controlador da malha que ele faz parte. Ele executa blocos de função Proporcional, Integral e Derivativo (PID), mas somente na sua própria malha, não em outras malhas. Esta nova arquitetura baseada na capacidade dos dispositivos de campo é chamada de *Field Control System* (FCS) (Figura 3.5) e é uma alternativa para a arquitetura DCS devido ao fato que o controle não está centralizado. Ela não trata cada dispositivo de campo como um periférico. Devido a sua natureza descentralizada, a arquitetura FCS tem vantagens como alta confiabilidade, grande escalabilidade e baixo custo. Esta arquitetura evoluiu da DCS e resultou num sistema que é mais distribuído e menos vulnerável a falhas (BERGE 2002).

Na arquitetura FCS os instrumentos da rede de campo (rede H1) estão conectados a rede dos hosts (rede *High Speed Ethernet* (HSE)) via *linking devices*. Portanto, há somente duas redes na FCS. A rede H1 opera a uma velocidade de 31,25 Kbps sendo de 10 a 20 vezes mais rápida que redes utilizando protocolos híbridos e a rede HSE com uma velocidade entre 1 a 2,5 Mbps.

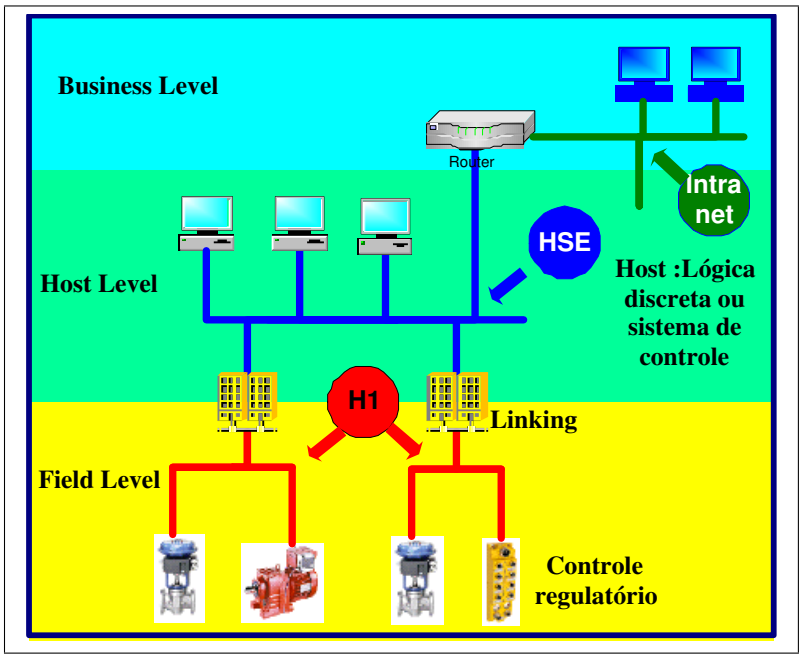


Figura 3.5: Arquitetura FCS.

Tipicamente, os instrumentos de campo executam controle regulatório que é uma das tarefas que exigem mais processamento na automação. O *linking device* ou controlador central pode executar uma lógica discreta ou controle seqüencial. Quando o controle é executado nos dispositivos de campo o número de controladores centrais requeridos diminui, e em alguns casos até é eliminado, diminuindo o custo do sistema.

Devido ao fato que na arquitetura FCS nenhum controlador está envolvido com várias malhas, um problema de uma simples falha afetar uma grande parte do sistema é basicamente eliminada. Contudo, na arquitetura FCS um controle centralizado pode freqüentemente ser encontrado executando controle e lógica discreta de E/S desde que estas tarefas são raramente executadas nos dispositivos de campo interligados em rede (protocolos AS-i ou Interbus Loop). Redundância deverá ser usada sempre que uma planta utilizar controle centralizado.

Pode ainda ser difícil compreender como pequenos dispositivos controladores de campo podem substituir uma unidade de controle em uma grande planta. O segredo atrás deste conceito é que cada dispositivo cuida de uma malha. O poder de processamento de centenas ou milhares de dispositivos ligados em rede excede

o processamento dos sistemas antigos. Desde que estes dispositivos trabalham simultaneamente, um verdadeiro sistema multitarefa é executado, o que não pode ser realizado por um simples processador. O resultado é uma performance excelente e mais dispositivos podem ser adicionados tornando o sistema mais potente.

(SCOTT & BUCHANA 2000) comentam que nas redes de instrumentação futuras não haverá necessidade de implementar complexos controladores principais, pois as operações de controle serão realizadas localmente. Então, os controladores principais terão funções de mais alto nível como manutenção de banco de dados. Isto é o que ocorre em nosso estudo de caso. Os DICs são capazes de realizar os testes, armazenarem os resultados por várias horas se necessário, enviar estes resultados a um servidor de banco de dados e auto-analisarem.

Esta arquitetura proporcionou o modelo de estrutura organizacional denominado "rede corporativa" (item 3.8)

3.8 Estruturas organizacionais na automação

A automação está presente na cadeia produtiva desde a fabricação do produto (ou sua extração) até a entrega ao cliente, passando pelos diversos meios de transportes. Componentes indispensáveis para conceito de automação no futuro são a capacidade de comunicação dos dispositivos e subsistemas aliados a uma metodologia de informação consistente. O aumento da comunicação ocorre tanto horizontalmente ao nível de campo como também verticalmente através de simultâneos níveis de hierarquia. *Fieldbus* com interface para *Ethernet* oferecem pré-condições ideais de rede transparente para todas as áreas de um processo produtivo.

A Automação Industrial baseia-se na pirâmide organizacional (Figura 3.6 (PROFIBUS 2002)), em que ilhas restritas de informações são criadas. Essas ilhas de informações caracterizam-se por sistemas onde o *hardware* e *software* utilizados são proprietários. Mas essa solução causa enormes prejuízos às empresas, uma vez que a conectividade e a integração com os outros equipamentos que não os do próprio fornecedor tornam-se muito complicadas e muitas vezes impossíveis de serem realizadas pelo alto custo da solução ou por uma simples incompatibilidade técnica.

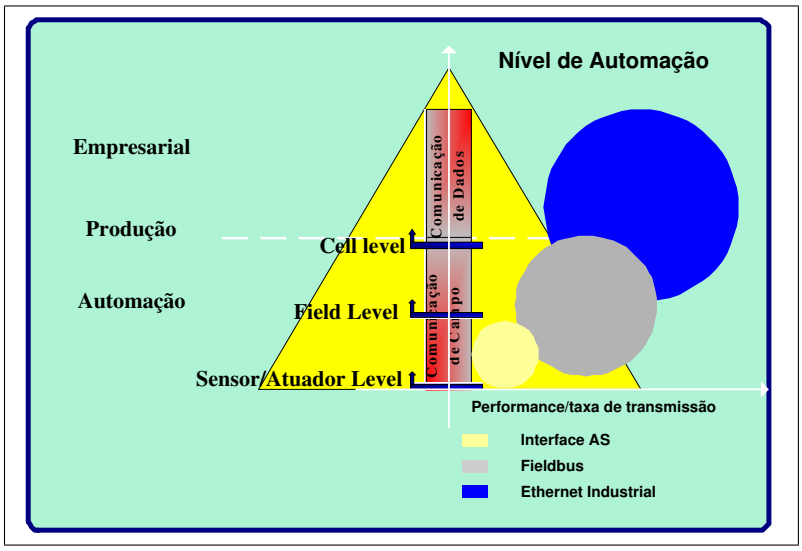


Figura 3.6: Piramide organizacional.

A criação dos denominados "gargalos de informações" também é uma das complicações geradas por essa estrutura. Muitas vezes a informação necessária para uma total integração já existe na fábrica e o problema ocorre quando os diversos sistemas devem ser integrados. A utilização de um protocolo padrão como o TCP/IP facilita muito a integração desses dados e mais do que isso, possibilita a integração entre equipamentos de fornecedores distintos.

Dessa forma, pode-se imaginar um novo modelo de estrutura organizacional, conforme Figura 3.7 (CAMARGO, LOPEZ, BUENO & CELANTE 2003), mas não como antigamente, quando eram criadas as ilhas de informações, mas sim uma arquitetura totalmente integrada com o protocolo TCP/IP como principal meio de transporte de informação.

Essa nova estrutura organizacional permitirá a integração do chão de fábrica com os diversos sistemas existentes, fazendo com que o fluxo de informação seja eficiente, diminuindo os custos de integração e manutenção.

Com o desenvolvimento da arquitetura de controle distribuído, os comitês internacionais incluíram além das especificações de rede, funções de controle e gerenciamento distribuído. A redundância será o principal benefício, pois os programas de controle poderão ser executados em vários dispositivos de campo e não somente no CLP (Padrão IEC 61804) (WOLLSCHLAEGER 2001).

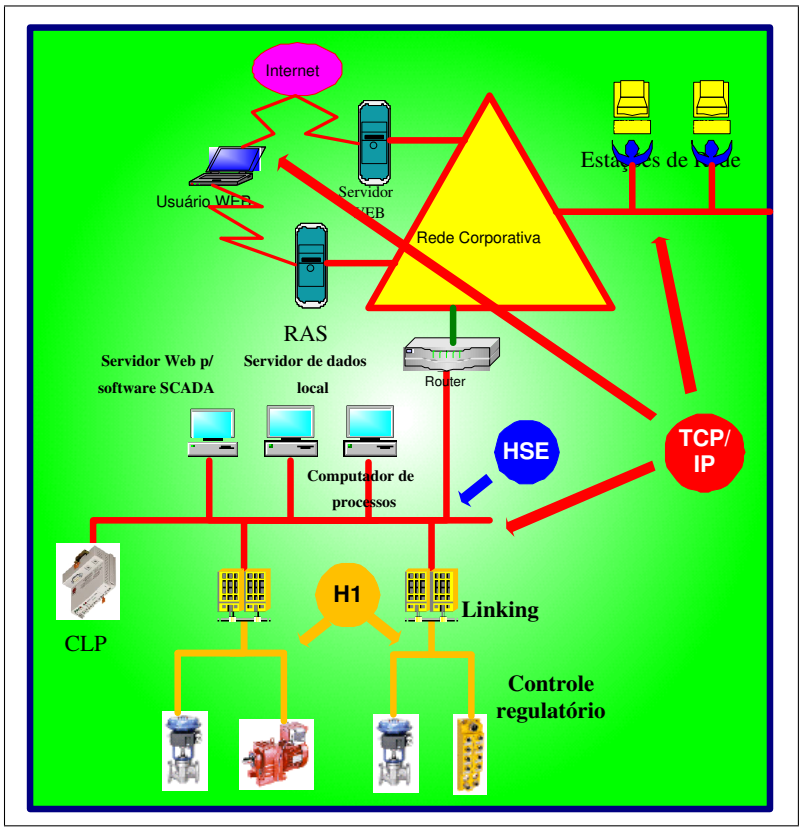


Figura 3.7: Rede corporativa.

Com bastante entusiasmo vemos a migração do foco da tecnologia de comunicação para a TI (Tecnologia de Informação) que está ditando o desenvolvimento no mundo da automação. Protocolos exclusivos de chão de fábrica estão adotando os princípios da TI, através dos protocolos TCP/IP, e estão alcançando grande integração com o nível de gerenciamento da corporação. A indústria de automação vem trilhando os mesmos caminhos do mundo corporativo, aonde a TI radicalmente transformou infra-estrutura, sistemas e processos. A implementação da TI no mundo da automação abre muitas novas perspectivas para comunicação de dados globalmente entre sistemas de automação.

Benefícios ao usuário, através do desenvolvimento das arquiteturas, são a grande motivação para o contínuo desenvolvimento dos *Fieldbus*. Como estes benefícios podem ser vistos:

- Os protocolos estão se tornando abertos em vez de proprietários, assegurando compatibilidade e expansibilidade a longo prazo, ou seja, proteção de investi-

mentos;

- Aumento do desempenho e melhoria da qualidade durante o setup e operação dos processos de automação;
- Redução dos custos na forma de informação regular de diagnósticos evitando paradas não programadas dos sistemas;
- Integração das informações empresariais com o chão de fábrica, fornecendo dados consistentes e on-line;
- Aumento de produtividade e flexibilidade de processos automatizados comparados a tecnologias convencionais.

3.9 Áreas de aplicação

Redes de comunicação digital são usadas em todas as áreas de automação. Em automação fabril, automação de processos e automação predial as redes executam diversas tarefas. Do mesmo modo, há diferenças distintas entre as tarefas executadas pelas aplicações nos diferentes setores industriais os quais possuem características únicas e conseqüentemente requisitos variados. O modo como os dispositivos são conectados, configurados e trocam dados também diferem. Assim sendo, os protocolos foram otimizados visando a atender estas diferentes características. Por exemplo, automação de processo e automação fabril são freqüentemente utilizadas em ambientes severos e perigosos aonde pessoas e máquinas de alto custo estão sujeitas a atmosferas explosivas ou aonde uma paralisação da produção pode causar muitos prejuízos econômicos. Esses equipamentos contrastam significativamente com os da automação predial, por exemplo, aonde manter os custos baixos é essencial (BERGE 2002).

3.9.1 Automação fabril

Linhas de Produção, como automotiva, de engarrafamento ou enlatamento, são predominantemente controladas usando lógica discreta e sensores, por exemplo, é necessário atuar sobre a válvula de engarrafamento rapidamente quando se completa

o nível de líquido em uma garrafa. O tipo de rede ideal é a que utiliza dispositivos de E/S discretos focados em pacote de dados pequenos e sem *overhead*, mas são inadequadas para *download* de grandes arquivos de configuração. Exemplos de protocolos destes tipos de rede são Interbus Loop, AS-i e Seriplex, os quais algumas vezes são denominados de "SENSORBUS" ou "BIT LEVEL BUSES" (Figura 3.8). Outros protocolos mais avançados orientados a lógica discreta são CAN, DeviceNet, LonWorks, Interbus, ControlNet, PROFIBUS DP e PROFIBUS FMS. Esses *Fieldbus* são referenciados como "DEVICEBUS" ou "BYTE-LEVEL BUSES". Automação fabril dá-se através de máquinas com movimentos rápidos, requerendo rápidas respostas em comparação a processos lentos. Tradicionalmente, estas tarefas têm sido executadas por CLPs.

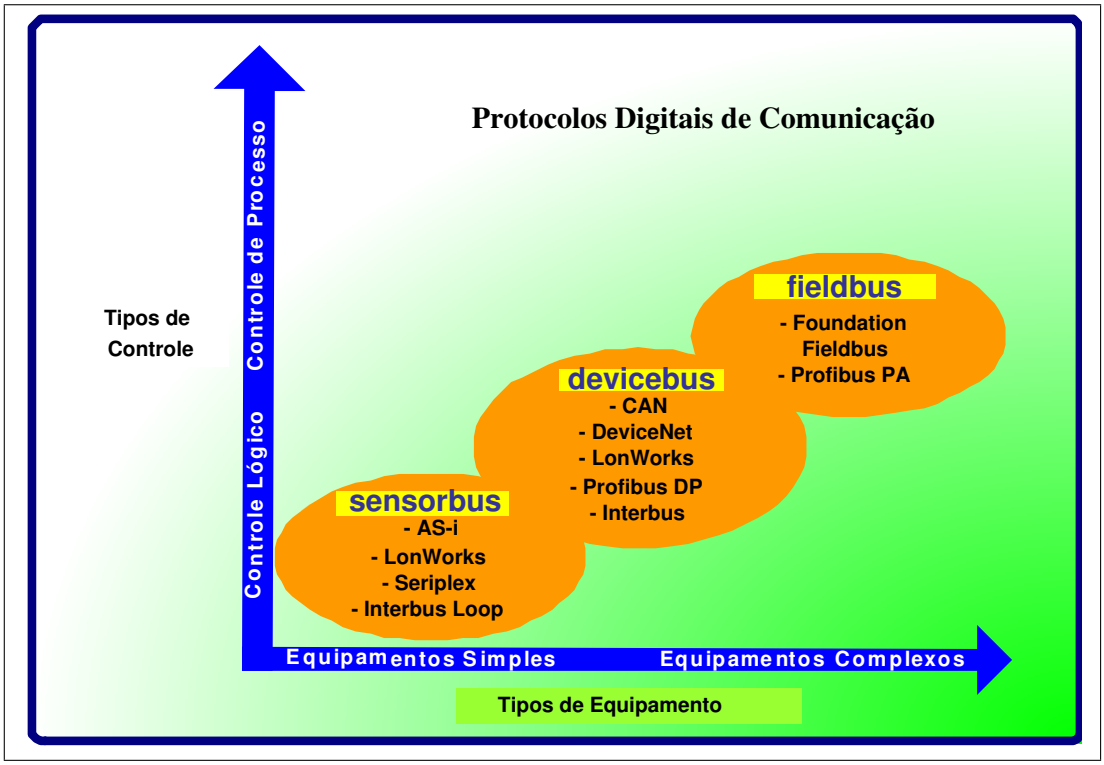


Figura 3.8: Protocolos digitais de comunicação.

3.9.2 Automação de processos

Segmentos de indústria como refinamento, papel e celulose, energia elétrica e química são dominadas por controle regulatório contínuo. Embora as medidas sejam analógicas

(transformadas para digital) há o uso de algum controle discreto. Protocolos relacionadas a processo incluem FOUNDATION Fieldbus, PROFIBUS PA e HART. Estes três protocolos foram especificamente projetados para instrumentação de campo energizada por barramento com parâmetros e comandos pré-definidos para informação de gerenciamento de ativos como identificação, diagnósticos, material de construção, funções de calibração e comissionamento. Em termos de tamanho as redes usadas em automação industrial são consideradas como LAN cobrindo áreas não maiores que um ou dois quilômetros em diâmetro. Automação de processos necessitam de alto desempenho, confiabilidade e operações seguras de planta. Tecnologias de transmissão à prova de falhas são definidas na norma IEC 61158-2. Isto significa que dispositivos de campos em áreas perigosas podem ser incorporados.

3.9.3 Automação predial

Não somente prédios residenciais, mas também as indústrias contêm instalações elétricas. A maior parte dos equipamentos consome energia elétrica, por exemplo, iluminação, aquecimento, ar-condicionado, ventilação e controles de acesso. Estes sistemas devem ser controlados e monitorados tanto quanto aqueles de automação industrial.

O protocolo LonWorks é ideal para automação predial.

3.10 Redes a nível de campo e a nível de *host*

Quando os barramentos de campo começaram a desenvolver, o processo industrial impôs um grande número de requisitos na rede ao nível de campo que não eram encontradas em outros tipos de rede, por outro lado, no nível *host* havia a necessidade de agrupar informações provenientes de tudo que era relacionado à automação.

3.10.1 Redes a nível de campo

Os protocolos dominantes são HART, Foundation Fieldbus H1 e PROFIBUS PA. O protocolo HART é significativamente diferente dos outros dois, pois é uma com-

binação de comunicação digital superposto num sinal convencional 4-20 mA. Desta maneira, o protocolo HART tem sido uma solução intermediária ideal na transição do sinal analógico para digital. HART é compatível com registradores analógicos e controladores enquanto ao mesmo tempo é possível tanto configuração remota quanto diagnóstico usando comunicação digital. O protocolo HART permite muitos dispositivos serem conectados no mesmo cabo, mas esta capacidade é freqüentemente não explorada, devido à pequena velocidade, tipicamente meio segundo por dispositivo. Tanto FOUNDATION Fieldbus H1 e PROFIBUS PA são completamente digitais e usam a mesma fiação, seguindo o padrão IEC 61158-2.

Neste nível, instrumentos aparecem em larga quantidade, freqüentemente centenas ou milhares. Mesmo havendo um limite de dispositivos a serem conectados a uma rede, uma planta de tamanho médio pode ter muitos cabos, mas muito menos que ligações ponto a ponto dos sistemas analógicos. Estas redes foram projetadas tanto para dados como para alimentação dos dispositivos.

Como contensão de custos, estas redes possuem velocidade moderada de transmissão não havendo necessidade de uso de cabos especiais, *hubs*, etc. Então é possível re-usar cabos de instalações antigas que desejem migrar para barramento de campo. Devido a velocidade moderada da rede, os dispositivos conectados não requerem CPUs de alta velocidade, conseqüentemente necessitam de baixo consumo, que implica em baixa queda de tensão ao longo da fiação, permitindo o uso de fiação longa. Outra vantagem deste tipo de barramento é que eles foram projetados para uso em ambientes hostis e eletricamente ruidosos encontrados em campo. Em ambientes de processo perigoso aonde fluídos inflamáveis estão presentes, segurança intrínseca é muitas vezes o método de proteção preferido.

3.10.2 Redes a nível de *host*

No nível *host* a *Ethernet* é o padrão predominante. Há muitos protocolos, como MODBUS TCP, PROFInet, FOUNDATION Fieldbus HSE, etc, projetados sobre o padrão *Ethernet*. Novas aplicações têm sido implementadas além dos dados tradicionais de controle das fábricas, pois uma grande quantidade de dados pode ser trafegada nas redes baseadas no padrão *Ethernet* e a baixo custo. Gráficos de tendência,

diagnósticos remotos, manutenção proativa e configuração são exemplos destas novas aplicações. Além disso, a *Ethernet* é um padrão bem conhecido e amplamente usado no nível corporativo. Um grande número de equipamentos a baixo custo já está disponível.

Mesmo ocorrendo falhas a rede no nível host ainda deve ser funcional, pois toda a planta é operada e supervisionada nesta rede. A *Ethernet* do nível *host* deve ser adaptada para aplicações em ambientes hostis. A rede deve ser tolerante à falhas com o uso de dispositivos redundantes.

As distâncias são menos importantes nas redes do nível *host* devido ao fato de elas estarem confinadas dentro das salas de controle e a distância que a *Ethernet* com cabeamento proporciona são suficientes para atender as necessidades destas redes. Uma vantagem de se adotar o padrão *Ethernet* é a possibilidade de utilizar várias mídias. Assim, o uso de fibras ópticas possibilita a *Ethernet* alcançar longas distâncias.

A rede no nível *host* engloba todos os subsistemas que um processo de automação pode ter. Em adição as funções de controle básico, os sistemas freqüentemente têm unidades de pacote para auxiliar funções como caldeiras ou compressores que já são comprados feitos. Eles possuem seus próprios controles que precisam ser integrados com o resto da planta. Por exemplo, as refinarias têm sistema de desligamento automático (*shutdown*), fábricas de papel e celulose devem possuir *scanner web*, e uma planta química deve ter um sistema de controle avançado. Subsistemas baseados no padrão *Ethernet* podem ser simplesmente conectados ao resto do sistema.

É importante lembrar que o padrão *Ethernet* não proporciona um sistema completo. *Ethernet* específica somente diferente opções de cabeamento para acesso ao barramento da rede. *Ethernet* não especifica o formato dos dados. Mesmo quando usado com outras tecnologias como TCP/IP e UDP o sistema ainda fica incompleto. Muitos desenvolvedores de sistemas de controle têm usado *Ethernet* por muitos anos, mas cada um implementou um formato de dados e funcionalidades diferentes. Mesmo com o uso do TCP/IP, muitas das redes *Ethernet* usadas em sistemas de controle são de fato proprietárias desde que outros dispositivos não podem acessar e interpretar a informação mesmo quando conectadas ao mesmo barramento.

Sistemas para *Ethernet* freqüentemente parecem o que não o são.

Devido aos requisitos opostos, diferentes características de rede são requeridas para o nível de campo e *host*. A rede de campo é lenta e inadequada para o nível *host*, e no nível *host* as redes têm alcance pequeno sendo inadequadas para uso em campo. As redes ao nível de campo executam funções dos protocolos tradicionais para instrumentos inteligentes e subsistemas de E/S, e no nível *host* as redes ocupam lugar das redes de controle e negócio. A tecnologia no nível *host* é a mesma tecnologia das redes de negócio, então elas podem ser integradas sem problemas. Um simples roteador entre as redes assegura desempenho, pois separa o tráfego de comunicação corporativo do tráfego de comunicação *host*.

Para uma integração fácil é importante selecionar uma arquitetura de rede homogênea na qual os protocolos do nível *host* e de campo sejam os mesmos, mas trafegando em diferentes mídias. Isto assegurará transparência e o mínimo de problemas de comunicação e interoperabilidade. Felizmente há protocolos disponíveis que apresentam tal situação. Boas combinações poderiam ser Foundation Fieldbus H1 e HSE ou PROFIBUS PA e PROFInet. Se um outro protocolo proprietário for usado no nível de controle ou em algum outro lugar, importantes funcionalidades e interoperabilidades podem ser perdidas no link entre os instrumentos e operadores. Isto pode forçar a engenharia a investir tempo mapeando parâmetros entre os diversos protocolos.

3.11 Exemplos reais de aplicação da tecnologia *Fieldbus*

Todos estes exemplos de aplicação foram obtidas nas Revistas Controle e Instumentação (CI), da editora VALETE.

Coamo Agroindustrial Cooperativa, Campo Mourão - PR

Objetivo: Modernização da planta de Esmagamento/Extração de soja.

Protocolos utilizados:

- Profibus PA e AS-Interface na instrumentação de campo;
- Profibus DP em dispositivo de proteção e comando de motores da Siemens;
- Protocolo *Ethernet* nas estações de operação e engenharia.

Fonte: CI n. 98.

Schincariol

Objetivo: Medição de Grau Plato (porcentagem em massa de sacarose presente em uma solução) usando transmissor de densidade.

Protocolo utilizado: Foundation Fieldbus.

Fonte: CI n. 98.

Acrinor, Pólo de Camaçari

Objetivo: Sistema de monitoração dos turbocompressores e dos circuitos de intertravamento ,sistema de controle de processo para produção de acrilonitrila.

Protocolo utilizado: Foundation Fieldbus.

Fonte: CI n. 98.

General Motors Corp.

Objetivo: Adotar um padrão para para suas operações de produção de veículos. O padrão deverá fornecer comunicação em tempo real entre os controladores das máquinas GM, os robôs e os equipamentos de controle de processo, assim como fornecerá as informações para os sistemas gerenciais.

Protocolo a ser utilizado: *Ethernet/IP*.

Fonte: CI n. 98.

Highland Spring, Grã-Bretanha.

Objetivo: Engarrafar água mineral.

Protocolos utilizados:

- Profibus-DP para interconexão dos controladores lógicos programáveis e painéis de operação;
- AS-Interface para comunicação dos sensores e atuadores da linha transportadora.

Fonte: C.I - 86

Realcafé, Espírito Santo

Objetivo: automatizar a fabricação de café solúvel.

Protocolo utilizados:

- Foundation Fieldbus para controle e captura dos dados de diagnóstico da instrumentação;
- Devicenet para os inversores, relés inteligentes e válvulas *on/off*;
- HART para instrumentação;
- *Ethernet* e *Modbus* para comunicação com outros sistemas.

Fonte: C.I - 86

Petrobrás, Plataforma de Pargo

Objetivo: Automatizar descarte de água.

Protocolos utilizados:

- *Ethernet TCP/IP* para interligação entre os painéis controladores;

- Profibus para conexão do controlador programável com estações remotas;
- AS-i para conexão das válvulas de retrolavagem via gateway com os controladores programáveis;

Fonte: C.I - 91

Complexo Industrial Ford Nordeste

Objetivo: Produção de carros.

Protocolos utilizados:

- *Ethernet TCP/IP* para comunicação entre CLPs e computadores da linha supervisória;
- Devicenet para para rede de dispositivos entre CLPs, sensores, válvulas e outros equipamentos;
- TimerSolda para os robôs;
- ControlNet nas linhas de transportadores entre CLPs, FlexI/Os e inversores de freqüência;
- SafetyBus como protocolo de segurança.

Fonte: C.I - 91

Tintas Coral, Mauá-SP

Objetivo: Gerenciamento e execução das ordens de serviço na planta de óleos e sintéticos .

Protocolos utilizados:

- *Ethernet* para comunicação com ambiente externo e sistema gerencial;
- Profibus para comunicação das máquinas de controle de campo.

Fonte: C.I - 91

3.12 Outras aplicações dos *Fieldbus*

No processo industrial, funções de engenharia para administração e otimização de dispositivos (MULLER et al. 2003) têm ganho importância quando comparado as funções de controle do processo. O termo *asset optimization* (otimização dos ativos) é freqüentemente usado para descrever técnicas que exploram o completo potencial dos ativos (instrumentos e válvulas, equipamentos mecânicos rotativos, equipamentos de processo e equipamentos elétricos) de uma fábrica. Isto é aplicado particularmente aos dispositivos de campo usados no processo de produção. Os dispositivos de campo eletrônicos atualmente possuem grande capacidade de processamento devido a microcontroladores cada vez mais sofisticados. Isto levou a uma ampla expansão de informações e funcionalidades inovativas, por exemplo, informações de diagnose e manutenção. Assim, uma nova geração de dispositivos de campo contém um conjunto de informações embarcadas que estão sempre crescendo. Este conjunto contém informações do dispositivo, parâmetros e atributos, todos pertencentes a funcionalidades e configuração do dispositivo.

Informações de diagnose e manutenção tem como objetivo principal diminuir o tempo nas atividades de manutenção, pois a concorrência cada vez mais acirrada no mercado não permite às empresas desperdício de tempo.

Neste contexto, as seguintes funções são implementadas nos DICs:

- O perfil do dispositivo deve informar o tipo de dispositivo, variáveis, parâmetros e suas semânticas, como também, métodos de acessar esses dados. O perfil pode ser acessado através do endereço IP do dispositivo. Ao ligar o dispositivo ele manda informações de que está ativo para a ferramenta de gerenciamento, assim como também, no caso de desligamento;

- O acesso às informações tem que ser independente de plataforma. O uso da linguagem XML torna a leitura do perfil independente da plataforma;

- *Status* dos dispositivos de campo continuamente monitorados. Em cada ciclo de teste de uma placa, juntamente com os resultados dos testes, segue o status do dispositivo de campo. Mesmo não havendo produção, num tempo pré-

programado pela ferramenta de gerenciamento, o DIC envia seu status, tanto de manutenção como de produção;

- Otimização do planejamento de manutenção: Os DICs trabalham com ferramentas de diagnose *on-line*. Quando da possibilidade ou ocorrência de algum problema, os DICs notificam a engenharia necessidade de planejamento de manutenção ou troca do dispositivo. O tipo de manutenção predominante é o pro-ativo. É a manutenção mais rentável, pois se consegue chegar no limite de utilização do aparelho. Em (BERGE 2002) são citados os diversos tipos de manutenção;

- Reconfiguração do sistema: Caso o DIC seja utilizado em outra linha de produção, ele detectará que foi trocado o módulo eletro-mecânico de acesso as placas a serem testadas e requisitará do *Manufacturing Execution Systems* (MES) o novo programa que será transferido automaticamente.

No primeiro contato com ferramentas de gerenciamento de ativo, as empresas verificam que o custo de investimento em tais tecnologias é alto, mas se forem considerados os tempos gastos por paradas de manutenções não programadas, o investimento valerá a pena (CI-86 2003). E esta é apenas a ponta do "*iceberg*" para uma nova aplicação denominada "Monitoramento Contínuo Baseado em *Internet* (MCR/BI)".

3.13 Perspectivas

Inicialmente o foco estava na tecnologia de comunicação. Nos dias de hoje busca-se a centralização em Tecnologia de Informação (TI). Os sistemas modernos de *Fieldbus* adotaram os princípios de TI para integração com o nível de gerência das corporações. Claramente percebe-se que a automação industrial segue os mesmos passos da automação de escritório (comercial), aonde a TI revolucionou processos, infra-estrutura e sistemas.

Comparando os sistemas industriais com o desenvolvimento da informática, notamos uma semelhança muito forte, como mostra a Tabela 3.1.

O uso da *Ethernet* está abrindo novas perspectivas na comunicação de dados entre sistemas de automação a nível global. Além disso, o uso da *Ethernet* determinística em alta velocidade, associada com o tipo de mídia da rede será capaz de resolver a maior parte dos problemas industriais. Por isso, *Fieldbus* consagrados, como PROFIBUS, passam a incorporar o protocolo *Ethernet* e oferecem novos protocolos como o PROFINET.

Tabela 3.1: Sistemas industriais X sistemas comerciais.

Informática	Sistemas Industriais
Computadores caros e operando sozinhos.	Controladores isolados.
Terminais burros conectados a um computador central.	Sensores/atuadores conectados a um elemento centralizador (CLP, PC..).
Computadores ligados a outros computadores.	Dispositivos inteligentes ligados a outros dispositivos inteligentes.
Computadores com sistemas operacionais diferentes (alguns abertos) ligados através de *Internet*, *Intranet*,....	Sistemas abertos ligados através de uma rede *Ethernet*, usando uma linguagem padrão como XML.

A grande tendência dos dispositivos industriais é que sejam sistemas abertos e integrados, em vez de sistemas proprietários, pois asseguram compatibilidade e expansibilidade, ou seja, proteção dos recursos investidos.

3.14 Conclusão

Embora tenha havido muitos problemas na busca de um padrão *Fieldbus*, esta tecnologia trouxe muito mais benefícios que a simples economia de cabo em relação aos sistemas analógicos tradicionais com laço de corrente 4-20 mA. Possibilitou o aumento da comunicação tanto horizontalmente no nível de campo como também verticalmente através de simultâneos níveis de hierarquia. A possibilidade de adquirir mais informações dos sensores e atuadores ocasionou o desenvolvimento de muitas aplicações como auto-diagnose, monitoração remota, etc. Estas aplicações juntamente com uso de segurança intrínseca permitem as fábricas trabalharem com uma margem de dano as pessoas, ambiente e equipamentos bem pequena.

Como os protocolos *Fieldbus* foram desenvolvidos para determinadas áreas es-

pecíficas é necessário ter bons conhecimentos das características de cada um deles. Somente após analisar criteriosamente a aplicação é que se deve definir o melhor protocolo a ser usado. O uso de protocolos não específico para determinada área acarretará perda de funcionalidade e desempenho.

Não há um único protocolo que satisfaça todos os requerimentos industriais. Implementar uma solução híbrida normalmente explora o máximo de cada barramento e freqüentemente é a melhor solução.

Capítulo 4

Protocolos de comunicação

No momento há mais de 50 diferentes nomes de *Fieldbus* disponíveis como Fondation Fieldbus (FF), Profibus, Interbus, WorldFip, LonWorks, CAN e muitos outros. Embora o *International Electrotechnical Commission* (IEC) tenha aprovado a proposta IEC61158 no final do ano de 1999 para internacionalização de um *Fieldbus* padrão, atualmente compreende oito tipos de protocolos. Com tantos *Fieldbus* é normal que os usuários fiquem confusos na escolha do *Fieldbus* que satisfaça a maior parte dos requisitos de um sistema (YABIN & NISHITANI 2003).

Por outro lado, a tecnologia *Fieldbus* e os produtos baseados no padrão IEC61158 estão disponibilizando mais funcionalidades em instrumentos e sistemas de controle em diferentes áreas devido as suas vantagens. As vantagens dos barramentos de campo são ilimitadas, mas devido à existência de muitos protocolos, o usuário não se sente seguro em investir em um deles e não obter o retorno esperado. As tecnologias existem e precisam ser usadas. Cabe ao interessado estudar as várias opções e definir por aquela que atende o maior número de necessidades do seu sistema. Nem sempre uma tecnologia é tão abrangente que satisfaça todos os requisitos do sistema, por isso, é necessário ter o conhecimento se estas tecnologias podem trocar dados entre si com uso de gateways (MANTOVANI 1998). Tecnologias que podem trocar dados entre si podem formar os sistemas híbridos, nos quais podemos explorar o que de melhor existe em cada protocolo.

Neste capítulo estudaremos alguns protocolos industriais pois podem ser a melhor solução para determinadas aplicações, principalmente em automação de sistemas

de controle e instrumentação. Utilizar um dos protocolos industriais foi uma das alternativas consideradas em nosso estudo de caso.

4.1 AS-i (*Actuator Sensor - Interface*)

4.1.1 Introdução

O protocolo AS-i (AS-I 1998) (AS-I 2001*a*) (AS-I 2001*b*) foi desenvolvido em 1991, por um consórcio de 11 companhias Euroupéias, como uma alternativa menos custosa para substituir a fiação para os sensores e atuadores (normalmente topologia tipo árvore). Tem provado ser extremamente confiável após anos de uso em numerosos setores industriais. Esta tecnologia está proposta no padrão IEC 62026-2.

A idéia original era desenvolver uma interface serial digital para conexão de sensores e atuadores com CLPs, PCs e CNCs através de um simples cabo de 2 vias que transmitisse dados, controle, configuração da arquitetura do sistema, energia para os dispositivos e monitoração da rede, fornecendo uma solução completa ao sistema.

Outros requisitos foram propostos:

- Sensores/atuadores inteligentes poderiam ser ligados e configurados via AS-i;
- Baixo custo de implantação;
- Interoperabilidade. Troca de dados entre diferentes produtos de diversos fabricantes;
- Em adição aos erros de comunicação, haverá o reconhecimento de erros localizados em periféricos, como curto-circuito, sobrecarga e ausência de tensão.

É usado no nível inferior dos barramentos de campo e controle, exatamente no SensorBus (Figura 4.1), focalizando os dispositivos discretos , suas interligações e controle. Os dispositivos analógicos são suportados, mas exigem tempo maior da rede.

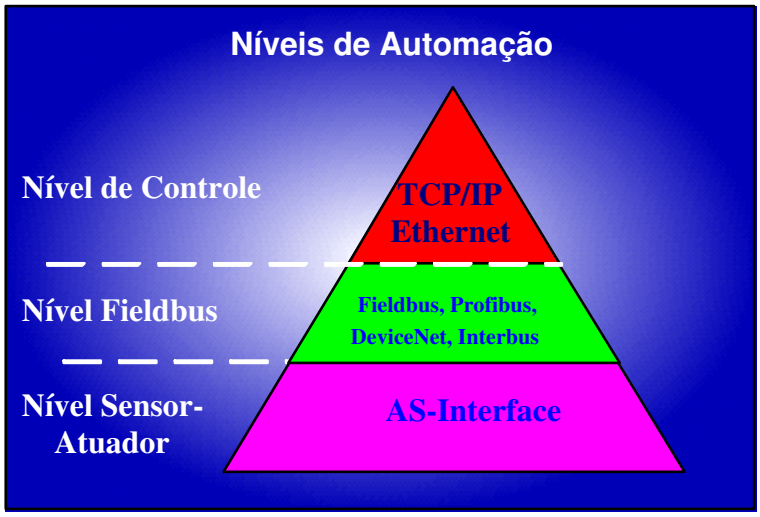


Figura 4.1: Níveis de rede para automação.

4.1.2 Características

Todos os elementos periféricos são conectados ao cabo AS-i. O elemento básico é o *chip slave* (o qual pode estar em um módulo ou sensor/atuador), através do qual os sensores e atuadores são conectados ao cabo AS-i. Em cada ciclo, 4 *bits* de informações são transferidos serialmente de cada escravo para o mestre. Tipicamente, o cabo AS-i é um cabo especial perfilado, composto de dois fios que transporta simultaneamente dados e alimentação para os elementos da rede, é denominado de *Yellow flat cable*. Existem também os *Black flat cable* e *Red flat cable* para tensões de 60 e 230 VAC respectivamente. A conexão é feita através de conectores vampiros que perfuram o isolamento, atingindo os fios internos. Como o isolante é autoregenerativo, não há necessidade de cuidados especiais após a retirada dos conectores. Por questões de segurança, atuadores conectados a paradas de emergência devem possuir fontes externas.

Benefícios:

- **Simplicidade**: Requer somente um simples cabo para conectar módulos de E/S de diferentes fabricantes. Os usuários não necessitam de conhecimento de sistemas de barramento e protocolos de comunicação. Não utiliza resistores terminais, arquivos ou *softwares* de configuração;

- **Redução dos custos**: Instalação e custos de operação são reduzidos em mais de 40% em relação aos cabos tradicionais de rede. Os *flats cables* eliminam a enorme quantidade de fios presentes nos sistemas discretos convencionais, poucos fios reduzem a quantidade de conduítes e custo de instalação;

- **Desempenho**: Tempo máximo de ciclo em $5\,ms$ para uma rede completa com 31 escravos. Isto torna a rede determinística, atendendo requisitos em tempo real para a maior parte dos controles e processos;

- **Confiabilidade Operacional**: Alta mesmo em ambientes industriais severos, provendo excelentes diagnósticos;

- **Fácil Expansibilidade**: A adição de periféricos é simples comparada a sistemas mais complexos. Conectar o módulo na fiação, endereçar e o sistema estará apto a receber o próximo dispositivo. AS-i suporta topologia Árvore, Barramento, Anel ou Estrela com até 100m de cabo ou 300m com repetidores;

- **Extrema Flexibilidade**: A expansão do sistema pode ser feita adicionando-se novos dispositivos simplesmente conectando o *flat cable* para cada novo segmento. Isto facilita a montagem e desmontagem para objetivos de transporte. Módulos de E/S podem ser movidos ao longo da rede sem afetar o programa de controle.

A Rede AS-i é do tipo mestre-escravo e utiliza somente um mestre por rede para controlar a troca de dados. O mestre chama cada escravo seqüencialmente e espera pela resposta. O protocolo AS-i devido ao fato de usar um formato fixo de transmissão elimina a necessidade de processos complexos para controle de transmissão ou para identificar tipos de dados. Isto permite uma varredura cíclica de 31 escravos em $5\,ms$.

Dois tipos diferentes de escravos, num total de 31, podem ser conectados ao Sistema AS-i mestre/escravo. O primeiro tipo é um módulo que permite conexão de atuadores e sensores de 24VDC. Estes módulos podem ser de perfil de usuário (IP67) ou perfil de aplicação (IP20) (4 entradas e 4 saídas) com no máximo de 248 participantes binários por rede. O segundo tipo é de sensores e atuadores dedicados.

Cada uma dessas unidades possui até 4 entradas e 4 saídas. Muitos destes escravos possuem dispositivos de indicação de falha que podem ser lidos pelos controladores.

Cada um desses dispositivos possui um único endereço na rede, de 1 a 31. O endereço pode ser trocado a qualquer tempo e é armazenado internamente numa memória não volátil e pode ser programado por um PLC usando bloco de função via Mestre AS-i ou dispositivos especiais de endereçamento.

Fontes AS-i provêem energia para os módulos de E/S e mestre. Cada módulo consome entre 15 a 100 mA e o mestre aproximadamente 100 mA. Cada módulo de entrada AS-i pode fornecer até 200 mA para os sensores, em tensão de 20 a 30VDC. Se a fonte para o sensor ou atuador for curto-circuitada ou mais corrente estiver sendo consumida que a típica, o módulo automaticamente desconecta-se da rede AS-i durante a falha, a qual é reportada ao mestre.

4.1.3 Comunicação

A troca de dados síncrona é inicializada pelo mestre. O mestre pode ser um CLP, um PC ou gateway conectado a um outro barramento. O mestre interroga um escravo e espera por uma resposta. Se uma resposta for recebida, ela é gravada numa tabela interna, aonde o endereço de todos os escravos e seus perfis são armazenados. O perfil de um nó é um número determinado pela especificação AS-i que representa um tipo particular de nó. Um módulo de 4 entradas, por exemplo, tem perfil 0.0, enquanto um sensor indutivo tem perfil 1.1. O processo de inicialização é automático e não requer *software* adicional. Com *software* adicional, o mestre pode detectar erros de endereçamento e reportá-los ao controlador.

Uma requisição AS-i contém chamada ao escravo, chamada do parâmetro e diagnóstico da chamada. Posterior a requisição, segue-se uma pausa (3 a 10 *bits*), a resposta do escravo e sua pausa (Figura 4.2) . Se a resposta do escravo não ocorrer, o mestre reconhece a falha e repete a requisição. Se a segunda requisição falhar, o processo passa para o próximo endereço. O mestre tentará acessar endereços em falha duas vezes nos próximos dois ciclos. Se mesmo assim não houver resposta, o mestre indica *bit* de erro de configuração. A rede fica ociosa após a resposta do escravo e a próxima chamada do mestre (2 *bits*). A eficiência é de 32% pois a taxa

de comunicação de 167 kbit/s passa a ser de 53,3 kbit/s. A pausa do escravo pode ser estendida desde que o tempo de ciclo não ultrapasse 5 ms.

Figura 4.2: Comunicacao AS-i.

Formato da mensagem

O *frame* do mestre possui 14 *bits*, enquanto o do escravo possui 7 *bits*. Na Figura 4.3 pode-se ver a estrutura dos mesmos.

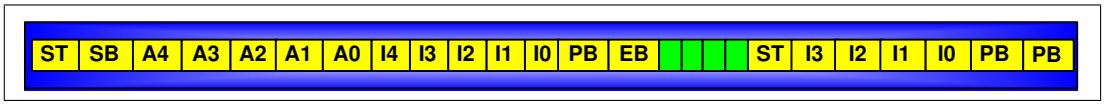


Figura 4.3: Formato da mensagem AS-i.

Na Tabela 4.1 temos a descrição dos campos.

Existem nove tipos diferentes de mensagens. São elas:

- **Troca de Dados.** Solicita valores de entrada de um escravo (0 0 A4 A3 A2 A1 A0 0 I3 I2 I1 I0 PB 1);
- **Escrita de Parâmetros.** Configuração dos parâmetros internos do escravo (0 0 A4 A3 A2 A1 A0 1 I3 I2 I1 I0 PB 1);
- **Deleta Endereço Operacional.** Faz com que o escravo passe a operar com endereço zero (0 1 A4 A3 A2 A1 A0 0 0 0 0 0 PB 1);
- **Atribuição de Endereço.** Novo endereço para o escravo, usado juntamente com a mensagem "Deleta Endereço Operacional" (0 0 0 0 0 0 0 A4 A3 A2 A1 A0 PB 1);

Tabela 4.1: Descrição dos campos AS-i.

Mestre	ST	Início	Início do *frame* de requisição do mestre.Sempre valor lógico 0.
Mestre	SB	Controle	0 - Dados, parâmetro e endereço.1 - Comandos.
Mestre	A0 a A4	Endereço	Endereço do escravo
Mestre	I0 a I4 a A4	Dados	Informação a ser requisitada
Mestre	PB	Paridade	A soma dos *bits* deve ser par, não considerar os ST e EB.
Mestre	EB	Fim	Fim do *frame*. Valor lógico 1.
Escravo	ST	Início	Início do *frame* do escravo.
Escravo	I0 a I4	Dados	Informação ligada a valores discretos de entrada, saída ou A/D.
Escravo	PB	Paridade	A soma dos *bits* deve ser par, não considerar os ST e EB.
Escravo	EB	Fim	Fim do *frame*. Valor lógico 1

- **Inicialização do Escravo.** Reset forçado do escravo (0 1 A4 A3 A2 A1 A0 1 1 1 0 0 PB 1);
- **Leitura de Configuração de E/S.** O mestre ler a configuração dos pinos, se são entradas ou saídas (0 1 A4 A3 A2 A1 A0 1 0 0 0 0 PB 1);
- **Leitura do ID do Escravo.** Ler perfil do dispositivo (0 1 A4 A3 A2 A1 A0 1 0 0 0 1 PB 1);
- **Leitura de Status.** Ler dois flags do escravo. S0 volatile_ address e S3 read_erro_non_volatile
  _memory (0 1 A4 A3 A2 A1 A0 1 1 1 1 0 PB 1);
- **Ler e Apaga Status.** Lê e apaga em seguida o *buffer* de status de um escravo (0 1 A4 A3 A2 A1 A0 1 1 1 1 1 PB 1).

A Nova especificação V2.1

Dentre as principais vantagens da nova especificação, temos o aumento de escravos acessados, de 31 para 62, mas com conexão de 4 entradas e 3 saídas. Os endereços passam a ser "A" e "B". Em adição aos erros de comunicação, pode-se reconhecer erros de periféricos como curto-circuito, sobrecarga e ausência de fonte de alimentação auxiliar. O tempo de ciclo ficou em $5\,ms$ para redes que não utilizam Técnica A/B e de $10\,ms$ para as que utilizam.

A nova especificação introduziu o conceito *AS-Interface Safety at Work*. Este conceito permite conectar componentes de segurança como chaves de emergência e barreiras de segurança diretamente a rede AS-i enquanto o sistema principal continua operacional.

4.1.4 Aplicações

Na Figura 4.4 [http://www.as-interface.com/presentations.asp] tem-se um sistema tradicional constituído de sensores de proximidade e motores de partida. Como as ligações são ponto a ponto, do painel de controle aos sensores e motores, a quantidade de cabos é grande, como também de cartões de E/S.

Na Figura 4.5 [http://www.as-interface.com/presentations.asp] temos o Sistema AS-i. Apesar do *hardware* AS-i ser mais caro, estimamos uma economia de aproximadamente 40% neste sistema em relação ao tradicional, além das vantagens mencionadas anteriormente.

4.1.5 Conclusão

O AS-i foi desenvolvidos para comunicação entre dispositivos discretos, com determinismo para aplicações em tempo real, área que outros barramentos não atingem com tal desempenho. Elimina a necessidade de cartões de E/S e alto custo de fiação.

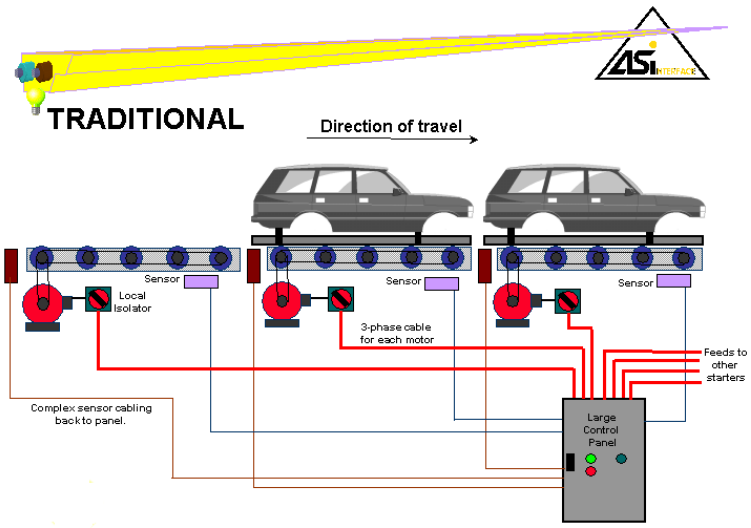


Figura 4.4: Sistema tradicional ponto a ponto.

4.2 CAN

4.2.1 Histórico

A rede CAN (*Controller Area Network*) foi desenvolvida originalmente na Europa, por Robert Bosch para utilização na indústria automobilística (EFEI 1998), (CIA 2003) e (SCOTT & BUCHANA 2000). Devido ao sucesso nos veículos automotivos sua aplicação foi estendida, encontrando espaço também na indústria de dispositivos de controle como sensores e atuadores. O CAN é um padrão internacional de comunicação serial versátil em tempo real, com critério de prioridade rígido, baseado na norma ISO11898 e ISO11519-2.

Devido ao fato de ser um protocolo padrão, atraiu fabricantes de semicondutores que desenvolveram dispositivos CAN a preços competitivos, deste modo, temos periféricos inteligentes que liberam a CPU para outras tarefas.

4.2.2 Características

O protocolo CAN possui as seguintes características:

- As mensagens são recebidas por todos os pontos, havendo uma varredura de

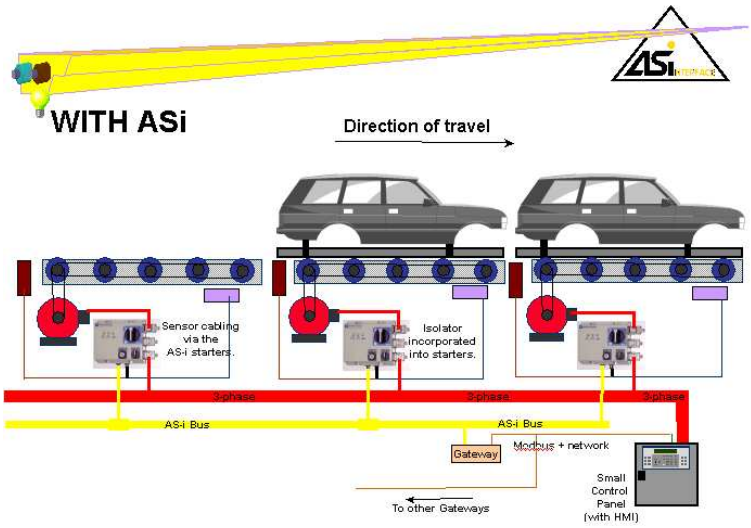


Figura 4.5: Sistema com AS-i.

erro antes de ela ser aceita;

- Comprimento da mensagem bruta: 47 - 150 *bits*. Utilizável: 11 - 94;
- Nós: máximo de 110;
- Distância/velocidade: 50 metros com velocidade máxima de 1Mbit/s e 500m com velocidade máxima de 100kbit/s;
- Prioridade: definida pelo usuário, assim a latência máxima é garantida para as prioridades máximas com tempo abaixo de $1\,ms$;
- Erros: Detecção e sinalização implementadas dentro do protocolo CAN. As mensagens corrompidas são re-transmitidas automaticamente;
- Protocolo multi-mestre que utiliza *NON-Destructive Collision Resolution*, ou seja, em caso de colisão o *bit* 0 no identificador é o dominante, definindo assim a prioridade dos dispositivos;
- Sistema flexível, de baixo custo e alto desempenho;
- Em sistemas embarcados, o protocolo CAN é o mais aceito na camada de aplicação;

- Utilizado em ambientes militares, onde as condições de trabalho são bastante agressivas.

4.2.3 Princípio da troca de dados

O protocolo CAN está baseado num mecanismo de comunicação *broadcast*, usando *Carrier Sense, Multiple Access with Collision Detect* (CSMA/CD) com *Non-Destructive Bitwise Arbitration* para evitar conflitos de acesso ao barramento (EFEI 1998) e (CIA 2003). Neste tipo de protocolo, dita orientada à mensagem (não há endereçamento de estações), cada mensagem é rotulada através de um identificador, o qual é único, e define tanto o dispositivo como também a prioridade da mensagem que é usada como árbitro em caso de uso simultâneo (quanto menor o valor do identificador, maior será a sua prioridade). Isto é muito importante quando várias estações competem para acesso ao barramento. Deste modo, toda estação transmissora efetua uma escuta (*hearback*) do barramento para verificar se o que ela transmitiu foi efetivamente colocado no barramento. Desta forma, ela compara o valor lógico que ela própria lançou no barramento, *bit* a *bit*, com o valor que efetivamente aparece. Isto é feito colocando-se os identificadores das mensagens numa lógica "E". O estado dominante "0" sobrepõe-se a um estado dominante "1". Caso duas estações ou mais tenham iniciado a transmissão simultaneamente, haverá um momento em que o *bit* zero da estação com maior prioridade predominará e a estação de menor prioridade, percebendo a colisão, interrompe imediatamente a transmissão, elas passam a ser receptores da mensagem e não tentam retransmitir até que o barramento esteja disponível novamente.

Como se observa na Figura 4.6, o nó dominante é o 2, ou seja, o de menor prioridade.

Isto permite diminuir o tempo de latência para sistemas que trabalham em tempo real.

Como conseqüência do esquema de identificadores, a flexibilidade de configuração dos sistemas é muito grande. Fica fácil adicionar estações puramente receptivas sem necessidade de modificação no *software* ou no *hardware*. Isto permite o conceito de eletrônica modular e também permite múltipla recepção e sincronização de um

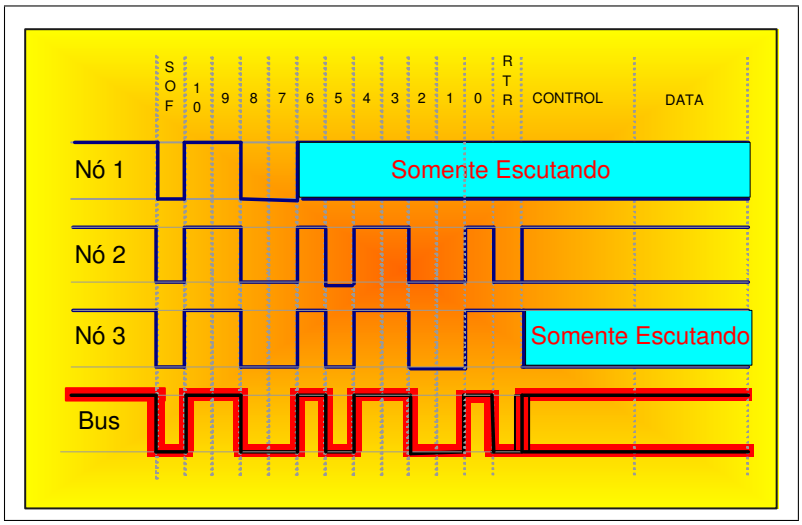


Figura 4.6: Prioridades no barramento CAN.

processo distribuído: os dados são transmitidos pela rede de tal modo que não é necessário para cada estação saber quem produziu o dado.

4.2.4 Formato do *frame* das mensagens

Existem dois formatos de *frame*: CAN2.0 A, conhecido como *Standard* CAN (Figura 4.7 (CIA 2003)) e o CAN 2.0B, conhecido como *Extended* CAN (Figura 4.8 (CIA 2003)). A diferença entre eles é o comprimento do identificador, 11 *bits* para o A e 29 *bits* para o B.

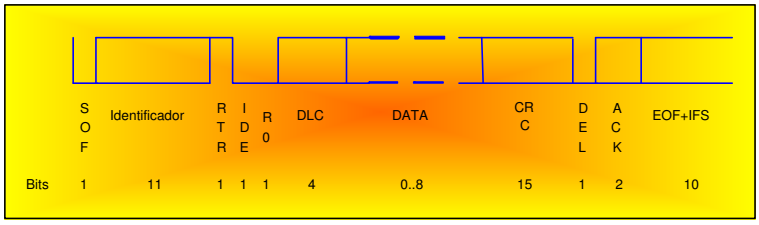


Figura 4.7: *Frame* CAN 2.0A com identificador de 11 bits.

Frame CAN 2.0A (*Standard* CAN)

- **SOF (*Start of Frame*)**. Indica o início do *frame* para obtenção do sincronismo. Constituído de 1 *bit*;

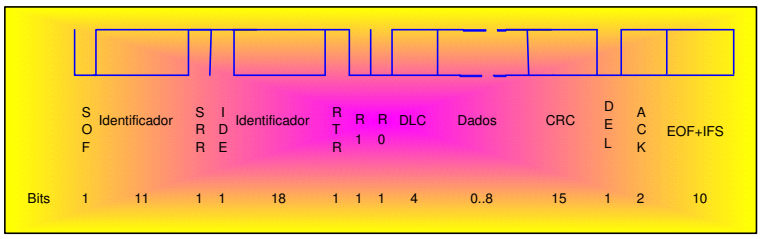


Figura 4.8: *Frame* CAN 2.0B com identificador de 29 bits.

- ***Arbitration field***. Campo de Decisão. Consiste do identificador, constituído de 11 *bits* e do *bit Remote Transmission Request* (RTR) que distingue se a mensagem é um *data frame* (caso seja 0) ou um pedido de envio de dados de outro nó do barramento, denominado de *remote frame* (caso seja 1);

- **IDE (*Identifier Extension*)**. Distingue entre o *frame* padrão e o *frame* estendido;

- ***Data frame* (5 *bits*)**. R0 reservado. O *Data Length Code* (DLC) contém quatro *bits* e indica o número de *bytes* que seguem no campo Data. Se a mensagem é um *frame* remoto, o DLC contém o número de *bytes* de dado requisitado;

- ***Data field***. Pode ter até 8 *bytes*;

- **CRC (*Cyclic Redundant Check*)**. É constituído de 15 *bits* e um *bit* recessivo delimitador. Garante a integridade do *frame*;

- **ACK (*Acknowledge*)**. Campo de reconhecimento. O nó que está transmitindo envia para a saída um *bit* recessivo e é sobrescrito como dominante por um receptor que recebeu a mensagem corretamente. As mensagens corretas são reconhecidas pelos receptores pelo resultado do teste de aceitação. O nó transmissor confere a presença deste *bit* e retransmite a mensagem caso não seja detectado o reconhecimento;

- **EOF (*End Of Frame*)**. Constituído de sete *bits* recessivos. Fim da mensagem;

- **IFS (*Intermission Frame Space*)**. Três *bits* recessivos que permitem os controladores se preparem para a próxima tarefa.

CAN Extended Frame

Promove a compatibilidade com outros protocolos de comunicação seriais automotivos e promove a compatibilidade com a versão 2.0 A.

Uma mensagem no formato CAN *extended frame* é muito parecida com a mensagem no formato *Frame* CAN 2.0A (*Standard* CAN). A diferença é no tamanho do identificador. O identificador é formado pelo identificador base (11 *bits*) e pelo identificador de extensão (18 *bits*). A distinção entre os dois formatos é feita pelo *bit* IDE que é transmitido como dominante no formato padrão e recessivo no formato estendido. O *Bit Substitute Remote Request* (SRR) é sempre transmitido no formato recessivo para garantir que quando os dois formatos coexistam no mesmo barramento, a mensagem no formato padrão tem prioridade sobre a do formato estendido.

Controladores CAN que suportam mensagens no formato estendido são também capazes de enviar e receber mensagens no formato padrão. Quando controladores CAN entendem somente o formato padrão numa rede, então somente mensagens no formato padrão podem ser enviadas. As mensagens no formato estendido podem ser mal interpretadas. Alguns controladores somente suportam o formato padrão, mas reconhecem e ignoram o formato estendido (versão 2.0 B passiva).

Detecção e sinalização de erros

Ao contrário de outros sistemas de barramento, o protocolo CAN não usa o reconhecimento de mensagens, em vez disso sinaliza erros tão logo eles ocorram. Três mecanismos são implementados ao nível das mensagens para detecção de erros:

- **CRC (*Cyclic Redundancy Check*).** O CRC protege as informações no *frame* pela adição de *bits* de cheque no final da transmissão. No receptor os *bits* de cheque são comparados com a soma mascarada dos *bits* recebidos. Se eles não concordarem há um erro de CRC;
- ***Frame checks.*** Este mecanismo verifica a estrutura do *frame* transmitido checando os *bits* contra um formato e tamanho fixo. Erros detectados pelo

cheque do *frame* são designados erros de formatação;

- ***Acknowledgement Error Checks (ACK).*** Como mencionado o *frame* transmitido é recebido por todos os nós. Se um dos nós reconhece a mensagem, um dos *bits* do ACK SLOT passa de recessivo (nível lógico 1) a dominante (nível lógico 0). Se o transmissor não reconhecer o *bit* dominante, um erro de ACK é indicado e retransmite a mensagem novamente.

O protocolo CAN também implementa dois mecanismos para detecção de erro ao nível de *bit*:

- ***Bit Monitoring.*** A habilidade do transmissor de detectar erros é baseada na monitoração dos sinais de barramento. Cada estação que transmite também observa o nível do bus e assim detecta diferença entre o *bit* enviado e o *bit* recebido. Isto permite uma detecção confiável de erros globais e erros locais na transmissão;

- ***Bit stuffing.*** A codificação dos *bits* individuais é testada ao nível de *bit*. A representação do *bit* usado pelo CAN é a codificação *Non Return to Zero* (NRZ), a qual garante máxima eficiência na codificação do *bit*. Os degraus de sincronização são gerados pelo *bit* de preenchimento (stuffing). Isto significa que após enviar cinco *bits* iguais consecutivos, o transmissor insere no *bit stream* um *bit stuff* com um valor complementar, o qual é removido pelo receptor. Se o receptor não complementar este *bit*, a mensagem é re-transmitida.

Se um ou mais erros são descobertos por uma das estações usando o mecanismo acima, a transmissão é abortada pelo envio de um *error flag*. Isto previne outras estações de aceitarem a mensagem e assim assegurar a consistência dos dados através da rede. Após a transmissão de uma mensagem errônea que foi abortada, o transmissor envia novamente a mensagem automaticamente. Há outra vez competição pelo acesso ao barramento.

O protocolo CAN provê mecanismos para distinguir erros esporádicos de erros permanentes e falhas locais nas estações. Isto é feito por uma estimativa estatística com o propósito de reconhecimento de uma estação defeituosa a qual entra num modo de operação no qual o resto da rede não seja afetado negativamente.

4.2.5 Controladores CAN

Muitos fabricantes de microcontroladores incorporaram o protocolo CAN. Isto prova a grande aceitação deste protocolo. São eles:

Tabela 4.2: Controladores CAN.

Philips	82C200	1.0	Basic CAN. Stand-alone.
Philips	82C15082C15082C150	1.0	SLIO
Motorola	MCAN	COP884	Basic CAN. On-chip
Nat.Semi	MCAN	2.0A	Basic CAN. On-chip
Siemens	81C90	2.0A	Full CAN.Stand-Alone
Intel	82527	2.0B	Full CAN.Stand-Alone
NEC	UPD72005	2.0B	Full CAN.Stand-Alone
Siemens	SABC167C	2.0B	Full CAN. On-chip
Motorola	TOUCAN	2.0B	Full CAN. On-chip
Motorola	MSCAN	2.0B	Basic CAN.On-chip

4.2.6 Comentários

No barramento CAN as colisões são detectadas rapidamente e o nó que possui maior prioridade é que envia a mensagem primeiramente. É um protocolo robusto que trabalha muito bem em ambientes ruidosos, como carros, ou equipamentos que não estejam operando corretamente.

A taxa relativamente alta de transmissão do barramento CAN permite uma grande quantidade de informações serem passadas entre instrumentos e controladores.

Um controlador secundário pode monitorar um controlador primário e removê-lo do barramento se ele não estiver operando corretamente.

Os controladores CAN podem ficar em stand-by e responder rapidamente a eventos, isto é bom para o uso em carros aonde o consumo deve ser moderado.

4.3 *LonWorks*

4.3.1 Histórico

A tecnologia de controle de rede LONWorks (*Local Operating Network*) , norma EIA 709.3, protocolo LonTalk (modelo OSI), desenvolvida pela companhia norte-america Echelon Corp. propõe uma plataforma única para produtos inteligentes aplicado em sistemas de controle distribuído a fim de que fabricantes distintos destes dispositivos possam manter total interoperabilidade entre eles. É uma tecnologia com controle de rede industrial aberto e descentralizado (ANSI/EIA 709.1) desde 1997 (MANTOVANI 1998) (ZUJUN, HONGMEI, LI & BAOQING 2002).

4.3.2 Características

As principais características do protocolo LonTalk são:

- Interoperabilidade garantida com variáveis de rede endereçáveis chamadas *Standard Network Variable Types* (SNVT's);
- Permite criação de redes com até 32385 nós;
- Trabalha com conceito mestre-escravo e dispositivos de campo inteligentes e distribuídos;
- Tecnologia de cabeamento flexível e robusta *Flexible Free Topology* (FFT), FTT-10 A e FFT-31;
- Taxa de transmissão: 1,25 Mbit/s em cabo coaxial (até 2700m), fibra ótica ou par trançado. 78 kbit/s em *Power link* (par trançado com alimentação embutida). 10 kbit/s em *power line* (rede elétrica). 4,8 kbit/s em RF;
- Opera em *Intranet* ou *Internet* (*LonWorks Network Service* (LNS));
- Resposta em tempo real: 7 a 13 ms;
- Fiação e custos de instalação reduzidos;

- Melhor resposta do sistema pela eliminação de congestionamento do meio de comunicação. Utiliza algoritmo *P-persistente Carrier Sense Multiple Access* (P-CSMA) (ZUJUN et al. 2002);

- Desenvolvimento de produtos simples, pois as ferramentas de desenvolvimento permitem integração da arquitetura;

- Sistema simples de instalar. Manutenção, diagnóstico e reparações mais rápidas em comparação com outras arquiteturas que suprem as mesmas necessidades;

- Bibliotecas em Visual Basic, Delphi e Visual C++ acessam as variáveis da rede, facilitando a implantação de *softwares* de controle e supervisão.

4.3.3 Princípio da troca de dados

Network Variable (NV) é a mais importante tecnologia em LonWorks, a qual simplifica a comunicação de rede em desenvolvimento de parâmetros . As NVs são linhas de código que definem entradas e saídas de dispositivos , como temperatura, valor de uma chave ou posição de um atuador (SNOONIAN 2003).

Com esta tecnologia, é fácil implementar redes de dispositivos inteligentes usando o protocolo LonTalk. É fácil, pois comunicação, controle, gerenciamento (*scheduling*) e suporte de E/S estão em um circuito integrado VLSI batizado Neuron, de custo aproximado de três (3) dólares. O projetista pode concentrar-se somente na aplicação, programando em linguagem Neuron C (baseada no C Ansi) orientada a eventos, assim a tarefa de desenvolvimento fica bastante simplificada. Soma-se a isto, um vasto suporte de *hardware* e *software* fornecido pelo fabricante Echelon.

O Neuron 3120 é constituído de:

- Três processadores de 8 *bits*;

- ROM de 10Kbytes contendo o *firmware* do chip;

- EEPROM de 512 *bytes* para código;

- RAM de 1Kb para armazenamento dos comandos de entrada ou dados;

- 11 pinos de E/S;
- Dois temporizadores/controladores de 16 *bits*;
- Interface de comunicação.

LonWorks oferece a qualquer um a possibilidade de usar outros processadores além do chip Neuron para tarefas industriais. Por exemplo, o protocolo LonWorks é executado hoje num processador MIPS 32-bit RISC , como também num Pentium Intel , Sistema Operacional Windows.

Suporta até 255 subnets, cada uma com 127 nós, perfazendo um total de 32385 nós, trocando dados entre si com o uso de transceptores adequados, pois o protocolo suporta comunicação distribuída, não havendo necessidade de um controle central.

4.3.4 *LonWorks* TCP/IP

A Figura 4.9 mostra LonWorks sobre uma arquitetura de rede IP (SHAHNASSER & WANG 1998). O servidor controla e monitora a rede LonWorks localmente e clientes podem controlar e monitorar a rede LonWorks remotamente. No servidor, um adaptador LonTalk acessa a rede LonWorks e obtém as variáveis da rede. Um cartão *Ethernet* conecta o servidor a rede IP e envia as variáveis da rede LonWorks para rede IP. No aplicativo do cliente, estas variáveis podem ser lidas e comandos de controle podem ser enviados. Então qualquer computador pode monitorar e controlar uma rede LonWorks usando uma conexão TCP/IP.

O monitor do servidor fica lendo a variável a ser controlada. Quando esta variável muda, é computado o instante da modificação e o novo valor da variável no banco de dados do servidor. O *buffer* do banco de dados do servidor também é atualizado. Depois de implementada uma conexão TCP/IP, o monitor do cliente lê o *buffer* do servidor e mostra estas variáveis na tela. O controle do cliente calcula novo ponto de operação, armazena o instante e o valor no banco de dados, implementa a conexão TCP/IP e envia os dados para o banco de dados do servidor. O controle do servidor verificando uma alteração no ponto de operação, grava a alteração no banco de dados e envia o novo ponto de operação da variável para a rede Lonworks.

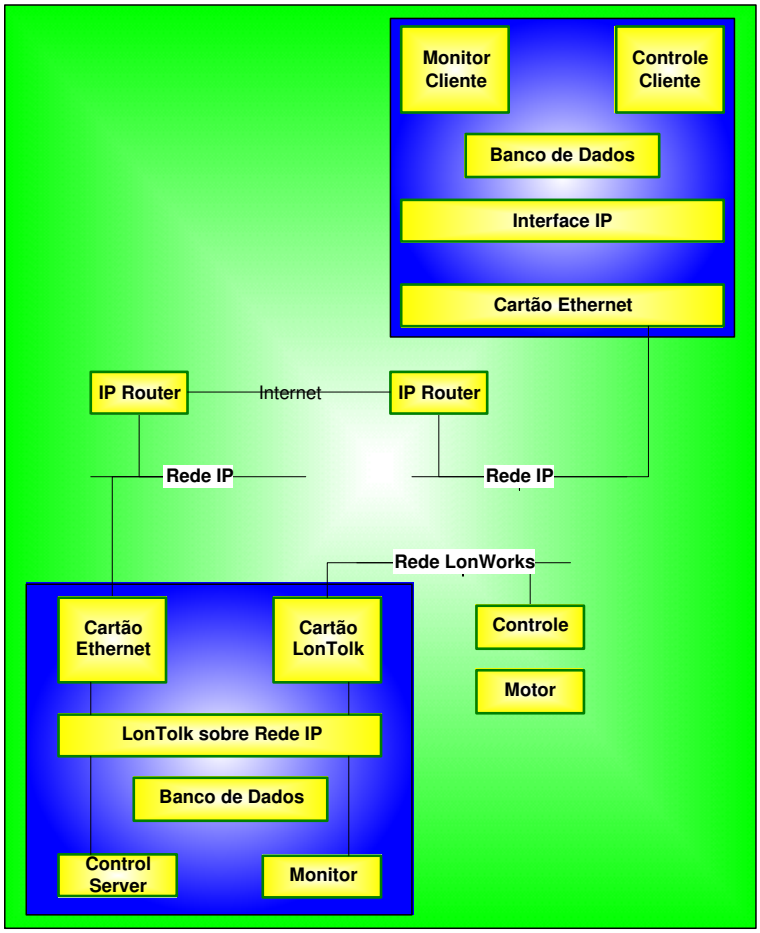


Figura 4.9: Protocolo LonWorks sobre TCP/IP.

O atraso IP é resultado do congestionamento da rede IP. O atraso LonWorks é causado pela latência do *software*.

4.3.5 Área de aplicações

- Automação residencial, predial (SNOONIAN 2003) ou industrial;
- Indústria;
- Geração de energia, e;
- Transportes (ZUJUN et al. 2002).

4.3.6 Desvantagens

As redes LONWorks ou suas aplicações apresentam algumas desvantagens. São elas:

- Alguns meios de comunicação , como RF, têm baixa velocidade de comunicação, insuficiente para aplicações com alto tráfego de informações e necessidade de resposta em tempo real;
- SNVT's limitada, o que faz com que muitos dispositivos sejam construídos de uma forma não padronizada;
- Soluções diferentes para um mesmo problema, o que faz com que um dispositivo não seja imediatamente substituído por um outro de diferente fabricante , mas deve-se levar em conta que alguns fabricantes fazem isso propositalmente.

4.3.7 Conclusão

A área de automação predial é o principal foco de atuação da tecnologia Lonworks, devido principalmente ao fato deste mercado ter começado após o da automação industrial e naturalmente mais receptivo a novas tecnologias. Mas isto não quer dizer que a mesma não possa ser aplicada em outras áreas de automação, analisando o potencial no tocante à facilidade de desenvolvimento e implementação, confiabilidade, custo e flexibilidade de instalação.

4.4 PROFIBUS (*Process field-bus*)

4.4.1 Histórico

A história do PROFIBUS remota ao ano de 1987, na Alemanha, quando um grupo de 21 companhias e institutos juntaram-se para criar um projeto *Fieldbus*. O objetivo era a realização e estabelecimento de um *Fieldbus bit* serial, o requerimento básico era a padronização da interface de dispositivo de campo. Para este objetivo, os membros associados da *Central Association for the Electrical Industry* (ZVEI) concordaram num conceito técnico mútuo de processo de automação e manufatura.

O primeiro passo foi a especificação do protocolo de comunicação PROFIBUS FMS (*Fieldbus Message Specification*), o qual foi feito sob medida para tarefas de demanda de comunicação, DIN 19245, 1991/1993. O passo seguinte foi dado em 1993 com a especificação de uma configuração mais simples e rápida do protocolo chamada PROFIBUS DP (*Decentralized Periphery*), padronizada pela norma EN50170.

É um sistema de comunicação digital, padronizado, aberto, de barramento robusto com uma ampla faixa de aplicações, particularmente nos campos de manufatura e processos de automação. PROFIBUS é apropriada para aplicações rápidas, de tempo crítico e de comunicação complexa. Hoje, a rede está presente nos níveis *Field Level* e *Cell Level*.

É padronizado pela norma IEC 61158, desde 1999, modelo *Fieldbus Intrinsically Safe COncept* (FISCO) e IEC 61784. Aplicação e aspectos de engenharia estão disponíveis na *PROFIBUS User Organization*. Isto assegura procedimentos de conexões simples entre diferentes fabricantes de componentes padronizados.

4.4.2 Características

O protocolo PROFIBUS tem um projeto modular e oferece várias tecnologias de comunicação, numerosas aplicações e perfis de sistemas, como também ferramentas de desenvolvimento de sistemas. Assim PROFIBUS cobre diversas aplicações específicas de produção e processos de automação. O número de sistemas de controle a automação PROFIBUS instalados é prova da grande aceitação desta tecnologia de barramento de campo.

O PROFIBUS usa o modelo OSI, Camada 2, ou Enlace, para definir procedimentos mestre-escravo e passagem do *token* para coordenação de muitos mestres no barramento. As tarefas da camada 2 também incluem funções, como segurança de dados e manipulação dos *frames*. A camada 7, de Aplicativo, forma a interface para o programa do usuário. O PROFIBUS oferece vários serviços para troca de dados cíclica e acíclica.

O PROFIBUS apresenta diferentes formas de aplicação do ponto de vista do usuário. Elas não são especificamente definidas, e sim demonstram utilidades como

resultado das aplicações freqüentes. Cada uma enfatiza principalmente resultados de uma típica combinação de elementos modulares de grupos de "transmissão de tecnologia" (quer dizer RS-485 ou barramento enegizado), "protocolo de comunicação" (DP-V0, DP-V1 ou DP-V2) e "perfis de aplicação" (PA, PROFIdrive ou PROFIsafe).

O PROFIBUS está dividido em três protocolos de comunicação, sendo cada um deles indicado para uma situação específica, são eles:

- PROFIBUS FMS;
- PROFIBUS DP; e
- PROFIBUS *Process Automation* (PA).

PROFIBUS FMS

Foi o primeiro protocolo de comunicação PROFIBUS. Foi projetado para comunicação de dados ao nível de células, por exemplo, entre CLPs ou entre um CLP e um PC. Possui as seguintes características:

- Projetado para troca de grande volume de dados;
- Protocolo universal que possui sofisticadas funções de comunicação entre dispositivos inteligentes, num tempo aceitável;
- Comunicação com dispositivos de campo é possível com o uso de interfaces FMS;
- Norma EN50170;
- Taxa de transmissão: 9,6 kbits/s a 11,5 Mbits/s.

PROFIBUS DP

É a solução mais simples, rápida, eficiente, cíclica, de baixo custo e determinística dentre os PROFIBUS. Está presente no *Field Level*, sendo desenvolvido especialmente para processo de troca de dados entre um barramento mestre e o designado

dispositivo escravo. É voltada para sistemas de controle onde se tem destacado o acesso de E/S distribuídos. É utilizado em substituição a sistemas convencionais 4 a 20mA ou HART ou em transmissão de 24 Volts. Requer um tempo de aproximadamente $1\,ms$ para transmitir 512 *bytes* de entrada e 512 *bytes* de saída varrendo 32 estações a uma velocidade de 12Mbit/s, a uma distância de 100 metros para 1000 pontos de E/S. É utilizado amplamente em controles com tempo crítico.

Atualmente 90% das aplicações envolvendo escravos PROFIBUS utilizam PROFIBUS DP. Esta variante é disponível em três versões: DP-V0(1993), DP-V1(1997) e DP-V2(2002).

A origem de cada versão foi de acordo com o avanço tecnológico e demanda das aplicações.

Tipos de dispositivos

Cada sistema DP pode conter três tipos diferentes de dispositivos:

- **Mestre DP Classe 1 (DPM1)**. Dispositivos mestres ou estações ativas determinam o tráfego de dados no barramento. Eles acessam o barramento e enviam mensagens sem uma requisição externa. Mestre DP Classe 1 é um controlador principal que troca informações ciclicamente com mais alta prioridade para transmissão de dados com valores e status de medição, assim como valores de *set-point* recebidos pelos equipamentos de campo. Os controladores lógicos programáveis (CLPs) e PCs são exemplos destes dispositivos mestres. Ciclicamente também se transmite uma seqüência de *bytes* de diagnósticos;

- **Mestre DP Classe2 (DPM2).** São as estações de engenharia e dispositivos de operação comumente usados em visualização, operação, manutenção e diagnose, como, por exemplo, terminais de programação, *notebooks* e *softwares* de supervisão. Usam tráfego de dados acíclicos com baixa prioridade de conexão;

- **Escravo**. São estações passivas e não possuem iniciativa de acesso ao barramento, somente comunicam quando da existência de uma requisição de um mestre. Um escravo DP é um dispositivo periférico, tais como dispositivos

de E/S, atuadores, IHM, válvulas, transdutores. Há também dispositivos que tem somente entrada, somente saída ou uma combinação de entradas e saídas.

As versões

A versão original, designada DP-V0, provê funcionalidades básicas do DP, incluindo configuração, parametrização, leitura de dados de entrada, escrita em saídas e leitura de dados de diagnósticos. Nesta versão um mestre DP lê e escreve ciclicamente seus escravos com tempo de ciclos em torno de $10\,ms$, dependendo da taxa de comunicação, que pode variar de 9,6 kbit/s a 12 Mbit/s. Por exemplo, em uma aplicação com 128 *bytes* de E/S, 1024 sinais analógicos, a 12Mbit/s temos um tempo de ciclo de $2\,ms$. Utiliza o meio físico RS485 ou fibra ótica, onde pode chegar a distâncias de até 80km (fibra sintética). Funções de diagnóstico facilitam a localização de falhas e são transmitidas ciclicamente. O PROFIBUS DP suporta a implementação de um mestre ou vários mestres, comportando até 126 elementos em uma rede (com quatro repetidoras no barramento) assim provendo um alto grau de flexibilidade durante a configuração de sistemas. Cada escravo pode enviar ou receber até 224 *bytes* de dados.

A arquitetura e filosofia do protocolo PROFIBUS assegura a cada estação envolvida nas trocas de dados cíclicas um tempo suficiente para a execução de sua tarefa de comunicação dentro de um intervalo de tempo definido. Para isto, utiliza procedimento de passagem de *token* entre estações mestres e o procedimento mestre-escravo para a comunicação com as estações escravas. Cada mestre fica com o *token* num tempo configurável e em seguida passa-o ao próximo mestre.

O procedimento mestre-escravo possibilita ao mestre que possui o *token* acessar os seus escravos através dos serviços de leitura e escrita.

O DP-V0 foi expandido para o DP-V1 que provê funcionalidades mais avançadas do DP, principalmente em termos de leitura, escrita e reconhecimento de interrupção em automação de processos, em particular na comunicação de dados acíclicos utilizada na parametrização, operação, visualização, supervisão e diagnóstico dos equipamentos de campo em conjunto com a comunicação cíclica. A comunicação acíclica é executada em paralelo à comunicação cíclica, porém com prioridade in-

ferior. O mestre classe 1 detém o *token* ao comunicar-se com seus escravos e no final do tempo de domínio do *token*, disponibiliza o mesmo ao Mestre classe 2. O Mestre classe 1 também pode executar troca de dados acíclicos com os seus escravos. Atualmente estas funções estendidas são amplamente usadas em operação on-line dos equipamentos de campo PA pelas estações de engenharia.

O DP-V2 provê funcionalidades mais sofisticadas, principalmente em termos de tecnologia de *drivers* e sistemas de segurança, assim como comunicação entre escravos com ciclo de barramento isócrono e gerenciamento de *clock*. Dá ênfase principalmente a automação fabril.

A comunicação escravo-escravo elimina o *overhead* causado pela necessidade de um mestre no sistema, sendo que um escravo pode agir como *Publisher* e a resposta do escravo pode ser direcionada aos demais escravos que agem como *Subscribers*. Isto pode reduzir em até 90% o tempo de resposta, dando mais flexibilidade às aplicações. Nos elementos escravos deve haver uma seleção local do endereço a ser utilizado na rede. O modo isócrono permite a sincronização do *clock* entre mestres e escravos, dando um maior controle no gerenciamento de mensagens de barramento, executando controle em tempo real e sincronizando estações, facilitando o *tracking* de eventos.

O perfil de aplicação PROFIBUS "PROFISAFE"(Perfil para Tecnologia Segura) utiliza DP-V2 e descreve mecanismos de comunicação segura entre periféricos sujeitos à falha segura (*Fail-Safe*) e controladores seguros. É baseado nos requisitos dos padrões e diretivas para aplicações com segurança orientada, como a IEC 61508 e EN954-1, bem como na experiência dos fabricantes de equipamentos com *Fail-Safe* e na comunidade de fabricantes de CLPs. Pode ser usado em sistemas de segurança com níveis AK6 e SIL3.

No PROFISAFE algumas medidas preventivas são tomadas, com o intuito de cercar as possíveis causas de falha e quando as mesmas ocorrerem, que aconteçam com segurança(**medidas usadas em nosso estudo de caso**):

- Numeração consecutiva de todas as mensagens: aqui se pretende minimizar a perda de comunicação, inserção de *bytes* no *frame* e seqüência incorreta;

- Sistemas de *watchdog timer* para as mensagens e seus reconhecimentos, ou seja, os atrasos;
- Proteção adicional do telegrama com a inclusão de 2 a 4 *bytes* de CRC, evitando a corrupção dos dados de usuários.

Além do exposto acima, alto grau de confiabilidade resulta da redundância em anel com uso de *Optical Link Modules* (OLM).

PROFIBUS PA

A utilização do PROFIBUS em dispositivos típicos e aplicações em controle de processo está definida segundo o perfil PROFIBUS-PA, usando tipicamente tecnologia de transmissão *Manchester Coded, Bus Powered - Intrinsically Safe* (MBP-IS) e protocolo de comunicação versão DP-V1 entre as estações. É usado em ambientes industriais de processos hostis (indústria química e indústria petroquímica). Os dispositivos são energizados via barramento, economizando cabos e horas de montagem na instalação.

Este perfil define objetos de comunicação, variáveis e parâmetros dos equipamentos para que haja interoperabilidade entre diferentes fabricantes, conseqüentemente possibilitando a monitoração do barramento. O Profibus PA é aplicado em transmissores de pressão, temperatura e posicionadores. É baseado no conceito de blocos funcionais que são padronizados de tal forma a garantir interoperabilidade entre os equipamentos de campo.

Os seguintes dispositivos de campo são especificados no perfil PA:

- Pressão e pressão diferencial;
- Nível, temperatura e vazão;
- Entradas e saídas analógicas e digitais;
- Atuadores e válvulas;e
- Analisadores.

O modelo *Fieldbus Intrinsically Safe COncept* (FISCO) foi desenvolvido pelo *Physikalisch-Technische Bundesanstalt* (PTB) (*German Federal Physical Technical Institute*) e é internacionalmente aceito para operações de barramento em áreas potencialmente explosivas e oferece considerável economia de tempo pois não há necessidades de gasto de tempo com cálculos, instalação e expansão de redes PROFIBUS PA. O modelo é baseado supondo que a rede é intrinsecamente segura e não requer qualquer outro cálculo de segurança se os quatros componentes principais de barramento (dispositivos de campo, cabos, acopladores de segmento e terminadores de barramento) estão dentro de limites pré-definidos com respeito à tensão, corrente, indutância e capacitância. Para isto, estes elementos devem ser certificados por órgãos autorizados em diferentes países, como exemplo o PTB. Se dispositivos padrões FISCO são usados, não somente é possível operar mais dispositivos numa mesma rede, como podem ser substituídos por dispositivos de outros fabricantes. Tudo isto, sem a necessidade de gastar tempo com cálculos e sistemas sem certificação.

Os valores e status de medição, assim como os valores de *set-point* recebidos pelos equipamentos de campo no PROFIBUS-PA são transmitidos ciclicamente com mais alta prioridade via mestre classe1 (DPM1). Já os parâmetros para visualização, operação e manutenção são transmitidos por ferramentas de engenharia mestre classe 2 (DPM2) com baixa prioridade através de serviços acíclicos pelo DP via conexão C2. Ciclicamente também se transmite uma seqüência de *bytes* de diagnóstico.

A descrição dos *bits* desses *bytes* está no arquivo*Device Description Data Files* (GSD) do equipamento e dependem do fabricante.

O tempo de ciclo (Tc) aproximado pode ser calculado como:

Tc $>= 10\,ms$ X número de equipamentos $+ 10\,ms$ (serviços acíclicos Mestre classe 2) $+ 1,3\,ms$ (para cada conjunto de 5 *bytes* de valores cíclicos).

Imaginemos a situação onde temos cinco malhas de controle com cinco transmissores de pressão e 5 posicionadores de válvula. Temos um tempo de ciclo de $100\,ms$.

Os benefícios confirmam o sucesso. Desde sua introdução, a cada ano, os números

de dispositivos instalados dobram: prova de eficiência e aceitação. As vantagens do PROFIBUS PA num ciclo de vida de um sistema de controle e automação são:

- Redução dos custos é resultado de:
  - Instalação e cabeamento;
  - Serviço e manutenção;
  - Segurança intrínseca.
- Proteção de investimentos é resultado de:
  - Padronização;
  - Dispositivos certificados;
  - Independência de qualquer fabricante.
- Alta praticabilidade é resultado de:
  - Diagnósticos on-line;
  - Uma grande faixa de produtos;
  - Interoperabilidade de dispositivos.

Princípio da troca de dados

No PROFIBUS, a camada 2 do modelo OSI (chamada de Enlace) é chamada de *Fieldbus Data Link*. O controle de acesso ao meio especifica o procedimento de transmissão de dados de uma estação quando esta tem o direito de transmitir.

O *Medium Access Control* (MAC) também é responsável por permitir que somente uma estação tenha o direito de transmissão por vez.

O protocolo PROFIBUS PA foi desenvolvido para combinar dois requisitos básicos:

- Durante a comunicação entre dois mestres de rede, o protocolo deve garantir que cada estação tenha o controle do tempo para transmissão dos dados de forma precisa e em intervalos;

- Por outro lado, a comunicação entre um mestre e um elemento escravo deve ser cíclica, em tempo real e a mais rápida possível, de forma simples e sem erros.

O MAC da rede PROFIBUS utiliza um procedimento de *token passing* quando há troca de dados entre elementos mestres de rede, e um procedimento mestre-escravo quando a comunicação é entre um elemento mestre e um escravo.

O procedimento *token passing* garante que um direito de acesso ao bus seja dado a cada mestre de rede, de tempos em tempos, de uma forma precisa. A *token message* que é a mensagem de um mestre para outro passando o direito de uso do bus, deve ser executada dentro de um tempo limite configurável por *software*.

Já o procedimento mestre-escravo permite que o mestre gerencie os *frames* de dados entre ele e os elementos escravos, fazendo o que se chama de *polling* entre as estações (**O** *polling* **foi uma solução proposta no nosso estudo de caso**).

Com esses métodos de acesso, é possível fazer as seguintes configurações de redes:

- Um sistema puro mestre-escravo;
- Sistemas multi-mestres (*token passing*);
- A combinação dos dois.

O *Token Ring* forma uma seqüência lógica de estações mestres formando um anel lógico, sendo que cada estação mestre que tem o direito de acesso ao barramento naquele momento, troca dados com os elementos escravos na comunicação mestre-escravo.

Adicionalmente, a comunicação mestre-escravo ponto a ponto é possível e também o estabelecimento de mensagens para outras estações ao mesmo tempo em *Broadcast* ou *Multicast*.

Um *frame* PROFIBUS pode acomodar até 246 *bytes*. Todos os campos, exceto o de dados, possuem um comprimento fixo de uma unidade denominada *UART Character*.

A comunicação cíclica e acíclica de dados entre o sistema de controle de processo (Mestre Classe 1) ou ferramentas de engenharia (Mestre Classe 2) e dispositivos de campo (escravos) pode somente acontecer se os mestres reconhecem os parâmetros do dispositivo específico e o formato dos dados.

Para atingir este objetivo, o PROFIBUS oferece uma solução padronizada: os parâmetros de comunicação de um dispositivo PROFIBUS são definidos como um dispositivo eletrônico na forma de arquivo de banco de dados (arquivo GSD) num formato pré-determinado. Utilizar o GSD nas ferramentas de configuração habilita o sistema de processo de controle a otimamente utilizar os parâmetros de comunicação do dispositivo.

As características de um dispositivo PROFIBUS para sua configuração são descritas usando *Electronic Device Description Language* (EDDL). EDDL é uma linguagem universal para fabricantes independente do dispositivo de campo.

Os arquivos EDDL informam ferramentas de configuração ou parâmetros do dispositivo para sistemas de controles de processo.

Áreas de aplicação do Profibus

Virtualmente usado em todas as áreas de automação, em automação de manufatura ou automação de processos, mas também em engenharia de tráfego, geração e distribuição de energia.

Exemplos de aplicação Profibus:

- Comunicação entre CLPs e estações remotas no descarte de água da plataforma de Pargo, Projeto Mar Azul da Petrobrás, e
- Interfaces Homem-Máquina.

4.4.3 Comentários

Através destas linhas podemos ver o avanço das versões do PROFIBUS e suas principais características. Suas melhorias proporcionaram vantagens às aplicações e

aos usuários. Existem vantagens potenciais da utilização desta tecnologia, principalmente de funcionalidades (transmissão de informações confiáveis, tratamento de status de variáveis, sistemas de segurança em caso de falha, equipamentos com capacidade de autodiagnose, uma grande variação de equipamentos, alta resolução nas medições, controle discreto em alta velocidade, aplicações em qualquer segmento, etc) e benefícios econômicos pertinentes às instalações (redução de 40% em alguns casos em relação aos sistemas convencionais), custos de manutenção (redução de até 25% em alguns casos em relação aos sistemas convencionais) e menor tempo de start-up.

4.5 *Fieldbus Foundation* (FF)

O protocolo FF alcançou um nível de integração em que tanto os instrumentos de campo como os controladores são parte do sistema, ou seja, arquitetura *Field Control System* (FCS). Instrumentos e sistemas não são duas ilhas separadas. FF H1 é usado no chão de fábrica e faz com que os dispositivos como transmissores e posicionadores sejam inter-operáveis. *FF High Speed Ethernet* (HSE) é usado na hierarquia de mais alto nível, aonde os dispositivos hosts e subsistemas são integrados. A capacidade do protocolo FF HSE vai muito além dos sistemas de E/S remoto e redes de nível de controle (FF 1996).

4.5.1 Características

O FF foi inicialmente definido pelo comitê de padronização ISA SP50, em 1994. É um protocolo com capacidade de aplicações de controle distribuídas na rede. O FF H1 opera a uma velocidade de 31,25kbit/s e interconecta dispositivos de campo como sensores, atuadores e módulos de Es/Ss. O FF HSE opera a 100 Mbit/s e provê integração em alta velocidade de controladores (CLPs), subsistemas H1 via dispositivos de *linking*, servidores e estações.

4.5.2 Benefícios

São benefícios alcançados com o uso do FF H1:

- **Disponibilidade de mais dados.** Múltiplas variáveis dos dispositivos são enviadas para a sala de controle;
- **Expansão da visualização dos processos e instrumentos.** Possibilitam ações corretivas na manutenção dos equipamentos;
- **Redução de equipamentos.** Distribuição de controle nos dispositivos de campo pode reduzir o número de equipamentos de controle e E/S, incluindo cartões, gabinetes e fontes;
- **Redução dos custos de fiação.** Muitos dispositivos são conectados a um simples par de cabos.

Além dos benefícios do FF H1, o FF HSE provê suporte ao controle na integração de todos os sistemas de controle e automação. São eles:

- **Padrão Ethernet.** Uso de equipamentos padrão *Ethernet*, como *switches*;
- **Alto desempenho.** Habilitação de funções de gerenciamento de ativos como diagnóstico, calibração, identificação e outras operações de manutenção proativa que alocam recursos aonde eles realmente são necessários;
- **Interoperabilidade entre subsistemas.** Subsistemas de diferentes fornecedores são integrados facilmente devido ao fato de o protocolo ser aberto;
- **Blocos de função.** Eliminam a necessidade de linguagem de programação proprietária.

4.5.3 Características

FF H1

O protocolo H1 consiste na camada do usuário, aplicativo, stack de comunicação e camada física, conforme Figura 4.10. O FF não usa as camadas 3, 4,5 e 6 do modelo OSI. A camada usuário não é definida pelo modelo OSI. O FF especifica um modelo da camada usuário significativamente diferente de outros modelos. Cada

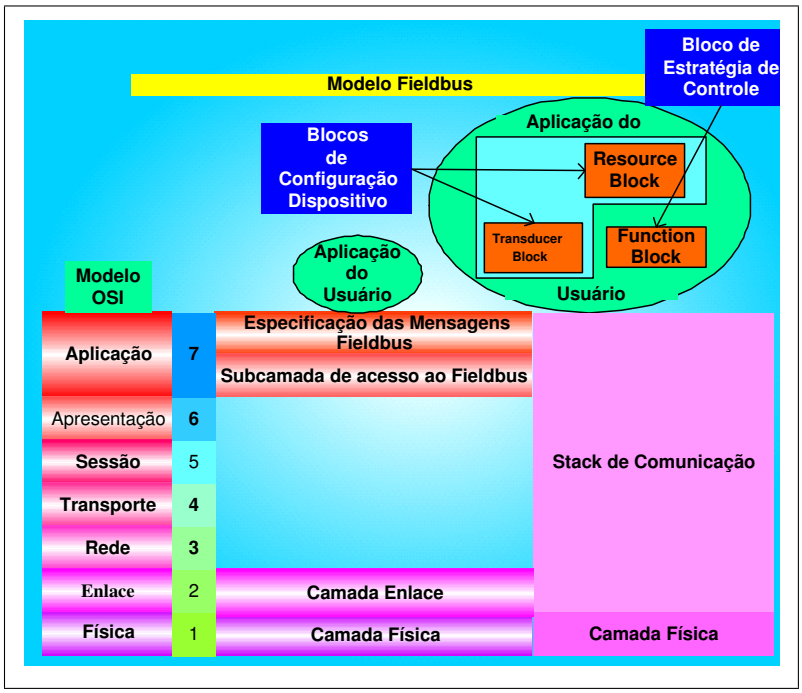


Figura 4.10: Camadas FF.

camada no sistema de comunicação é responsável por uma parte da mensagem que é transmitida no barramento.

A camada física foi definida pelos padrões ISA S50.02-1992 e IEC 61158-2:2000 (ed. 2.0). A camada física recebe mensagens do stack de comunicação e converte as mensagens em sinais físicos para transmissão na mídia e vice-versa. Sinais do barramento são codificados usando técnica *Manchester Biphase-L technique*. O sinal é denominado síncrono serial devido à informação do clock está embutida no fluxo de dados serial. Os dados são transmitidos a uma velocidade de 31,25 kbit/s com variação de 1 V modulada na tensão contínua de alimentação que pode variar de 9 a 32V. O FF suporta o modelo FISCO ou modelo tradicional de Entidade. O comprimento do barramento é determinado pela velocidade de comunicação, tipo e tamanho do cabo e opção de energia no barramento.

A camada enlace controla a transmissão das mensagens no barramento através de escalonamento determinístico ao barramento denominado *Link Active Scheduler* (LAS). O LAS tem uma lista de todos os dispositivos que necessitam ser acessados ciclicamente. Um barramento pode ter vários LAS, se o corrente LAS falhar, um dos outros se tornará o LAS. A comunicação começa quando o LAS envia uma

mensagem *Compel Data* (CD) para um dispositivo, e este publica os dados do *buffer* para todos os dispositivos no barramento. Qualquer dispositivo configurado para receber os dados é denominado *subscriber*. No intervalo de enviar uma nova CD todos os dispositivos tem chance de enviar mensagens desde que ele possua o *token*. A lista de todos os dispositivos que estão respondendo a passagem do *token* chama-se *Live List*. Esta lista é atualizada automaticamente pelo LAS.

A sub-camada de acesso ao *Fieldbus Access Sublayer* (FAS) usa características da camada de enlace para provê um serviço para a Especificação das Mensagens *Fieldbus* (FMS). Os tipos de serviço podem ser:

- **Cliente/Servidor.** Ocorrem entre dois dispositivos para troca de set-points, alarmes, etc. Faz uso do *token*;
- **Informação por Exceção.** Um dispositivo envia uma mensagem para um grupo de dispositivos. Por exemplo, alarmes para operadores de console. Faz uso do *token*;
- **Publisher/Subscriber.** Comunicação de um para muitos. Por exemplo, envio de variáveis. Faz uso da *Compel Data*.

A subcamada de aplicação FMS permite as aplicações dos usuários enviarem mensagens para outros usuários usando um conjunto padrão de mensagens, como serviço de comunicação, formato da mensagem, etc.

O FF definiu uma camada do Usuário baseada em blocos. Os blocos são representações funcionais de diferentes aplicações. Os tipos de bloco estão definidos na Figura 4.10. *Resource Block* contém características do dispositivo como nome do dispositivo, etc. *Function Block* provê o comportamento do controle do sistema. Estes blocos podem ser Analog Input (AI), Proportional/Derivative (PD), etc. *Transducer Blocks* é a entidade que armazena os parâmetros associados com os sensores e atuadores.

Um dispositivo *Fieldbus* é um dispositivo de E/S remoto que tem muitos blocos de função no qual dados se comunicam. Cada bloco tem um perfil, isto é, um número. Por exemplo, um bloco PID padrão tem código 0108 em hexadecimal. A

engenharia de processos constrói a estratégia de controle e mais tarde a engenharia de instrumentos aponta o bloco para o dispositivo. Basta a engenharia de instrumentos confirmar com o arquivo *Capability Files* do instrumento que o mesmo suporta o perfil 0108. Isto significa que você pode usar o mesmo bloco para um transmissor, posicionador ou controle central sem ter que usar outro bloco.

Para alcançar interoperabilidade a tecnologia de descrição de dispositivos (DD) é usada em adição a blocos de função padrão. DDs são plataformas que operam independentemente do sistema. DDs são similares aos drivers usados em PCs para operar diferentes impressoras e outros dispositivos. Qualquer sistema de controle pode operar com dispositivos se eles tem DD.

Um dispositivo é fornecido com três arquivos de suporte:

- ***Device Description Tokenize.*** Os fornecedores constroem uma DD importando o padrão DD fornecido pelo FF, e adicionam características específicas como procedimentos de diagnóstico e calibração;
- ***Device Description Services* (DDS).** Estas bibliotecas estão armazenadas no host e são usadas para ler as descrições do dispositivo. Note que DDS lêem as descrições e não os valores, que são lidos dos dispositivos via FMS;
- ***Capability Files.*** Diz ao host que recursos o dispositivo tem em termos de blocos de função.

Stack de comunicação FF HSE

A seguir apresentamos algumas características do stack de comunicação FF HSE:

- Suporta dispositivos de linking (redes H1 para HSE), dispositivos de *Ethernet*, gateways para interface com outros protocolos e dispositivos host como estações de operação;
- Primeiro protocolo baseado em *Ethernet* com alto nível de redundância ao nível do host.

4.5.4 Comentários

A principal característica do protocolo FOUNDATION Fieldbus é a integração de componentes de um sistema. Os instrumentos de campo são partes integrais do sistema como os controladores também são, isto é, instrumentos de campos e sistemas não são duas ilhas separadas. O protocolo FOUNDATION H1 é usado a nível de campo possibilitando que transmissores e posicionadores sejam interoperáveis, o protocolo FOUNDATION HSE é usado no nível mais alto da hierarquia do sistema, aonde os dispositivos *hosts* e susbsistemas estão interligados através de uma rede.

O FF foi o primeiro protocolo a implementar um controle cada vez mais distribuído. Redes de instrumentação futura não necessitarão ter controladores complexos, pois os instrumentos de campo poderão fazer o controle localmente. As operações de baixo nível de instrumentação realizadas pelo controlador principal serão substituídas por operações de mais alto nível como controle avançado, configuração centralizada, filtragem de alarmes e manutenção de um banco de dados global.

4.6 Quadro comparativo

A seguir apresentamos tabelas comparativas entre diversos barramentos de campos (YABIN & NISHITANI 2003), (FELSER & SAUTER 2002) e (CI-94 2004).

Tabela 4.3: Organização, ano e padronização.

	Organização	Ano	Padronização
Asi-Bus	AS-Interface Consortium www.as-interface.com	1993	IEC 62026/2 IEC 50295 IEC 947
CAN	CiA - CAN in Automation www.can-cia.org	1995	EN 50325-4
ControlNet	ControlNet International www.controlnet.org		ControlNet International
DeviceNet	Open DeviceNet Vendors Association www.odva.org	Março/1994	ISO 11898 e 11519
Fieldbus Foundation	FieldbusFoundation /IEC www.fieldbus.org	1994	ISA SP50IEC 61158
Hart	Hart Communication Foundation www.hartcomm.org		Bell202 communication standardIEC 801-3IEC 801-4HCF revisão 5.0
INTERBUS	InterbusClub	1984	DIN 19258EM 50254IEC 61158
LonWorks	LonMark www.lonmark.org	Março/1991	EIA-709.1 - define o protocolo comum;EIA-709.2 - define os transceivers para que vários produtos possam se comunicar;EIA-709.3 - define um meio físico com topologia livre.
Modbus *Ethernet* TCP/IP	Modbus.org www.modbus.org	1978	IEE 802.3
PROFIBUS DP/PA	Profibus International www.profibus.com	DP-1994,PA-1995	DIN 19245IEC 61158IEC 61784EM 50170
SafetyBus	SafetyBus Club www.safetybus.com		DIN 19250DIN VDE 0116 EM - 954

Tabela 4.4: Interoperabilidade, interconectividade e número de nós.

	Interoperabilidade	Interconectividade	Max. de nós
Asi-Bus	Homologados pela AS-Interface.	Sim via gateways para Profibus DP, DeviceNet, Modbus e Interbus.	31 nós por segmento de rede para versão 2.0 e 62 para versão 2.1.
CAN	Não		110
ControlNet	100% com produtos ControlNet, conexão via interface para *Ethernet I/P*, *Ethernet TCP/IP*, DeviceNet e FF.	100% com produtos ControlNet, conexão via interface para *Ethernet* I/P, *Ethernet* TCP/IP, DeviceNet e FF.	99
DeviceNet	100% com outros produtos DeviceNet(ODVA). A partir de uma rede ControlNet, *Ethernet* I/P, *Ethernet* TCP/IP é possível coletar, configurar e controlar dispositivos em diferentes segmentos de rede DeviceNet.	100% com outros produtos DeviceNet(ODVA). A partir de uma rede ControlNet, *Ethernet* I/P, *Ethernet* TCP/IP é possível coletar, configurar e controlar dispositivos em diferentes segmentos de rede DeviceNet.	64 dispositivos inteligentes, combinando sinais discretos e analógicos. Máximo: (8192 sinais digitais ou 4096 sinais analógicos).
Fieldbus Foundation(FF)	Normas Fieldbus Foundation.	Linking Devices que interfaceam de forma nativa com FF H1, FF HSE. Gateways para Modbus TCP/IP, DeviceNet e Profibus DP.	16 nós para ligação a dois fios sem segurança intrínseca, 16 instrumentos com 2 fios e com 4 barreiras de segurança intrínseca e 32 instrumentos com 4 fios.
Hart	Através de conjunto de comandos universais e "Common Practice", além de DDS para comandos específicos.	Interfaces Fieldbus Foundation, Profibus e Modbus.	Ponto a ponto, limitado a capacidade do controlador multidrop: 15 dispositivos.
INTERBUS		Permite a comunicação com inversores de freqüência, IHM, tags de RF, válvulas, transmissores, leitores de código de barras, entre outros.	4096 pontos em 254 nós da rede distribuídos em 12,8km de cabos. Distância entre módulos de 400m utilizando cabo de cobre.
LonWorks	Garantida através das variáveis "tipadas", as SNVTs.	Com gateways apropriados.	Até 255 subnets, cada uma com 127 nós, perfazendo um total de 32385 nós.
Modbus *Ethernet* TCP/IP	Protocolo padrão nos principais equipamentos. Fácil desenvolvimento através de bibliotecas disponíveis abertamente.	Acessório para conexão física - padrão *Ethernet*.	Ilimitado - Cascateamento de *Switches*.
PROFIBUSDP/ PA	100% com produtos Profibus, ou através de gateways que podem ser até CLPs.	Através de funções de roteamento entre redes disponibilizadas para cada modelo específico de CLP/SDCD, através do uso de Proxy para *Ethernet* (Profinet) ou através de gateways disponíveis no mercado.	DP: 126 separados por segmentos de 32 nós. PA: 10 (para áreas classificadas) ou 30 (áreas não classificadas). FMS: igual ao DP.
SafetyBus	Sim, dual channel.	Sim, Time - driven.	64 no máximo com 3.500 m de comprimento e topologia linear.

Tabela 4.5: Velocidade, tipo de tráfego e segurança intrínseca.

	Velocidade	Tipo de tráfego	Segurança intrínseca
Asi-Bus	160 kbit/s, com ciclo máximo de $5\,ms$ para a versão 2.0 e $10\,ms$ para versão 2.1 .	Cíclica e polling.	Não.
CAN	Máximo de 1Mbit/s para 50 m, 100kbit/s para 500m e 5kbit/s para 1000m.	Síncrono.	Não.
ControlNet	5Mbit/s com determinismo e repetibilidade.	Determinismo e repetibilidade (variáveis de controle) e não determinismo (configuração e supervisão).Modelo produtor - consumidor, multicast e multi-master.	Sim.
DeviceNet	Máximo de 125 kbit/s para 500m, 500 kbit/s para 125m.	O modelo produtor-consumidor permite priorizar o tráfego na rede (polling, mudança de estado, cíclico). Modelo produtor - consumidor, polling, cíclico, mudança de estado e multi-mestre.	Em desenvolvimento.
Fieldbus Foundation	31,25 kbit/s no nível H1 do FF para 1000 m e 100 Mbit/s no nível HSE.	Determinístico (para variáveis de processo) e não determinístico (para parametrização e diagnósticos).Single/Multi-master, token pass.	Impedância maior que 400 ohms para uma freqüência variando de 7,8 a 39KHz. Padrão FISCO.
Hart	1200 bits/s.	Polling, burst.	Até 4 equipamentos por barreira de segurança.
INTERBUS	Pontos digitais: 256 pontos distribuídos em 16 nós com tempo de acesso igual a $1,9\,ms$. Pontos analógicos: 128 pontos distribuídos em 16 nós com tempo de aceeso igual a $7,4\,ms$. Taxa de transmissão de 500 kbit/s e 2 Mbit/s full duplex.	Cíclica, controlada por mestre e determinística.	Sim.
LonWorks	1,25Mbit/s em cabo coaxial, fibra ótica ou par trançado até 2700m. 78kbit/s em Power link (par trançado com alimentação embutida). 10kbit/s em power line (rede elétrica). 4,8kbit/s em RF.	-	-
Modbus *Ethernet* TCP/IP	100 Mbit/s. Alto throughput através de sistemas Publisher Subscriber	Não determinístico.	Não.
PROFIBUS DP/PA	DP: de 9.6kbit/s a 12 Mbit/s(93,75 kbit/s em 1200m). PA: 31,25 kbit/s em 1900m. FMS: 9.6 kbit/s a 12 Mbit/s.	Mestre/escravo, ponto a ponto e multicast. Cíclica para transferência de dados e acíclica para gerenciamento de parâmetros.	Profibus DP: Sim através da utilização do meio físico RS-485-IS (com cabos de segurança e barreira de segurança intrínseca);Profibus PA: Sim, através da utilização dos modelos EX para os DP/PA link/couplers, de acordo com modelo *Fieldbus Intrinsically Safe Concept* (FISCO)
SafetyBus	Até 500 kbit/s.	-	Sim.

Tabela 4.6: Redundância, interação e inteligência distribuída.

	Redundância	Interação com PCs, PLCs e SDCDs	Inteligência distribuída
Asi-Bus	Não.		
CAN	Sim.		Não.
ControlNet	Sim. Todos os módulos de E/S para ControlNet oferecem a possibilidade de conexão em rede redundante.	Permite a comunicação entre CLPs, entre CLPs e Sistemas de E/S e com SDCDs.	Sim, com módulos de E/S com tecnologia DeviceLogix, Inversores de Freqüência com tecnologia DriveLogix, e através de interfaces para Foundation Fieldbus.
DeviceNet	N/A	Permite a comunicação com PCs,CLPs e com SDCDs.	Sim, com módulos de E/S com tecnologia DeviceLogix.
Fieldbus Foundation	Redundância na fonte do nó, condicionador de sinal e Link Active Scheduler (LAS); via acopladores redundantes, CPU, Co-processador Lógico, OPC Server e Power Supply. No FF HSE redundância da rede.	O nó é integrado ao controlador principal, mas há os que utilizam gateways ou linking devices que usam a rede *Ethernet* para interface - via conexão na camada de transferência rápida de dados, definida pela Fieldbus Foundation, sobre a camada *Ethernet* chamada, FF HSE.	Os limites de blocos de função e do fornecedor do nó variam dependendo do fabricante. Todas as funcionalidades são distribuídas até os equipamentos de campo através de blocos de função: 20 blocos por dispositivo de campo; 200 blocos por linking devices, por exemplo.
Hart	Não.	Normalmente integrado ao controlador principal ou através de multiplexadores. Interação através de sinal 4-20 mA e nos equipamentos mais modernos via comunicação serial.	Blocos de função que rodam algoritmos de controle PID.
INTERBUS	Sim.	Sim.	Através de módulos inteligentes.
LonWorks	É possível existir redundância de módulos.	Total. Nos PC's através de cartões de interface de rede. Nos PLC's através de módulos de comunicação apropriados.	Sim.
Modbus *Ethernet* TCP/IP	Sim, em anel.	Protocolo padrão nos principais equipamentos.	Sim, através de processadores distribuídos.
PROFIBUS DP/PA	Sim, existindo um profile específico para este fim, o que possibilita a redundância de CPU/mestre de rede, de meio físico (tanto elétrico quanto óptico) quanto dos dispositivos de E/S.	Sim, através do Profibus para CLPs / Profibus FMS.	PA: Sim, através da utilização de escravos inteligentes, sendo possível a implementação de estratégias de transferência de controle Bumpless entre a CPU com o Mestre Profibus e o Escravo inteligente.
SafetyBus	Sim, multi-master bus.	Através dos seguintes protocolos: DeviceNet, Profibus, Interbus, Modbus, CANopen e ControlNet.	Sim.

Tabela 4.7: *Ethernet*, *softwares* e aplicações.

	Ethernet	*Softwares*	Aplicações
Asi-Bus	Via *gateway*.	*Software* de configuração / vários fabricantes.	Rede de sensores, máquinas, manufaturas, todos os segmentos da indústria.
ControlNet	Sim.	Requer *software* de configuração dependendo do controlador.	Qualquer tipo de indústria.
DeviceNet	Sim.	Requer *software* de configuração dependendo do controlador.	Qualquer tipo de indústria.
Fieldbus Foundation	100 Mbit/s ou 1 Gbit/s.	Para supervisão e controle, específicos para batelada, para controle avançado (Redes Neurais, Controle Preditivo, Controle Multivariável, Lógica Fuzzy, Sintonia de Malha, gerenciamento de ativos, auditoria de malhas, intertravamento e sequenciamento).	Para qualquer tipo de controle e manufatura.
Hart	Conexão via *gateway*.	Gerenciamento de ativos, configuradores, calibração, diagnósticos básicos e avançados. Exemplos: CONF301, CONF402, HPC301 e PSION.	Para qualquer processo que necessite de instrumentação inteligente de campo: químico e petroquímico, açúcar e álcool, papel e celulose, automação predial, alimentos e bebidas, cimento, vidro e fibra ótica, metalurgia, mineração e indústrias de processo em geral.
INTERBUS	*Gateways* Interbus e E/S diretamente na rede Ethernet.	*Softwares* para configuração, monitoração e diagnósticos da rede para CLPs, como o CMD e *softwares* PC Worx.	Rede utilizada em qualquer segmento da indústria , no Brasil principalmente em indústrias alimentícia, automobilística , manuseamento e armazenamento de materiais, e máquinas de empacotamento.
LonWorks	É possível transferência de informações através de roteadores.	Existem *softwares* para instalação de redes LON, bem como *softwares* para programação dos dispositivos LON.	Automação Industrial, Automação Predial, Transportes, Utilidades e Automação Residencial.
Modbus Ethernet TCP/IP	Sim.	Sem necessidade de *software* configurador, ferramentas padrão de informática para manutenção e diagnóstico.	Processo e manufatura.
PROFIBUS DP/PA	O Padrão Profinet contempla a integração dos protocolos Profibus e a *Ethernet* Industrial de forma homogênea e harmoniosa, inclusive considerando a integração de redes Profibus existentes através de Proxies.	Normalmente o *software* de configuração do controlador contempla as funções de configuração de rede e em casos onde isso não acontece, existem *softwares* padrões no mercado com tal funcionalidade. Com relação aos supervisórios, basicamente todos possuem drivers de comunicação / interface OPC para conexão com a rede Profibus através de placas amplamente disponíveis no mercado. Em termos de Profibus PA, existem configuradores baseados em EDDL e FDT/DTM.	Indústrias de processo e manufatura, implementação de sistemas de automação e controle distribuídos (CPUs + remotas de E/S em rede), interligação de sistemas (CLPs e SDCDs) com inversores de freqüência, sistemas de pesagem, leitores de código de barras, *encoders*, *robots*, CCMs inteligentes, sistemas de medição de energia, implementação de sistemas, instrumentação de segurança com remotas de E/S em rede em SIL 3 / AK (Profisafe), implementação de redes de instrumentação de campo (Profibus PA), etc
SafetyBus	Sim, através de CPU.	PSS SW PG (PSS - Range).	Prensas, linhas de transferência, células robotizadas, aeroportos, teleféricos e máquinas em geral.

Tabela 4.8: Orientação do protocolo.

	Tipo de protocolo
Asi-Bus	Orientado a E/S.
CAN	Orientado a mensagem.
ControlNet	
DeviceNet	Orientado a mensagem.
Fieldbus Foundation	Orientado a mensagem.
Hart	
INTERBUS	Orientado a E/S.
LonWorks	
Modbus Ethernet TCP/IP	
PROFIBUS DP/DA	DP - Orientado a E/S PA - Orientado a mensagem.
SafetyBus	

Capítulo 5

Integração via *Internet* e *World Wide Web*

5.1 Introdução

A *Internet* e a *World Wide Web* (WWW) revolucionaram o computador e o mundo das comunicações como nada antes. A invenção do telégrafo, do telefone, do rádio e do computador foram os precursores para esta integração sem precedentes jamais vista. A *Internet* e a WWW são mecanismos para a disseminação de informações a nível mundial, são meios de cooperação e interação entre indivíduos e seus computadores sem consideração da localização geográfica.

A *Internet* e a WWW representam um dos exemplos mais bem sucedidos de benefícios proveniente de investimentos à pesquisa e ao desenvolvimento de infra-estrutura de informação. A *Internet* começou a surgir com as pesquisas sobre comutação de pacotes nas redes de comunicação digital. O primeiro grande serviço oferecido pela *Internet* foi o correio eletrônico, só superado em volume de tráfego na rede pelo WWW, o serviço mais bem sucedido da *Internet*. Governos, indústrias e universidades de vários países (principalmente USA, Inglaterra e França) foram sócios em evoluir e desdobrar esta tecnologia emocionante. Hoje, os termos "WWW" e "@" fazem parte do dia a dia das pessoas.

Infelizmente, os caminhos para padronização dos protocolos *Fieldbus* não foram os mesmos da *Internet* e WWW. Muitos protocolos *Fieldbus* tornaram-se "pro-

prietários" e os protocolos padrões do IEC (por si só) não são capazes de trocar informações entre si. Tanto que, em 1996, quando foi inicializada a "guerra dos *Fieldbus*" , quando a minuta do FOUNDATION Fieldbus foi colocada para votação como padrão, no IEC, os serviços da *Internet* como correio eletrônico e WWW já eram usados em mais de 120 países, com mais de 12 milhões de endereços IP registrados e com mais de 600.000 sites. Os números por si só mostraram que o uso de padrões abertos, como *Ethernet*, TCP/IP e conceitos baseados na *Web* tornariam-se muito importante nos futuros sistemas de comunicação das indústrias. Tanto que, alguns padrões do IEC foram encapsulados pelos protocolos TCP/IP, como o FOUNDATION FIELDBUS HSE.

A WWW introduziu outras "tecnologias abertas" que estão facilitando ainda mais o processo de integração entre os diferentes *Fieldbus*, como as linguagens HTML e XML, protocolo HTTP e navegadores *Web*. A adoção de tecnologias *Web* introduz novos conceitos e oferece novas oportunidades de serviços em várias aplicações de um sistema de automação. Contudo, é necessário avaliar criticamente as diferentes tecnologias *Web* com respeito ao ambiente em que elas podem ser usadas (WOLLSCHLAEGER 2000).

Há muito entusiasmo em relação à *Internet* e à WWW. A *Internet* une o "mundo das informações". A WWW torna a *Internet* fácil de utilizar e lhe confere a aparência e o funcionamento de programas multimídias. As organizações vêem a *Internet* e a *Web* como algo crucial às suas estratégias de sistemas de informação.

O item 5.2 relata a história, conceitos e os principais protocolos de comunicação utilizados pela *Internet* e pela *World Wide Web*. O item 5.3 descreve as principais ferramentas disponíveis para *Web*. O item 5.4 descreve o modelo de referência TCP/IP de quatro camadas. Os itens 5.5 e 5.6 descrevem como a *Ethernet*, *Internet* e WWW podem ser aplicadas a uma arquitetura de automação.

5.2 História da *Internet* e da *World Wide Web*

Em 1957, a União das Repúblicas Socialistas Soviéticas lançaram o primeiro satélite artificial terrestre, o Sputnik. Em resposta, o Estados Unidos da América, através

do *Department of Defense* (DoD) criaram a *Advanced Research Projects Agency* (ARPA) como órgão de controle em ciência e tecnologia aplicável para fins militares. Uma possível pretensão da ARPA era criar uma rede capaz de automaticamente se reconfigurar e encontrar caminhos alternativos caso um ou mais nós da rede saíssem de operação. O objetivo era interconectar todo o sistema dos mais de 500 silos com mísseis balísticos intercontinentais localizados nos Estados Unidos. Assim sendo, a rede deveria estar disponível para receber autorização de ataques e conseqüentemente lançar os mísseis, mesmo que algumas cidades ou silos da rede tivessem sofrido um ataque nuclear, possibilitando assim o rápido contra-ataque dos americanos.

Em agosto de 1962, no *Massachusetts Institute of Technology* (MIT), J.C.R. Licklider escrevia os primeiros memorandos sobre as interações sociais que poderiam ser originadas pelo uso das redes de comunicação digital. Licklider discutia o conceito de "*Galactic Network*". Ele visionava um conjunto de computadores interconectados através do globo no qual qualquer pessoa poderia acessar rapidamente dados e programas. No espírito, o conceito era muito parecido com a *Internet* de hoje. Licklider foi o primeiro "cabeça" do programa de pesquisa em computadores da ARPA e convenceu seus sucessores, como Lawrence G. Roberts, da importância da criação de redes de comunicação digital.

Em Julho de 1961, Leonard Kleinrock do MIT publica o primeiro artigo sobre teoria de comutação de pacotes. Kleinrock convenceu Roberts da possibilidade teórica de comunicações em rede usando comutação de pacotes em vez da comutação de circuitos. Este foi um dos pontos chave da evolução das redes de comunicação digital. A outra etapa chave era interconectar os computadores. Em 1965, Roberts junto com Thomas Merrill, conectaram o computador TX-2 do *MIT Lincoln Lab*, com o computador AN/FSQ-32 da *System Development Corporation* (Santa Monica, CA) através de uma linha telefônica dedicada de 1200 bps (sem rede de pacotes), criando a primeira rede WAN. O resultado desta experiência era a concretização que os computadores poderiam ser utilizados por vários usuários independentes conectados ao mesmo tempo, executando programas e extraindo informações de máquinas remotas, mas que a técnica de comutação de circuitos por redes telefônicas era totalmente inadequado para o trabalho. A convicção de Kleinrock da necessidade de comutação

de pacotes foi confirmada.

Em 1966 Roberts foi à ARPA desenvolver o conceito de redes de computadores e rapidamente propôs o plano ARPANET, publicado em 1967. Na conferência onde apresentou o artigo, havia também um artigo sobre comutação de pacotes escrito por Donald Davies e Roger Scantlebury do *National Physical Laboratory* (NPL), Middlese, England. Scantlebury disse à Roberts sobre o trabalho do NPL assim como o trabalho de Paul Baran da (RAND). Todo o trabalho do MIT (1961-1967), da RAND (1962-1965) e do NPL (1964-1967) foram feitos em paralelo sem nenhum dos pesquisadores saberem da existência dos outros trabalhos. A palavra "pacote" foi adotada do trabalho do NPL e a velocidade de linha proposta a ser usada no projeto da ARPANET foi promovida de 2,4 kbps a 50 kbps.

Em 1968 a *Bolt Beranek and Newman, Inc.* (BBN), através do trabalho de Bob Kahn, recebeu a incumbência de construir o protocolo *Interface Message Processors* (IMP), um dos componentes chaves da comutação de pacote. Em 1969 quatro computadores estavam interconectados usando protocolos *Host-to-Host* (*University of California Los Angeles* (UCLA), *Standford Research Institute* (SRI), *University of California Santa Barbara (UCSB)* e *University of Utah*.

Em Dezembro de 1970 o *Network Working Group* (NWG) finalizava o protocolo *host-to-host* denominado de *Network Control Protocol* (NCP). Em 1972, Kahn demonstrava a primeira aplicação quente da ARPANET, o correio eletrônico. A ARPANET crescia em direção da *Internet*.

Enquanto a ARPANET se desenvolvia, as organizações mundiais estavam implementando suas próprias redes tanto para comunicações intra-organização (dentro da organização) como para inter-organização (entre organizações). Apareceu uma ampla variedade de *hardware* e *software* de rede. Um desafio era obter a comunicação entre essas diferentes redes.

A idéia de rede em arquitetura aberta, foi primeiramente introduzida por Bob Kahn logo após ter chegado no DARPA em 1972. Seu trabalho era desenvolver um sistema em rádio, com pacotes, confiável e que mantivesse efetiva comunicação mesmo com interferências externas. Esse projeto era denominado "Internetting".

O protocolo NCP não tinha a capacidade de endereçar redes (e máquinas) e confiava na ARPANET para entregar os pacotes com fidedignidade, portanto algumas mudanças no NCP teriam que ser feitas. Assim, Kahn juntamente com Vint Cerf (um dos projetistas do NCP) decidiram desenvolver uma nova versão do protocolo que atendesse as necessidades do ambiente de arquitetura aberta. Eventualmete este protocolo foi denominado de *Transmission Control Protocol* (TCP). Na verdade o TCP de Kahn continha o TCP e o *Internet Protocol* (IP) usado nos dias de hoje. Kahn seguiu quatro regras básicas para desenvolver o antigo TCP: nenhuma mudança interna deveria ser feita em qualquer rede para conexão a *Internet*, os pacotes seriam re-transmitidos caso eles não chegassem ao destino, caixas pretas poderiam ser usadas para conectar as diferentes redes (*gateways* e roteadores) e não haveria controle global de operação.

Embora a *Ethernet* estivesse em desenvolvimento na empresa XEROX PARC ao mesmo tempo, a proliferação das LANs não foram pressentidas , muito menos os PCs e estações de trabalho. O modelo original era uma rede a nível nacional, assim um endereço IP de 32 bits foi usado.

Quando os computadores *desktop* surgiram percebeu-se que o protocolo TCP era muito grande para ser executado em computadores pessoais. David Clark e seu grupo de pesquisas no MIT compactaram o TCP e o implementaram em uma das estações de trabalho desenvolvida pela XEROX PARC.

O protocolo TCP trabalhava bem para transferência de arquivos e *login* remoto, mas para aplicações de voz, nos anos 70, ficou claro, que alguns pacotes perdidos não poderiam ser corrigidos pelo TCP. Assim reorganizaram o TCP em dois protocolos, o IP que provia somente o endereçamento e envio de pacotes individualmente, e separado, o TCP que estava focado no fluxo de controle e recuperação de pacotes perdidos. Para aplicações que não requeriam os serviços do TCP, uma alternativa foi desenvolvida, chamada *User Datagram Protocol* (UDP) provendo acesso direto aos serviços básicos do IP.

A transição do protocolo *host* NCP da ARPANET para o TCP/IP ocorreu no dia 1 de Janeiro de 1983. A ARPANET foi dividida em MILNET para fins militares e ARPANET para fins de pesquisa.

O número de *hosts* crescia devido a proliferação das LANS, já não era mais fácil manter uma tabela com o nome e endereço de todos os *hosts*. Em 1984, Paul Mockapetris, da *University of Southern California / Information Sciences Institute* (USC/ISI), desenvolve o *Domain Name System* (DNS). O DNS permitiu um mecanismo de distribuição escalonável definindo uma hierarquia no nome dos *hosts* (ex.: www.acm.org) para um endereço *Internet*.

Em 1985 a *Internet* já estava bem estabelecida como tecnologia suportando uma ampla comunidade de pesquisadores e começou a ser usada por outras comunidades (por exemplo, educação e comércio) como ferramenta de comunicação diária. Em 1988 foram proibidos os backbones para uso que não fossem de pesquisa e educação, surgia a *Internet* para o mundo de negócios.

Em 24 de Outubro de 1995, o *Federal Networking Council* (FNC) com unanimidade aprovou uma resolução definindo o termo "Internet". Esta definição foi desenvolvida em conferências com os membros da *Internet* e comunidades proprietárias de direitos intelectuais. RESOLUÇÃO: O FNC concorda que a linguagem seguinte reflete a nossa definição do termo "Internet". "Internet" refere-se à um sistema de informação global que – (i) está logicamente interconectado globalmente por um único espaço de endereços baseado no protocolo IP ou suas extensões subseqüentes; (ii) é capaz de suportar comunicações usando o conjunto de protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP) ou suas extensões subseqüentes, e outros protocolos compatíveis IP; e (iii) estabelece, uso ou acesso, tanto público ou privado, níveis de serviços em camadas de comunicação e infra-estrutura relacionada descrita aqui.

O maior serviço de comunicação baseado na *Internet* é o *World Wide Web* (WWW) idealizado no *Conseil Européen pour la Recherche Nucléaire* (CERN) (www.w3.org) por Tim Berners-Lee em 1989, e mais tarde aperfeiçoado por ele e Robert Cailliau em 1990.

A *World Wide Web* permite aos usuários de computador localizar e visualizar documentos baseados em multimídia (isto é, documentos com texto, imagens gráficas, animações, áudio e vídeo) sobre quase qualquer assunto.

Lee desenvolveu o primeiro servidor *Web* denominado "httpd", o primeiro pro-

grama cliente (navegador *Web* e editor) denominado "WorlWideWeb" e a primeira versão do "*Hyper Text Markup Language*" (HTML). O HTML é uma linguagem de marcação que especifica o formato do texto exibido em um navegador *Web* como o *Microsoft Internet Explorer* ou o *Netscape Communicator* (DEITEL et al. 2003).

A *Web* não é idêntica a *Internet*, é somente um dos muitos serviços de comunicação baseados na *Internet*. A relação entre elas pode ser entendida usando uma analogia com um sistema rodoviário global (Figura 5.1). Na *Internet*, como em um sistema rodoviário, três elementos são essenciais: as conexões físicas (rodovias e cabos), comportamento geral (regras de trânsito e protocolo *Internet*) e serviços (entrega de correio e o WWW). Maiores detalhes sobre esta analogia podem ser obtidas em (http://public.web.cern.ch/public/Content/Chapters/AboutCERN/Achievements/World WideWeb/WebHistory/WebHistory-en.html).

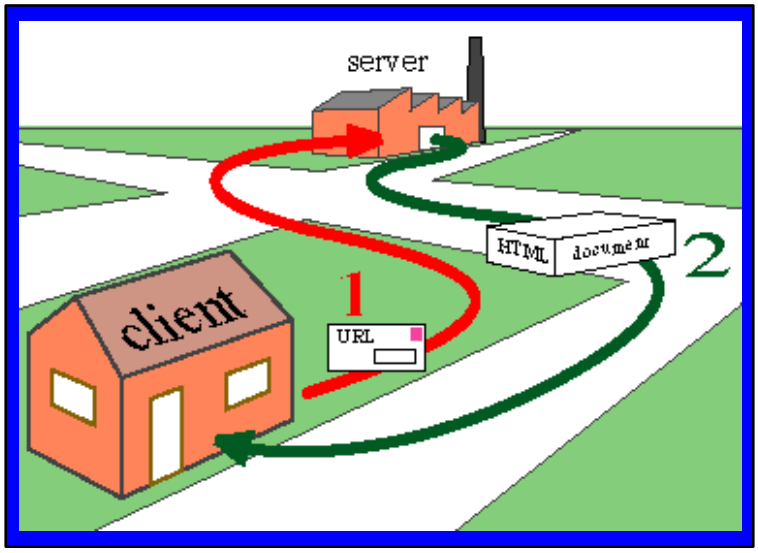


Figura 5.1: Analogia *Web/Internet* com sistema rodoviário global.

A *Web* é um mundo de informações disponíveis no *click* de um *mouse*. Para usá-la, necessitamos de um computador, uma conexão para *Internet* e um navegador. Ao digitarmos o endereço da página *Web*, conhecido como *Uniform Resource Locator* ou Localizador Universal de Recursos (URL), páginas de informações são mostrados na tela do computador. A função do navegador é interpretar e transformar os arquivos HTML em palavras e gráficos na tela do computador. Caso necessitemos de mais informações, tudo o que se tem a fazer é clicar num *hyperlink* (elemento

de hipermídia formado por um trecho de texto em destaque ou por um elemento gráfico que, ao ser acionado (geralmente mediante um clique de mouse), provoca a exibição de novo hiperdocumento). Todos os documentos *Web* estão armazenados nos computadores denominados servidores, representado pela imagem da fábrica. Usuários podem consultar estes documentos através de seus computadores, representado pela casa, através de uma requisição. Todos os computadores envolvidos na *Web* são conectados pela *Internet*, representado pelas rodovias.

Em 1993, o *National Center for Supercomputing Applications* (NCSA), na Universidade de Illinois, lançou a primeira versão do navegador "Mosaico". Foi o primeiro navegador com interação baseada no sistema Windows. A *Web* já era responsável por 1% do tráfego da *Internet*, o restante era acesso remoto, e-mail e transferência de arquivos. Detalhes sobre o desenvolvimento dos primeiros navegadores e servidores *Web* podem ser encontrados no site do CERN (http://public.web.cern.ch).

A primeira conferência mundial da *Web* (*First International World-Wide Web*) aconteceu no CERN em 1994, e neste mesmo ano, em Outrubro, Tim Berners-Lee fundou uma organização chamada *World Wide Web Consortium (W3C)* no *Massachusetts Institute of Technology, Laboratory for Computer Science* (MIT/LCS) em coloboração com o CERN, com suporte do DARPA e *European Comission* dedicada a desenvolver tecnologias inter-operáveis não proprietárias para *World Wide Web*. Um dos objetivos principais do W3C é tornar a *Web* internacionalmente acessível independentemente de deficiências, linguagens ou culturas.

No final de 1994, a *Web* tinha 10.000 servidores , dos quais 2.000 eram comerciais, e 10 milhões de usuários.

O ponto chave do grande sucesso da *Internet* e da *Web* é que elas envolvem padrões abertos em vez de proprietários, portanto, uma ampla comunidade de "Internautas" podem trabalhar juntos para criar e desenvolver tecnologias relacionadas a elas, muito diferente do desenvolvimento das redes em ambientes industriais.

5.3 Ferramentas já disponíveis para a *Web*

As seguir são relacionadas algumas das mais importantes realizações do W3C em termos de ferramentas disponíveis para a WWW. Mais informações sobre o trabalho do W3C podem ser obtidas no site www.w3.org.

- **Portable Network Graphics (PNG) 1.0**. É um padrão de imagem desenvolvido pelo W3C em resposta a uma decisão da corporação UniSys em começar a cobrar direitos autorais sobre o formato *Graphics Interchange Format* (GIF) (DEITEL et al. 2003).

- **Cascading Style Sheets (CSS)**. Permite especificar a apresentação dos elementos em uma página *Web* (fontes, cores, espaçamento) separadamente da estrutura do documento (cabeçalhos de seção, corpo do texto). Esta separação entre estruturas e apresentação simplifica a manutenção e *lay-out* de um documento *Web*.

- **Extensible Markup Language (XML)**. É uma linguagem para troca de dados, facilitando a interoperabilidade entre dispositivos. Os documentos XML só contém dados, sem instruções formatadas; desse modo, os aplicativos que processam os documentos XML devem decidir como exibir os dados do documento. Por exemplo, o PDA (*Personal Digital Assistant*) pode exibir um documento XML de maneira diferente de um computador *desktop* exibiria este documento. A XML permite criar marcação para quase todos os tipos de informação. Essa extensibilidade permite criar linguagens de marcação inteiramente novas para se descrever tipos de dados específicos, incluindo fórmulas matemáticas (MathML), reconhecimento de voz (VoiceXML), química (CML - *Chemical Markup Language*), indústria (*Business To Manufacturing Markup Language* (B2MML) definida pela ISA) e outras.

Além das realizações do W3C, outras ferramentas importantes para a *Web* foram desenvolvidas por empresas e associações. São elas:

- **JavaScript**. JavaScript é uma linguagem criada originalmente pela Netscape para criação de *scripts* (programas) que aprimoram a funcionalidade e aparência

das páginas *Web*. O *script* pode estar tanto do lado do cliente quanto do lado do servidor. Os *scripts* estão contidos no corpo do documento XHTML. O navegador *Internet Explorer* contém um interpretador JavaScript que processa os comandos escritos em JavaScript (DEITEL et al. 2003).

- **Active Server Pages (ASP)**. O *script* (programa) do lado do servidor utiliza informações enviadas por clientes, informações armazenadas no servidor, informações armazenadas na memória do servidor e informações da *Internet* a fim de criar páginas *Web* dinâmicas para os clientes. Os documentos *Extensible HyperText markup Language* (XHTML) são documentos estáticos, ou seja, todos os clientes vêem o mesmo conteúdo quando o documento é solicitado. As ASP são uma tecnologia da Microsoft para enviar documento dinâmico da *Web* para o cliente, incluindo XHTML, controles ActiveX (componente que estende as funcionalidades dos scripts, como acesso a árvore de diretórios), scripts do lado do cliente e miniaplicativos Java (ou Java applets, isto é, programas em Java do lado do cliente que são incorporados em uma página *Web*). A Active Server Page processa a solicitação (que freqüentemente inclui a interação com um banco de dados) e retorna os resultados para o cliente - normalmente na forma de um documento XHTML, mas outros formatos de dados podem ser retornados, por exemplo, imagem e dados binários. Os scripts do lado do servidor têm um espectro mais amplo das capacidades programáveis que seus equivalentes do lado do cliente. Por exemplo, os scripts do lado do servidor podem acessar a estrutura de diretórios de arquivo do servidor, ao passo que os scripts do lado do cliente não podem acessar o diretório de arquivo do cliente (segurança na *Web*) (DEITEL et al. 2003).

- **Commom Gateway Interface (CGI)**. É um padrão par interfacear aplicações externas com servidores HTTP. Cada vez que um usuário remoto modifica os parâmetros de controle usando um navegador *Web*, o servidor irá executar o programa CGI em tempo real (YANG & EAGLESON 2002).

- **Simple Object Access Protocol (SOAP)**. Muitos aplicativos utilizam a *Internet* para transferir dados. Alguns desses aplicativos são executados em clientes com pouco poder de processamento; dessa forma, tais aplicativos invocam métodos em outra máquinas para processar dados. Muitos desses aplica-

tivos utilizam especificações proprietárias de dados e protocolos, que dificultam a comunicação com outros aplicativos. A maioria desses aplicativos também reside atrás de *firewalls* de rede, que freqüentemente restringem a comunicação de dados em relação ao aplicativo. A IBM, a Lotus Development Corporation, a Microsoft, a DevelopMentor e a Userland Software desenvolveram o SOAP para tratar destes problemas. O SOAP é um protocolo baseado em XML que permite aos aplicativos se comunicarem facilmente pela *Internet* usando documentos XML denominados de mensagens SOAP (DEITEL et al. 2003).

- **Servidor Object Linking and Embedding (OLE) for Process Control (OPC)**. A grande motivação para se criar o padrão OPC é a necessidade de se estabelecer um mecanismo padrão de comunicação entre diferentes fontes de dados, sejam eles provenientes de equipamentos de campo ou até mesmo de outros arquivos de dados. A arquitetura da Informação em um Processo Industrial de Manufatura, envolve três níveis:

  - Gerência de Campo: com o advento dos equipamentos de campo inteligentes, uma grande variedade de dados provenientes desses equipamentos, tais como: dados de configuração e controle, podem ser disponibilizados para usuários ou mesmo para outras aplicações.
  - Gerência de Processo: a utilização de sistemas de controle do tipo SCADA e Sistemas de Controle Distribuído (SDCD) permite o controle descentralizado de processos industriais. Os dados fornecidos podem ser considerados conjuntamente de modo a permitir uma gerência efetiva e integrada de todo o processo industrial.
  - Gerência do Negócio: é a integração das informações de chão de fábrica e dos dados de gerência individual de cada processo controlado com os dados corporativos da empresa, administrativos e de aspectos financeiros.

  Os dados e informações podem ser utilizados por aplicações cliente de modo a otimizar a gerência e a integração de todo o processo de manufatura. Para realizar essa integração de modo efetivo, os fabricantes precisam ter acesso aos dados de chão-de-fábrica, dos processos industriais e integrá-los aos seus sistemas corporativos. Devem também poder utilizar as ferramentas disponíveis

nos sistemas SCADA, banco de dados utilizados, planilhas eletrônicas, etc., para atingir os objetivos da gerência integrada dos processos industriais. A chave para a realização dessa integração é uma arquitetura aberta e efetiva de comunicação baseada no acesso a dados e não no tipo de dados.

O OPC implementa dois grandes módulos: OPC Server e OPC Client. Enquanto o OPC Server especifica interfaces padrão de acesso direto aos equipamentos ou aplicações, o OPC Client especifica a interface padrão para as aplicações terem acesso aos dados coletados. A grande aceitação do padrão OPC pelas indústrias em todo o mundo trouxe vários benefícios:

  - Fabricantes de *hardware* têm de desenvolver apenas um conjunto de componentes para acesso aos seus equipamentos;
  - Desenvolvedores de *software* não têm de re-escrever *drivers* por causa de mudanças nos equipamentos;
  - Os usuários têm mais opções para desenvolver sistemas internacionais e integrados;
  - Simples de desenvolver;
  - Flexibilidade para "acomodar" as características de múltiplos fabricantes;
  - Alto nível de funcionalidade;
  - Permite uma operação eficiente.

- *Distributed Commom Object Model* **(DCOM)**. O OPC foi escrito baseado na tecnologia DCOM da Microsoft. O DCOM foi especialmente desenvolvido para colher dados através da rede, como também dividi-los através das aplicações nas estações de trabalho na rede *Ethernet* no nível *host*.

5.4 Modelo de referência *Internet* de quatro camadas

Normalmente descrevemos um modelo de referência como um conjunto de camadas. O modelo de referência TCP/IP possui quatro ou cinco camadas conforme Figura 5.2. Ressaltamos que este modelo não é baseado no modelo OSI da ISO, pois tanto

a *Ethernet* como os protocolos TCP/IP surgiram bem antes da criação do modelo OSI. Maiores detalhes sobre este modelo podem ser obtidos em (MOKARZEL & CARNEIRO 2004) e (SCRIMGER et al. 2002).

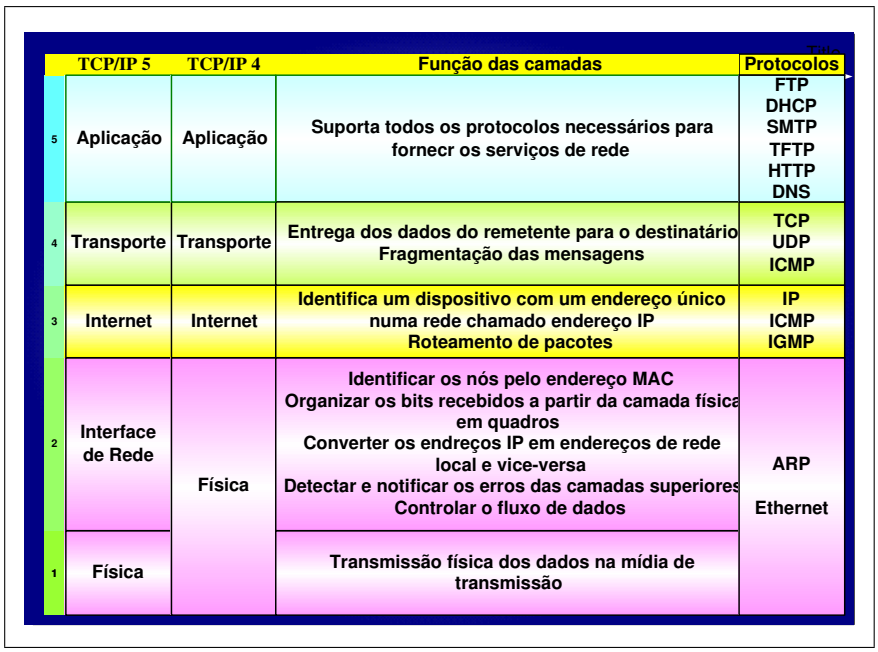


Figura 5.2: Modelo de referência *Internet* de 4 ou 5 camadas.

Atualmente a camada física ou acesso a rede mais popular para a LAN é a *Ethernet* (IEEE 802.3) que utiliza o cabo UTP categoria 5e (MOKARZEL & CARNEIRO 2004). A rede *Ethernet* é atualmente subdividida em quatro camadas, são elas:

- Especificação de mídia;
- Sub-camada física (PHY). O PHY é um componente dedicado, como o CS8900A da *Crystal semiconductor Corporation* ou o Realtek8019 da *Realtek Semiconductor Corporation*;
- Sub-camada de controle de acesso à mídia (MAC 802.3);
- Sub-camada de controle lógico do link (LLC);

Semelhante ao modelo ISO da OSI, em uma rede *Ethernet* as estações utilizam quadros (*frames*) para transmissão e recepção de dados. Um *frame* é uma seqüência de bits separada por delimitadores. No *frame* estão:

- **Preâmbulo**. Seqüencia de 64 bits para iniciar o processo de comunicação;
- **Destino**. Endereço físico único de 48 bits (MAC) do nó receptor do pacote;
- **Origem**. Endereço MAC do nó remetente do pacote;
- **Tipo/Tamanho**. Tipo de protocolo de nível mais alto transportado por este pacote;
- **Dados**. Dados transportados fornecidos pela camada *Internet*. Mínimo de 64 bytes e máximo de 1500. Uma informação de 64 bytes possui uma eficiência de 71%;
- ***Frame Check Sequence* (FCS)**. Verificação da integridade do quadro.

Ao receber um pacote, a camada *Ethernet* verifica o FCS. Se o FCS estive correto e o endereço de destino for o seu ou endereço de *broadcast* (FF:FF:FF:FF:FF:FF) esta camada envia o pacote para a camada superior. É bom lembrar que a camada *Ethernet* descarta qualquer pacote cujo FCS esteja incorreto.

Na camada física ou acesso a rede também reside o *Address Resolution Protocol* (ARP). Na realidade, o ARP separa o limite entre a camada *Internet* e a interface de rede (ou física). A função do ARP é converter os endereços IP em endereços físicos MAC. Quando um nó precisa descobrir o endereço físico de outro nó, ele transmite por *broadcast* um pacote especial. Esse pacote contém uma solicitação de endereço físico para um determinado endereço IP. Uma vez que o pacote é uma mensagem de *broadcast*, todos os dispositivos da rede recebem o pacote, mas somente aquele que possui o endereço IP destinatário é que responde com seu endereço IP e endereço físico. A funcionalidade do ARP está limitada à rede local, uma vez que a maioria dos roteadores não encaminha o tráfego de *broadcasting*.

Na *Internet*, a cada dispositivo é associado um endereço inteiro de 32 bits, denominado endereço IP. Ao contrário do endereço *Ethernet* que é fixo para cada dispositivo e não pode ser trocado, o endereço IP para cada dispositivo de rede pode ser estático ou dinâmico. Se uma LAN é conectada a *Internet* via roteadores, então o endereço IP deve respeitar regras de uso da autoridade InterNIC. Devido ao crescimento da *Internet*, passos estão sendo dados para implementação de um novo

formato IP, conhecido como Ipv6, que estenderá o número de bits usados para 48. Endereços IP podem ser "*unicast*" (um único destino), "*multicast*" (destinado a um grupo) ou "*broadcast*" (todos recebem).

Na camada *Internet* temos o protocolo *Internet Protocol* (IP), o *Internet Control Messaging Protocol* (ICMP) e o *Internet Group Management Protocol* (IGMP). O IP fornece os serviços de endereçamento e empacotamento. O IP identifica *hosts* locais ou remotos. Se o caminho para a rede de destino utilizar um tamanho diferente de pacote, o IP fragmenta o pacote para que ele seja transmitido sem erros. O IP então remonta ou empacota o pacote no *host* destino. Além disso, o IP descarta pacotes ultrapassados. Por fim, o IP envia os pacotes designados aos protocolos da camada de nível mais alto. IP não garante a entrega do pacote. IP é executado sobre *Ethernet* ou outra tecnologia LAN ou WAN. O ICMP é utilizado para informar ou diagnosticar problemas durante a transmissão. O IGMP é o responsável pelo gerenciamento do uso de *multicast* ou entrega seletiva sem broadcast.

A camada de transporte fragmenta as mensagens que chegam da camada aplicativo em segmentos e passa-os com o endereço de destino para a próxima camada *Internet*. Ela também fornece uma comunicação lógica entre os processos do aplicativo em execução em *hosts* diferentes. Os protocolos de transportes suportados pela camada de transportes são o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP). O protocolo TCP estabelece uma comunicação confiável e é orientado para conexão. O TCP estabelece uma conexão entre o processo na origem e o processo no *host* de destino antes de enviar os segmentos dos dados reais. Uma vez que a conexão seja estabelecida, os dados podem se transferidos em ambas as direções entre os dois *hosts* - um processo denominado de transferência de dados *full duplex*. Pelo fato de o TCP ser um protocolo confiável, ele resolve todos os problemas fundamentais de rede, como controle de congestionamento, seqüencia e controle de fluxo. O TCP trabalha somente no modo ponto a ponto e é usado em aplicações como *Terminal Emulation* (Telnet), *File Transfer Protocol* (FTP) e *Hypertext Transfer Protocol* (HTTP). Em uma aplicação industrial, o TCP poderia ser usado para carga de programas ladder entre estações de trabalho e CLPs, ou mensagens ponto a ponto entre dois CLPs. O protocolo UDP fornece uma entrega de dados não confiável. Essencialmente, o UDP envia os dados ao cliente receptor e espera

que ele seja recebido. As camadas de aplicativo que utilizam algoritmos próprios de checagem podem fazer uso do UDP, como por exemplo, programação de memória flash em dispositivos de rede. O UDP é menor e mais rápido que TCP. O UDP pode operar nos modos *unicast*, *multicast* e *broadcast*. Muito mais informações sobre estes protocolos podem ser obtidas em (SCRIMGER et al. 2002) ou (SCRIMGER 2002).

Os protocolos *Ethernet* e TCP/IP tornaram-se a maior plataforma de computação no mundo. Faz parte de sistemas operacionais distribuídos como Windows NT e Windows 2000, sendo uma escolha popular para rede de PCs. Muitas organizações possuem equipamentos de diversos fabricantes com TCP/IP disponível. O TCP/IP tornou-se uma escolha lógica para integração destes equipamentos numa LAN (BROOKS 2001). Mas o uso do TCP/IP não garante que a *Ethernet* possa conversar efetivamente, somente garante que a mensagem será transferida com sucesso entre os dispositivos. Para estabelecer este entendimento, um protocolo em comum na camada de aplicativo é necessário. A camada de Aplicativo deve fornecer atributos e serviços para aplicação em ambos os dispositivos. Aplicações típicas da *Internet* são *File Transfer Protocol* (FTP), *HyperText Transfer Protocol* (HTTP) e *Simple Mail Transfer Protocol* (SMTP). Embora esses serviços tenham sido estabelecidos sob orientação do IETF, a situação na indústria de automação não é tão simples. Cada vendedor de equipamentos na indústria de automação que trabalha com *Ethernet* TCP/IP implementou seu próprio protocolo na Camada de Aplicação. Como resultado, equipamentos de diferentes fabricantes conectados a mesma intranet de chão de fábrica podem fisicamente coexistir na LAN, mas não são inter-operáveis. Isto trouxe de volta todos os problemas relativos a padronização nos barramentos de campo.

5.5 *Ethernet* e *Internet* em aplicações industriais

Tanto (MUSKINJA et al. 2003) como (BROOKS 2001) descrevem o uso do protocolo TCP/IP em ambientes industriais. (MUSKINJA et al. 2003) descrevem os benefícios alcançados em unir os protocolos industriais com o *Ethernet* TCP/IP. Esses benefícios seriam o desenvolvimento de dispositivos com custo mais baixo que os baseados em protocolos proprietários, facilidade de uso e interoperabilidade entre

diferentes dispositivos de diversos fabricantes. (BROOKS 2001) descreve técnicas e mecanismos que são usadas para implementar um conjunto de serviços e objetos em uma rede baseada em *Ethernet* TCP/UDP/IP.

Uma arquitetura de automação industrial deve prover aos usuários três serviços básicos. São eles (BROOKS 2001):

- **Controle.** É o mais importante. Envolve a troca de dados (determinística) entre dispositivos de controle e E/S. Redes que atuam neste serviço devem transmitir dados com níveis de prioridade;
- **Capacidade de Configuração.** Essa funcionalidade tipicamente envolve uma ferramenta de programação de vários dispositivos do sistema;
- **Coleção de Dados.** Para visualização, análise e manutenção.

As redes que provêem os três serviços possuem flexibilidade e eficiência para um melhor desempenho de todo o sistema.

As redes que são baseadas no modelo Produtor/Consumidor (aonde o dado é identificado, em vez do endereço de fonte e destino, ou seja, um produtor irá transmitir, via broadcast, um pacote de dados no barramento, enquanto todos interessados consumidores podem pegar este pacote no barramento e usar os mesmos dados) podem suportar Controle, Configuração e Coleção de dados. Camadas de Aplicação que usam objetos distribuídos e serviços de comunicação Produtor/Consumidor satisfazem os requisitos das arquiteturas de automação.

No estabelecimento destes serviços não se pode assumir que um só nível de rede pode fornecer todos os requisitos da aplicação. Então uma arquitetura de rede deve provê consistência de dados entre os vários níveis de rede.

A arquitetura típica de automação (Figura 5.3) inclui um nível de informação (*Information Level*), que normalmente possui um segmento de rede *Ethernet*. Um nível de controle (*Control Level*), com alto índice de determinismo e média redundância, e um nível de dispositivos (*Device Level*) que requer baixo volume de dados mas com barramento robusto provendo energia e dados simultaneamente (como DeviceNet da *Open DeviceNet Vendors Association* (ODVA)).

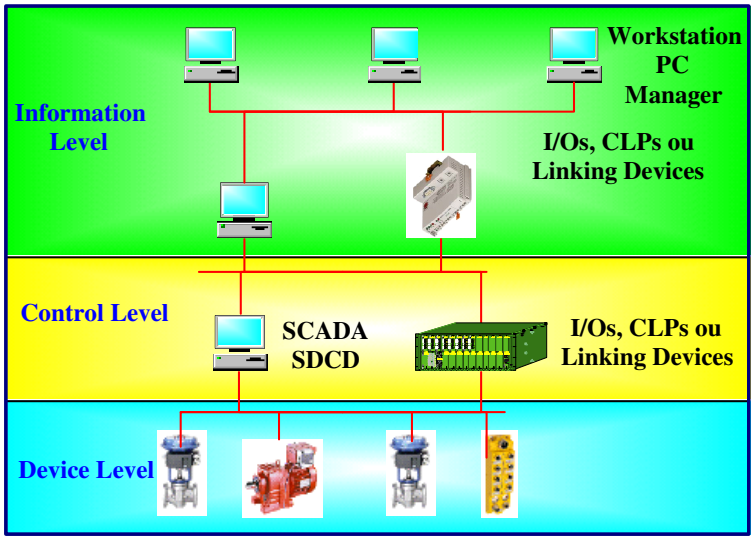


Figura 5.3: Típica arquitetura de automação com multi-camadas.

A ODVA e ControlNet International adicionaram o mais novo membro na família *Control and Information Protocol* (CIP), o *Ethernet Industrial Protocol (Ethernet/IP)*. Isto implementou um conjunto completo de Controle, Configuração e Coleção de Dados na rede *Ethernet* usados nos níveis de Informação e Controle na arquitetura típica da Figura 5.3.

Os benefícios que a maioria dos usuários *Ethernet/IP* esperam obter são (maiores detalhes em (MUSKINJA et al. 2003) e (BROOKS 2001)) :

- Interoperabilidade entre os dispositivos;
- Robustez. Os protocolos que trabalham acima da camada física, como o TCP/IP, são mais que depurados e suportam sofisticadas transferências de dados;
- Custos de infra-estrutura de rede tendem a cair de preço;
- Usar um protocolo estabelecido permite usuários e desenvolvedores de produtos preservarem seus investimentos em tecnologia, por exemplo, usuários de DeviceNet podem usar os mesmos dados dos usuários de Control-Net;
- A interface com os usuários, na camada de aplicativo, de diferentes protocolos

implementados em versões TCP/IP é praticamente a mesma, conseqüentemente a curva de aprendizado é bem mais rápida;

- Companhias podem instalar arquitetura aberta como *"Foundation Fieldbus High Speed Ethernet"* (FFHSE) com a segurança que dispositivos de outros fabricantes são interoperáveis, e podem ser intercambiáveis, se eles já eram no protocolo primário;

- Fácil uso dos protocolos para e-mail (SMTP) e para páginas *Web* (HTTP).

Um exemplo: havendo a necessidade de monitoração do status de uma unidade remota pode-se implementar um servidor HTTP neste dispositivo, no qual uma página *Web* é dinamicamente gerada com informações e dados relevantes. A maior vantagem do uso dos servidores *Web* é a possibilidade de ver as informações das páginas *Web* de qualquer navegador padrão, em qualquer plataforma capaz de exibir páginas HTML. A necessidade de desenvolver uma *Graphics User Interface* (GUI) é drasticamente reduzida.

Outro exemplo é relacionado ao uso de e-mails. Uma unidade remota pode enviar uma coleção de dados usando mensagens de e-mail para avisar uma variação anormal de parâmetros.

A adoção do TCP/IP assume uma importância em particular no nível mais baixos da rede (chão de fábrica): se todos as unidades estão em conformidade com o protocolo TCP/IP, métodos padrões de comunicação e serviços podem ser aplicados as redes locais como também controle remoto e monitoração de sistemas.

A seguir listamos alguns benefícios alcançados com implementação do protocolo CIP em *Ethernet/IP*:

- Compartilhamento de objetos e perfis pelos protocolos Controlnet, DeviceNet e *Ethernet IP (Industrial Protocol)* pois usam a mesma camada comum de aplicativo. Permitindo operação *plug-and-play* sem uso de *software* para dispositivos complexos de vários fabricantes;

- Provê aos usuários serviços de mensagem implícita (controle) e explícita (informação). Fornece todos os tipos de serviços que são essenciais no controle

das redes, ou seja, transferência de dados via *polling*, cíclica, ponto a ponto, *multicast* e mudança de estado;

- Introdução no mercado de mais de 500 produtos interoperáveis.

Todos estes benefícios não seriam alcançados se a *Ethernet/IP* necessitasse de uma nova construção de infra-estrutura de rede com específicos desenvolvedores da mídia física. Similarmente, estes benefícios não seriam alcançados se não houvesse conexão da *Ethernet/IP* com a rede corporativa. Compatibilidade com protocolos *Internet* é um DEVER. Isto significa que a *Ethernet* e TCP/IP estarão em todos os lugares.

Um dos argumentos que tem sido tradicionalmente usado contra o uso da *Ethernet* em sistemas de controle é que a *Ethernet* não é determinística. Envasar uma garrafa requer muito mais precisão comparado ao acesso de um arquivo armazenado num servidor remoto (MUSKINJA et al. 2003). Determinismo habilita usuários a predizer com exatidão o tempo de entrega de dados. Além disso, os usuários necessitam de uma alta garantia que os dados sempre chegarão neste tempo (repetibilidade). Muitos refinamentos na tecnologia *Ethernet* melhoraram o determinismo, repetibilidade e desempenho numa larga extensão (BROOKS 2001).

O problema fundamental em se aplicar a *Ethernet* no chão de fábrica estava no mecanismo half-duplex *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) ou "Método de Acesso Múltiplo com Verificação de Portadora e Detecção de Colisão pelo qual a rede *Ethernet* define o compartilhamento do meio físico e corrige erros de colisão dos dados. O problema é que este mecanismo é fundamentalmente não determinístico. Neste mecanismo a estação emissora verifica a existência de portadora antes de transmitir. Não havendo portadora a estação começa a transmitir. Mas pode ocorrer de duas estações transmitirem ao mesmo tempo e ocorrer uma colisão. O CSMA/CD de cada estação verificando a ocorrência da colisão, aguarda um tempo aleatório, conhecido como *Backoff* para nova tentativa de transmissão. O problema é que quanto mais colisões ocorrem, o tempo aleatório fica maior, o que causa sérios problemas para determinar quando as mensagens serão enviadas de fato. Quanto mais estações estiverem compartilhando o mesmo barramento, maior a probabilidade de ocorrer colisões. Também, quanto mais as

aplicações acessarem o barramento, maior a possibilidade de colisões, podendo ocorrer até o monopólio do barramento.

Uma rede *Ethernet* com taxa de utilização de 35% tem uma taxa líquida de transmissão de 2,5Mbps, e acima de 60% a rede já está congestionada. Recomenda-se uma taxa de colisão máxima de 10% como aceitável. Essa taxa de utilização pode ser lida via sistema operacional Windows NT (MORAES 2004).

Estão sendo introduzidos os seguintes refinamentos na *Ethernet* para que ela possa ser usada em controle de tempo real (POZZUOLI 2003) (BROOKS 2001).

- **IEEE 802.3x**. Operação *Full Duplex*. Permite uma efetiva conexão de 10Mbit/s (100Mbit/s) em cada direção, ou seja, 20Mbit/s (200Mbit/s) em cada dispositivo. Faz a *Ethernet* muito mais determinística.

- **IEEE 802.3p**. Priorização de Mensagens. Padrão implementado dentro de *switches* e *stack* TCP/IP para priorizar mensagens de alarme. Uma simples mensagem com atraso pode causar perda de produção.

- **IEEE 802.3q**. Virtual LAN. Característica que permite usuários configurar um grupo de *switches* de tal modo que as portas são subdivididas em grupos. Todos os pacotes recebidos por uma porta do grupo serão transmitidos para todas as outras portas dentro do mesmo grupo. Cada grupo forma uma VLAN. Evita que pacotes de broadcast sejam enviados para todas as estações ("Tempestades de *Broadcast*"). Outro benefício é aumento de segurança, pois pode separar equipamentos por grupos.

- **IEEE 802.3u**. *Fast Ethernet*. Sistemas de controle e automação que utilizam encapsulamento de dados *Ethernet* em 64 bytes e pequenos microcontroladores não são beneficiados pela *Fast Ethernet*, pois o desempenho destes dispositivos está limitado a velocidade dos processadores. Uma área em que a *Fast Ethernet* pode ser notada é na melhoria da recuperação de colisões.

- **IEEE 8021.w**. *Rapid Spanning Tree Protocol*. Permite a criação de arquiteturas de rede em anel tolerante a falhas que se reconfiguram em ms ao contrário dos 10 segundos no caso do protocolo original *Spanning Tree* 802.1D.

5.6 Automação industrial integrada à *Web*

(WOLLSCHLAEGER 2000),(WOLLSCHLAEGER 2001) e (WOLLSCHLAEGER et al. 2002) apresentam trabalhos relacionados ao uso da *Web* na indústria de automação. Tais trabalhos mostram as motivações, conceitos, pré-requisitos e detalhes de implementação para integração dos sistemas de controle e automação com a *Web*. Atenção especial é dada aos servidores *Web* baseado em XML.

Como discutido no capítulo 3, um único padrão *Fieldbus* dificilmente atenderá todas as necessidades de uma indústria. *Local Area Network* (LAN), na maior parte baseadas em *Ethernet* e protocolo TCP/IP, são usadas para interconectar diferentes tipos de redes *Fieldbus*. Portanto, o mapeamento dos componentes *Fieldbus* para as LANs tem sido amplamente discutido.

Integrar os sistemas de automação e controle com os últimos desenvolvimentos técnicos (*Web* e telefonia móvel) é uma decisão estratégica para as indústrias. Soluções baseadas na *Web* são disponíveis para uma ampla faixa de aplicações, por exemplo, em interfaces homem-máquina e ferramentas de engenharia. As tecnologias *Web*, tais como servidores *Web*, navegadores e gráficos tipo PNG, também são importantes para a integração vertical, preenchendo o vácuo existente entre o chão de fábrica e o nível corporativo.

A tecnologia de Servidores *Web* promete fazer com que clientes e servidores heterogêneos possam compartilhar aplicações usando módulos que são descritos, publicados, localizados e invocados através de uma rede de forma transparente. *Web Services* são ideais para projetos de integração. É a próxima geração de sistemas abertos de automação de chão de fábrica.

As principais razões deste crescimento são (WOLLSCHLAEGER et al. 2002):

- **Integração com sistemas baseados em navegadores** *Web***;**
- **Independência de plataforma;**
- **Sem pré-requisitos de** *hardware* **e** *software***;**
- **Sem custos de instalação;**

- **Acesso mundial;**
- **Sem limitações do número de usuários;**
- **E tecnologia já aceita pelos usuários.**

Em geral, a integração *Web* é uma decisão tomada em direção a protocolos e modelos abertos (WOLLSCHLAEGER 2001). Alguns problemas ainda devem ser resolvidos para a implementação de sistemas de alta qualidade apesar de todas as vantagens oferecidas. Um dos principais problemas é a questão da segurança (WOLLSCHLAEGER et al. 2002).

5.6.1 Conceito de integração

(WOLLSCHLAEGER et al. 2002) descrevem o conceitos de integração *Web*.

Um modelo geral de arquitetura *Web* é mostrado na Figura 5.4. A arquitetura consiste de três camadas. A camada mais baixa contém os *Fieldbus* e seus dispositivos. A camada intermediária ou portal contém o Servidor *Web*, que armazena as informações para os clientes que estão na camada superior. A camada intermediária mapeia os protocolos proprietários *Fieldbus* para o protocolo público HTTP. Assim, o portal contém a lógica de negócios, que transforma requisições dos usuários em aquisições de dados específicas e gerencia tarefas usando comunicação *Fieldbus*. Na camada superior, as soluções de gerenciamento baseadas na *Web* fazem uso intensivo dos navegadores *Web*.

Para implementar esta arquitetura, diferentes tipos de problema tem que ser resolvidos. O primeiro problema (P1 Figura 5.4) é o desenvolvimento do método de troca de dados entre a interface *Fieldbus* e o servidor *Web*. Desde que tecnologias baseadas em *Personal Computers* (PC) têm sido amplamente adotadas , conceitos como *OLE for Process Control* (OPC), baseadas no *Component Object Model* (COM) , podem ser usadas. Outros conceitos são baseados na linguagem JAVA. Como visto na Figura 5.4, os dispositivos de campo tem que implementar um servidor *Web* com interfaces apropriadas, de modo que eles podem ser acessados diretamente por protocolos IP e HTTP. Contudo, esta estratégia é independente do *Fieldbus* usado. Isto é um **CONCEITO PROMISSOR** para os fabricantes de dispositivos.

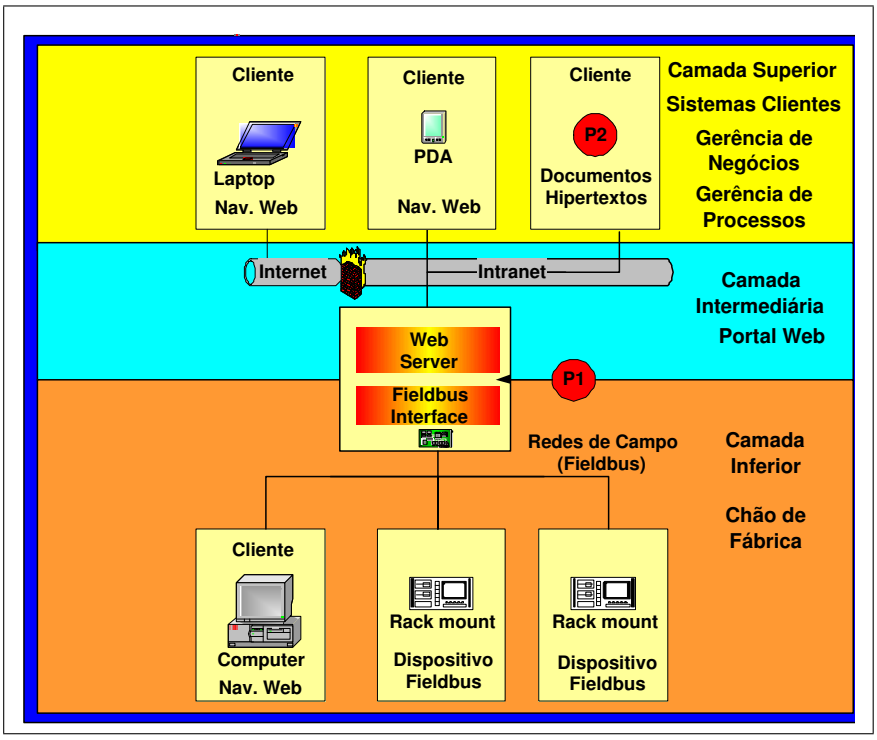


Figura 5.4: Arquitetura genérica de integração *Web*.

O segundo problema é o desenvolvimento do *front-end* para implementação das funções de gerenciamento (P2, Figura 5.4). Documentos hipertextos são usados em sistemas baseados na *Web*. Eles podem conter informações heterogêneas como gráficos, formulários, arquivos multimídia e muito mais. Isto permite o uso de soluções baseadas na *Web* para apresentar as informações de gerenciamento heterogêneas.

O terceiro problema é a gerência dos componentes de *software*. Enquanto a implementação dos componentes nos *websites* podem ser facilmente feitas pelos integradores de sistemas ou usuários, o desenvolvimento dos componentes de gerenciamento é usualmente executadas pelos fabricantes de dispositivos.

Desde que todas estas tarefas descritas na arquitetura de integração *Web* requerem uma excessiva troca de dados entre elas, interfaces de *software*, estrutura de arquivos e métodos de interação com objetos tem que ser definidos. Essas interfaces na maior parte das vezes são específicas dos fabricantes de dispositivos ou dos sistemas de barramento. Especialmente no desenvolvimento de sistemas heterogêneos, a deficiência de interfaces de troca de dados exclusivas ocasionam um problema de

integração de ferramentas de diferentes vendedores e sistemas.

Uma possível solução para este problema seria a introdução de uma descrição geral dos dados com sintaxe específica ao contexto e fácil acesso as interfaces para troca de dados entre os diferentes ambientes. Soluções baseadas no HTML já existem. O HTML é usado na descrição de documentos heterogêneos, que são encontrados em todos os lugares da *Internet*. A *eXtensible Markup Language* (XML) está substituindo as soluções tradicionais baseadas na HTML. A XML expande a descrição da linguagem HTML com o uso de marcadores, tipos de dados e estruturas. A XML possui algumas vantagens em relação ao HTML como uma nítida separação entre a descrição dos dados, os dados e sua representação nos navegadores *Web*. Em adição, declarar informações de sintaxe e semântica em arquivos separados (*Document Type Definition* (DTD)) permite re-usar a a estrutura da descrição em diferentes contextos. Isto provê um número de benefícios quando usando o mesmo arquivo XML para diferentes tarefas. Diferentes cenários (telas) podem ser implementados em cima dos mesmos dados. A descrição pode ser hierarquicamente organizada. Dependendo das funções a serem executadas, os dados XML podem ser filtrados e associados a componentes de *software*. A seleção da informação necessária e a definição dos detalhes de apresentação podem ser executadas por folhas de estilo, usando a *eXtensible Style Language* (XSL). Usar diferentes folhas de estilo permite acesso a dados específicos ao contexto. Outro método de acesso a arquivos XML é feito por analisadores sintáticos baseado em *Document Object Model* (DOM) que constroem uma estrutura de árvore que contém dados de um documento XML.

A descrição XML de um componente *Fieldbus* consiste de diferentes partes. Primeiramente, define-se a sintaxe e a semântica apropriadas relacionada ao *Fieldbus*. Isto inclui os marcadores e atributos, seqüencias e tipos de dados. Então, um modelo de conteúdo da descrição *Fieldbus* é criado. Este modelo contém um conjunto de Definições de Tipo de Documento (DTDS), esquemas, descrição dos tipos de dados e regras de transformações (DEITEL et al. 2003). DTDS e esquemas verificam se um documento XML é válido. Segundo, os arquivos XML são criados. O terceiro passo é desenvolver filtros (remover ou não um elemento de um arquivo) e regras de formatação em ordem de gerar a tela desejada em função do contexto. Isto pode ser feito através das folhas de estilo.

A transferência de dados em arquivos XML causará perda de eficiência do mecanismo de transporte assim como processamento extra do computador para leitura e escrita destes arquivos, principalmente quando se trata de pequenos sistemas embarcados. Em compensação outros benefícios são alcançados, principalmente no lado dos servidores. São eles:

- A transformação de arquivos XML usando folhas de estilo (XSLT) permitem gerar qualquer formato de saída desejado. Típicos exemplos são HTML para navegadores *Web* ou *Wireless Markup Language* (WML) para navegadores em celulares móveis *Global System for Mobile Communications* (GSM). Ambos os navegadores usam os mesmos dados, somente outra folha de estilo é gerada.
- *Softwares* como *parses* (analisadores sintáticos) XML são de código abertos e sem custo, disponíveis para linguagens como C, C++ e Java.
- Uso do bem conhecido protocolo HTTP como protocolo de encapsulamento dos arquivos XML. Como o protocolo HTTP usa o protocolo TCP como meio de transporte haverá a garantia que os dados serão entregues.

A seguir são feitas algumas observações sobre o uso da linguagem XML utilizada em nosso estudo de caso. São elas:

- Os marcadores a serem utilizados são da linguagem B2MML;
- Os arquivos XML enviados dos DICs para o Servidor *Web* só são montados na hora em que são transmitidos. Estes arquivos são montados com a união dos dados dos resultados dos testes das placas eletrônicas lidos de um arquivo seqüencial com um formulário;
- Os arquivos XML enviados não possuem DTD (Definição de Tipo de Documento) ou esquemas;
- Os analisadores sintáticos utilizados pelos DICs na recepção dos arquivos verificam a marcação B2MML e extraem os dados. Como não estamos usando DTD (Definição de Tipo de Documento) ou esquemas os analisadores são não

validadores. Lembramos que nossa aplicação está baseada no uso de microcontrolador de 8 bits, então estamos simplificando ao máximo o uso do XML e seus recursos para não aumentar a carga de processamento de nosso microprocessador;

- O Servidor *Web* utilizado em nosso microcontrolador de 8 bits não é capaz de unir arquivos XML com folhas de estilo. Portanto, na geração das páginas HTML não usamos os dados do arquivo XML, e sim as variáveis das estruturas da linguagem C. De modo que as páginas dinâmicas dos resultados de teste são feitas com a união de um formulário com a estrutura de dados da linguagem C.

5.7 Conclusão

O uso da *Ethernet* 100Mbps, *switches* e operação *Full Duplex* têm reduzido a probabilidade de colisões fazendo o uso da *Ethernet* uma opção de baixo risco a ser utilizada em sistemas de E/S.

O uso da *Ethernet* TCP/IP em muitas redes fez com que houvesse uma grande oferta de dispositivos baseados nesta rede, com custo baixo, compatíveis entre si e com grande capacidade de vazão de dados. O grande passo a ser dado é a comunicação distribuída em objetos. Assim sendo, dispositivos com rede *Ethernet* no chão de fábrica terão que operar com aplicativos de informações e suporte ao controle, tudo na mesma rede. Requisito de interoperabilidade na mesma rede será inevitável. Alcançar este objetivo requer o uso de um protocolo de comunicação com diversas características, são elas:

- Baseado em TCP/IP e UDP/IP;
- Implementação em uma estrutura de dados orientada a objetos;
- Baseado em padrões industriais abertos;
- Provimento de um modelo efetivo de mensagens de E/S;
- Permitir controle e informação existirem na mesma rede;

- Reunir diversos requisitos das indústrias de automação e controle, e;
- Ser aceito e implementado por muitos fabricantes.

A integração *Web* em sistemas de comunicação fabril são um grande passo para estabelecimento de um protocolo em comum, desde o nível de chão de fábrica até o nível de informação em uma típica arquitetura de automação. Protocolos como HTTP, linguagem como HTML e descrição de dados XML são tecnologias em direção a arquiteturas abertas.

O uso de uma linguagem de marcação como o XML provê um número de vantagens: criação de marcação para quase todos os tipos de informações, um documento XML é legível tanto por seres humanos como por máquinas, analisadores sintáticos verificam a sintaxe dos documentos XML e extraem os dados, folhas de estilo filtram informações relacionadas ao contexto, etc. XML representa um padrão em métodos de descrição universal, permitindo acesso aos dados independente da plataforma. Também muito importante, é que a linguagem XML é também usada por outras tecnologias *Web* que podem oferecer um novo cenário de aplicação a arquitetura de automação, como telefonia móvel e monitoração remota.

Mas alguns problemas ainda têm que ser resolvidos, tais como segurança relacionadas a *Web*, assim como *overhead* nos mecanismo de transporte de dados na redes de comunicação, sobrecarga no processamento de arquivos XML e a não possibilidade de usar *Ethernet* em áreas intrinsecamente segura devido a baixa capacidade de fornecimento de corrente pelos dispositivos *Ethernet* juntamente com os dados. Mas como a velocidade da rede *Ethernet* está cada vez maior e os microprocessadores estão cada vez mais velozes e consumindo menos corrente, isto deve ter uma influência menor no futuro.

Capítulo 6

Dispositivos de implementação para sistemas baseados na *Internet*

6.1 Introdução

Com o rápido crescimento da *Internet*, o desenvolvimento de sistemas de controle baseados na *Web* ou na *Internet* têm despertado um imenso interesse. Tais sistemas de controle tipicamente incluem controladores programáveis que podem ser monitorados e controlados pela *Internet*. Navegadores *Web* permitem usuários remotos ou operadores monitorar o status dos sistemas de controle ou processos de produção, modificar parâmetros de controle e enviá-los aos controladores programáveis usando protocolos da *Internet*, como *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP) e *Hypertext Transfer Protocol* (HTTP). Embora os sistemas de controle baseados na *Web* sejam relativamente novos, eles têm aberto um mundo novo de aplicações, tais como:tele-produção, tele-medicina, segurança remota e monitoração de pacientes nos cuidados da saúde (YANG & EAGLESON 2002).

Neste capítulo, apresentamos várias propostas para implementação do *hardware* de tais sistemas de controle. Especificamente, um sistema baseado na *Ethernet* TCP/IP. Lembrando, que deve-se estabelecer um compromisso entre o desempenho desejado e o custo de implementação no momento de se decidir qual das soluções deve ser a adotada para implementação do *hardware*.

Neste capítulo, o item 6.2 refere-se a implementação de sistemas embarcados para

Internet realizado por grupos de pesquisas com objetivos de monitoração remota, o item 6.3 descreve a arquitetura de sistemas de controle baseado na *Internet*, o item 6.4 descreve as principais características dos microcontroladores/micropocessadores, PC industriais e CLPs pois podem representar possíveis soluções para conexão com a *Internet*, o item 6.5 descreve como deve ser feito um *hardware* mínimo para conexão com a *Internet* usando microcontroladores e PHY *Ethernet*, como também indica vários módulos de sistemas embarcados com conexão à *Internet* integrada que podem ser adquiridos no mercado.

6.2 Trabalhos relacionados

Nos últimos anos, diversos grupos de pesquisa e indústrias desenvolveram sistemas de controle para monitoração via *Web*. Estes sistemas são baseados em computadores pessoais (PC) ou em sistemas embarcados.

(POSTOLACHE, GIRÃO, PEREIRA & RAMOS 2002) propõem o projeto e implementação de um sistema com vários sensores baseado na *Internet* para monitoração remota da qualidade das águas dos rios em Portugal. Eles propõem o uso de duas arquiteturas para implementação do sistema. São elas:

- A primeira arquitetura faz uso de uma interface serial RS232 de um computador pessoal (PC) para receber os parâmetros enviados por uma porta *Universal Serial Asynchronous Receiver Transmitter* (USART) de um microcontrolador. Os dados recebidos são processados pelo *software* LabVIEW e publicados na *Web* usando ferramentas de publicação LabVIEW Web;
- A segunda arquitetura é baseada na utilização de um microcontrolador PIC16F 877 da Microchip que automaticamente publica os parâmetros através de páginas *Web* armazenadas em uma memória EEPROM serial externa ao microcontrolador.

(YANG & EAGLESON 2002) têm como objetivo a implementação de um sistema de controle embarcado baseado na *Web* para monitorar o desempenho de sistemas e modificação de parâmetros de controle através de navegadores *Web*. O sistema

embarcado utilizado é o BL2000 da Z-World Inc. O módulo eletrônico possui um processador Rabbit 2000, entradas e saídas analógicas e conectividade *Ethernet*. A programação é feita em linguagem C.

No livro de (MOKARZEL & CARNEIRO 2004) temos um excelente trabalho como implementar um sistema embarcado mínimo para *Web* utilizando microcontroladores da família PICC 18 da Microchip Technology ou MPS430F149 da Texas Instruments. Este trabalho tanto contém o desenvolvimento do *hardware* quanto do *software*. Além disso, possui diversas referências para opções de compra de sistemas embarcados já prontos para uso na *Internet*.

6.3 Arquitetura de sistemas de controle baseados na *Internet*

A arquitetura de sistemas de controle baseado na *Internet* consiste em um componente de comunicação e um de medida. Estes componentes estão organizados, respectivamente, em dois blocos operacionais: um bloco de instrumentação e controle e um bloco de comunicação, conforme Figura 6.1 (POSTOLACHE et al. 2002).

O bloco de instrumentação e controle representa o principal elemento de qualquer sistema de monitoração e inclui os sensores, circuitos de condicionamento de sinais e o microcontrolador que executa aquisição e o processamento dos dados adquiridos em ordem de extrair valores numéricos dos parâmetros medidos.

O bloco de comunicação envia dados do processo monitorado ou do status de funcionamento do sistema de controle através de páginas *Web* solicitada por clientes da *Internet* que utilizam navegadores *Web*. O microcontrolador pode enviar as páginas *Web* diretamente para os clientes com o uso de um controlador *Ethernet* (PHY) ou indiretamente através de um PC.

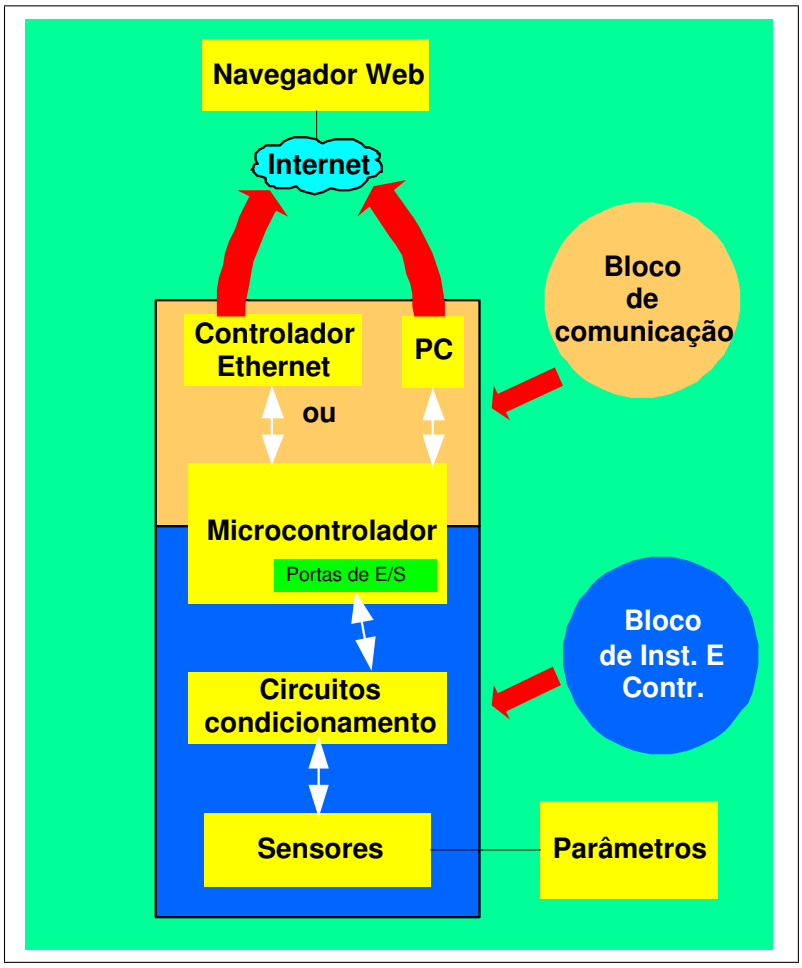


Figura 6.1: Arquitetura de sistemas de controle baseados na *Internet*.

6.4 Soluções propostas de *hardware* para implementação de sistemas de controle baseados na *Ethernet*

Como descrito no item 6.3, as arquiteturas de sistema de controle baseados na *Internet* podem fazer uso de microcontroladores, mais especificamente microcontroladores ou microprocessadores, computadores pessoais (PC) ou uso combinado dos dois.

Lembrando, que deve-se estabelecer um compromisso entre o desempenho desejado e o custo de implementação no momento de se decidir qual das soluções deve ser a adotada para a implementação do *hardware*.

A seguir serão fornecidas algumas informações sobre microcontroladores, PC In-

dustriais e CLPs.

6.4.1 Microprocessadores e microcontroladores

A evolução da automação industrial é a evolução dos microcontroladores / microprocessadores (mC/P). A evolução ocorre principalmente através da miniaturização, aumento da eficiência (maior capacidade de processamento com menos consumo) e flexibilidade.

Os mC/P surgiram para substituir soluções analógicas que utilizavam amplificadores operacionais ou mesmo soluções digitais que utilizavam inúmeros circuitos integrados. Como os microcontroladores (mC) são programáveis, problemas como flexibilidade, altos custos de implantação e confiabilidade passam a ser resolvidos mais facilmente. Com a evolução da micro-eletrônica, surgiram as redes industriais, filtros digitais substituindo filtros analógicos, a autodiagnose em dispositivos de campo e o reconhecimento de padrões através do uso de redes neurais e lógica *fuzzy*.

A evolução também trás benefícios para a logística. Um mesmo equipamento, através da programação, pode ser usado para executar várias funções, reduzindo o número de tipos de equipamentos em uso, facilitando o treinamento, configuração e manutenção. Conseqüentemente reduzindo os custos. Aliás os custos podem ser duplamente reduzidos, pois equipamentos conectados a *Internet* podem ter configuração e programas atualizados a partir de qualquer parte do mundo (SOUZA & MATA 2004).

O que difere basicamente entre um equipamento pequeno, de baixo custo e consumo de um equipamento com maior capacidade de processamento é o chip mestre do sistema, que pode ser um mC ou um microprocessador (mP). mCs/Ps são circuitos integrados digitais que fazem aquisição de dados em formato binário, processa-os e os devolve através de interfaces apropriadas. Os dois possuem CPU (Unidade Central de Processamento), sendo que no mP o poder de processamento é bem maior do que o do mC, mas os mCs possuem periféricos, tais como memórias de programas e dados, temporizadores, portas de entradas e saídas digitais (E/S), portas seriais, portas analógicas, etc. com custo menor que um mP. Em http://cmpmedia.globalspec.com/LearnMore/Semiconductor /Microprocessors_Microcontrollers podemos fazer

pesquisas de modelos e fabricantes de mCs/Ps em função das características dos periféricos desejados, assim como computadores industriais também. Na Tabela 6.1 temos as principais diferenças entre mPs e mCs. Já estão disponíveis no mercado mCs DSPs (processador digital de sinais) com arquitetura otimizada para processamento digital de sinais e os mesmos periféricos dos mCs. Os mCs DSPs são usados em celulares e PDAs. O baixo consumo dos DSP's pode beneficiar barramentos de campo que transportam dados e energia simultaneamente (SOUZA & MATA 2004). Hoje, existem módulos de mCs com *Ethernet* e começam a surgir no mercado, com grande evolução tecnológica mCs com controladores *Ethernet* (PHY) já embutido neles, como o *ColdFire* MCF5282 da Motorola. Isto permite monitoração em qualquer parte do planeta.

Tabela 6.1: Principais diferenças entre microprocessadores e microcontroladores.

	Microprocessadores	Microcontroladores
CPU (Capacidade de Processamento)	MAIOR	MENOR
Memória Interna	NÃO	SIM
Periféricos, E/S, Seriais, etc.	NÃO	SIM
Consumo	MAIOR	MENOR
Custo	MAIOR	MENOR

Usar um mC ou mP está baseado numa análise de custo/benefício. Os mCs apesar de serem mais limitados que os mPs em desempenho, satisfazem aplicações que não necessitam interfaces complexas, mas sim tamanho pequeno, baixo consumo e custo, viabilizando projetos que requeiram tais requisitos.

O fato é que microcontroladores em funções dos barramentos internos de processamento (8, 16 ou 32 *bits*) e microprocessadores possuem mercados bem definidos em função da relação custo/benefício. Por exemplo, fica inviável usar um microcontrolador de 32 *bits* em uma máquina de lavar roupa, já que o mesmo controle pode ser feito por um microcontrolador de 8 *bits* com memória limitada que custa em torno de 1 a 4 dólares (fonte de www.digikey.com). A desvantagem destes pequenos microcontroladores é a pouca capacidade de memória, portanto, devem ser programados em linguagem *Assembly*. Aumentando-se um pouco a complexidade dos sistemas faz-se necessário o uso de linguagens de alto nível, como C, C++, BASIC e

Java, pois aumentam a produtividade, eficiência e portabilidade, mas necessitam de mais memória para armazenamento dos programas. Se os microcontroladores de 8 *bits* não satisfizerem os requisitos dos sistemas, como capacidade de processamento, deve-se optar por microcontroladores de 16 ou 32 *bits*. Módulos eletrônicos para sistemas embarcados baseadas em processadores Intel, como o 586, já estão na faixa de preço de algumas centenas de dólares.

Por possuir características próprias, a área de automação e controle industrial impõe certos requisitos que a evolução dos mCs/Ps devem satisfazer. São eles:

- Mais desempenho, armazenar mais dados, mais funcionalidades e consumir menos energia para atender os requisitos de eficiência energética e requisitos de segurança para áreas classificadas;
- mCs com mais periféricos possuem plataformas mais flexíveis e otimizadas para atender um leque maior de aplicações;
- Miniaturização a fim de que os microcontroladores sejam utilizados em sensores inteligentes de campo.

6.4.2 PCs Industriais

PCs industriais são computadores pessoais especialmente fabricados para atender as adversidades dos ambientes industriais. Estas adversidades são representadas por: temperatura tipo ambiente militar, corrosão, umidade, interferência eletromagnética, vibrações e poeira. A Tabela 6.2 mostra como estas vantagens são evidentes (RODRIGUES 2004).

Tabela 6.2: Comparativo atual de PC industriais X comerciais.

CONDIÇÕES DE TRABALHO	INDUSTRIAL	COMERCIAL
TEMPERATURA	0 a 60° C	15 a 30° C
CHOQUE	5,0 G	0,5 G
VIBRAÇÃO	17 a 500Hz c/ 1,0G	Não Suporta
Umidade Relativa	10 a 95%	15 a 80%
Interferência Eletromagnética	Suporta	Não Suporta

Os PCs devem ser praticamente imune à falhas, ou seja, um alto MTBF (tempo mínimo entre falhas) . Os seguintes requisitos são necessários para atingir estes objetivos:

- Substituição dos discos magnéticos por cartões de memória *Flash*;
- Fonte chaveada com MTBF $>$ 50.000 horas;
- Uso de componentes militares;
- Montagem com tecnologia *Surface Mounting Device* (SMD).

Em um processo industrial, o tempo de parada deve ser o menor possível, por isso os PCs devem apresentar um pequeno MTTR (Mínimo tempo para reparo). Os seguintes requisitos são necessários para atingir estes objetivos:

- Led's para indicação das fontes de alimentação;
- Fácil acesso ao interior do microcomputador;
- Módulos periféricos removíveis;
- As CPUs estão localizadas em placas que são inseridas na vertical, ao contrário dos micros comerciais que são horizontal.

Uma das principais vantagens dos PCs é a possibilidade do uso de placas de aquisição de dados oriundas de diferentes fabricantes, estas placas podem ser usadas em diversas áreas como:

- Controle e monitoração industrial;
- Ensaios de testes de laboratório;
- Medidas e testes automáticos em sistemas;
- Telecomunicações, etc.

De uma maneira reduzida são as seguintes vantagens dos PCs, em relação aos CLPs e sistemas microcontrolados:

- Absorção de tecnologia proveniente dos micros comerciais;
- *Hardware* padronizado implica em manutenção mais fácil;
- Possibilidade de modernização;
- Ótimo para ser usado em aplicações de supervisão;
- Uso de ferramentas SOFT LOGIC que simulam no PC um CLP, com suporte as Linguagens do Padrão IEC 61131-3;
- Possibilidade de controle e supervisão ao mesmo tempo, eliminando a sala de controle climatizada que fica fora do chão de fábrica.

Apesar de todas as vantagens apresentadas os PCs apresentam o inconveniente de possuírem preços equivalentes ou maior que os dos CLP e sistemas microcontrolados.

6.4.3 Controlador Lógico Programável (CLP)

In 1968, um dos líderes da indústria de automóveis escreveu uma especificação de projeto para o primeiro controlador programável. O principal objetivo era eliminar os altos custos associados com as freqüentes substituições dos inflexíveis sistemas de controle baseados em relés. A especificação também era conhecida como "computador industrial em estado sólido" que poderia ser facilmente programado por técnicos de manutenção e engenheiros de sistemas de controle e automação. Era esperado que o controlador programável poderia reduzir o tempo para configuração dos sistemas e provesse fácil expansibilidade para futuras melhorias e trocas no processo produtivo. Em resposta a esta especificação de projeto, muitos fabricantes desenvolveram controladores baseados em computadores denominados de controladores programáveis (HUGHES 2001).

O CLP (Controlador Lógico Programável) foi criado para atender projetos de automação nas indústrias, através de facilidade de programação, interface direta

com atuadores e sensores, interface com IHMs e inversores de freqüência utilizando protocolos industriais. Possuem facilidade de diagnóstico e manutenção, alto índice de robustez, modularidade e capacidade de conexão com módulos de expansão.

A evolução dos CLPs está diretamente ligada com a dos mCs. Hoje, possuem a mesma capacidade de processamento de um computador pessoal simples, ou seja, clocks na centena dos MHZs, interface USB e serial , tela *touch screen*, *Ethernet* 100Mbps e muito mais.

A programação dos CLPs é feita através de linguagens de alto nível, mais intuitivas e amigáveis, como Blocos Lógicos (*Function Block Diagram* - FBD), Ladder (*Ladder Logic* - LAD) e Lista de Instruções (*Statement List* - STL).

Os CLPs são classificados como micros, pequenos, médios e grandes principalmente em função dos números de E/S. Micro CLPs geralmente têm até 32 E/S, pequenos CLPs até 256 E/S, médios CLPs até 1024 E/S e grandes CLPs têm mais de 1024 E/S.

Os seguintes componentes constituem um sistema de micro-automação:

- Micro CLP, com processador, fonte e E/S;
- IHM;
- Sensores e Atuadores de Campo;
- Opcionalmente:Expansão de monitoramento via celular ou linha telefônica.

Vantagens dos CLPs:

- Confiabilidade comprovada;
- Alto MTBF (Tempo mínimo entre falhas);
- Continuidade no controle;
- Respostas em tempo real;

- Adaptam-se muito bem a ambientes hostis.

Desvantagens dos CLPs:

- Sistemas fechados;
- Peças de reposição quase sempre oriundas dos fabricantes;
- Difícil integração com outros equipamentos.

6.5 *Hardware* mínimo para conexão *Ethernet*

Esta seção foi baseada nos trabalhos de (MOKARZEL & CARNEIRO 2004), (AXELSON 2003) e pesquisas na *Internet* dos fabricantes de módulos embarcados para *Internet*.

6.5.1 Diagrama em blocos

Para termos acesso a *Internet* um *hardware* mínimo deve ser construído. Os microcontroladores são a melhor opção em relação ao custo, mas devem atender alguns requisitos mínimos, como:

- **Memória de Programa**: não deve ser inferior a 32Kbytes;
- **Memória de Armazenamento de Dados**: não deve ser inferior a 2Kbytes de RAM;

  A Tabela 6.3 (RAJBHARTI 2002) mostra o uso de memória de programa e dados para implementação do *stack* TCP/IP e outros protocolos para a família de microcontroladores PIC C18 da Microchip Technology.
- **Portas de Entrada/Saída (E/S) ou Barramentos**: a comunicação com o controlador *Ethernet* usa no mínimo 8 *bits* para dados, 4 para endereços e 2 de controle, totalizando uso mínimo de 14 *bits*;
- **Velocidade de Processamento**: um processador muito lento ou carregado com outros sistemas pode deixar a resposta lenta, aconselha-se a usar um microcontrolador RISC.

Tabela 6.3: Uso de memória pelo PICC18 para o *stack* TCP/IP

Módulo	Memória de Programa (*words*)	Memória de Dados (*byte*)
MAC (Ethernet)	906	5
SLIP	780	12 + *buffers* transmissão e recepção
ARP	392	0
ARPTask	181	11
IP	416	3
ICMP	318	0
TCP	3323	42
HTTP	1441	10
FTP Server	1063	35
DHCP Client	1228	26
MPFS	304	0
Stack Manager	334	12+ICMP *Buffer*

Os dois componentes principais para um *hardware* mínimo de conexão a *Internet* são o microcontrolador, por exemplo o PIC18F8720 da Microchip Techonology ou o MSP430F149 da Texas Instruments, e o controlador *Ethernet* (PHY), por exemplo, o RTL8019AS da Realtek Semiconductors Corp. ou o CS8900A da Cirrus Logic (Figura 6.2). O controlador *Ethernet* executa as funções da camada física. Além dos componentes semicondutores, necessitamos de um transformador de isolação (separa eletricamente o PHY da rede) e um conector RJ-45. A interface para os controladores citados acima é bastante simples feita através de dados de 8 *bits*, lógica de seleção de endereço se necessário e *bits* de controle. Dependendo do microcontrolador usado, esta interface pode ser feita pelo barramento de dados, endereço e controle, ou usando um conjunto de pinos de entradas/saídas (E/S) dos microcontroladores. Normalmente os PHY trabalham como área de E/S, para isso são necessários os *bits* de controle IOR e IOW. Como veremos a seguir, os módulos da Rabbit Semiconductors usam interface usando barramentos do microcontrolador Rabbit 3000, enquanto os microcontroladores da Microchip usam conjuntos de E/S. O esquema elétrico do diagrama em blocos da interface do microcontrolador Rabbit

3000 para diversos PHY, como também, dos microcontroladores da Microchip série PIC18F para o controlador *Ethernet* (PHY) podem ser encontrados respectivamente em www.rabbitsemiconductors.com e www.microchip.com.

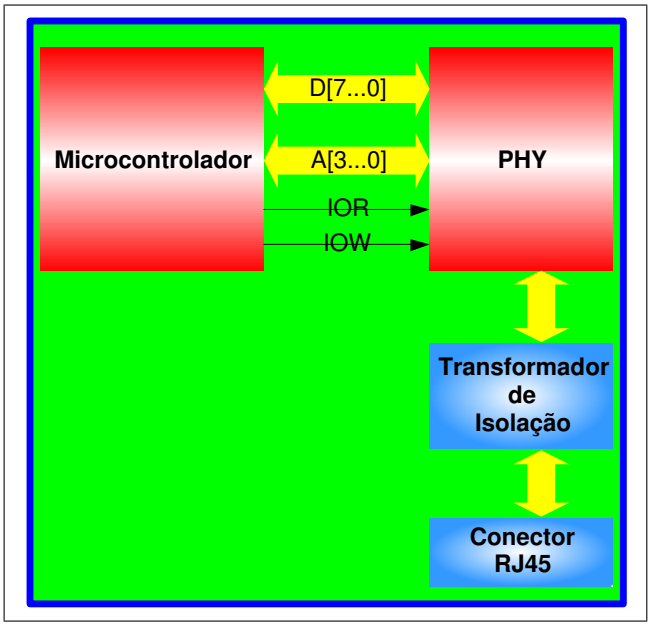


Figura 6.2: Diagrama em blocos de um hardware mínimo para conexão à *Internet*.

6.5.2 Construção ou aquisição do *hardware*

A seção 6.5.1 mostrou um diagrama de blocos para construção de um *hardware* mínimo para conexão a *Internet* usando microcontroladores de fácil aquisição no mercado. Normalmente, os fabricantes dos microcontroladores fornecem esquemas elétricos dos *kits* de desenvolvimento de acesso a *Internet* em seus sites. O problema é que fazer o layout da placa de circuito impresso, comprar os componentes e soldá-los na placa pode ficar com custo mais elevado do que comprar um módulo com conexão à *Internet* já pronto. Principalmente, se a intenção é fabricar poucos módulos. Então, é importante verificar algumas opções que podem facilitar o desenvolvimento de um projeto. Nesta seção vamos abordar alguns módulos, uns baseadas em *hardware* mínimo e outros que são sistemas dedicados.

A seguir são descritas várias opções para aquisição de *hardware* complementando o trabalho de (MOKARZEL & CARNEIRO 2004).

Módulo da Modtronix

A *Modular Electronic Solutions* desenvolveu dois sistemas baseados no *Application Notes da Microchip* para conectividade *Internet* usando o microcontrolador PIC18Fxxx, um denominado SBC44EC e o outro SBC45EC (Figura 6.3), a diferença entre eles são os pinos de acesso que o microcontrolador usa para acessar o controlador *Ethernet*.

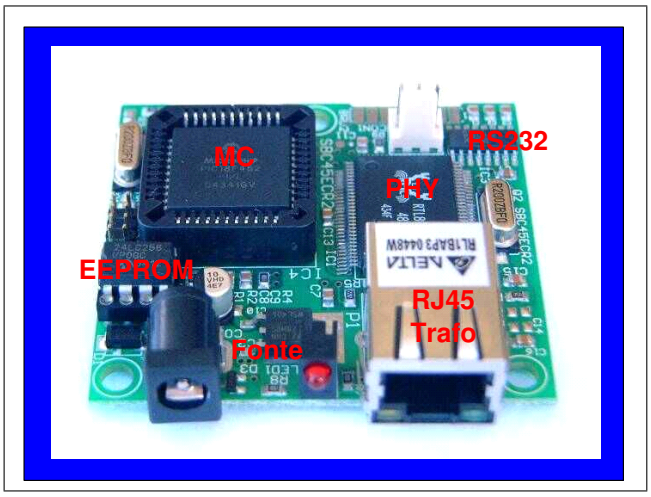


Figura 6.3: Módulo SBC45EC da Modtronix.

O módulo SBC45EC contém um microcontrolador Microchip PIC18F452 ou PIC18 F458, um controlador *Ethernet* Realtek RTL8019AS, uma interface RS232 e uma memória EEPROM 24LC256. O servidor *Web* Modtronix já vem instalado e para mudar a página *Web* basta usar o *Hyperterminal* do sistema operacional Windows.

Todo o código fonte e tutoriais para o Servidor *Web* Modtronix podem ser baixados sem nenhum custo no site do fabricante. Todo o *software* foi codificado em C fazendo usa dos compiladores HiTech ou Microchip MPLAB C18 para gerar o código fonte. Uma versão para estudantes do compilador Microchip MPLAB C18 está disponível sem nenhum custo no site da Microchip e pode ser usada para compilar novos programas. O código compilado gerado com a versão para estudantes é um pouco maior que a criada com a versão comercial.

Aplicações típicas deste módulo incluem:

- Servidor *Web*;
- Correio eletrônico;
- Conversor *Ethernet* para RS232;
- Conversor *Ethernet* para RS485;
- Controle remoto via servidores *Web* ((POSTOLACHE et al. 2002));
- Interface *Ethernet* para I2C, SPI e CAN.

Características:

- CPU PIC18F452;
- Servidor *Web* Modtronix;
- Memória *Flash* de 32Kbytes, memória SRAM de 1536 bytes e EEPROM serial 24LC256 de 32Kbytes. O servidor *Web* Modtronix usa aproximadamente metade da memória disponível EEPROM, deixando bastante espaço para as páginas *Web* dos usuários;
- 13 pinos de E/S dos quais 3 podem ser configurados como interrupções externas;
- Um *LED* vermelho para indicar que a placa está energizada;
- Um *LED* amarelo para indicar que o pacote está sendo transmitido;
- Um *LED* verde para indicar recepção de pacotes;
- Um *LED* vermelho para de indicação de estabelecimento de conexão;
- Uma interface serial RS232;
- Conectores de expansão para sinais da CPU como I2C, SPI, RS232, etc;
- Alimentação entre 7 a 30V DC;
- 10 BaseT *Ethernet*;

- Conector ICSP (*In circuit Serial Programming*);
- Suporta MAC, IP, ARP, ICMP, TCP, HTTP, FTP, DHCP e IP.

Módulo da Olimex

A Olimex desenvolveu seus módulos de conexão à *Internet* baseado no microcontrolador de 16 *bits* MSP430 da Texas Instruments. Uma das principais características deste microcontrolador é o baixo consumo.

O *kit* de desenvolvimento EasyWeb2 (Figura 6.4) é bastante completo e possui as seguintes características:

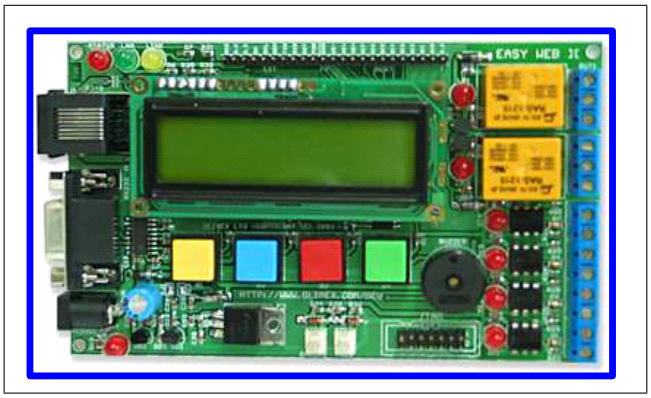


Figura 6.4: Módulo EasyWeb2 da Olimex.

- Microcontrolador Texas de 16 *bits* MSP430F149 de baixo consumo;
- TCP/IP de código aberto;
- Controlador de *Ethernet* CS8900 10 Mbps da Cirrus Logic, Inc;
- Conector RJ-45;
- 4 entradas com acopladores ópticos 4N37;
- 2 saídas com relés;
- Leds indicativos de status da LAN, relés, foto-acopladores e fonte;
- 4 chaves *push-button*;

- *Buzzer*;
- LCD de 2 linhas por 16 colunas;
- Memória *flash* de 64Kbytes para armazenamento de páginas *Web*;
- Dimensão: 138mm X 83mm.

O EasyWeb3 (Figura 6.5) basicamente não possui os foto-acopladores, reles e chaves *push-button*.

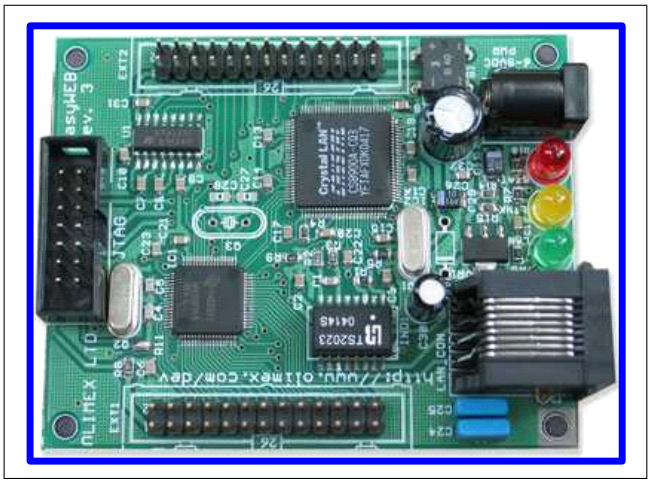


Figura 6.5: Módulo EasyWeb3 da Olimex.

Kit de desenvolvimento da Microchip

O PICDEM.net (Figura 6.6) é um *kit* de desenvolvimento da Microchip (www.microchip.com) que contém o *hardware* necessário para implementação de um sistema embarcado para *Ethernet* 10-Base T. O *kit* contém um microcontrolador de 8 *bits* PIC16F877 e um PHY da RealTek RTL8019S. Todo o código fonte necessário para aplicações *Internet* pode ser obtido gratuitamente no *site* da Microchip.

O usuário tem acesso imediato a *Internet* após configurar o endereço IP. A memória *Flash* do microcontrolador permite modificações ao programa de demonstração para adicionar novas aplicações. O PICDEM.net possui as seguintes características:

- Microcontrolador de 8 *bits* PIC16F877;

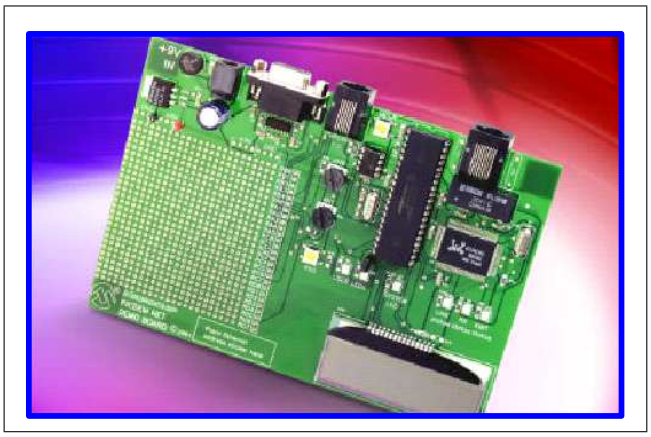


Figura 6.6: Módulo PICDEM.net da Microchip.

- *Stack* TCP/IP;
- Servidor *Web* com HTML;
- EEPROM serial 24L256 para armazenar páginas *Web*;
- Porta Serial RS232 para download da página *Web* via Xmodem usando o HiperTerminal do sistema operacional Windows;
- Possui conector para interface entre o microcontrolador e o *In-Circuit Debugger*, caso haja necessidade de re-programação da memória *Flash*;
- PHY Realtek RTL8019AS;
- Conector RJ-45;
- *Display* de LCD 2 linhas por 16 colunas;
- Área para expansão de circuitos.

Módulo da IINCHIP

Uma opção bastante interessante em termos técnicos é o circuito integrado W3100A, da IINCHIP (www.iinchip.com), que já possui a pilha TCP/IP integrada. A vantagem no uso desta opção é a não sobrecarga do microcontrolador responsável pelo controle do sistema, por outro lado, é um componente a mais no sistema e não temos acesso ao código fonte. O W3100A é colocado entre o PHY e o microcontrolador,

conforme Figura 6.7. O W3100A contém os protocolos TCP, IP, UDP, ICMP e ARP incluindo o endereço MAC, observe que não foi mencionado o protocolo SMTP. A comunicação entre o microcontrolador e o W3100A pode ser feita através de um barramento de endereços e dados ou através de uma porta serial I2C.

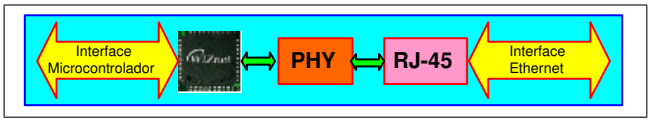


Figura 6.7: Diagrama de blocos para o W3100A.

Além do circuito integrado, a IINCHIP disponibiliza dois módulos baseados no circuito integrado W3100A. O módulo IIM7000 integra o W3100A com o controlador *Ethernet* Realtek RTL8201L (Figura 6.8). O módulo IIM7010 integra o W3100A, controlador *Ethernet* Realtek RTL8201L e conector RJ-45 (Figura 6.9).

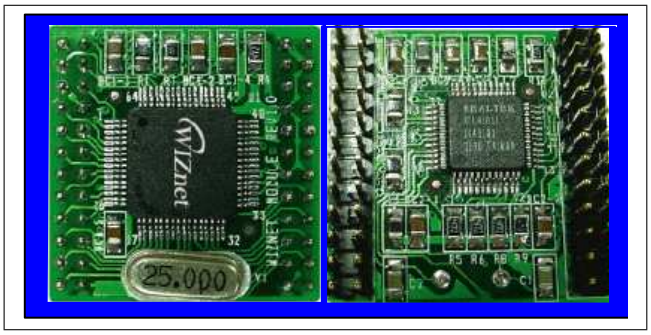


Figura 6.8: Módulo IIM7000 da IINCHIP.

Módulo da Atmel

A Atmel (www.atmel.com) oferece várias soluções para conectividade *Web*. As primeiras soluções foram baseadas na família de microcontroladores *Flash* 8051. Esta família de microcontroladores provê várias soluções embarcadas para *Internet* para vários tipos de redes como *Public Switch Telephone Network* (PSTN), LAN, *General Packet Radio Service* (GPRS) e links seriais.

A Atmel desenvolveu uma pilha do protocolo TCP/IP, livre de *royalties*, para família dos microcontroladores 8051, habilitando uma implementação fácil e sem

Figura 6.9: Módulo IIM7010 da IINCHIP.

custos de serviços *Internet* como HTTP e FTP.

O *kit* de desenvolvimento Web PSTN51S (Figura 6.10) é baseado no microcontrolador de 8 *bits* Atmel T89C51AC2 com memória *Flash* de 32Kbytes. Este *kit* de desenvolvimento possui um modem analógico V.32bis para conexão *dial-up* PSTN e uma porta serial RS232 para conexão direta com celulares GPRS.

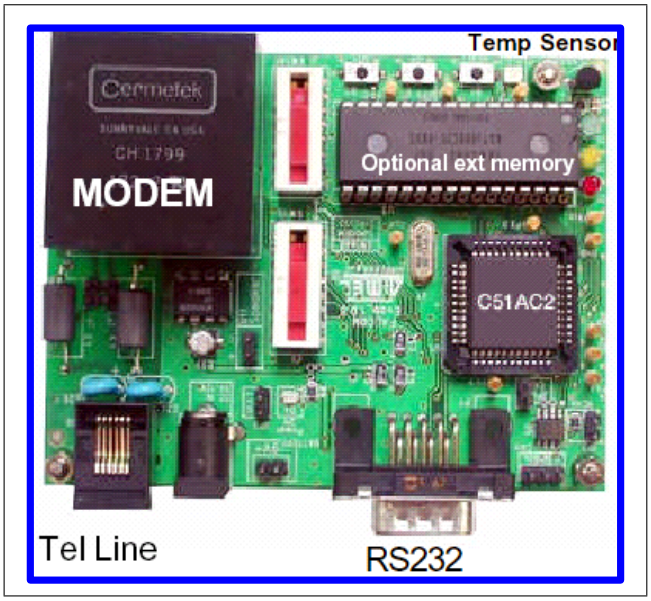


Figura 6.10: Módulo Web PSTN51S Atmel.

Em parceria com a WIZnet, líder em circuitos integrados contendo a pilha TCP/IP, a Atmel oferece um conjunto de ferramentas baseada nos microcontroladores C51. O Web LAN51H (Figura 6.11) é um *kit* de desenvolvimento para conectividade *Internet*

com microcontroladores Atmel *Flash* 80C51 e WIZnet IInchip.

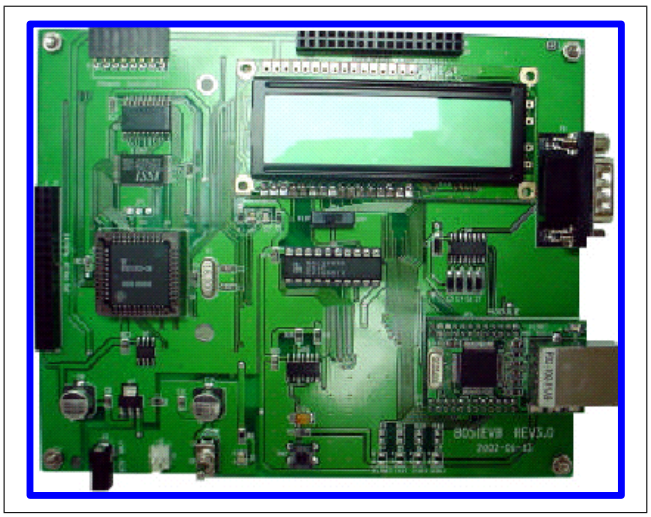


Figura 6.11: Módulo Web LAN51H da Atmel.

O *kit* de desenvolvimento Web LAN51H possui as seguintes características:

- Microcontrolador Atmel T89C51RD2;
- *Flash* de 64Kbytes para memória de programa;
- *Stack* TCP/IP no circuito integrado WIZnet W3100A;
- Protocolos: TCP, IP, UDP, ICMP, DHCP e ARP;
- MAC integrado;
- Transferência de até 400 Kbps usando C51 padrão;
- *Ethernet* PHY: Realtek RTL8201;
- Conector RJ-45 com transformador integrado;
- Software FLIP para gravação de programa na memória *Flash*;
- 4 Kbytes para o protocolo TCP;
- 4 Kbytes para o protocolo UDP;
- LCD;
- Configuração de rede através de interface serial;

- Programa monitor para fácil desenvolvimento de programas.

O *kit* Web LAN51H está disponível para três aplicações específicas. São elas:

- @Web LAN51H RC para aplicações de controle remoto;
- WebLAN51H VOIP para aplicações de transmissão de voz;
- Web LAN51H WC para aplicações de imagem (Figura 6.12).

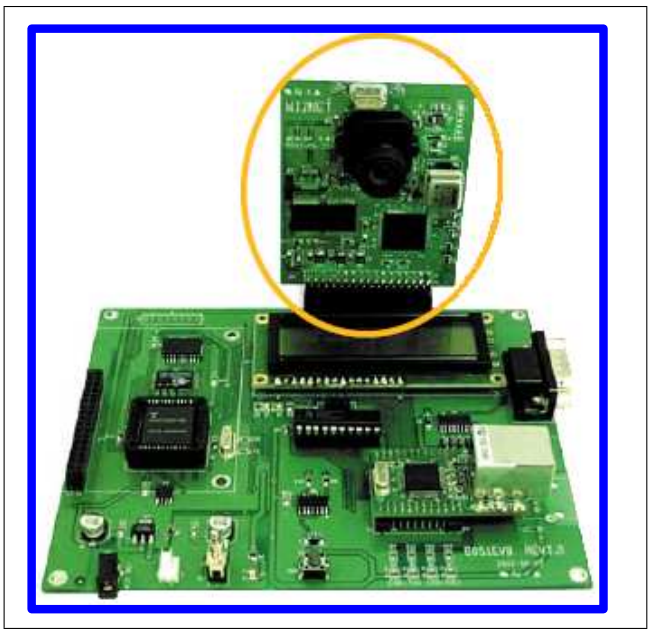


Figura 6.12: Módulo Web LAN51H WC da Atmel.

Módulos da Rabbit

Baseado em seus dois microprocessadores de 8 *bits*, o Rabbit 2000 e o Rabbit 3000 (ambos aperfeiçoamentos do antigo microprocessador Z80), a Rabbit Semiconductors possui vários módulos para conectividade *Internet*.

O que difere basicamente os módulos são: o microprocessador, a freqüência de oscilação do cristal em que está conectado o microprocessador, o controlador *Ethernet* PHY (10 BaseT ou 100 BaseT), a capacidade de memória *Flash* de programa, capacidade de memória RAM não volátil para dados e a quantidade de pinos de entrada e saída do microprocessador.

Acompanha os *kits* de desenvolvimento um CD contendo o compilador Dynamic C, manual dos microprocessadores Rabbit 2000, Rabbit 3000, *stack* TCP/IP, servidor *Web*, exemplos de código de todos os periféricos dos módulos, como seriais, teclado e *display*, e esquemas elétricos. Todo os exemplos de *software* estão escrito em linguagem C e assembly, fazendo uso dos compiladores Dynamic C ou Dynamic C Premier (contém RTOS) da Z-World para geração dos códigos fontes. Estes compiladores não são totalmente compatíveis com o ANSI C.

O módulo RCM3200 (Figura 6.13) usado no desenvolvimento do nosso estudo de caso, possui as seguintes características:

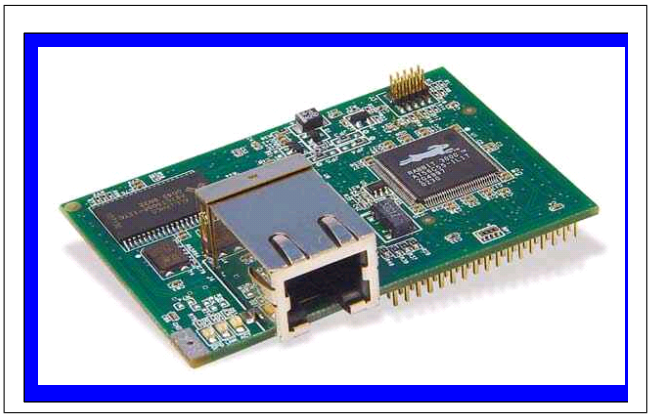


Figura 6.13: Módulo RCM3200 da Rabbit Semiconductors.

- Microcontrolador Rabbit 3000 a 44.2MHz;
- Controlador *Ethernet* PHY 10/100 Base-T (AX88796L da ASIX) e conector RJ-45;
- 512Kbytes de memória *Flash* para programa ;
- 256Kbytes de memória RAM não volátil para dados (deve-se somente conectar uma bateria de 3 Volts para não haver perda dos dados);
- 52 portas de E/S;
- 6 portas seriais (SCI ou SPI);
- *Real Time Clock*;

- Operação entre 3,15 a 3,45V DC, consumo de 244mA a 3,3V;
- Tamanho do módulo 69 X 47 X 22 mm;
- Preço para uma unidade: US$ 89.

Módulo da Dallas

A Dallas Semiconductor (Maxim) desenvolveu o módulo DSTINIm400 (Figura 6.14) baseado no microcontrolador DSC80C400 de 8 *bits*. Uma das principais características deste módulo é a possibilidade de ser programado utilizando-se a linguagem JAVA ou a linguagem ANSI C.

Um dos principais testes realizados com este módulo foi a verificação da portabilidade da linguagem JAVA. Utilizamos as mesmas APIs do protocolo MODBUS TCP/IP, disponíveis sem nenhum custo em www.sourceforge.net/projects/jamod , tanto no módulo da Dallas quanto em um computador tipo PC. As APIs foram compiladas pela *Java 2 Platform Standard Edition* para criação das classes e o respectivo pacote ".jar". No PC, basta executar o programa Java gerado pelo compilador, normalmente "java nomedoprograma.jar". No módulo da Dallas, é necessário converter os arquivos das classes em um formato que seja reconhecido pela máquina virtual do módulo da Dallas. Depois de instalado os programas no PC e no módulo da Dallas, foi estabelecida a comunicação entre eles sem nenhum problema.

O módulo DSTINIm400 (Figura 6.14) possui as seguintes características:

Características principais:

- Microcontrolador: Dallas DS80C400;
- 1MB de memória *Flash* para armazenar o programa;
- 1MB de memória RAM não volátil para armazenar os dados;
- Três portas seriais;
- Uma porta CAN2.0B;
- *Clock* em tempo real;

Figura 6.14: Módulo DSTINIm400 da Dallas Semiconductors.

- Suporte para um controlador *Ethernet* PHY (*Ethernet Physical Layer Transceiver*) externo. O PHY implementa a camada física.
- Preço: US$ 69.

Módulo da Lantronix

O XPort é a opção mais compacta para conectividade *Ethernet*, e também a mais completa. o XPort possui as seguintes características:

Figura 6.15: Módulo Xport da Lantronix.

- Inclui os protocolos: TCP, IP, UDP, ARP, ICMP, SNMP, TFTP, Telnet, DHCP, BOOTP e HTTP;
- Microcontrolador DSTni-LX 186;
- PHY 10/100Base T;

- 256Kbytes de RAM;
- 512Kbytes de *Flash* (dos quais 384 Kbytes são para páginas *Web*);
- Programação através de interface serial RS-232;
- Três linhas de E/S programáveis;
- Conector RJ45;
- Temperatura de operação: -40° a +85°C;
- Tensão de alimentação: 3,3VDC.

A grande desvantagem deste sistema é que o *software* fornecido pelo fabricante não é de código aberto, impossibilitando que alterações mais sofisticadas sejam efetuadas.

Módulo da Motorola

Uma opção bastante interessante , mas ainda não disponível quando iniciamos o desenvolvimento do nosso estudo de caso é o microcontrolador MCF5282 ColdFire da Motorola. Devido este microcontrolador de 32 *bits* operar com 60 MIPS, podemos explorar a velocidade da *Fast Ethernet*. As principais características deste microcontrolador são (MOTOROLA 2004):

- Dados e endereços em 32 *bits*;
- *Software* compatível com a família 68000;
- Suporte a depuração de programas;
- 64 Kbytes de memória RAM não volátil para dados;
- 2 Kbytes de memória cache;
- 512 Kbytes de memória *Flash*;
- *Fast Ethernet Controller* (10/100Base T);
- Módulo CAN2.0B;

- Três *Universal Asynchronous/Synchronous Receiver Transmitters* (UARTs);
- Módulo I2C;
- SPI;
- Conversor Analógico de 10 *bits*;
- Até 142 portas de E/S.

A NetBurner Inc. (www.netburner.com) fábrica módulos *Ethernet* baseado no microcontrolador MCF5282 ColdFire (Fig 6.16). Este módulo, além dos recursos do MCF5282, possui as seguintes características:

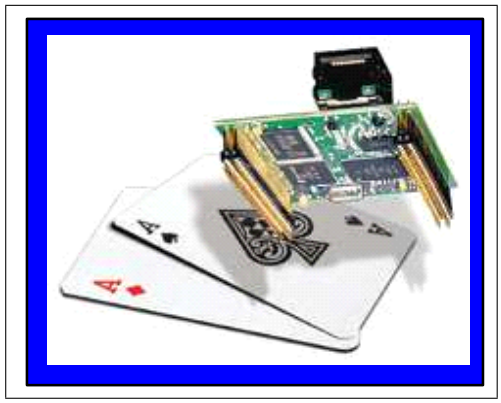


Figura 6.16: Módulo Mod5282 Processor Board da NetBurner.

- 8MB de memória RAM;
- Conector RJ-45 para *Ethernet*;
- Dois conectores de 50 pinos para acesso aos pinos do microcontrolador;
- Consumo de 500mA a 3,3 VDC;
- Dimensões: 51 X 66mm ;
- Temperatura de Operação: 0° a 70°;
- Preço: US$145.

A Tabela 6.4 resume as principais características dos sistemas embarcados estudados (observação: mC significa microcontrolador, mP: microprocessador, MA: módulo para aplicação, ou seja, pode ser soldado em uma placa, KD: *kit* de desenvolvimento).

6.6 Dispositivos inteligentes de campo (instrumentos virtuais)

No mundo do controle de processo, quando falamos em instrumentos baseados em tecnologia digital, ou seja, com plataforma de *hardware*, sistema operacional, comunicação em rede e informações de diagnósticos, faz mais sentido referir-se aos sistemas embarcados como Dispositivos Inteligentes de Campo (DICs) ou instrumentos virtuais (IV) (PINCETI 2004).

A descentralização baseada em dispositivos inteligentes de campo permite uma nova estrutura no processo de automação (SCHNEIDER 2003). Os DICs são compostos de:

- Um ou mais sensores;
- Um conversor A/D;
- Um microprocessador;
- Uma interface homem máquina; e
- Uma ou mais portas de comunicação.

As vantagens de uma inteligência distribuída são evidentes:

- Reduzem os custos iniciais de desenvolvimento dos projetos, além de permitir que um mesmo dispositivo de comunicação seja usado em muitas aplicações diferentes;
- Facilitam expansão e reconfiguração;

Tabela 6.4: Sistemas embarcados para *Internet*.

Fabricante	Site (www. .com)	Comentários
Embedded Ethernet.com	embeddedethernet	MA.PHY.Trafo.RJ-45.
Rabbit Semiconductors	rabbitsemiconductors	MA.KD. Até 1MB de memória *Flash* e 1MB de RAM. mP Rabbit 3000 de 8 *bits*. Linguagem C.
Olimex, Ltd.	olimex.com	KD. mC MSP430 da Texas Instruments. 4 entradas fotoacopladas, 4 chaves *push button*, LCD e EEPROM. Linguagem C
Dallas	maxim-ic	MA. KD. uC 8 *bits* família 8051 com endereçamento de memória até 16MB. Linguagem JAVA
Microchip Technology, Inc.	microchip	KD. uC PIC16F877. EEPROM serial 32K (24L256). Área de teste. LCD. Linguagem C.
Lantronix, Inc.	lantronix	Várias soluções para conectividade *Internet*, destacando-se o MA XPort, de tamanho extremamente pequeno. Protocolos TCP, IP, UDP, ARP, ICMP, SNMP, TFTP, Telnet, DHCP, BOOTP e HTTP. mC MCU DSTni-LX 186. Programação serial.
Netburner, Inc.	netburner	MA.MD. uC Motorola ColdFire 5270 , 512 KBytes de *Flash*, 2MBytes de SDRAM e 10/100 *Ethernet*.
Netmedia, Inc	siteplayer	MA. Pequeno, mas não possui o conector RJ-45, possui PHY, servidor *Web*. É baseada na família 8051. Programação serial.
WIZNet, Inc.	iinchip	MA. o uC W3100A já possui a pilha TCP/IP integrada. Com ou sem RJ-45.
ATMEL Corporation	atmel	MC. KD. uC T89C51RD2. *Stack* TCP/IP no c.i WIZNET W3100A. RJ-45. LCD. Programação serial.

- Proporcionam maior flexibilidade, seja quanto ao tamanho, configuração ou aplicação do sistema;
- Diminuem os custos de instalação, poupando fiação;
- Permitem que produtos e sistemas antes incompatíveis possam interoperar.

Merece atenção especial no estudo de caso a vantagem da flexibilização. Como a maior parte das indústrias do distrito industrial da ZFM faz o controle de qualidade em cima de aparelhos eletrônicos, mais especificamente, nas placas eletrônicas, os DICs devem possuir uma instrumentação que recebam vários módulos eletromecânicos que acessem estas placas. Por exemplo, um DIC pode testar placas eletrônicas desde televisores de 14"a 29", ou testar, *display* de celulares preto e branco como coloridos.

6.7 Conclusão

Na escolha dos equipamentos para conexão a *Internet* deve-se tomar o cuidado de não super dimensioná-los ou não prever futuras expansões. Aplicações de automação e controle podem fazer o uso de PCs ou CLPs com ou sem conexão *Ethernet*. Se a necessidade for muito específica, o circuito eletrônico de instrumentação e controle pode ser instalado numa placa PCI de um computador PC ou ser uma placa de um sistema embarcado, ambos podem ter conexão *Ethernet*, mas lembrando que os sistemas embarcados são mais baratos, em compensação possuem menor capacidade de processamento e memória. O número de E/S necessárias à aplicação, complexidade da lógica de controle e tempo de respostas determinarão a melhor escolha da tecnologia a ser implementada.

Capítulo 7

Ambiente de programação

Antes de discutirmos linguagens de programação é necessário conhecer o significado da expressão "sistema embarcado". Segundo (MAURER 2002), sistema embarcado é o *hardware* e *software* de um computador que formam um componente de algum sistema maior e do qual é esperado funcionamento sem nenhuma intervenção humana. Tipicamente, é um computador pequeno e de baixo custo, com um mínimo de recursos, executando uma aplicação específica. O sistema embarcado pode estar executando uma aplicação sem sistema operacional. Segundo (MORTON 2001), sistemas embarcados são sistemas eletrônicos que contém um microprocessador ou microcontrolador, mas nós não pensamos neles como computadores - o computador está oculto, ou embarcado, no sistema.

O item 7.1 trata das características das linguagens de programação para sistemas embarcados, e que nem sempre a escolha da linguagem de programação é definida exclusivamente por méritos técnicos. Este item também esclarece alguns pontos importantes sobre as linguagens Java e C/C++. O item 7.2 introduz o sistema operacional em tempo real (RTOS) pois a linguagem C ANSI não possui programação concorrente. O item 7.3 mostra as principais características da linguagem XML que é usada principalmente para troca de dados na *Web* entre diferentes plataformas de sistemas.

7.1 Selecionando a linguagem de programação

Segundo (MAURER 2002), as linguagens de programação dos sistemas embarcados devem possuir as seguintes características:

- Ser mínima em execução e pequena em tamanho, pois todo código executável deve estar contido em uma pequena memória ROM ou *Flash*, operando com o mínimo de memória RAM e executada em unidades centrais de processamento (UCP) de custo reduzido;
- Essas linguagens necessitam ser capazes de acessar detalhes específicos do *hardware* em muito baixo nível. Isto reduz o tamanho do programa associado com a interação do *hardware* especializado comum em sistemas embarcados para o qual *drivers* (programa ou rotina usada para interfacear e gerenciar um dispositivo de entrada/saída ou outros periféricos) não existem;
- O código fonte criado por estas linguagens devem ser de fácil leitura e manutenção pois os sistemas embarcados estão sempre evoluindo;
- As linguagens de programação para sistemas embarcados devem ser desenvolvidas para criar códigos fontes extremamente confiáveis;
- As linguagens para sistemas embarcados necessitam de controle de exceção por causa da natureza autônoma destes sistemas. Quando uma exceção ocorrer em tempo de execução o sistema deve recuperar-se e continuar a operação;
- As linguagens de programação para sistemas embarcados que operam com bateria devem minimizar as operações do processador e movimento de dados a fim de diminuir o montante de energia consumida.

Selecionar uma linguagem de programação pode estar baseado em fatores que não têm nada a ver com méritos técnicos da linguagem (NAIDITCH 1998). Esses fatores são:

- Disponibilidade, maturidade e eficiência dos compiladores;

- Disponibilidade de engenheiros com conhecimento da linguagem e *hardware* do sistema embarcado utilizado;
- Se a linguagem permite desenvolver facilmente interfaces com sistemas já existentes, como banco de dados;
- Existência de um legado de *software*;
- Como a linguagem pode refletir no custo final do sistema.

Pelos fatores expostos acima, em nosso estudo de caso, somente duas linguagens participaram do processo de seleção: C e Java. Além dos fatores expostos acima, o processo de seleção deve ser feito por pessoas que tenham reconhecido conhecimento das linguagens, mas que separem muito bem fatores técnicos de fatores emocionais.

Tanto (NAIDITCH 1998), (MOREIRA et al. 1998), (MAURER 2002) fazem estudos comparativos entre várias linguagens. Vamos relacionar alguns itens que necessitam de melhor esclarecimento, pois estão relacionadas com nosso estudo de caso. São eles:

C é mais eficiente que Java, pois está mais perto do código assembly. Devido ao fato de Java fazer uso intensivo de estrutura de dados dinâmica (alocação de memória) e por ter um automático *garbage collection* (limpeza de uma parte da memória cujos dados não estão mais em uso) e geralmente ser interpretada é mais lenta que C. O último problema pode ser suavizado se um compilador nativo (*just-in-time* (JIT)) estiver disponível. Em (http://shootout.alioth.debian.org) pode-se verificar que na maior parte dos *benchmarks* a linguagem C executa programas mais rapidamente que implementações em Java.

(MOREIRA et al. 1998) fazem uma comparação de computação numérica entre Java e C/C++. Os resultados para puro Java são desapontadores. Descaracterização da linguagem JAVA podem aumentar o seu desempenho, mas não são boas práticas de programação. Por exemplo, desabilitar checagem de *arrays* em tempo de execução.

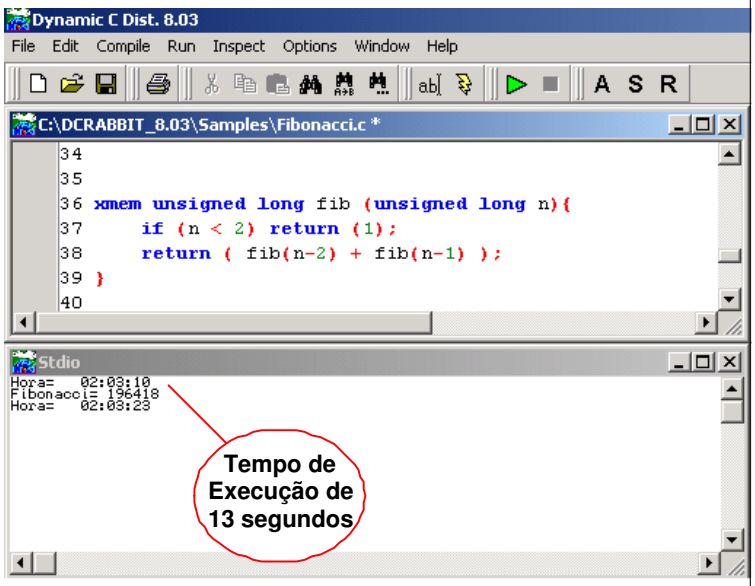
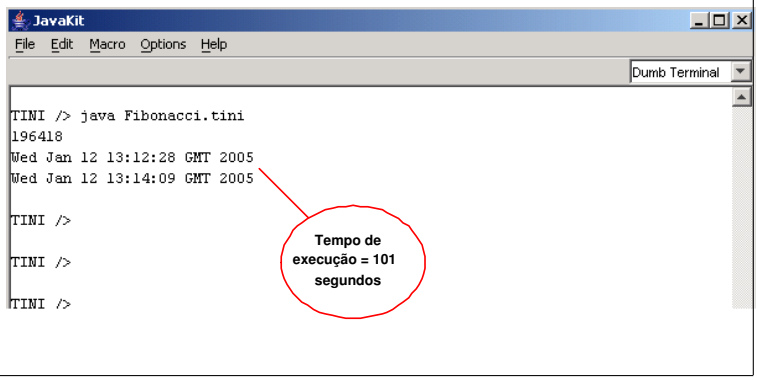
Fizemos um teste comparativo entre dois módulos microcontrolados de 8 bits na mesma faixa de preço (em torno de US$50),conforme Tabela 7.1, usando algoritmo

de Fibonacci . Os resultados para o conjunto Dallas DSC400 com Java forma desapontadores em nossos testes, também. O conjunto Rabbit RCM3200 com C foi muito superior ao conjunto Dallas DSC400 com Java. O tempo de execução do algoritmo foi de 101 segundos para o conjunto DSC400 com Java e 13 segundos para o conjunto RCM3200 com C. Esse é o principal fator que determinou a escolha da linguagem C em nosso estudo de caso. Quanto mais rápido o programa for executado, dentro das limitações de *hardware*, mais placas poderão ser testadas na linha de produção, refletindo no custo das mesmas.

- **Java é uma grande linguagem porquê o mesmo código pode ser executado nos sistemas operacionais Windows, Solaris, UNIX, Macintosh ou sobre outra plataforma**. A independência da plataforma JAVA é realmente uma maravilha. Mas esta independência não tem muito haver com a linguagem em si. Mas sim pelo fato que os *bytecodes* (instruções especiais ou códigos de operação) podem ser interpretados por qualquer *Java Virtual Machine* (JVM). A JVM é um computador abstrato construído de acordo com as especificações da Sun Microsystems (LEE, TAK, MAENG & KIM 2000). A JVM decodifica, interpreta e executa os bytecodes. A grande vantagem do JAVA é a portabilidade através dos tipos numéricos pré-definidos. Assim um tipo "int" tem sempre 32 bits. Já em C e C++ podem ser de qualquer tamanho. Se uma arquitetura não suportar implementação dos tipos numéricos Java, então os tipos devem ser implementados em *software*, os quais podem significativamente diminuir o desempenho do sistema;

- **Usar serviços de Sistemas Operacionais (SO) para programação concorrente não é tão bom como usar construtores de código concorrente.** Ao contrário do Java, C e C++ não possuem programação concorrente (código que pode executar muitas tarefas "ao mesmo tempo"). Para programação concorrente, C e C++ freqüentemente utilizam serviços de SO como semáforos que são difíceis de depurar. Com o uso de semáforos há o risco de bloquear um item e esquecer de desbloqueá-lo, de modo que outras tarefas podem também ficar bloqueadas quando tentam acessar o item compartilhado, a esta situação chamamos de *deadlock*, ou seja, o sistema fica em conflito;

Tabela 7.1: Teste comparativo RCM3200 com C e DSC400 com Java.

Módulo Microcontrolado	Resultado
RCM3200 com linguagem C	
Dallas DSC400 com Java	

- **Programação procedural tem sido substituída pela programação orientada a objetos.** Apesar de ter sido criado para o uso em programas com muitas linhas de código, o C++ torna-se uma opção interessante para programas pequenos em desktops (HOLZNER 2002), pois pode-se fazer sobre-carga de funções, ou seja, chamar a mesma função com parâmetros diferentes. Mas isto causa perda de performance (GHAHRAMANI & PAULEY 2003). O uso da linguagem procedural parece ser mais natural em sistemas embarcados em tempo real (NAIDITCH 1998);

- **Java é uma linguagem simples, pequena e fácil de aprender.** Embora a linguagem Java não possua muitos comandos e seja simples de usar, possui uma imensa biblioteca com mais de 150 classes e interfaces. Um esforço considerado deve ser feito para determinar qual a melhor classe a ser usada. Por outro lado, C torna-se mais complexo com aumento do programa e perde portabilidade por não ter uma biblioteca básica;

- **Linguagens de programação deveriam dar aos programadores liberdade para fazerem o que eles querem.** A grande vantagem do C/C++ é a facilidade de interação com o *hardware* do dispositivo para satisfazer as necessidades de controle, mas trazendo responsabilidades quando do uso avançado de ponteiros. Em C e C++ ponteiros podem apontar para qualquer endereço. Não causa nenhuma surpresa, que o excessivo abuso de ponteiros é responsável pela maioria dos erros de programação em C e C++. Estes ponteiros podem apontar para dados errados ou itens que não mais existem. A linguagem Java foi radical em evitar o uso de ponteiros. Linguagens como Ada permite o uso de ponteiros com segurança. Na linguagem Ada, verificações são feitas automaticamente para garantir que um ponteiro não aponta para um objeto do tipo errado ou programa que não mais existe.

O uso da linguagem C por hora, não quer dizer que num futuro próximo as linguagens JAVA ou C++ não possam ser utilizadas em nosso estudo de caso. Pois tanto (HARDIN 2000) como nós observamos o seguinte panorama durante o desenvolvimento deste trabalho:

- "A complexidade dos Sistemas Embarcados" está aumentando a cada dia, pois

o leque de aplicações está evoluindo bastante em áreas de telecomunicação, automação industrial, automação predial e sistemas automotivos. Assim sendo, os programas necessitam ser desenvolvidas de uma forma mais rápida. O uso de linguagens orientadas a objeto juntamente com ferramentas UML tornam isto possível. Com toda essa complexidade os sistemas precisam ser mais robustos. Mas tudo isto leva a uma falta de padronização no mercado. Existem muitos processadores, sistemas operacionais, compiladores, etc. Estes fatores estão levando ao uso da linguagem Java;

- Programas com muitas linhas de código são mais facilmente gerenciáveis com o uso de linguagens orientadas a objeto;
- Suporte a rede. Programas feitos em Java para rede são facilmente lidos;
- Alguns *bugs* (erros de programação) são cometidos devido ao uso excessivo de ponteiros nas linguagens C e C++.

(HIGUERA-TOELDANO & ISSARNY 2000) citam que Java é uma linguagem candidata a sistemas embarcados, mas que apresenta alguns problemas em aplicações de tempo real. Várias API foram ou estão sendo desenvolvidas para resolverem estes problemas. (BOLLELLA & GOSLING 2000) descrevem resumidamente a especificação para Java em tempo real (RTJS). E os primeiros frutos para aplicação em tempo real já começam a ser colhidos, como por exemplo, em aplicações para sistemas de aviação (SHARP, PLA, LUECKE & HASSAN II 2003).

7.2 Sistema Operacional MicroC/OS-II

Em plataformas de desenvolvimento simples, ou seja, de baixo custo e consumo mínimo de energia, todas as funções são consideradas igualmente importantes. Normalmente o sistema não estabelece prioridades. As funções são tipicamente escritas em C e executadas a partir de alguma memória ROM ou *Flash*. Não há sistema operacional e talvez não mais que 0,5K de memória RAM para armazenamento das variáveis do programa. Na mesma proporção do aumento de necessidades aumenta a necessidade de estabelecer prioridades. Torna-se mais importante ter alguma espécie

de mecanismo de decisão que seja parte do sistema embarcado. Os sistemas mais avançados atualmente têm um minúsculo e organizado sistema operacional sendo executado, gerenciando todas as tarefas do sistema. Eles são denominados de *Real-Time Operating System* (RTOS).

Mais de uma tarefa pode estar sendo executada concorrentemente em um dispositivo embarcado e é trabalho do RTOS direcionar este tráfego. Mas o quê é tempo real? Na indústria de sistemas embarcados usamos o termo "tempo real" para nos referirmos a um sistema que deve computar um resultado, baseado em certas entradas, num tempo aceitável máximo, isto é, deve responder em tempo real.

O C ANSI não possui programação concorrente. Utilizam serviços de Sistemas Operacionais multi-tarefa. Neste item será estudado o *software* uC/OS-II, um *kernel* (núcleo) preemptivo de tempo real. Os seguintes benefícios são obtidos com uso de um *kernel* preemptivo:

- Facilita a construção dos programas, pois permite que as tarefas sejam usadas novamente;
- Permite que as tarefas de mais alta prioridade tenham melhores respostas de tempo a eventos;
- Gerência o uso de recursos compartilhados.

Para uma descrição detalhada do *kernel* uC/OS-II consulte o livro (LABROSSE 2002) que inclui um disquete contendo o código fonte do *kernel*. Por simplicidade, deste ponto em diante nós vamos referir ao *kernel* simplesmente como uC/OS. Interfaces para diversos microcontroladores e microprocessadores podem ser obtidas em (www.ucos-ii.com). O uC/OS é um *kernel* preemptivo multi-tarefa em tempo real. Está escrito em C com exceção de um pequeno módulo em assembly que é específico para cada CPU. Isto significa que o uC/OS pode ser adaptado facilmente para diversos processadores. De fato para uma ampla faixa de processadores, desde simples micro-controladores de 8 bits como o 8051, até CPUs de 32 bits. Uma desvantagem de se ter escrito o uC/OS em linguagem C é uma pequena perda de velocidade

e eficiência do tamanho do código. Contudo, isto é um pequeno preço a se pagar em relação à versatilidade e portabilidade que a linguagem C propicia. O uC/OS pode ser configurado para ser pequeno e eficiente em CPUs de 8 bits com limitada memória e velocidade de processamento, ou pode ser configurado para grandes sistemas com ilimitados recursos. Há muitos núcleos em tempo real disponíveis para microcontroladores de 8 bits e 16 bits como os Motorola 68HC11 e 68HC12, mas o uC/OS tem uma boa virtude que é oferecer muitos recursos em relação ao pouco *hardware* exigido para funcionamento. Uma outra vantagem é que o código fonte está disponível, desde sua criação em 1992, e impecavelmente bem escrito e muito bem documentado no livro (LABROSSE 2002).

As tarefas podem ser codificadas independentemente umas das outras como elas tivessem controle completo sobre a CPU. A Figura 7.1 mostra um fluxograma para um sistema que usa um *kernel* preemptivo. As tarefas são de dois tipos: as relacionadas a inicialização e outras de execução que estão contidas em um loop sem fim. Se a CPU estiver executando uma tarefa que não é a de mais alta prioridade, o *kernel* pode sobrepor uma tarefa por outra de mais alta prioridade.

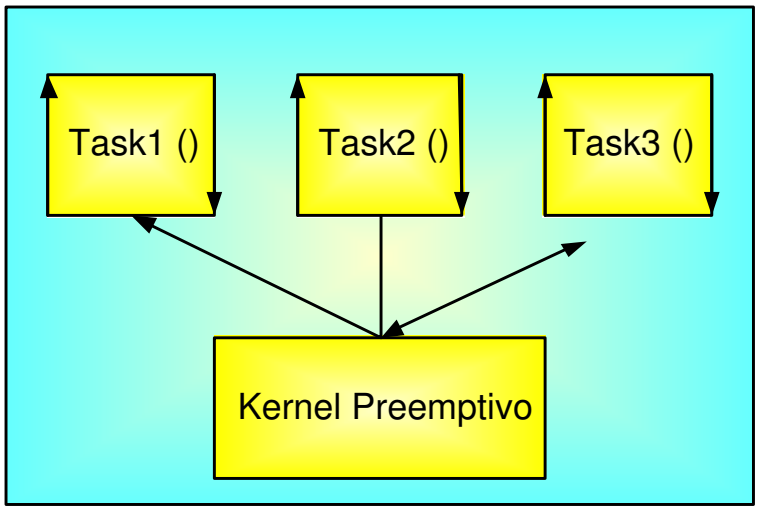


Figura 7.1: Fluxo de um programa multi-tarefa preemptivo.

Um *kernel* preemptivo além de facilitar a construção de programas, permite o aproveitamento das tarefas e possibilita que as tarefas de mais alta prioridade tenha melhor tempo de resposta a eventos. Para ilustrar esta situação, vamos ver o tempo de resposta a um evento que é detectado através de uma interrupção, mas servido

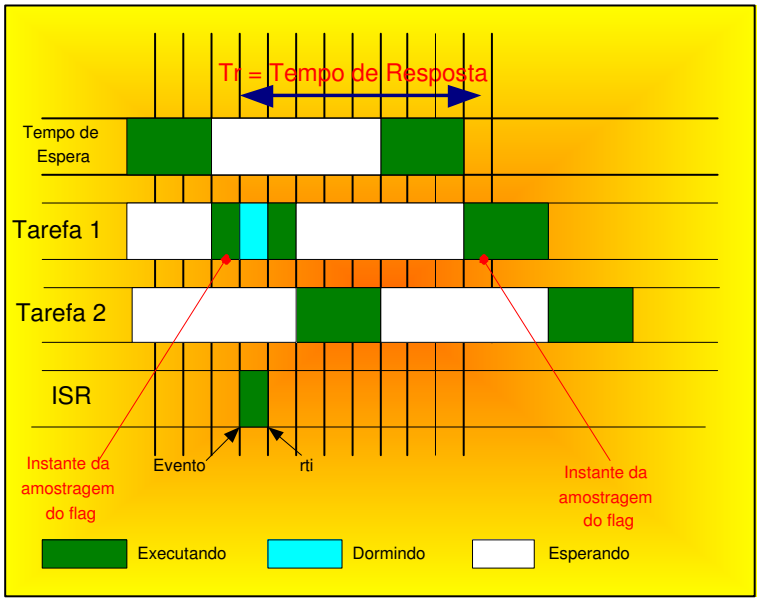


Figura 7.2: Tempo de resposta em um *scheduler de varredura contínua.*

por uma tarefa. A Figura 7.2 mostra como isto acontece num *scheduler* de varredura contínua. O evento é primeiramente detectado pelo circuito de interrupção. Então a rotina do serviço de interrupção seta o flag que é amostrado pela Tarefa 1. Se o flag é setado, a Tarefa 1 responde ao evento. Na figura, o evento ocorre no pior tempo possível, logo após a Tarefa 1 amostrar o flag do evento. Portanto o flag não é detectado até que a Tarefa 1 faça a próxima amostragem na próxima porção de tempo destinada a ela, e o tempo de resposta (Tr) é aproximadamente igual ao tempo de execução de todas as tarefas.

O único modo de melhorar o tempo de resposta com um organizador de tarefa por divisão de tempo é colocar o código de serviço do evento na rotina de interrupção, mas isto faz o tempo de resposta para outras tarefas e interrupções ser mais longo.

A Figura 7.3 mostra o tempo de resposta para um *kernel* preemptivo. Quando o *Interrupt Service Routine* (ISR) é completo, o controle necessariamente não volta a tarefa interrompida, que neste caso foi a Tarefa 2. O ISR libera a Tarefa 1 e como ela tem mais alta prioridade que a Tarefa 2, o *kernel* executa primeiramente a Tarefa 1. Portanto, o tempo de resposta é efetivamente reduzido.

Quando projetamos tarefas para um *kernel* preemptivo devemos ter em mente que as tarefas são executadas assincronamente e que qualquer tarefa pode ter sua

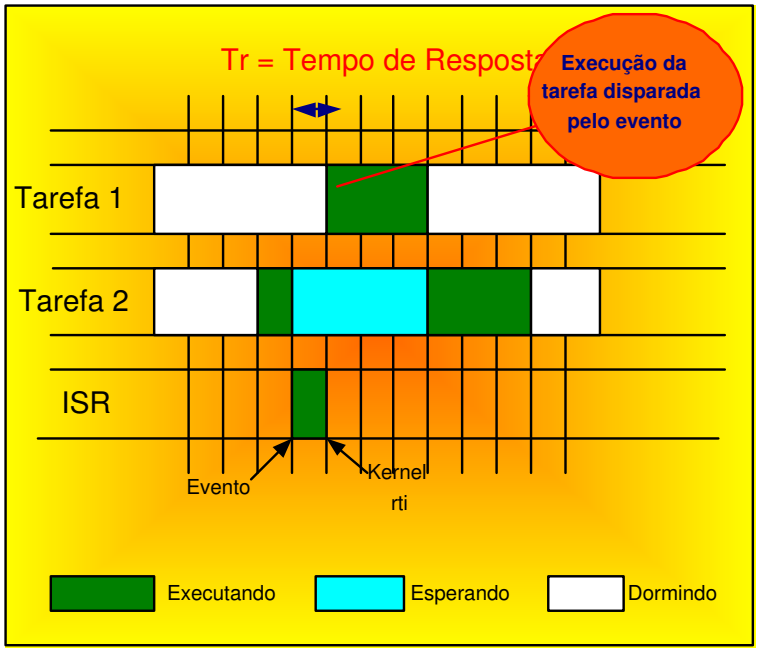


Figura 7.3: Fluxo de um programa multi-tarefa preemptivo.

execução tomada por uma outra de mais alta prioridade. Devido a isso, é necessário aprendermos diferentes técnicas para comunicação entre tarefas e proteção de recursos compartilhados. Um *kernel* deve incluir serviços de comunicação entre tarefas como semáforos (*semaphores*), mensagens (*messages*), e filas (*queues*). Semáforos são usados também como chaves para prover acessos exclusivos a recursos compartilhados. Em nosso estudo de caso são considerados recursos compartilhados o display da IHM, banco de dados e conversor A/D. As tarefas relacionadas a rede *Ethernet* TCP/IP possuem prioridade menor em relação as tarefas envolvidas nos testes, pois é nosso objetivo testar mais placas num menor tempo possível. Pequenos atrasos na transmissão dos dados de testes da placa para o Portal *Web* não são críticas pois temos um banco de dados local no sistema embarcado.

7.3 XML

Em 1969, foi inventada a primeira linguagem de marcação, a *Generalized Markup Language* (GML) e foi usada para dar suportes a aplicativos que processavam documentos. Em 1974 foi criada a Standard GML, e em 1980 ela foi padronizada pela ISO para atender aos requisitos de independência dos sistemas. Nos fim dos

anos 90, uma configuração da SGML, chamada *eXtensible Markup Language* (XML) (GRAVES 2003), foi projetada para fornecer extensibilidade no ambiente *World Wide Web*. A XML estimula a separar o conteúdo, a apresentação e os processos.

A XML é uma linguagem de representação de dados que provê um formato para descrever propriedades dos dados estruturadamente. Em outras palavras, descreve o conteúdo do documento. O XML é um padrão aberto, independente dos dispositivos.

A XML foi projetada para ser distribuída pela *Web*. Os dados são representados com o uso da estrutura de um documento XML. Os elementos e atributos fornecem informações sobre os dados e os estruturam de uma maneira que podem ser transferidos pela *Web*, manipulados por aplicativos diferentes e alterados para atender diversas necessidades.

A Figura 7.4 mostra um exemplo de um arquivo XML. Um documento é composto de tags (palavras encapsuladas por sinais "<" e ">") e dados formado por caracteres (texto não marcado) que são combinados para formar elementos. A primeira linha de nosso exemplo, "<?xml version=1.0?>" é uma instrução de processamento. Em um documento XML existe somente um elemento externo. O elemento externo em nosso exemplo começa com uma tag de abertura "<TestResult>" e termina com uma de fechamento "</TestResult>". Elementos podem conter outros elementos aninhados em seu interior, denominados de sub-elementos.Por exemplo: "<EquipmentID>DIC123T</EquipmentID>". Significa que a identificação do equipamento é DIC123T.

A XML é uma tecnologia aberta e amplamente suportada para troca de dados. Ela permite criar sua própria marcação para quase todos os tipos de informações. Essa extensibilidade permite criar linguagens de marcação inteiramente novas para descrever tipos de dados específicos, incluindo fórmulas matemáticas, estruturas moleculares, etc.

Todos os elementos implementados na Figura 7.4 são padronizados no padrão XML *Business To Manufacturing Markup Language* (B2MML), conforme modelo de dados e atribuições definidos no padrão "ANSI/ISA 95.00.02 Enterprise/Control System Integration".

Figura 7.4: Exemplo de um arquivo em XML.

Os documentos XML são altamente portáveis. Abrir um documento XML não exige nenhum *software* especial, qualquer editor de textos que suporta caracateres ASCII/UNIC
ODE será suficiente. Uma característica importante da XML é que ela é legível tanto por seres humanos como por máquinas.

Processar um documento XML, que geralmente tem a extensão **.xml** , exige um programa de *software* que se chama analisador sintático de XML (ou um processador de XML). Os analisadores verificam a sintaxe de um documento XML e podem suportar o modelo de objeto de documento (DOM) ou a *Simple API for XML* (SAX). Os analisadores baseados em DOM constroem uma estrutura em forma de árvore que contém os dados do documento XML na memória. Isso permite manipular os dados do documento. A árvore DOM tem um único nó raiz que contém todos os os outros nós no documento. Cada nó é um objeto que tem propriedades, métodos e eventos. As propriedades associadas a um nó fornecem acesso ao nome do nó, ao valor, aos nós filhos, etc. Os métodos permitem criar, excluir e acrescentar nós, carregar documentos XML, etc. Os analisadores baseados em SAX processam o documento e geram eventos conforme o analisador encontra marcas, texto e comentários. Esses eventos contêm dados do documento XML. Os programas de *software* podem "ouvir"

esses eventos para recuperar os dados particulares do documento.

Um documento XML pode fazer referência a um documento opcional que define a estrutura do documento XML. Esse documento pode ser uma definição de tipo de documento (DTD) ou um esquema. Se um documento XML seguir fielmente sua DTD ou seu esquema correspondentes, então o documento de XML é válido. Os analisadores que não podem obedecer à conformidade do documento em relação a DTD e ao esquema são analisadores não-validadores. Se o analisador de XML (validador ou não-validador) puder processar com sucesso um documento XML que não tem uma DTD e um esquema , o documento XML é bem formado (isto é, está sintaticamente correto). Por definição, um documento XML válido também é um documento bem-formado.

Os documentos da *Extensible Stylesheet Language* (XSL) especificam como os programas devem exibir os dados de um documento XML. Um subconjunto da XSL, a XSL *Transformations* (XSLT) fornece elementos que definem regras para transformar dados de um documento XML par produzir outro documento XML (por exemplo, XHTML).

7.3.1 Principais benefícios da linguagem XML

- Separação da apresentação dos dados estruturados. Isto simplifica o desenvolvimento e manutenção;
- É usada na *Web* para transferência de dados entre diferentes aplicativos;
- A XML quando utilizada juntos com as *eXtensible Stylesheet Language* (XSL) ou *Cascading Style Sheets* (CSS) disponibilizam mecanismos de fácil visualização em navegadores *Web*. Este mesmo documento pode der manipulado por aplicativos, como por exemplo, em banco de dados;
- Novos padrões com base em XML são úteis em áreas cujas informações tenham uma organização complexa. Padronizam objetos em diversas áreas, como a B2MML;
- Os analisadores XML e outras ferramentas genéricas já estão disponíveis;

- Possuem suporte dos navegadores *Web* que, por já estarem difundidos, reduzem o custo da distribuição;
- Bancos de dados, como Oracle, dão suporte à tecnologia XML;
- Permite que as empresas definam protocolos para a transferência de dados independentemente da plataforma;
- Padrões de compressão do HTTP 1.1 podem ser usados para o XML;
- O XML permite a integração de dados estruturados de diversas fontes, tais como banco de dados.

7.4 Conclusão

A escolha de um sistema embarcado para desenvolvimento de um projeto depende de fatores técnicos, de fatores econômicos e mão de obra especializada que possua conhecimento técnico sobre o sistema embarcado a ser utilizado.

Quando poucas unidades do projeto irão ser produzidas o custo do *hardware* pode ter uma influência menor na decisão do sistema embarcado a ser utilizado. Quando muitas unidades serão produzidas o custo do *hardware* é decisivo, diferenças de centavos de dólares podem representar milhares de dólares no final de uma ano de produção.

As empresas que possuem um orçamento reduzido tem dificuldade em adquirir compiladores que custam milhares de dólares, e normalmente estes compiladores só dão direito a uma instalação ou mesmo adquirir algumas unidades de compiladores que custam algumas centenas de dólares mas que também só dão direito a uma instalação. O uso de sistemas embarcados que utilizam a linguagem Java não apresentam este problema pois as ferramentas de programação podem ser obtidas sem nenhum custo, mas esta linguagem é considerada mais lenta que a linguagem C, inviabilizando seu uso para determinadas aplicações, apesar de outras inúmeras vantagens que Java apresenta.

Ao contrário da linguagem Java, a linguagem C não possui a característica de programação concorrente, necessitando de serviços dos sistemas operacionais em tempo

real (RTOS). Estes sistemas operacionais não possuem mecanismos automáticos de acesso exclusivo a um item protegido, de modo que muito atenção é necessária durante o desenvolvimento dos programas para que os sistemas não fiquem bloqueados.

O uso de ferramentas não proprietárias, como a linguagem XML, junto com o uso de linguagens mais robustas como Java, parecem ser o caminho ideal para desenvolvimento de sistemas embarcados que a cada dia se tornam mais complexos, mas alguns problemas como perda de desempenho e falta de determinismo da linguagem Java ainda devem ser resolvidos.

Capítulo 8

Banco de Dados em Sistemas Embarcados

Segundo (GRAVES 2003) o banco de dados é um conjunto de dados armazenados de maneira que persistam e possam ser manipulados. Por persistência queremos dizer que os dados permanecerão em seus locais depois que o trabalho que os utiliza for encerrado e o computador for desligado. O sistema de banco de dados é composto de um banco de dados e do ambiente ao seu redor, incluindo o *software*, sistema operacional, *hardware* e pessoas que o utilizam.

Segundo (OLSON 2000), muitos sistemas embarcados executam múltiplas tarefas e necessitam algum tipo de sistema operacional para gerenciamento destas tarefas. Sistemas embarcados freqüentemente requerem sofisticados serviços de gerenciamento de banco de dados.

Uma das estratégias chaves em escolher um banco de dados para um sistema embarcado é ter completo conhecimento dos requisitos da aplicação. As características dos bancos de dados variam muito dependendo do fornecedor. Alguns não atenderão todos os requisitos de uma aplicação, outros até mais. A escolha do banco de dados dever ser aquela que atenda, se não a todos os requisitos, pelo menos a maior parte deles.

Depois de escolher o sistema operacional, plataforma de *hardware* e *software* do banco de dados para o sistema embarcado, devemos projetar um sistema que seja de operação confiável, que tenha nenhuma ou pouca intervenção humana. Ao contrário dos computadores *desktops* sistemas embarcados não podem ficar perguntando a um

operador o que fazer quando a aplicação encontrar algum problema.

Técnicas de processo de transação como "*write-ahead logging*" (guarda o bloco do arquivo (por exemplo 1024 bytes) a ser alterado antes de ser gravado, caso ocorra algum problema durante a gravação, fica fácil recuperar o arquivo antes da gravação) e "*two-phase locking*" (bloqueamento dos dados até que todas as alterações sejam finalizadas de modo que outro usuário não pode modificá-los durante este intervalo) são implementadas em muitos sistemas embarcados do que em sistemas de banco de dados relacionais (compostos por dados que se correspondem entre si) em computadores pessoais. Muitos bancos de dados em sistemas embarcados utilizam estas técnicas para tornar os serviços de transação mais robustos. Normalmente, estas características são configuráveis, permitindo os desenvolvedores incluir ou excluir estes serviços. Em nosso estudo de caso, apesar do nosso banco de dados não ter serviços de leitura e escrita concorrentes, implementamos este serviço através de semáforos do sistema operacional em tempo real (RTOS), de modo que nós mesmos desenvolvemos a técnica *two-phase locking*.

Oslon classifica os bancos de dados em três tipos: bancos de dados relacionais cliente-servidor, banco de dados cliente-servidor orientado a objeto e bibliotecas de banco de dados embarcadas.

Alguns fornecedores de banco de dados cliente-servidor relacionais, como Centura Software e Pervasive, oferecem seus bancos de dados empresariais para desenvolvedores de sistemas embarcados. Os bancos de dados cliente-servidor relacionais são bastantes populares, devido ao fato de muitos programadores já conhecerem a *Structured Query Language (SQL)* e o desenvolvimento de banco de dados relacional. A principal desvantagem deste sistema é o tempo gasto de comunicação entre os clientes e servidores e a complexidade adicional na instalação e manutenção de um servidor embarcado.

Outros fornecedores oferecem banco de dados orientado a objetos para sistemas embarcados. Esses produtos parecem ser uma boa escolha para usuários de sistemas embarcados, mas apresentam sérios problemas de gerenciamento de memória e de comunicação, pois foram desenvolvidos para sistemas Unix. Com grande dificuldade estes produtos são adaptados para diferentes arquiteturas de sistemas embarcados.

Banco de dados orientado a objetos são populares devido a integração com as linguagens C++ e Java e por conterem toda a complexidade de desenvolvimento de banco de dados. A grande desvantagem é a carga de comunicação imposta no relacionamento cliente-servidor.

Outros fornecedores de banco de dados fornecem bibliotecas para uso embarcado. Essas bibliotecas são compiladas junto com a aplicação. Eles não requerem a linguagem SQL para manipulação dos dados. Uma das grandes vantagens de usar estas bibliotecas é a rápida execução do código, pois as operações relativas ao banco de dados não necessitam se comunicar com um servidor em separado. Outra vantagem é a alta confiabilidade devido ao fato de poucos componentes serem executados no sistema embarcado. As bibliotecas apresentam a desvantagem de requererem desenvolvimento de interface de comunicação com outros sistemas.

São fatores a ser considerado na escolha de um banco de dados:

- **Suporte a Plataforma**. Banco de dados para sistemas embarcados que operam com Linux ou VxWorks, em plataforma Pentium, podem ser facilmente encontrados, pois as opções disponíveis no mercado são muitas. Desenvolvedores que usam sistemas embarcados menos populares terão poucas possibilidades de escolha. Este fator deve ser fortemente considerado se o sistema a ser desenvolvido for muito dependente de um banco de dados;

- **Recursos Alocados**. Devemos também considerar área de memória a ser usada pelo banco de dados no sistema embarcado. O sistema embarcado suportará um banco de dados com relacionamento cliente-servidor ou melhor será usar as bibliotecas embarcadas que alocam menos recursos de memória;

- **Serviços Requeridos**. O banco de dados será acessado por uma simples tarefa ou por tarefas concorrentes. Tarefas concorrentes necessitam do uso de um gerenciador de tarefas para acesso aos arquivos do banco de dados;

- **Recuperação de Dados**. Alguns bancos de dados fornecem serviços de recuperação dos arquivos após a ocorrência de algum *crash* (falha no sistema);

- **Preço**. Alguns sistemas de banco de dados embarcados são distribuídos sem custo, pois os fornecedores cobram pelos seus serviços, outros cobram pela

venda de licenças e outros sobre os produtos que os desenvolvedores vendem, numa forma de *royalties*.

Após a escolha do banco de dados, os seguintes cuidados devem ser tomados durante o desenvolvimento da aplicação que utilizará o banco de dados embarcado:

- **Representação dos Dados.** Se a aplicação e o banco de dados usam diferentes representações de valores, cada operação de leitura e escrita irá requerer uma translação de dados. Bancos de dados normalmente usam uma estrutura em C que é transladada para os registros dos arquivos. Poucos sistemas de banco de dados, geralmente aqueles baseados em bibliotecas, deixam armazenar os dados no formato nativo do programa, neste caso não há translação dos dados;

- **Chaves**. Os dados devem ser armazenados de acordo com as consultas que devem ser feitas. Deve se considerar o uso de chaves para classificação dos registros. Se mais de uma chave for usada, os bancos de dados normalmente duplicam alguns dados para efetuar uma busca mais rápida, mas usam mais área de memória RAM ou disco magnético para armazenamento;

- **Configuração.** Podemos configurar a maior parte dos bancos de dados para sistemas embarcados de acordo com a nossa necessidade. Essa configuração normalmente inclui memória usada para *caches* secundários, se o dado dever ser escrito no disco ou em memória e controle de acesso dos usuários aos arquivos. Em geral, devemos desabilitar qualquer sub-sistema que não é necessário a aplicação e desse modo, salvar tempo de execução e espaço de memória;

- **Predicabilidade**. Significa que o sistema continua em operação sem nenhuma intervenção manual após um *crash*. Por exemplo, se um sistema de banco de dados entrar num processo de auto-recuperação após um *crash*, a aplicação deve somente noticiar o usuário que um recuperação foi iniciada. O sistema deve executar todas as ações apropriadas para recuperação dos arquivos do banco de dados, sendo tudo transparente ao usuário. O desenvolvedor do sistema deve ser "fanático" em verificar o status de operações como abertura,

leitura, escrita, exclusão e fechamento de arquivos. Caso ocorram erros deve tomar ações apropriadas para resolver estes problemas;

- **Disputa por Dados**. Se muitas tarefas tentam acessar o mesmo objeto do banco de dados ao mesmo tempo, algumas serão forçadas a esperar enquanto as outras completam o processamento. Um meio de melhorar este problema é restringir o acesso somente ao registro que está sendo disputado e não ao arquivo todo.

- **Transferência de dados entre memória e discos magnéticos**. Mover dados entre memória e discos magnéticos causam gargalos de desempenho ao sistema. Uma maneira de melhorar este problema é usar *buffers* de memória RAM para armazenar dados provenientes de escrita e leitura de um disco magnético, isto reduz o número de escritas e leituras ao disco. Se o fator custo permitir usar memória RAM para substituir os discos magnéticos pode ser uma boa opção, principalmente para banco de dados pequenos, mas rotinas de monitoração da tensão da bateria de *back-up* da memória RAM devem ser rotineiramente executadas.

- ***Deadlock.*** Situação quando um primeiro processo quer acessar um recurso de posse de um segundo processo e este, por sua vez, quer acessar o recurso de posse do primeiro processo, mas nenhum pode usar o recurso do outro, pois, para isto, cada um espera que o outro libere o recurso, o que nunca vai ocorrer. Sistemas de banco de dados embarcados possuem mecanismos que verifiquem a existência de *deadlock*. Aplicações com um único usuário ou operações de leitura com vários usuários não apresentam problema de *deadlock*.

8.1 Banco de dados em nosso estudo de caso

Em nossa aplicação, estamos fazendo o uso da biblioteca fornecido pelo próprio fabricante do *hardware* (módulo Rabbit RCM3200). Deve-se mais uma vez lembrar, que não temos suporte há acessos de escrita e leitura concorrentes, sendo necessário uso de ferramentas como o sistema operacional uC/OS para este gerenciamento. Bibliotecas como a da FreeSoftware apresentam o mesmo problema de

falta de gerenciamento. As duas grandes vantagens das bibliotecas são aproveitadas em nossa aplicação: (i) execução rápida, pois não necessitamos de um servidor em separado e (ii) aumento da confiabilidade, pois estas bibliotecas apresentam poucos componentes em sua instalação.

A seguir relacionamos os principais pontos na implantação do nosso banco de dados. São eles:

- **Desempenho.** Como estamos usando somente memória RAM não volátil para armazenamento dos dados, não se têm problemas relacionados à transferência de dados para discos magnéticos. Cuidados com o uso de memória RAM não volátil são verificar constantemente o nível de tensão de bateria de *backup* e rotinas de testes de escrita e leitura para verificação do estado da memória;

- **Suporte a plataforma.** Exemplo de criação de arquivos, leitura e escritas nos mesmos são bem documentadas pelo desenvolvedor do ambiente de programação Dynamic C usado no nosso caso de uso. O banco de dados fornece aproximadamente 120 erros que possam ocorrer durante a execução das funções relacionadas com o banco de dados;

- **Alocação de recursos.** O uso de bibliotecas embarcadas utiliza uma quantidade pequena de memória de programa, aproximadamente 24Kbytes;

- **Estimação do desempenho.** Escritas e leituras de registros nos arquivos do banco de dados são feitos em menos de $15\,ms$. Como cada ciclo de teste dos nossos produtos tem em média aproximadamente $20\,s$ de duração, isto significa, que em situações normais de operação, escritas e leituras só ocorrerão a cada $20\,s$ no banco de dados. Este não é um aspecto crítico no nosso sistema;

- **Serviços requeridos.** Sincronização na leitura e escrita dos dados deve ser fortemente considerada. A leitura de um registro não pode ser interrompida por uma escrita, assim como, a escrita em um registro não pode ser interrompida por uma leitura. Caso isto ocorra o ponteiro do arquivo será corrompido, causando um *crash* no arquivo. A concorrência das tarefas deve ser considerada. Por exemplo, a tarefa de testes estando em dormência, e neste intervalo o

sistema tenta ler um registro do arquivo de testes, mas a tarefa de testes volta a ser executada pelo sistema operacional e tenta escrever um novo registro no arquivo de testes, ocorrerá um *crash* no ponteiro do arquivo. A solução é usar um semáforo que indica recurso compartilhado. Se a tarefa que envia os registros dos arquivos de teste para o Portal *Web* (aonde ocorre a leitura no arquivo) for executada e no meio da leitura do arquivo a tarefa de testes tomar o controle do gerenciador de tarefa de volta e quiser escrever no arquivo, encontrará o arquivo bloqueado. A tarefa de testes encontra o arquivo bloqueado e é mais prioritária que a tarefa de enviar o arquivo XML que pode desbloquear o arquivo. A solução encontrada é que a tarefa de testes ao verificar que o arquivo está bloqueado execute um comando para entrar em estado de dormência, neste intervalo a tarefa que envia os registros de testes para o Portal *Web* tem tempo suficiente para finalizar a leitura do registro e liberar o recurso bloqueado. A tarefa de testes voltando a ser executada encontra o recurso compartilhado desbloqueado e pode assim escrever os resultados dos testes no arquivo;

- **Preço.** Em nosso caso, o banco de dados já faz parte do pacote de desenvolvimento do módulo microcontrolado Rabbit RCM3200.

- **Uso de chaves.** Não há uso de chaves. Nosso banco de dados é não relacional. Os dados são acessados na forma *"Fist in, First out"*. Significa que o primeiro registro de teste a ser gravado é o primeiro a ser enviado para o portal *Web*.

- **Configuração.** O tamanho da memória RAM foi um dos requisitos mais importantes para escolha de nosso sistema embarcado, justamente para atender as necessidades de nosso banco de dados.

- **Prognosticar falhas.** Nosso banco de dados fornece status de todas as operações de abertura, leitura, escrita, exclusão e fechamento de arquivos, entre outras.

- **Segurança.** O aspecto mais importante no nosso projeto é a confiabilidade das informações. Caso a memória de registros analíticos atinja ocupação de 100 %, as informações passarão a ser gravadas sinteticamente. Infelizmente, as medidas dos testes são perdidas, mas a totalização de produção e os tipos de

erros ocorridos nos displays sob testes são mantidos. A cada 0,6 *s* o gerenciador de tarefas verifica se há algum registro no arquivo analítico ou sintético a ser enviado para o Portal *Web*. Caso haja algum registro a ser enviado o DIC abre uma conexão com o Portal *Web* e envia o registro. Somente após receber a confirmação do Portal que o arquivo foi recebido com sucesso é que o DIC exclui do arquivo o registro enviado.

8.2 Conclusão

O uso do banco de dados fornecido pela ferramenta de desenvolvimento de nosso sistema embarcado acrescido do uso do sistema operacional em tempo real uC/OS de Jean Labrosse para sincronização dos acessos aos registros dos arquivos do banco de dados e armazenamento de dados em memória RAM não volátil satisfazem os requisitos de nossa aplicação. Mas a recuperação de arquivos proveniente de *crash* é uma importante função a ser implementada e deve ser objeto de estudo. O uso de uma memória *Flash* serial deve também ser considerada para aumentar a capacidade de armazenamento das informações na eventual impossibilidade de transmitir os dados para o Portal *Web*.

Capítulo 9

Estudo de caso - desenvolvimento de um Dispositivo Inteligente de Campo para a Philips MDS

A Empresa Philips MDS produz *displays* de celulares no Distrito Industrial da Zona Franca de Manaus apresentados na Figura 9.1. Outras empresas utilizarão estes *displays* na montagem de aparelhos celulares.

Figura 9.1: *Displays* produzidos pela Philips.

A seqüência atual do processo produtivo (montagem e inspeção) pode ser visualizada pelo esquema da Figura 9.2. No momento, o sistema de testes opera de forma

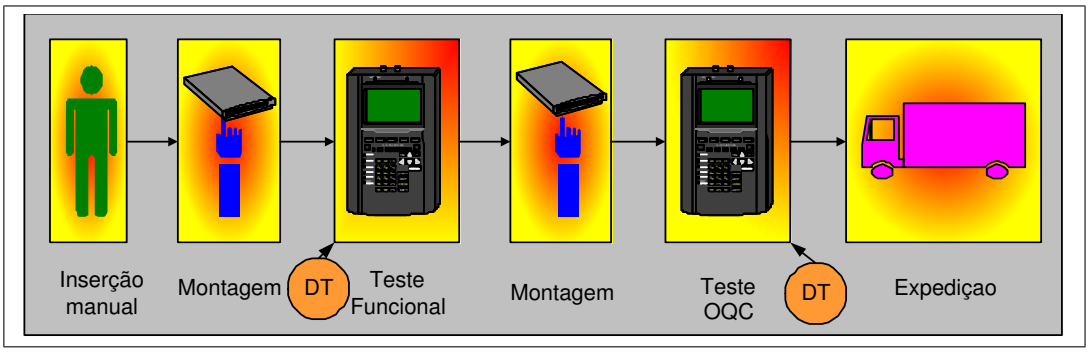


Figura 9.2: Processo de montagem e inspeção.

autônoma e *off-line*, necessitando de passos manuais para a totalização de defeitos. A geração de relatórios ocorre com um dia de atraso em relação aos testes.

Deseja-se ligar os Dispositivos de Teste (DTs) em rede para permitir que a totalização seja feita de forma a reduzir os erros de operação e possibilitar ações imediatas, graças à disponibilização *on-line* dos dados dos testes para fins de análise e estatística do processo. A introdução de novas funcionalidades não deverá aumentar o tempo de teste dos *displays*, o que acarretaria diminuição da produção.

Os DTs atuais por não apresentarem possibilidade de conexão em rede serão substituídos por novos DTs com interface de rede. Um banco de dados nos DTs será desenvolvido para não haver perda dos dados de produção na eventual indisponibilidade da rede. A arquitetura a ser utilizada será a de Integração *Web* com uso de rede *Ethernet*, protocolos *Transmission Control Protocol / Internet Protocol* (TCP/IP), *Hypertext Transfer Protocol* (HTTP) e *File Transfer Protocol* (FTP). Os resultados dos testes e a configuração dos DTs serão enviados para os computadores de controle de produção e armazenamento de dados (Portal *Web*) através de arquivos XML utilizando o protocolo *Hypertext Transfer Protocol* (HTTP) como meio de transporte. A rede *Internet* além de atender os requisitos do projeto, já está implantada no nível corporativo e facilitará o processo de integração com outras unidades do grupo, clientes e fornecedores se assim desejado.

No mundo do controle de processo, quando falamos em eletrônica embarcada , ou seja, com plataforma de *hardware* microcontrolada (capítulo 6), sistema operacional (capítulo 7), comunicação em rede (capítulo 2), protocolos de comunicação

(capítulos 3 e 4), capacidade de armazenar dados (capítulo 8) e fornecimento de diagnósticos, faz mais sentido referirmos aos DTs como Dispositivos Inteligentes de Campo (DICs).

O DIC enviará informações tanto para o Sistema de Execução de Manufatura (MES) quanto para a *Internet*. A Engenharia de Projetos da Philips MDS, através do uso de navegadores *Web*, poderá visualizar parâmetros elétricos do *display* sob teste, diretamente do DIC (*Internet* ou *Intranet*) ou indiretamente através do MES. Engenheiros de manutenção serão avisados quando da necessidade de manutenção pela implantação de funções de diagnose nos DICs. A Engenharia de Produção, através do MES, com informações provenientes dos DICs, acompanhará a produção *on-line* e assim poderá tomar medidas para melhorar a eficiência da produção. Operadores da linha de produção não necessitarão utilizar planilhas de papel para registrar defeitos de aparência nos *displays*, basta digitar no DIC o tipo de defeito ocorrido.

Neste capítulo é descrito passo a passo o desenvolvimento de um dispositivo inteligente de campo (DIC) para aquisição de dados em chão de fábrica.

As seguintes etapas serão desenvolvidas:

- Levantamento de requisitos dos usuários;
- Especificação de requisitos;
- Definição da arquitetura do sistema;
- Escolha da CPU;
- Escolha da linguagem de programação;
- Implementação das camadas de acordo com o modelo de referencia *Internet*;
- Projeto do *hardware* do DIC;
- Etapas no desenvolvimento do *software*:
  - Diagramas de caso de uso;
  - Diagramas de seqüência;

  - Máquina de estado;

- Integração do Sistema e Testes;
- Avaliação dos benefícios alcançados oriundos das informações dos DICs.

9.1 Levantamento dos requisitos dos usuários

A primeira etapa do projeto consistiu em obter um detalhado entendimento do processo. Este entendimento consiste em uma organização sistemática de dados coletados em entrevistas feitas com os futuros usuários do DIC com o objetivo de definir o que eles realmente almejam.

Estas entrevistas definem uma sistemática de aproximação, organização e documentação dos requisitos do sistema, estabelecendo e mantendo uma concordância entre os usuários e a equipe de desenvolvimento do projeto na modificação, inclusão e exclusão de requisitos do sistema.

Após a revisão e aceitação do documento dos requisitos do usuários, o analista escreve a primeira versão da especificação de requisitos.

Entrevista efetuada com a Engenharia de Projetos e Engenharia de Produção

A Empresa Philips MDS produz *displays* de celulares (Figura 9.1) no Distrito Industrial da Zona Franca de Manaus. A seqüência atual do processo produtivo pode ser visualizada pelo seguinte esquema, conforme a Figura 9.2.

Na primeira etapa é feita a montagem manual de determinada parte do produto, pois caso ocorram defeitos nesta fase, parte do que foi montado pode ser reaproveitado. Após o teste funcional desta primeira etapa, executado pelos DTs, novos acessórios são adicionados ao produto, normalmente em relação à aparência final do mesmo. Em seguida uma nova Operação de Controle de Qualidade (OCQ) será realizada pelos DTs e finalmente o produto será embalado para expedição.

De acordo com a Engenharia de Projetos e Engenharia de Produção, o dispositivo de testes (DT) deverá ter as seguintes funcionalidades:

- Verificar os parâmetros do *driver* controlador do *display*, tais como correntes e tensões;
- Auxiliar o operador a escolher o melhor ajuste de contraste para o *display* através de uma tela (imagem) padrão;
- Auxiliar o operador a verificar o perfeito funcionamento dos *pixels* do *display* mostrando seqüencialmente várias telas pré-programadas;
- Programar o registrador do *driver* do *display* responsável pelo contraste do *display*. Este registrador encontra-se em uma memória *Flash* ou *One Time Programmable* (OTP). O valor de programação do registrador é variável, pois vários *drivers* de *display* mesmo recebendo valores iguais no registrador de contraste podem gerar tensões diferentes para o *display* de cristal líquido, portanto diferentes contrastes. Um dos pontos principais do sistema é verificar se realmente o valor gravado foi o desejado. Esta verificação é feita indiretamente, ou seja, pela tensão gerada pelo *driver* do *display*;

São diretivas dos DTs:

1. O DT deverá ser de tamanho pequeno. O controle não poderá ser feito utilizando-se um microcomputador PC, pois o mesmo não caberá nas mesas da linha de produção.
2. O acabamento não pode conter detalhes cromados para não refletir luz nos olhos do operador.
3. Os DTs deverão ser interligados em rede para permitir a totalização *on-line* dos tipos de defeitos, armazenamento das medidas elétricas, controle da produção e todos os outros benefícios provenientes de uma automação.
4. Como são testados vários tipos de *displays*, é necessário, quando da troca de linha de produção, que o DT detecte a troca do módulo eletro-mecânico de testes. Desse modo, consegue-se agilizar o processo de configuração, pois a transferência do programa para o DT será feito via rede, mas nada impede que esta transferência seja feita localmente. Atualmente, troca-se a memória *Flash* do equipamento quando o DT é transferido para outra linha de produção.

5. Hoje, quando se deseja modificar o intervalo de medida de um parâmetro é necessário regravar a *Flash* do dispositivo de teste. Um dos benefícios provenientes do uso de uma rede é poder criar no banco de dados uma tabela de parâmetros que poderá ser transferida para os DTs de forma automática. As tabela de parâmetros serão documentadas por sua versão. Qualquer modificação nos parâmetros, por menor que seja, ocasionará alteração na versão da tabela de parâmetros.

6. Em cada conjunto de testes de um *display*, deverá ser enviado ao banco de dados Oracle (imposição Philips MDS) um relatório com o resultado dos testes, a versão da tabela de parâmetro em uso, o status do DT e outras informações a serem definidas no decorrer do projeto. Estes dados servirão para a geração de relatórios, histogramas, etc.

7. Mesmo não havendo testes, o status do DT deverá ser enviado ao portal *Web* em intervalos de tempo regulares pré-programados. Este mesmo intervalo pré-programado servirá de referência para o controle da produção. Por exemplo, se um ciclo médio de produção for de $20\,s$, mas ao atingirmos $40\,s$ e nenhum *display* for colocado no DT para teste acontecerá uma "parada automática" de produção. Dados como tempo de teste de cada *display* de celular e o tempo total em que ele permanece no DT serão utilizados para avaliar a eficiência da produção.

8. O DT deverá fornecer os resultados dos testes para um *Data Logger*, ou outro meio de visualização, via rede.

9. Cada DT deverá ter uma tabela interna que reflita o seu perfil: número de série, identificação na rede, versão do *software*, versão do *hardware*, módulo mecânico acoplado, versão da mecânica, versão de parâmetros, linha e posto de produção.

10. Durante a configuração do DT, a engenharia de produção poderá introduzir via IHM ou rede dados como linha, posto, etc.

11. No DT atual existe somente um potenciômetro para controle do contraste do *display*, um LED para indicar aprovação ou não do *display*, uma chave

push-button para ativar o início dos testes e outra para a parada dos testes. O novo DT deverá ter , além do que tem no atual, um visor para que haja interação com o usuário, um teclado numérico para digitação de códigos de erros utilizados atualmente e outras teclas que serão definidas no decorrer do projeto da IHM.

12. Ao ser ligado o DT deverá mostrar uma mensagem que ele está em condições de funcionamento e em seguida solicitar o posicionamento do *display* para a realização dos testes.

13. O DT deverá ter dois modos de operação: um para produção e outro para engenharia.

14. No modo de engenharia ou depuração os resultados dos testes não são gravados no banco de dados local e nem enviados ao banco de dados Oracle. O DT inicia o primeiro teste após reconhecer a colocação do *display*. Aparecerá no *display* da IHM de forma contínua o número do teste e o valor da medida até ser pressionada a tecla <ENTRA> para passar ao teste seguinte. Se for pressionada a tecla <CANCELA> os demais testes não serão realizados e o *display* deverá ser retirado do módulo eletro-mecânico.

15. No modo produção ou automático não é necessário teclar <ENTRA> para passar de um teste para outro. As medidas não são mostradas no *display*. Haverá sempre a gravação dos resultados dos testes no banco de dados, independentemente dos testes serem aprovados ou da ocorrência de falha elétrica no *driver* ou problema de aparência detectado pelo operador.

16. Para iniciar os testes basta o operador posicionar o *display* no módulo eletro-mecânico do DT de forma correta. Quando o DT reconhecer que o *display* está posicionado começam os testes e aparece no *display* a mensagem "Testando...".

17. O operador detectando qualquer problema de aparência teclará <CANCELA> e em seguida digitará o código do erro. O DT fará a consistência do código digitado pelo operador com uma tabela de erros para verificar se realmente o código de erro digitado existe. Esta tabela de erros, assim como a tabela de parâmetros, poderá ser transferida automaticamente, via rede, para os DTs.

18. Após a realização dos primeiros testes e não sendo detectado nenhum defeito elétrico ou anormalidade, o DT enviará uma tela para o *display* que visualmente facilita o ajuste de contraste do *display*. O DT calcula o melhor valor de contraste em função dos últimos valores programados nos testes anteriores. Este valor é enviado para o registrador de contraste do *driver* do *display*. O operador verificando que este valor é aceitável, ou seja, o contraste do *display* está bom, tecla <ENTRA> para prosseguimento dos testes. Porém, caso seja necessário, o operador poderá usar as teclas de incremento e decremento para selecionar o melhor contraste do *display*.

19. Ao finalizar o ciclo de testes, o DT deverá informar o tempo de duração do testes, caso o *display* tenha sido aprovado, ou então, exibir o número do teste e o respectivo valor da medida em que o *display* foi reprovado.

20. O ciclo de produção é finalizado quando o operador coloca um novo *display* para teste ou quando ocorre uma parada automática. Inicia-se, assim, um novo ciclo de produção.

21. Se for necessário fazer um novo teste em um *display* que já havia sido testado e o resultado gravado no banco de dados Oracle, o operador deverá teclar <RETESTE> antes ou durante o novo teste para avisar ao banco de dados que se trata de um re-teste. Caso contrário, o banco de dados interpretará que dois *displays* foram testados e não um, ocasionando um erro no número de *displays* produzidos.

22. O DT deverá ter um banco de dados local com dois arquivos. Um arquivo analítico e um sintético. O arquivo analítico possui registros que incluem os resultados de cada ciclo de teste. Os registros são formados por campos que abrangem o número do teste, valores das medidas e código de status do resultado do teste. Por exemplo, teste de número “200”, com valor de tensão de “8,324V” e aprovado, será armazenado na forma “200”, ”8,324V” e “0.0”. “0.0” indica aprovado, “0.1” indica reprovado por erro de medida limite inferior, “0.2” indica reprovado por erro de medida limite superior, “20.01” indica erro de aparência por formação de bolha no *display* e assim por diante. Caso a rede esteja fora de operação, haverá um momento, estipulado pela Philips

MDS em 2 horas, em que não mais será possível armazenar os resultados no arquivo analítico, pois 100% da capacidade deste arquivo será atingida. Então, passam a ser armazenados no arquivo sintético somente o tempo inicial e final de produção, número de ciclo de testes e o número de vezes de ocorrência de determinado erro. As medidas de cada teste não são mais armazenadas. Neste tempo espera-se solucionar o problema na rede de dados que estiver impedindo a transferência destas informações para o banco de dados Oracle.

Entrevista realizada com a Engenharia de Manutenção

Os DTs além de fornecerem dados de produção e medidas elétricas dos *displays* sob teste para o Sistema de Operações de Manufatura (*Manufacturing Execution Systems (MES))*, também deverão enviar pela rede informações sobre seu estado de funcionamento, tais como: tensão das fontes de alimentação, *off-set* dos circuitos de medição de corrente e tensão e tempo de abertura da válvula do módulo eletromecânico. Estas mesmas informações podem ser enviadas para um *data logger* via interface serial RS-232. Caso uma dessas informações esteja fora de certos limites especificados, deverá ser enviado um aviso de necessidade de manutenção. Há dois tipos de manutenção: a proativa indica que os valores medidos estão dentro de limites considerados ainda pouco críticos, possibilitando a Engenharia de Manutenção planejar futuras paradas sem interromper o processo produtivo; a manutenção corretiva indica que os valores das medidas estão fora de limites especificados e o dispositivo deve ser retirado da linha de produção, neste caso, há uma parada do processo produtivo.

Entrevista realizada com os Operadores de Produção

1. O DT deve ter uma interface com teclado agradável, pois são realizados aproximadamente 1000 testes em cada turno. Os operadores fazem muitos movimentos repetitivos. O teclado de membrana causa dores nas pontas dos dedos. Um teclado parecido com os utilizados pelos caixas do sistema bancário seria o ideal.

2. A inserção do *display* de celular a ser testado no DT deve ser feita de forma rápida.

3. O DT deve ter cantos arredondados para evitar acidentes de trabalho.

4. O DT deverá ter um LED vermelho para indicar que o *display* foi reprovado e um LED verde para indicar que o *display* foi aprovado. É melhor visualizar um LED do que ler um *display* de cristal líquido na IHM.

5. Na ocorrência de qualquer problema de aparência no *display* sob teste, o código do problema a ser digitado no DT deve ser o mesmo da tabela usada no processo manual.

Após as entrevistas realizadas com as engenharia de projetos, produção, manutenção e operadores, foi proposta uma IHM com as seguintes características iniciais:

- Um *display* de LCD de 2 linhas × 16 colunas;
- Uma chave para ligar e desligar o DT;
- Um teclado com as seguintes teclas: 0 a 9, incremento, decremento, cancela, entra, menu, para baixo, para cima, para esquerda, para a direita e re-teste. Algumas dessas teclas podem ter funções alternativas.

Observação sobre o funcionamento de algumas teclas:

- **<CANCELA>.** Quando teclada durante um teste indica que o operador digitará em seguida um código de erro;
- **<RE-TESTE>.** Indica que o *display* está sendo testado novamente. O ideal é que cada *display* tivesse um código de barras para sua identificação. Esta etapa chamada de rastreamento, será executada em um futuro próximo, pois envolve outros problemas que não estão relacionados ao DT;
- **<Incremento> e <decremento>.** Serão usadas para ajuste do contraste do *display*.

9.2 Especificação de requisitos

Na maioria das vezes, não é fácil prever as dificuldades de implementação de um sistema, principalmente quando se deve integrar várias áreas e tecnologias. O desenvolvimento de qualquer sistema técnico complexo requer especificações bem definidas. As especificações apresentam uma compreensão clara do comportamento de todo o sistema. Assim, os requisitos do sistema podem ser verificados e erros de projeto podem ser detectados e eliminados ainda em um estágio inicial.

A especificação de requisitos é um documento que deve ser aprovado tanto pelos usuários do sistema quanto pelos analistas, arquitetos, desenvolvedores e testadores do sistema. Toda a equipe de desenvolvimento é responsável pela definição e revisão de todos os requisitos.

Segue abaixo, a especificação de requisitos para o projeto.

9.2.1 Abreviações mais utilizadas

DT	Dispositivo de Teste.
HTTP	*HyperText Transfer Protocol*. Refere-se ao protocolo que possibilita o serviço de *Web* da *Internet/Intranet*.
IHM	Interface Homem-Máquina.
NUTELI	Núcleo de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e de Informação da Universidade Federal do Amazonas.
MES	*Manufacturing Execution Systems*. Sistema de Operações de Manufatura.
Stack TCP/IP	Refere-se ao conjunto de protocolos de acesso, configuração e administração da rede local.
STU	Sistema de Testes Unificado. Refere-se a segunda versão do sistema de testes desenvolvido pelo NUTELI para a Philips-MDS.
UFAM	Universidade Federal do Amazonas.

9.2.2 Especificações

A seqüência atual dos testes de *displays* na fábrica da Philips MDS pode ser visualizada pelo seguinte esquema, conforme Figura 9.3.

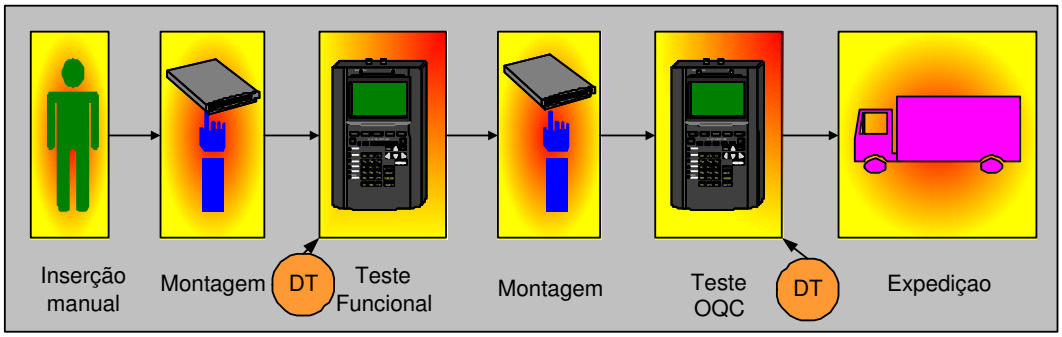


Figura 9.3: Processo de inspeção e montagem.

No momento, o sistema de teste opera de forma autônoma e *off-line*, necessitando de passos manuais para a totalização de defeitos. A geração destes relatórios já nasce com um dia de atraso. Deseja-se ligar os DTs em rede, permitindo que a totalização seja feita de forma a minimizar os erros de operação e possibilitar ações imediatas, graças a disponibilização *on-line* dos dados dos testes para fins de análise e estatística do processo. Esse desejo foi transformado, neste documento, em especificações.

As especificações aqui descritas foram feitas com base em visitas à fábrica e reuniões com as engenharias e operadores da linha de produção da Philips-MDS em Manaus, além de uma análise do ambiente operacional da fábrica.

Requisitos obrigatórios

Os novo DTs devem possuir os seguintes requisitos obrigatórios:

- Interface para rede;
- Capacidade para armazenar o resultado do testes localmente em um banco de dados;
- Capacidade de transferir os resultados dos testes para um MES;

- Capacidade de armazenamento de páginas *Web*, para acesso de qualquer ponto da *Internet*;
- Interface serial para um *data-logger*;
- Capacidade de gerar sinalizadores de condições críticas.

Requisito opcional

- O DT deve possuir um *software* de predição para ajuste de contraste do *display*, ou seja, com base nos últimos valores de calibração do contraste do *display* o DT deverá calcular um valor inicial ótimo para esta operação.

Condições de operação

- Rede de comunicação padrão IEEE 802.5;
- Rede de alimentação de energia elétrica separada de outros equipamentos industriais alimentada por *no-breaks*;
- Temperatura de operação entre 25°C e 70°C;
- Humidade entre 5% e 95%.

9.2.3 Ambiente

Restrições de *hardware*

- A infraestrutura da rede já deverá estar instalada para aplicação deste sistema de aquisição de dados.

Restrições de *software*

- Navegador *Web* com protocolo HTTP versão 1.0. Se no cliente, o navegador *Web* utilizar o protocolo HTTP com versão superior a 1.0 algumas requisições podem não ser entendida pelo servidor HTTP 1.0. Maiores detalhes em http://www.w3.org/Protoc
  ols/rfc2616/rfc2616_sec3.htmlsec#3.1;

Restrições de interfaces

- Haverá um responsável pelas atualizações do perfil de cada DT.

9.2.4 Requisitos funcionais

Os seguintes requisitos funcionais do sistema, agrupados por classe, são listados a seguir:

- **Requisitos de confiabilidade:**
  - **/RF10/** Perda zero dos dados de testes até o limite de tempo de duas horas;
  - **/RF11/** Capacidade de armazenamento analítico dos relatórios de teste em não menos que duas horas. Após atingir esta capacidade de memória os dados deverão ser armazenados sinteticamente.
- **Requisitos de segurança:**
  - **/RF20/** Necessidade de digitação do nome do usuário e senha para acesso às páginas *Web* dos DTs;
  - **/RF21/** Isolamento da rede de DTs da rede corporativa;
- **Requisitos dos dados operados pelo DTs:**
  - **/RF30/** Os DTs enviarão os resultados dos testes a partir de um protocolo a ser especificado;
  - **/RF31/** Os parâmetros dos testes poderão ser atualizados via rede;
  - **/RF32/** Reconhecer automaticamente o módulo eletromecânico acoplados ao DT específico a cada *display*;
  - **/RF33/** Na troca de módulo eletromecânico o DT deverá requerer da rede o download do programa de teste específico para o *display* a ser testado;
  - **/RF35/** Em cada relatório de teste ou em um tempo pré-programado, o DT enviará seu status de operação: se "Produzindo", em "Parada Automática", "Parada Manual" ou em "Falha".

9.2.5 Requisitos da IHM

- **/RIHM10** Teclado tipo bancário;
- **/RIHM11** *Display* de cristal líquido de 2 linhas × 16 colunas;
- **/RIHM12** Teclas de 0 a 9;
- **/RIHM13** Tecla de ENTRA;
- **/RIHM14** Tecla de CANCELA;
- **/RIHM15** Tecla de INCREMENTO;
- **/RIHM16** Tecla de DECREMENTO;
- **/RIHM17** Tecla de RE-TESTE;
- **/RIHM18** Tecla de LIMPA;
- **/RIHM19** Teclas de NAVEGAÇÃO (Esquerda, Direita, Cima e Baixo);
- **/RIHM20** Chave de Liga/Desliga.
- **/RIHM21** LED verde, indicativo de ciclo de teste "OK";
- **/RIHM22** LED vermelho, indicativo de ciclo de teste "Não OK";

9.2.6 Requisitos de qualidade

Os requisitos de qualidade do DIC estão relacionados na Tabela 9.1.

Significado dos requisitos de qualidade:

- Funcionalidade. Uso cômodo e prático;
- Confiabilidade. Capacidade de uma unidade funcional desempenhar, sem falhas ou avarias, dada tarefa sob certas condições e dentro de um período determinado;
- Interconectividade. Capacidade de conexão de plataformas de *hardware* distintas e de diferentes portes através de uma rede padrão;

- Portabilidade. Capacidade de implementação da mesma funcionalidade em diferentes plataformas de *hardware* e *software*;
- Modularidade.Capacidade de inclusão, eliminação e alteração de funções, módulos e mesmo novos centros com impacto mínimo sobre os demais componentes do sistema;
- Expansibilidade. Capacidade de crescimento incremental de *hardware* (adição / substituição) e de *software* (adição de novas funcionalidades).

Tabela 9.1: Requisitos de qualidade.

Item	Requisitos	Alto	Normal	Baixo	Irrelevante
RQ1	Funcionalidade	X			
RQ2	Confiabilidade	X			
RQ3	Interconectividade		X		
RQ4	Portabilidade				X
RQ4	Modularidade			X	
RQ4	Expansibilidade		X		
RQ5	Segurança		X		

9.2.7 Ambiente de testes

- **Laboratório**
  - O DT será desenvolvido inicialmente nos laboratórios do NUTELI, UFAM.
- **Unidade fabril**
  - Após os testes de laboratório o produto deve passar pelo processo de homologação a ser descrito pela Philips-MDS.

9.2.8 Ambiente de desenvolvimento

- ***Software***
  - Compilador C Dynamic C Premier 7.32P;
  - Sistema operacional RTOS uC/OS-II de Jean Labrosse.
- ***Hardware***
  - Módulo microcontrolado RCM3200 da *Rabbit Semiconductors*.

9.3 Arquitetura do sistema

As soluções de arquiteturas disponíveis foram examinadas no capítulo 3. Duas arquiteturas são candidatas ao desenvolvimento do sistema:

1. Arquitetura FCS com uso de barramentos de campo proprietários (Figura 9.4).

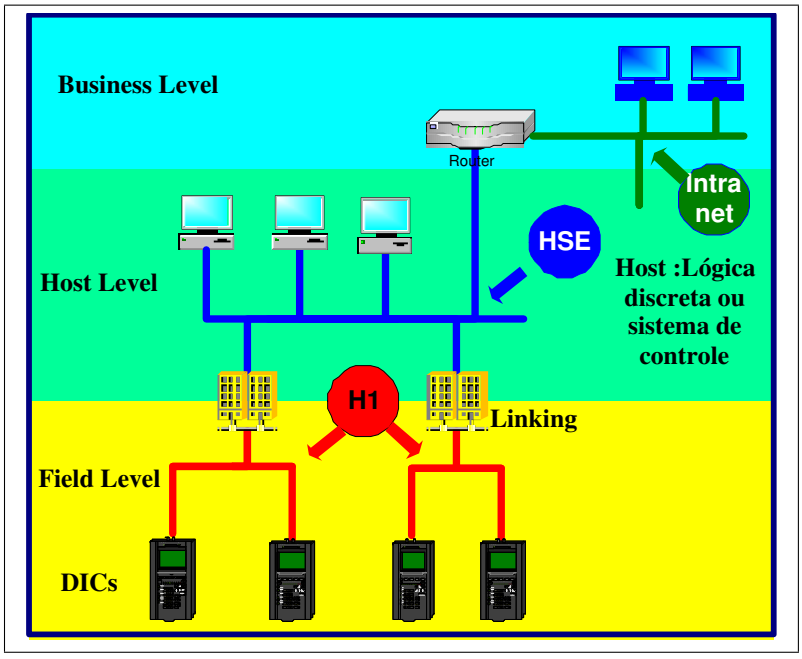


Figura 9.4: Arquitetura FCS.

2. Arquitetura de Integração *Web* (Figura 9.5).

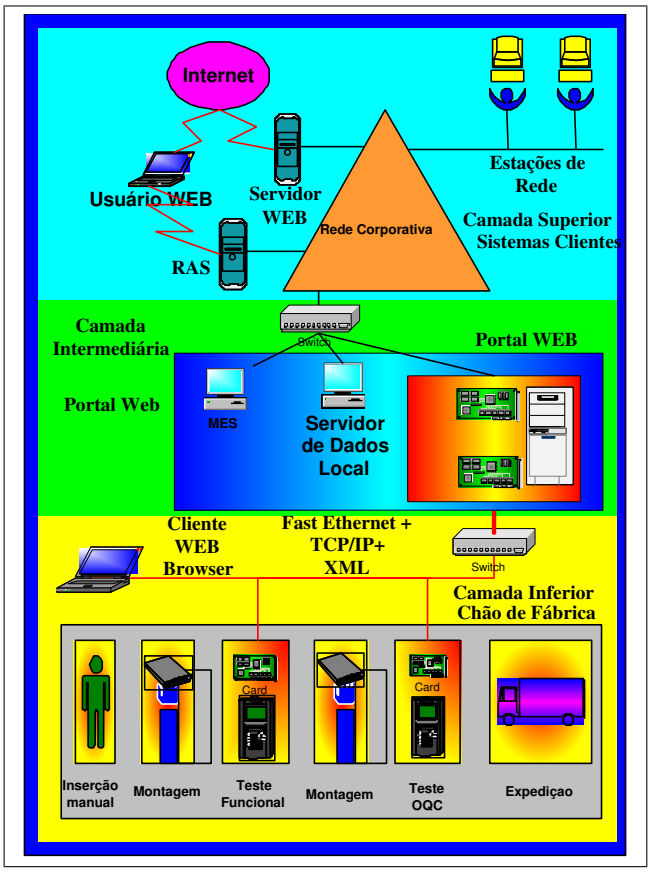


Figura 9.5: Arquitetura de integração WEB proposta.

Como não há necessidade de determinismo na transmissão dos dados na rede e como no chão de fábrica da Philips não existem sistemas implementados com Arquitetura FCS com uso de protocolos proprietários (capítulo 4), não faz sentido a utilização da mesma, pois esta originará dependência de produtos por parte de fornecedores.

Por outro lado, além de atender as especificações do projeto e não provocar dependência por parte de fornecedores, a arquitetura de integração *Web* já está presente no nível corporativo.

A Arquitetura de Integração além de atender a taxa de transferência dos dados do chão de fábrica para o portal *Web*, possui outros benefícios como:

- Independência de fornecedores, pois é uma arquitetura aberta;
- Curva de aprendizado com tempo menor, pois os usuários já possuem algum

conhecimento de alguns ítens da arquitetura, usados no dia-a-dia, especialmente dos navegadores *Web*;

- Diminuição dos custos de integração e manutenção;
- Facilidade de expansão do sistema.

9.4 Escolha da CPU

Plataformas de desenvolvimento mais simples, ou seja, aquelas que possuem memória de programa *Flash* com capacidade menor que 16Kbytes ou 32Kbytes e memória de armazenamento de dados menor que 0,5Kbytes têm inúmeras soluções encontradas no mercado: incontáveis fornecedores de microcontroladores baseado na famosa linha Intel 8051, o famoso 68HC08 da Motorola, os versáteis PIC fornecidos pela Microchip e assim por diante.

Sistemas mais complexos necessitam de memória de armazenamento de programas e dados na ordem de centenas de Kbytes ou mais. Estes sistemas mais complexos podem fazer bom uso de computadores pessoais (PC) ou sistemas embarcados com maior capacidade de memória e processamento.

Não é considerado o uso de PC industriais devido à restrição imposta pela Philips. O tamanho considerável dos PCs causaria um transtorno muito grande na mudança de *lay-out* das linhas de produção, pois as mesas dos operadores são muito pequenas. Além do que eles apresentam custo em pelo menos uma ordem de grandeza mais elevada do que os sistemas embarcados.

Devido ao menor custo que os PC industriais, os sistemas embarcados baseados em microcontroladores de 16 *bits*, 32 *bits* ou microcontroladores de 8 *bits* com endereçamento de memória superior a 64Kbytes são uma boa solução para sistemas mais complexos desde que satisfaçam as restrições e requisitos do sistema.

São restrições impostas pela Philips:

- **Tamanho.** Por apresentarem tamanho bastante reduzido, não implicarão em mudanças de *lay-out* das linhas de produção;

- **Tempo de Teste.** Com a introdução de novas funcionalidades nos DTs, como interface de rede, o tempo de ciclo de cada teste não deverá aumentar. Como os testes visuais executados pelo operador são da ordem de alguns segundos, haverá tempo suficiente para envio dos dados dos teste pela rede, mesmo usando microcontroladores de 8 *bits*, de acordo com experimentos realizados;

- **Memória de armazenamento de dados.** A memória de armazenamento de dados deve ser calculada para o pior caso, ou seja, quando não for possível transmitir os dados pela rede.

  Estimativa para a capacidade da memória de dados:

  - Tempo mínimo de um ciclo de testes por *display* = $20\,s$

  - Número de testes executados em duas horas = (2)(60)(3) = 360

  - Número de bytes necessários para armazenar os resultados de um único testes (utilizando estrutura em linguagem C) = 400 bytes

  - Tamanho mínimo de memória de dados = (400)(360) = 144 Kbytes.

- **Tempo de desenvolvimento**. Prazo para a entrega do primeiro protótipo: 8 meses. Módulos de sistemas embarcados já prontos para uso facilitarão o desenvolvimento do projeto;

Sistemas embarcados atendem as especificações de projeto:

- **Conectividade Ethernet TCP/IP mais protocolo HTTP.** Alguns sistemas embarcados já apresentam o *stack* TCP/IP mais protocolo HTTP, SMTP e FTP implementados, possibilitando o uso da Arquitetura de Integração *Web*;

- **Memória de programa.** Existência de microcontroladores com endereçamento de memória superior a 64Kbytes de programa suportarão a implantação dos protocolos TCP/IP, HTTP, FTP e SMTP , armazenamento de telas a serem enviadas ao *display* e mais o programa de controle de teste.

9.4.1 Soluções propostas

Todas as soluções propostas são baseadas em microcontroladores, já que o uso de PCs está descartada devido às restrições de tamanho. Das soluções propostas no item 6.5 três foram selecionadas. Outras não foram selecionadas por ainda não estarem disponíveis quando da inicialização do desenvolvimento do projeto de nosso estudo de caso, como os microcontroladores ColdFire da Motorola.

Solução proposta no. 1 — Módulo Dallas DSTINIm400

Características principais:

- Microcontrolador: Dallas DS80C400;
- 1MB de memória *Flash* para armazenar o programa;
- 1MB de memória RAM não volátil para armazenar os dados;
- Três portas seriais;
- Uma porta CAN2.0B;
- Clock em tempo real;
- Suporte para um controlador *Ethernet* PHY (*Ethernet Physical Layer Transceiver*) externo.O PHY implementa a camada física.
- Preço: US$ 69.

Solução proposta no.2 — Módulo Rabbit RCM3200

Características principais:

- Microcontrolador: Rabbit 3000 a 44.2MHz;
- 512Kbytes de memória *Flash* para armazenar o programa;
- 256Kbytes de memória RAM não volátil para armazenar os dados;

Figura 9.6: Módulo DSTINIm400.

- Seis portas seriais, duas configuráveis como *Serial Peripheral Interface* (SPI). SPI é um tipo de comunicação de dados serial síncrona *full duplex* com 3 sinais (*Serial Clock* (SCK), *Master Out Slave In* (MOSI) e *Master In Slave Out* (MISO)) que é implementada entre microcontroladores e dispositivos periféricos. Consiste de registradores de deslocamento que transmitem e recebem dados simultaneamente. Um dispositivo atua como mestre e outro como escravo;
- *Clock* em tempo real;
- Porta *Ethernet* 10/100 Base-T, RJ45, 3 LEDs;
- Preço: US$ 89.

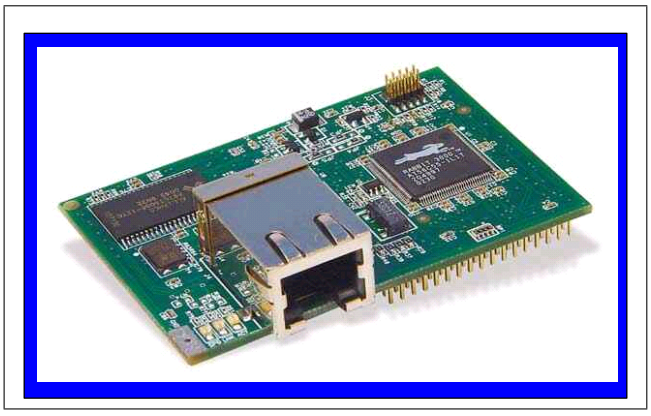


Figura 9.7: Módulo Rabbit RCM 3200.

Solução proposta no. 3 — Microcontrolador Motorola MC68HC812A4 e controlador de *Ethernet* Realtek RTL8019AS

Características principais:

- Microcontrolador: Motorola MC68HC812A4 (16 *bits*);
- Controlador *Ethernet*: RTL8019AS Realtek, *Full-Duplex* 10BaseT;
- Capacidade de acesso a até 4MB de memória RAM externa;
- Duas portas seriais SCI;
- Uma porta serial SPI;
- Conversor A/D de 8 *bits*.

9.4.2 Solução adotada

Devido ao tempo de projeto bastante limitado, foi descartado o uso do microcontrolador Motorola MC68HC812A4 com o controlador de *Ethernet* Realtek RTL8019AS. Esta opção exigiria um projeto complexo que envolveria o uso de:

- Memória *Flash* para armazenamento do programa;
- Memória RAM não volátil para armazenamento das variáveis de programas e dados de produção;
- *Clock* em tempo real;
- Implementação em *software* do *stack* TCP/IP e protocolos HTTP e FTP;
- Desenvolvimento do *lay-out* da placa de circuito impresso.

O uso de módulos microcontrolados com conectividade *Ethernet* TCP/IP já prontos adquiridos no mercado é a melhor solução. Tanto o módulo da Dallas quanto o módulo da Rabbit são equivalentes em termos de custo. Apesar de o módulo da Rabbit ser um pouco mais caro, ele apresenta o conveniente de já possuir o controlador de *Ethernet* e o conector RJ-45, no caso do módulo da Dallas será necessário

implementar o controlador de *Ethernet* e o conector RJ-45. O módulo da Dallas já vem com a bateria para memória RAM não volátil, mas isso não chega a não ser nenhum problema para o módulo da Rabbit, pois uma simples bateria "CR2032, de 3V" é de fácil aquisição no mercado, pois é utilizada em computadores PC. Ambos os módulos requerem a utilização de um circuito conversor A/D em separado, já que nenhum deles possui esta funcionalidade.

O fator decisivo na escolha da melhor solução é o desempenho. Verificou-se que o microcontrolador Rabbit 3000 é mais veloz que o microcontrolador Dallas DS80C320. Um quadro comparativo é apresentado na Figura 9.8 (Tabela obtida em `http://www.rabbit semiconductor.com/products/benchmarks/index.shtml`. Como estamos fazendo a comparação com o módulo DSTINIm400 necessitamos fazer a comparação do microcontrolador DS80C400 com o microcontrolador DS80C320. O ciclo de instrução do DC80C320 é de $121\,ns$ e o do DS80C400 é de $54\,ns$. Portanto, o DS80C400 é 2,24 vezes mais veloz que o DS80C320. Mesmo assim, o microcontrolador Rabbit 3000 continua sendo mais veloz que o DS80C400.

Rabbit 3000 Microprocessor Benchmark Comparisons			
Operation/Program	**Dhrystone 1.1 1,000s/sec**	**Whetstone 1,000s/sec**	**Sieve (milliseconds)**
Rabbit 3000 @ 50 MHz Dynamic C	6,570	813	53
AMD 188ES @ 40 MHz Borland 3.31 C	3,603	61	120
Zilog ez80 @ 40 MHz Zilog C compiler	2,914	20	158
Dallas DS80C320 (8051) @ 33 MHz Keil C	1,251	140	160
Phillips 80C51 @ 33 MHz Keil C	598	61	350

Figura 9.8: Desempenho comparativo de diversos chips microcontroladores.

Uma outra vantagem dos módulos da Rabbit é o fato de apresentarem baixa emissão de EMI. Os módulos baseados no Rabbit 3000 têm tipicamente menos que $10\,dB\,\mu V/m$ a $3\,m$, e em alguns testes tem sido alcançado $0\,dB$.

9.5 Escolha da linguagem de programação

Algumas considerações não técnicas devem ser levadas em conta na escolha do conjunto linguagem de programação mais CPU. São elas:

- Custo de cada módulo para fins de desenvolvimento;
- Custo do compilador;
- Possibilidade de fazer várias instalações do compilador;
- Documentação e legado de *software* do fabricante;
- Conhecimento da linguagem por parte da equipe de desenvolvimento.

A escolha do módulo microcontrolado Rabit RCM3200 nos dá como única possibilidade usar a linguagem C para a sua programação, obviamente também é possível utilizar o *assembly*. Pode-se escolher o C Ansi padrão ou Dynamic C Premier. Dynamic C Premier é projetado para trabalhar com microcontroladores Rabbit e é baseado no C ANSI. Um fator favorável à escolha do Dynamic C Premier foi o legado de *software* oferecido pelo fabricante, como comunicação serial, conversão analógica/digital, banco de dados, protocolos TCP/IP, HTTP, etc. Este legado de *software* está muito bem documentado e exemplificado, o que facilita o desenvolvimento de projetos. Além do fato, que o compilador Dynamic C Premier já contém as bibliotecas do Sistema Operacional em tempo real RTOS μC/OS de Jean Labrosse. Com o uso deste RTOS, a programação pode ser dividida em tarefas, tornando a manutenção do *software* mais fácil e as tarefas podem ser priorizadas umas em relação às outras.

9.6 Implementação das camadas do modelo TCP/IP

9.6.1 Camada Física (Interface de Rede + Física)

Opção adotada

A solução adotada para esta camada foi definida quando da escolha do módulo microcontrolado RCM3200. A camada de rede é responsável por duas funções bastantes distintas. A primeira programa e gerencia a comunicação com o PHY (ASIX AX88796L *10/100 BASE 3-in-1 Local CPU Bus Fast Ethernet Controller with Embedded SRAM*). A segunda função é controlar a comunicação do servidor HTTP com a rede *Ethernet*. A camada de rede executa as seguintes tarefas (MOKARZEL & CARNEIRO 2004):

- Na transmissão:
    1. Recebe os pacotes para serem enviados das camadas superiores.
    2. Controla esse pacote até ser transmitido.
    3. Acha o endereço MAC da placa de rede destino.
    4. Preenche as informações do cabeçalho *Ethernet*.
    5. Solicita espaço no buffer do PHY para enviar o pacote.
    6. Transfere o pacote para o PHY.
    7. Informa a quem solicitou a transmissão que o pacote foi enviado.
- Na recepção:
    1. Verifica quando existe pacote novo no PHY para ser tratado.
    2. Lê as informações do pacote.
    3. Caso o pacote esteja íntegro, verifica se o MAC de destino é o do servidor ou se a mensagem é de *broadcast*.
    4. Armazena o IP e o MAC na tabela ARP.
    5. Retira o cabeçalho *Ethernet*;

6. Passa o pacote para a camada superior ou para uma de suas funções de tratamento de mensagem.

A topologia utilizada é a do tipo estrela pois os módulos RCM3200 da *Rabbit Semiconductors* suportam somente esta topologia. Os DICs são conectados a um *hub* por meio de um cabo tipo par trançado. Existe um computador Mestre no portal *Web* que isola a rede de chão de fábrica da rede corporativa.

Meio físico: Cabo UTP de par trançado 100 Base TX.

Vantagens: Fácil expansão, gerenciamento e manutenção. Se um nó for removido do sistema, todos os outros continuam ativos.

Desvantagens: A rede para se o *hub* apresentar problemas, mas os DICs continuam ativos.

Restrição

Distância máxima de alcance com uso de hubs repetidores é de $400\,m$.

Merece atenção especial o número de DICs que podem ser interligados em função da quantidade de dados transferidos por cada um. A seguir é apresentado o cálculo do número máximo de DICS que podem ser interligados em rede com taxa de utilização de 35% (MORAES 2004), conforme (POZZUOLI 2003).

Cálculo do número de DICs produtores conectados ao Portal *Web*:

1. Largura de banda do dispositivo consumidor: BW $_{Consumer}$ (Mbits/segundos)
2. Tamanho do quadro (frame) gerado pelo DIC produtor: FS (em *bits*)
3. Período de reportagem do DIC produtor: T $_{Producer}$ (ms)

Dado um Switch Ethernet com 16 portas *full-duplex*:

- BW $_{Consumer}$ = 100Mb/s.

O número máximo de quadros em *bits* com utilização de 35% da porta é definida como:

- BW_{MAX} = 35.000.000 $bits/ms$ (Este número representa o total de tráfego incluindo CRC e cabeçalhos).

Para esta análise assuma o seguinte:

- O tamanho do quadro gerado pelo dispositivo produtor = 1000 bytes (com overhead XML).
- Período de reportagem (para 400 testes serem transmitidos em 2 minutos conforme especificação do cliente) = $120ms$ / 400 = $0{,}3ms$.

Portanto, pode ser calculado que cada dispositivo produtor consumirá:

- 1000 bytes X 8 *bits* X 1/0,3 = 26.667 *bits* da banda.

Portanto podemos concluir o seguinte:

- Cada DIC consome (26.667)/(35.000.00) = 0,076% da banda em cada segundo.
- (1/0,00076)= 1312 . Isto implica 1312 DICs produtores serem usados antes que 35% da largura de banda seja alcançada.

Poderiam ser usados, tranqüilamente, módulos microcontrolados com *Ethernet* a 10 Mbps. Neste caso, em vez de 1312 passaríamos a ter no máximo 131 DICs conectados em rede. Isto não seria um problema, pois não se tem mais do que 20 linhas de produção, cada uma operando com dois DICs. Mas o inconveniente dos módulos microcontrolados com controladores *Ethernet* 10Mbps é que a CPU deles opera com *clock* de 29MHz, inferior aos módulos que possuem controlador *Ethernet* 100Mbps cuja CPU opera com *clock* de 40MHz. Conseqüentemente o tempo de teste de cada *display* iria aumentar, tornado-os inapropriados para o nosso uso.

Em relação ao controlador *Ethernet* não temos problemas em enviar relatórios de testes a cada 0,3 s. O problema é que ler o registro do arquivo analítico do banco

de dados local para uma estrutura em linguagem C e transformá-la em arquivo XML(*parse*) tem um custo de aproximadamente 600 ms. Isto implica em dobrar o tempo de 2 minutos para 4 minutos para envio dos 400 testes. Isto sem contar o custo de abrir um soquete, transmitir a informação e fechar o soquete (aproximadamente 150 ms). Dependendo da aplicação, microcontroladores de 8 *bits* podem não ser beneficiados pelo uso de *Ethernet* a 100 Mbps.

9.6.2 Camada *Internet*

Nesta camada reina o protocolo IP. A camada *Internet* é bastante simples. Ela funciona como um empacotador/desempacotador no qual entra datagrama de um lado (proveniente da camada transporte) e sai pacote do outro ou entra pacote de um lado (camada interface de rede) e sai datagrama do outro.

O Protocolo IP provê comunicação entre os nós em diferentes tipos de rede. O IP especifica o endereço de origem e o endereço de destino do pacote de dados. O endereço especificado é um endereço IP de 32 *bits*. O IP fragmenta os pacotes e cada um fica independente do outro. Não há garantia que o pacote será entregue e não há *hand-shaking*.

9.6.3 Camada Transportes

De acordo com a especificação de requisitos (item RF10) a rede não deverá apresentar perda de dados. A confiabilidade na transmissão das informações é um dos aspectos mais importantes do sistema. A camada de transportes define se os dados serão transportados mais rapidamente com menos confiabilidade, ou mais demoradamente com mais confiabilidade.

Soluções

Há dois protocolos que podem ser usados na camada de transporte: *Transmission Control Protocol* (TCP) e *Use Datagram Protocol* (UDP).

A comunicação lógica fornecida pela camada de transporte pode ser não-orientada à conexão (UDP) ou orientada à conexão (TCP).

O UDP não assegura uma entrega de dados confiável, ou seja, não há garantia de entrega dos dados ao seu destino. O UDP não cuida de alguns problemas fundamentais de rede como congestionamento e fluxo de dados.

O protocolo TCP é confiável, ele resolve todos os problemas fundamentais de rede, como controle de congestionamento, seqüência e controle de fluxo. O TCP trabalha somente no modo ponto a ponto e é usado em aplicações como *Terminal Emulation* (Telnet), *File Transfer Protocol* (FTP) e *Hypertext Transfer Protocol* (HTTP). Há uma conexão entre o remetente e um único receptor antes do envio dos dados. É chamada de conexão ponto a ponto. É estabelecida uma transferência de dados *full duplex* entre os nós.

Opção adotada

Devido à necessidade de confiabilidade, a adoção do protocolo TCP. O conjunto de protocolos TCP/IP garante transmissão confiável e com o uso de um código de checagem de erro nas mensagens a possibilidade de não detectar uma corrupção dos dados é muito remota.

De fato, a escolha do protocolo HTTP, na camada de aplicação, para enviar os relatórios de testes no formato XML faz uso do protocolo TCP.

9.6.4 Camada de aplicação

Há muitas aplicações disponíveis para o conjunto de protocolos TCP/IP. As mais comuns são SMTP para envio de emails, FTP para transferência de arquivos e HTTP para visualização de páginas *Web*.

O *software* do DIC consiste em uma aplicação primária e uma aplicação secundária. A aplicação primária cuida do recebimento da aplicação secundária (programa de testes) via protocolo FTP. Quando o equipamento é ligado, a aplicação primária espera por $30\,s$ a transferência do programa de testes via FTP. Após este intervalo, não havendo nenhuma transferência de programa, é iniciada a aplicação secundária. A aplicação secundária verifica se o módulo eletromecânico conectado está em conformidade com ela. Não estando, a aplicação secundária envia um pedido

para o portal *Web* transferir o programa correspondente ao módulo mecânico conectado. A aplicação secundária executa um reset forçado. O DIC é reiniciado com a aplicação primária esperando a transferência do programa da aplicação secundária.

O protocolo HTTP serve tanto para visualização das páginas *Web* contidas nos DICs, como para envio e recebimento de arquivos XML entre os DICs e o portal *Web*.

9.7 Projeto do *hardware* do DIC

9.7.1 Arquitetura do Sistema

A arquitetura do sistema consiste em componentes para alimentação do *display*, comunicação com o *display*, medida dos parâmetros do *display* e comunicação *Ethernet* TCP/IP organizados em sete blocos operacionais (Figura 9.9):

1. Corel module Rabbit RCM3200 de 8 *bits*:

   Representa o principal bloco no sistema de aquisição de dados. O microcontrolador realiza a aquisição, o processamento dos dados adquiridos para extrair os valores numéricos dos parâmetros medidos e automaticamente publica os valores medidos na *Ethernet* TCP/IP, seja para o portal *Web* ou para algum outro cliente conectado na rede. O módulo RCM é caracterizado por:

   - Microcontrolador Rabbit 3000 operando a 44MHz;
   - 512Kbytes de memória *Flash* de programa;
   - 256Kbytes de RAM não volátil;
   - 52 linhas de E/S compartilhadas com cinco portas seriais;
   - Controlador *Ethernet* ASIX AX88796L *10/100 BASE 3-in-1 Local CPU Bus Fast Ethernet Controller with Embedded SRAM*.

   Estas características satisfazem os requisitos mínimos , em termos de capacidade de processamento, para um sistema de aquisição de dados baseado na *Web*.

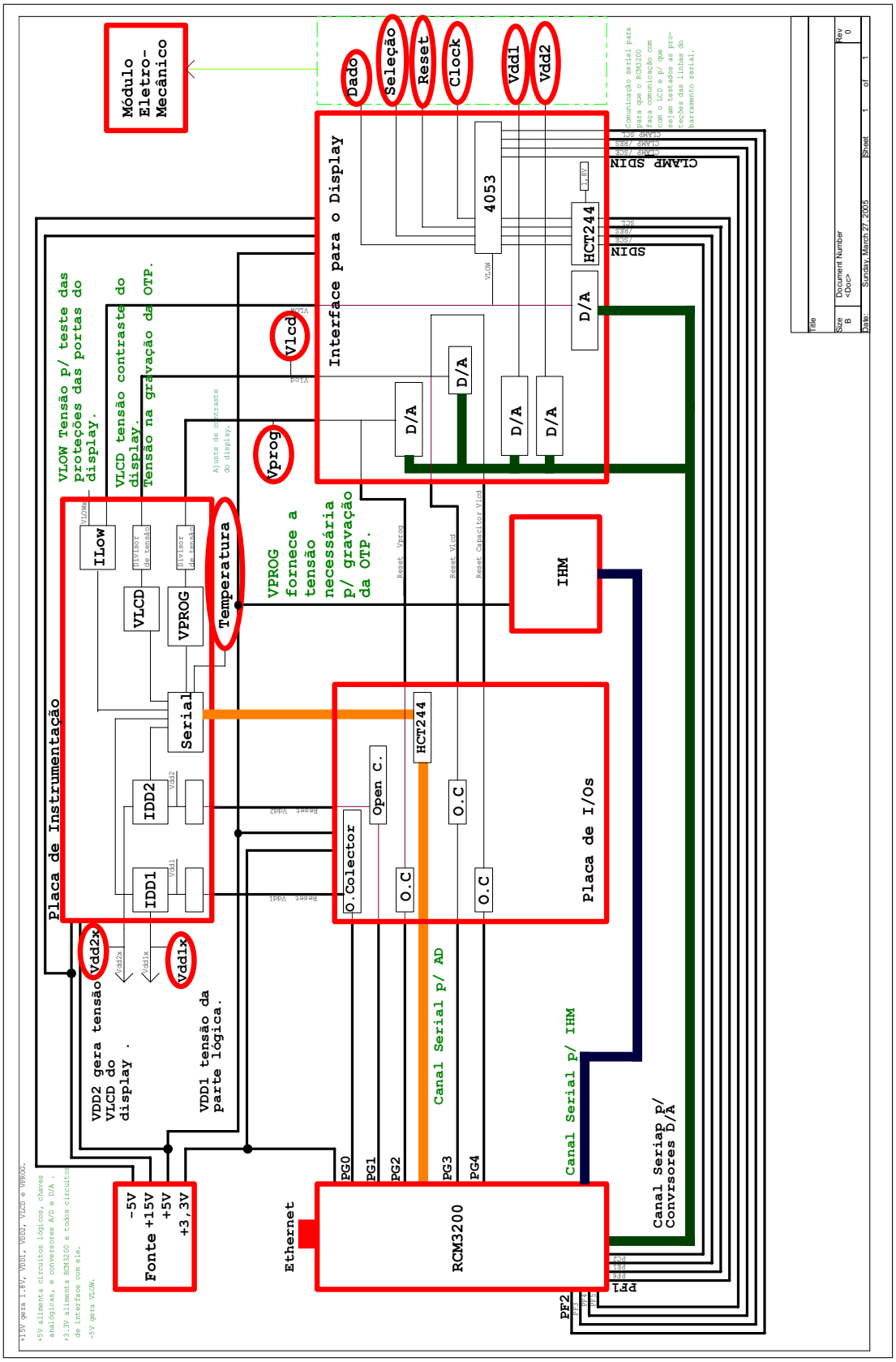


Figura 9.9: Diagrama de blocos do hardware.

2. Placa de instrumentação;

   A placa de instrumentação possui circuitos de condicionamento das tensões e correntes provenientes do *display*. Estes circuitos são:

   - Conversor de corrente-tensão IDD1. Mede consumo da parte lógica dos *displays*. É constituído por um circuito amplificador diferencial formado por amplificadores operacionais de precisão OP297GP (*Dual Low Bias Current Precision Operational Amplifier* e OP97FP *Low Power, High Precision Operational Amplifier*). A principal características destes operacionais é o baixo offset de tensão de entrada . A saída deste circuito é conectada diretamente ao conversor ADS7870 com o uso de diodos de proteção.
   - Conversor de corrente-tensão IDD2. Mede consumo da corrente que fornece contraste para o *display*. Possui a mesma constituição do conversor de corrente-tensão IDD1.
   - Conversor de corrente-tensão ILow. Mede consumo dos diodos de proteção das portas do *driver* do *display*. Possui a mesma constituição do conversor de corrente-tensão IDD1.
   - Conversor de temperatura tensão. Fornece o valor da temperatura ambiente para fins de ajuste de contraste do *display*. É constituído do circuito integrado LT1025 da *Linear Technology* que fornece uma tensão proporcional a temperatura ambiente (o incremento de 10mV corresponde ao incremento de 1°C). A saída deste circuito é conectada diretamente ao conversor ADS7870.

   Todas as demais tensões envolvidas no processo, sejam provenientes das fontes de alimentação ou dos *displays* sob teste, são conectadas através de divisores resistivo aos conversor ADS7870.

   Os circuitos de condicionamento dos sinais elétricos fornecem tensões entre 0 a 2,5V para o conversor A/D ADS7870. Este conversor possui 12 *bits* de resolução com 8 canais de ganho programável. O ADS7870 operando com tensão de referência interna de 2,5V possui resolução de 2,5V/4096 = 0,6uV,

satisfazendo o requisito do sistema em relação as medidas de tensão que requerem precisão de 10 mV. O ganho dos circuitos conversores corrente-tensão são de 324X. De modo que podemos medir correntes com passo de 0,6 uV / 324 = 0,0019 uA, satisfazendo o requisito do sistema em relação as medidas de corrente que requerem precisão de 0,1uA.

3. Placa conversora de nível entre o módulo RCM3200 e demais partes do circuito.

   Devido ao fato das portas do microcontrolador RCM3000 operarem com tensão de 3,3V e outras partes do sistema operarem com 5V,assim como o *displays* sob testes operam com tensão de até 1,8V, são necessárias várias conversões de níveis de tensão para funcionamento do sistema.

4. Fonte de alimentação.

   As fontes são todas reguladas e possuem LEDs indicativos de funcionamento. Usamos um transformador toroidal para diminuir a interferência eletro-magnética.

5. Placa de interface para o dispositivo a ser testado.

   Fornece para os *displays* sob teste:

   - Tensões Vdd1, Vdd2, Vprog, Vlcd e Vlow via conversores D/A seriais PCF8591;
   - Ajusta os níveis lógicos do módulo RCM3200 para níveis lógico do *driver* do *display*.

6. IHM.

   A comunicação com a IHM é feita através de interface serial SCI RS-232. Entre as portas do microcontrolador e a IHM existe um circuito integrado MAX232 para ajuste de nível de tensão.

7. Módulo eletromecânico.

   Fornece uma tensão para o conversor ADS7870 com o propósito de fornecer a sua identificação. Estes módulos são projetados com características físicas de acordo com as dimensões de cada *display*, assim como as do conector elétrico do *display*.

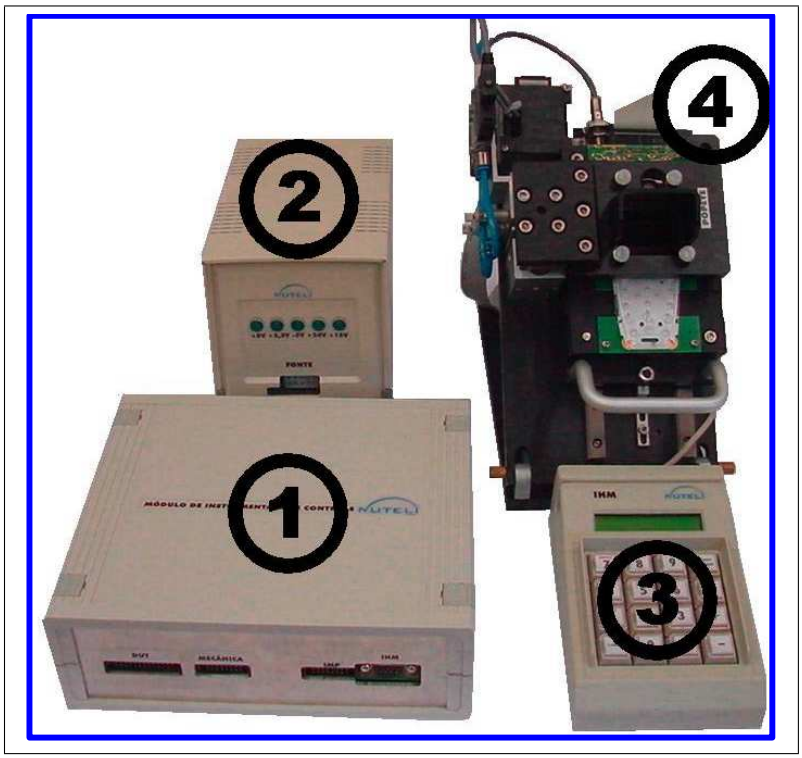


Figura 9.10: Módulos do sistema.

9.7.2 Apresentação dos módulos do sistema

Na figura 9.10 estão indicados todos os módulos que fazem parte do DIC. São eles:

1. **Módulo de Instrumentação e Controle**. É responsável pela conexão entre todos os módulos do DIC. Neste módulo são realizadas todas as medidas dos testes. É formado de (Figura 9.11):

   - Corel Module RCM3200;
   - Placa de Instrumentação e Controle;
   - Placa conversora de nível e E/S;
   - Interface para o *display*.

2. **Fonte de Alimentação.** É responsável pela geração de todos os níveis de tensão necessários para o funcionamento do DIC. Os níveis de tensão possuem LEDs indicativos de seu funcionamento (Figura 9.12).

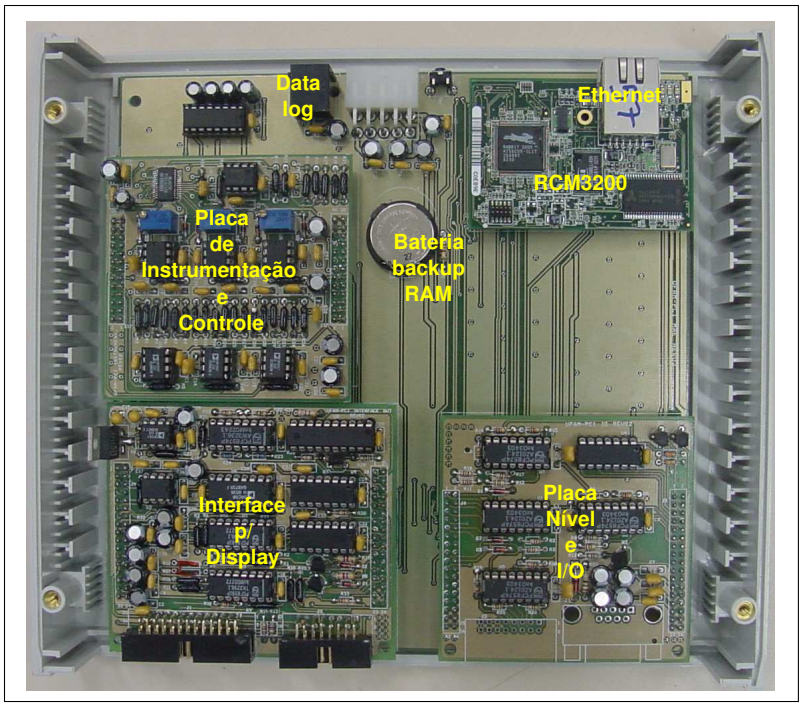


Figura 9.11: Módulo de instrumentação e controle.

Figura 9.12: Fonte de alimentação.

3. **Interface Homem-Máquina (IHM).** Através da IHM (Figura 9.13) o usuário poderá efetuar operações no DIC, como por exemplo configurar a linha em que ele se encontra, o posto, a hora, etc. A seguir estão relacionadas todas as teclas da IHM, bem como a função de cada uma no DIC:

Figura 9.13: Interface Homem-Máquina.

- **<+>.** Incremento do contraste do *display*;
- **<->.** Decremento do contraste do *display*;
- **ALT.** Habilita funções alternativas do teclado;
- **ANULA**;
- **ENTRA**;

- **RETESTE.** Indica que o *display* já foi contabilizado na produção;
- **Número 0**;
- **Número 1**. Primeira função: número 1, segunda função: OFF;
- **Número 2**. Primeira função: número 2, segunda função: Para baixo;
- **Número 3**;
- **Número 4**. Primeira função: número 4, segunda função: Para esquerda;
- **Número 5**. Primeira função: número 5, segunda função: CLEAR;
- **Número 6**. Primeira função: número 6, segunda função: Para direita;
- **Número 7**. Primeira função: número 7, segunda função: MENU;
- **Número 8**. Primeira função: número 8, segunda função: Para cima;
- **Número 9**.

A IHM também possui um *display* de cristal líquido para exibir as informações relacionadas ao teste, bem como instruções para que o teste seja realizado de maneira correta.

4. **Módulo eletromecânico**. Local aonde será colocado o *display* para teste (Figura 9.14). Basicamente é constiuído de um chassis, válvula pneumática e placa contendo os contatos elétricos para o *display*.

   São fornecidos quatro módulos mecânicos conforme Figura 9.15.

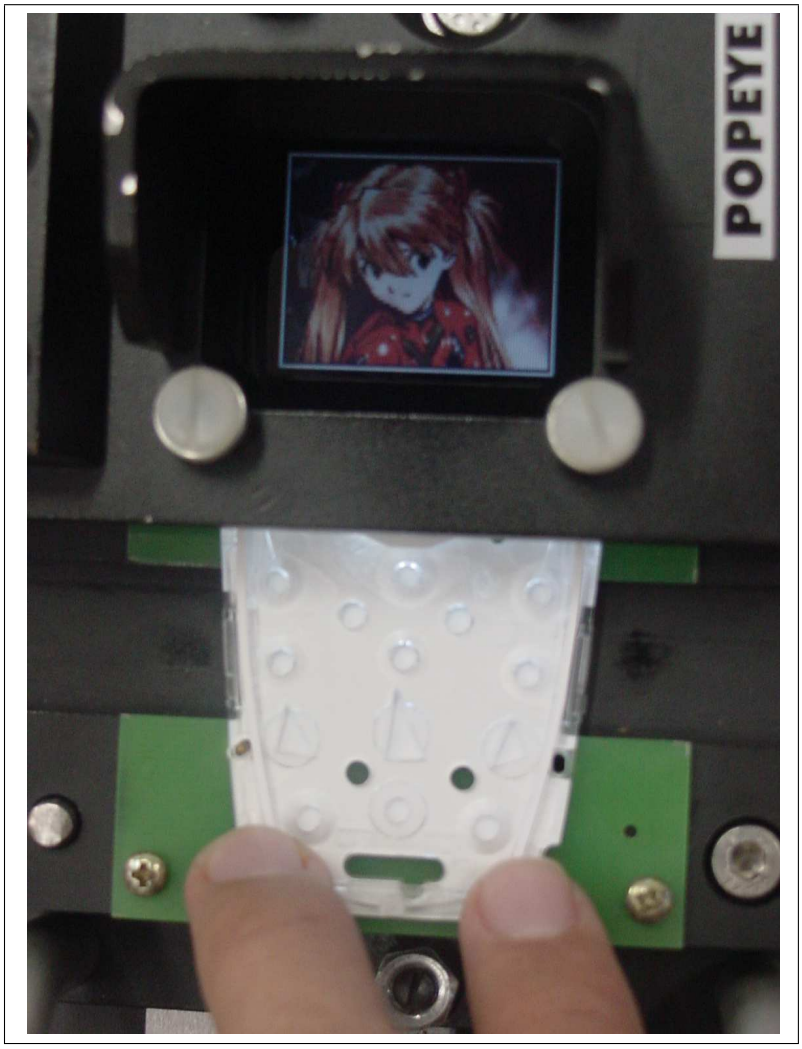


Figura 9.14: Módulo mecânico.

Figura 9.15: Módulos mecânicos disponíveis.

9.8 Etapas no desenvolvimento do *software*

Depois da etapa da especificação dos requisitos, começa a etapa de modelagem do *software* do DIC usando-se algumas características da linguagem *Unified Modeling Language* (UML) (DOUGLASS 2000) (DOUGLASS 2001) (DOUGLASS 2002).

Recentemente, o desenvolvimento de *software* em sistemas embarcados tornou-se mais complexo. Usuários dos sistemas requerem um tempo de desenvolvimento mais rápido com novos níveis de confiabilidade e desempenho. O custo de desenvolvimento do projeto deve ser cuidadosamente gerenciado. Todas estas exigências têm aumentado a pressão sobre a equipe de desenvolvimento do projeto. Desenvolvimento interativo tem se tornado um meio efetivo e popular de responder a estas pressões. Desenvolvimento interativo é uma forma de desenvolvimento em fases. Gerentes de projeto freqüentemente usam fases para quebrar projetos em seções dentro de um ciclo de vida do projeto. Cada fase tem um objetivo bem determinado e um escopo definido.

O *Rational Unified Process* (RUP) define quatro fases de desenvolvimento interativo(Figura 9.16) (http://www-128.ibm.com/developerworks/rational/library/2830.html#N1005C). Cada fase provê ao time de desenvolvimento um foco específico para um controle contínuo dos requisitos, arquitetura, projeto e codificação. Essas fases são:

- **Começo** (*Inception*). Define o escopo e ciclo de vida do projeto.
- **Elaboração** (*Elaboration*). Abrandamento dos riscos e criação de uma arquitetura básica.
- **Construção** (*Construction*). Desenvolvimento do sistema tão eficiente quanto possível.
- **Transição** (*Transition*). Treinamento dos usuários e aceitação do produto.

Em cada interação, disciplinas como gerenciamento de projeto, requisitos, análise e projeto, codificação, integração e teste são executadas.O grau de como cada uma dessas disciplinas é executado varia de acordo com a fase do projeto. O objetivo

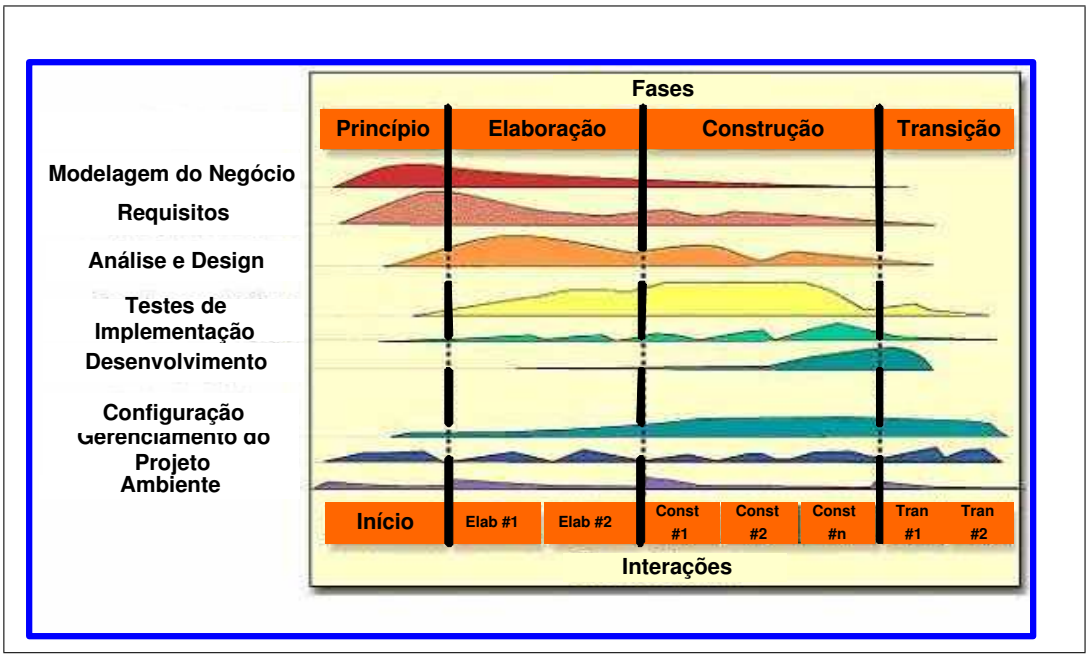


Figura 9.16: Fases de desenvolvimento interativo Rational.

de cada interação é produzir uma versão executável do sistema. Em cada interação todos os membros do time de desenvolvimento têm um entendimento claro dos objetivos da interação, escopo e critérios de evolução.

Como no desenvolvimento de dispositivos inteligentes normalmente encontramos um patamar maior de complexidade no *software*, somente alguns diagramas serão mostrados em cada fase de interação.

Fase interativa - princípio

Primeiramente, é feito a modelagem do sistema através de entrevistas com os usuários, conforme item 9.1. O analista começa a definir os requisitos. Os requisitos são revisados e aceitos. A primeira versão da especificação de requisitos é criada (item 9.2).

O analista cria o primeiro diagrama de caso de uso. Os diagramas de caso de uso mostram a comunicação do sistema com objetos externos (atores) na execução de uma função do sistema. Um diagrama de caso de uso pode conter outros diagramas de caso de uso. Um dos diagramas de caso de uso é mostrado na Figura 9.17.

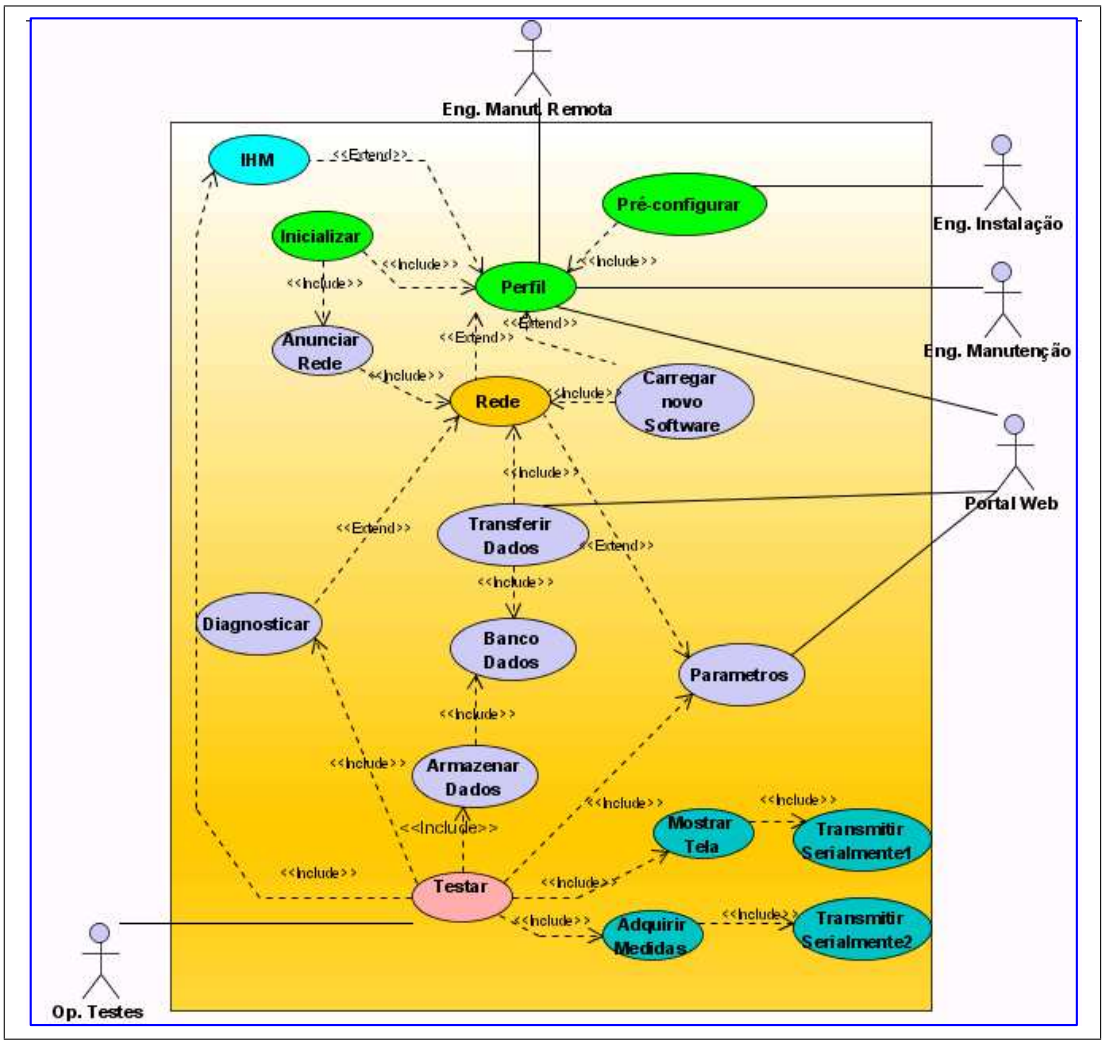


Figura 9.17: Diagrama de caso de uso principal.

O diagrama de caso de uso não deve necessitar de explicações para o seu entendimento. Por exemplo, a função "Testar" inclui funções como "Adquirir Medidas" e "Mostrar Tela" no *display* que está sendo testado. Verificamos também que o operador interage com esta função, assim como a função "Testar" usa as funções de "Diagnosticar", "Parâmetros", "Armazenar Dados" e "IHM". Lembramos que a função testar pode ter outros sub-diagramas que vão detalhando cada vez mais as funções do sistema.

As instâncias dos diagramas de caso de uso são chamadas de cenários. Um cenário pode ser representado através de um diagrama de seqüência. A Figura 9.18 mostra um dos diagramas de seqüência referente como o operador deverá interagir com o

sistema como descrito no item 9.1.

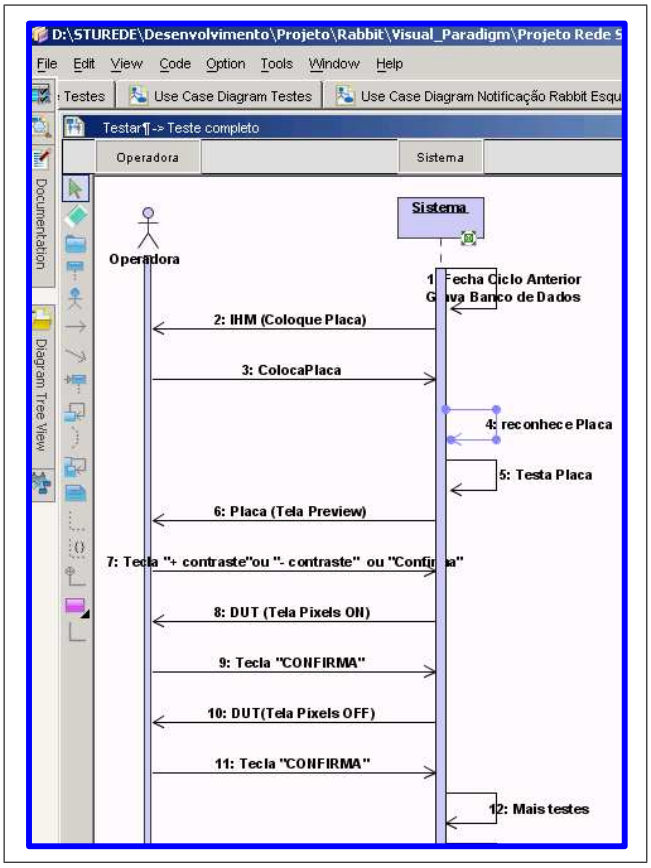


Figura 9.18: Diagrama de seqüência - operadora.

O analista submete os diagramas de caso de uso aos demais membros da equipe de desenvolvimento de projeto para revisão. O arquiteto do sistema com o gerente de projetos determinam quais requisitos devem ser detalhados. O analista detalha estes requisitos e diagramas de caso de uso. Nova revisão é feita pela equipe de desenvolvimento, e uma nova versão de requisitos está pronta para ser incorporada ao sistema.

Fase interativa - elaboração

Quando os diagramas de caso de uso estão completos, o arquiteto identifica os casos de uso e cenários que representam riscos para a arquitetura do sistema. O gerente de projetos junto com o arquiteto definem objetivos baseados na prioridade dos diagramas de caso de uso. O ajuste de prioridades assegura a equipe de desenvolvimento a

ordem na qual os requisitos devem ser detalhados, projetados e implementados. O analista adiciona detalhes aos requisitos se necessário. O gerente de projeto revisa os requisitos com a equipe de desenvolvimento e testes e estima o esforço necessário para implementar e testar os requisitos.

Implementação e testes dos requisitos são inicializados de acordo com suas prioridades, normalmente os primeiros testes realizados são os que representam mais riscos para a arquitetura. Desenvolvedores completam as análises dos diagramas de caso de uso e seqüencia e começam a projetar e implementar os componentes e banco de dados com o uso de uma máquina de estado.

Verificamos que a Máquina de Estado (Figura 9.19) possui sete tarefas concorrentes. Quando o sistema é inicializado as sete tarefas são construídas e inicializadas. São elas:

- **Notificação.** É a tarefa menos prioritária. Avisa ao Portal *Web* que o DIC entrou em operação enviando o perfil e estado de operação.
- **Desligamento.** É a mais prioritária. Após sua inicialização, suspende a si mesma. A rotina de teclado ativa a tarefa de desligamento após o operador teclar um pedido de desligamento. A tarefa de desligamento suspende todas as outras tarefas e só desliga o aparelho após transmissão de todos os testes no banco de dados ou caso verifique que não é possível transmitir os dados devido a um problema operacional na rede.
- **Envia testes XML.** Primeiro verifica se há arquivo sintético a ser enviado para o Portal *Web*, caso haja, transforma o arquivo sintético num arquivo XML e o envia ao portal *Web* usando o protocolo HTTP. Depois verifica se a registro no arquivo analítico a ser enviado, caso haja, transforma estes registros em um arquivo XML e o envia ao portal *Web* usando o protocolo HTTP.
- **Escuta.** Fica escutando se há requisições do portal *Web*, como envio de nova configuração (perfil) ou nova tabela de parâmetros para os testes. Todas estas transações são feitas via arquivos XML com uso do protocolo HTTP.
- **HTTP.** Envia páginas HTML para o browser que a requisitou. Podem ser do

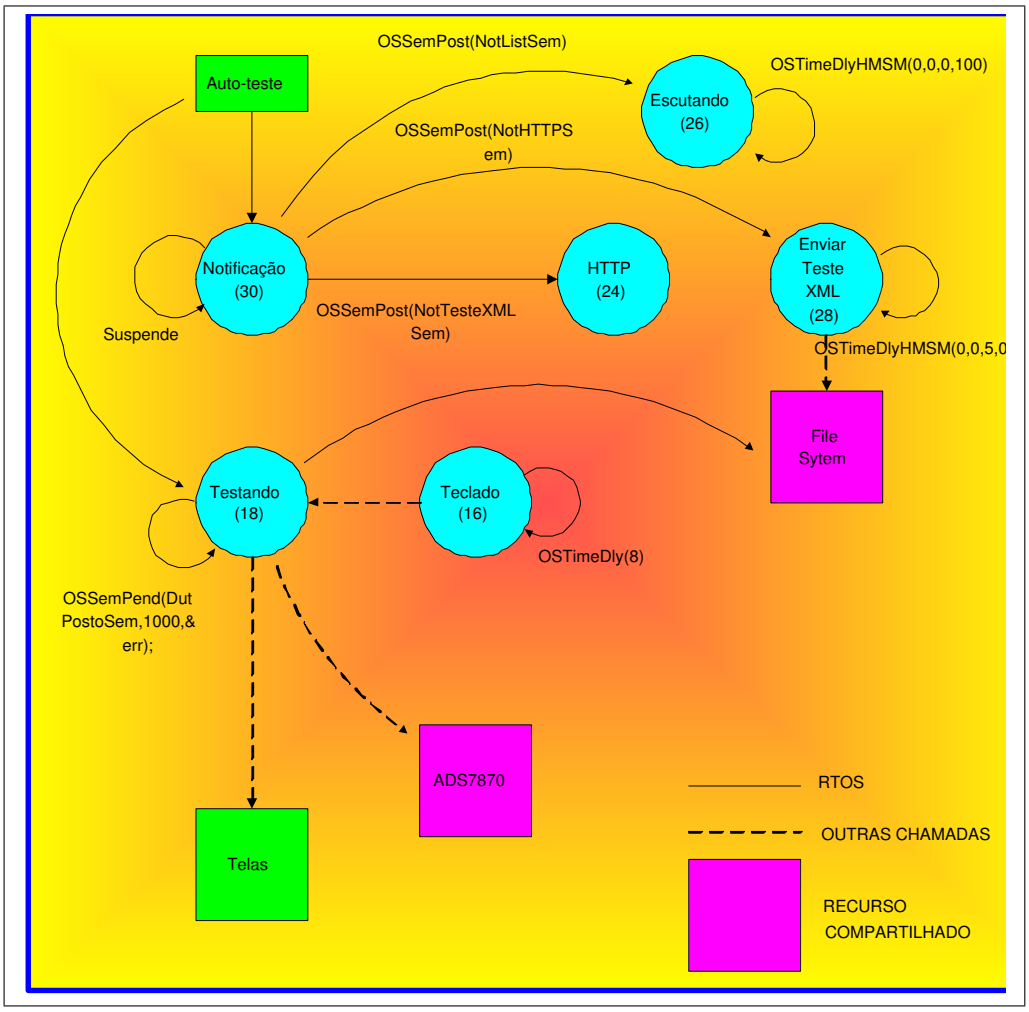


Figura 9.19: Diagrama da máquina de estado.

perfil do DIC, valores da tabela de parâmetros armazenada no DIC, valores das medidas do último ciclo de testes realizado e diagnóstico.

- **Testando.** Executa os testes no *display*. A transmissão de arquivos XML para o portal *Web* não afeta o desempenho desta tarefa, pois as tarefas relacionadas a rede possuem prioridades menor que esta tarefa.

- **Teclado.** Fica verificando se a IHM envia dados. Esta tarefa é executada a cada $100\,ms$, aproximadamente.

Fase interativa - construção

Ao finalizar a codificação de um requerimento, o desenvolvedor informa a engenharia de testes que este requerimento pode ser testado para fins de aprovação.

A codificação do programa no ambiente de desenvolvimento Dynamic C pode aparecer como o seguinte:

- **Definição das páginas HTML**

```
{ HTTPSPEC_FUNCTION, ‘‘teste.html’’ , 0, fnBufferTesteHTMLGet, 0,
NULL, NULL},

{ HTTPSPEC_FUNCTION, ‘‘/parametros.html’’,
0,fnBufferParametrosHTMLGet, 0, NULL, NULL},

{ HTTPSPEC_FUNCTION, ‘‘/configuracao.html’’, 0,
fnBufferConfiguracaoHTMLGet, 0, NULL, NULL},

{ HTTPSPEC_FUNCTION, ‘‘/indice.html’’, 0, fnBufferIndiceHTMLGet, 0,
NULL, NULL},

{ HTTPSPEC_FUNCTION, ‘‘/autodiagnose.html’’, 0,
fnTestesAutodiagnoseHTMLGet, 0, NULL, NULL}
```

- **Criação das Tarefas para o RTOS**

```
OSTaskCreateExt(fnTaskHTTP, (void *)0, 24, TASK_8_ID, 2048,(void
*)0, OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);  Criação da tarefa
HTTP, prioridade 24

OSTaskCreateExt(fnTaskDesligando,(void *)0,10,2, 256,(void *)0,
OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);  Criação da Tarefa de
Desligamento

OSTaskCreateExt(fnTaskTeclado,(void *)0,TASK_1_PRIO,TASK_1_ID,
256,(void *)0,OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR); Criação da
Tarefa Teclado

OSTaskCreateExt(fnTaskTestes, (void *)0, 18, TASK_3_ID, 512,
(void*)0,OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR); Criação da
Tarefa Testes
```

```
OSTaskCreateExt(fnTaskNotificacao, (void *)0, 30, TASK_4_ID,
256,(void *)0,OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR); Criação da
Tarefa Notificação

OSTaskCreateExt(fnTaskTemTeste,(void *)0,28,TASK_6_ID,2048,(void
*)0,OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR); Criação da Tarefa
Tem Teste

OSTaskCreateExt(fnListen, (void *)0, 26, TASK_7_ID, 2048, (void
*)0,OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR); Criação da Tarefa
Escuta

OSStart(); começa multi-tarefa
```

Como toda equipe de desenvolvimento do sistema, a engenharia de testes (ET) também deve estar envolvida com a evolução dos requisitos. É um erro comum excluí-los das revisões dos requisitos. A ET deveria rever os requisitos para assegurar que eles não são ambíguos e que são completos. Quando a implementação dos requisitos é baseada em interações, a ET revê os requisitos, planeja a interação e estima o esforço requerido para o plano de testes dos requisitos. Estas estimativas são enviadas ao gerente de projeto que finaliza o escopo da interação.

A ET recebe as funções para teste de acordo com os riscos dos requisitos. São elas:

- **Tempo de Execução dos Testes**. Primeiramente, o sistema é concebido de modo a funcionar *stand-alone*, ou seja, sem rede *Ethernet* TCP/IP. As medidas feitas com uso do circuito de instrumentação e interface que envia as telas para o *display* sob teste satisfazem os requisitos do sistema , principalmente em relação a duração do teste. O sistema é aproximadamente 40% mais rápido que o sistema de testes implementado atualmente na fábrica da Philips. O tempo de execução dos testes foi reduzido de $20\,s$ para menos de de $12\,s$ para um dos modelos de *display*. Deve-se levar em conta que quase metade do tempo dos testes é para aprovação do contraste do *display* e inspeção visual das telas para verificação dos pixels pelo operador. Essas "folgas de tempo" darão a possibilidade de implementar as rotinas da rede sem alterar significativamente o tempo de teste, devido ao uso do *kernel* preemptivo.

- **Transmissão de arquivos XML pela rede Ethernet TCP/IP**. Um arquivo XML é transferido pela rede em um tempo aproximado de 150 ms, ou seja, tempo de iniciar a conexão com o Portal *Web*, enviar o arquivo XML, receber a confirmação do Portal e encerrar a conexão. O tempo dos testes dos *displays* não foi alterado pois as tarefas relacionadas a rede só são executadas quando a tarefa de testes entra em estado de dormência.

  Para executarmos estes testes são necessários desenvolvimentos de pequenos programas que simulem o recebimento dos arquivos pelo Portal *Web*. A Figura 9.20 mostra um programa escrito em linguagem Java recebendo relatórios de testes em XML enviados por um DIC.

- **Banco de Dados**. Em seguida são testadas as bibliotecas relativas ao banco de dados local a ser implementado no DIC. Funções de escrita e leitura nos arquivos do banco de dados são realizadas em menos de 15 ms. A transformação dos registros do arquivo analítico para arquivo XML (*parse*) está levando um tempo considerável, aproximadamente 600 ms. Apesar de não constituir um problema para este estudo de caso, esta função deverá ser melhorada para diminuição da carga de processamento no microcontrolador causado pelo uso da linguagem XML.

- **Escuta**. Esta tarefa recebe configurações e parâmetros do Portal *Web*. Como se trata de informações que são usadas durante um ciclo de testes, elas são consideradas como compartilhadas. Só poderão ser atualizadas quando não houver *displays* sob teste. Neste caso utilizamos o serviço de semáforo do RTOS.

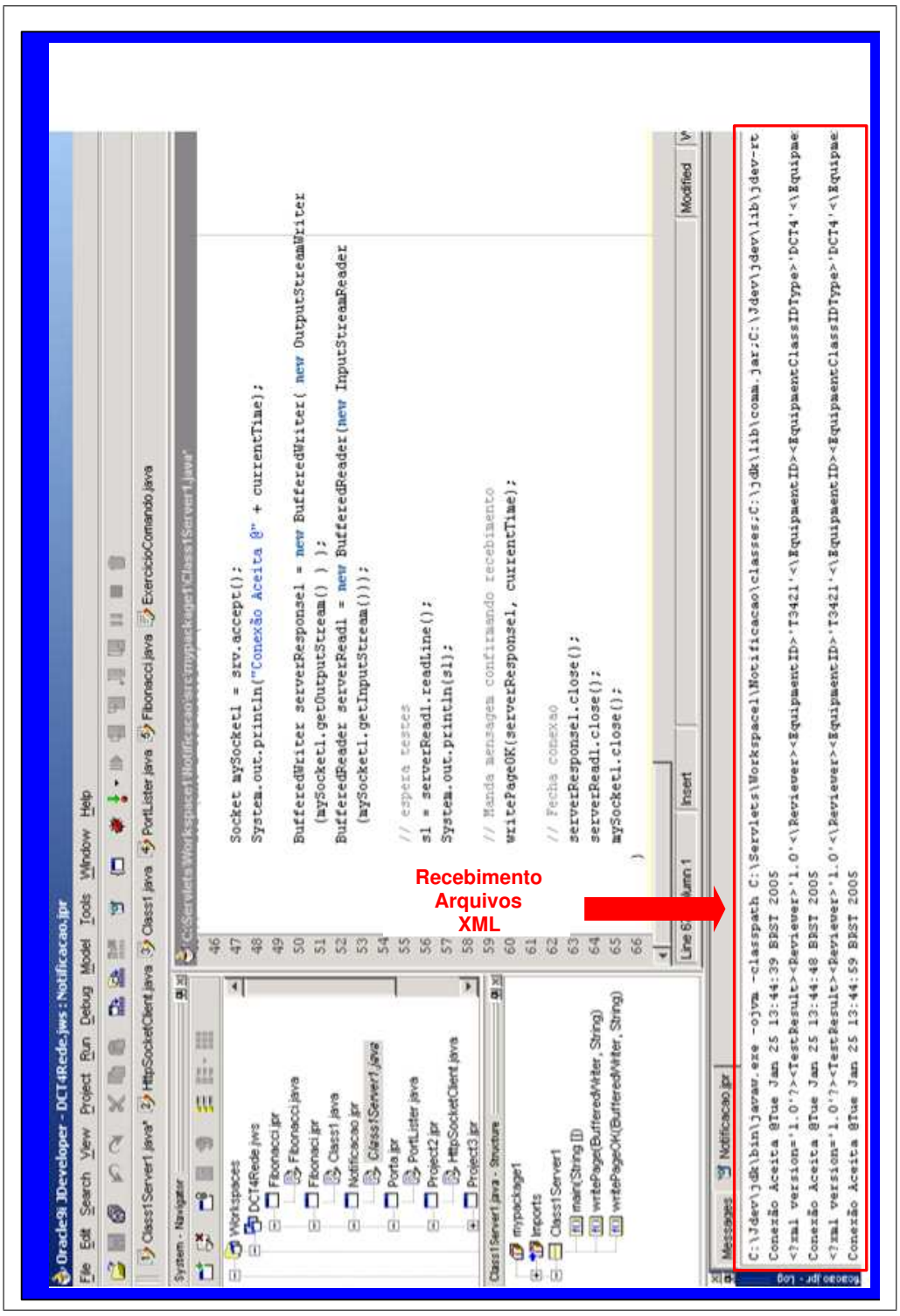


Figura 9.20: Programa Java recebendo arquivo XML do DIC.

- **Servidor HTTP**. Foram armazenadas na memória *Flash* do RCM3200 cinco páginas *Web*. Para acessar as páginas *Web* contidas no servidor HTTP do DIC basta digitar em um navegador *Web*, por exemplo, `http://10.10.1.1/indice.html`, para acessar a página de menu do DIC . As outras páginas serão acessadas com um simples clique no *hyperlink*. As páginas *Web* são:

  1. Índice. Tamanho de 623 bytes;

Figura 9.21: *Web Service* - índice.

  2. Perfil.Tamanho de 1043 bytes;
  3. Tabela de parâmetros.Tamanho de 3414 bytes;
  4. Resultado dos testes.Tamanho de 4181 bytes;
  5. Diagnóstico.Tamanho de 1369 bytes.

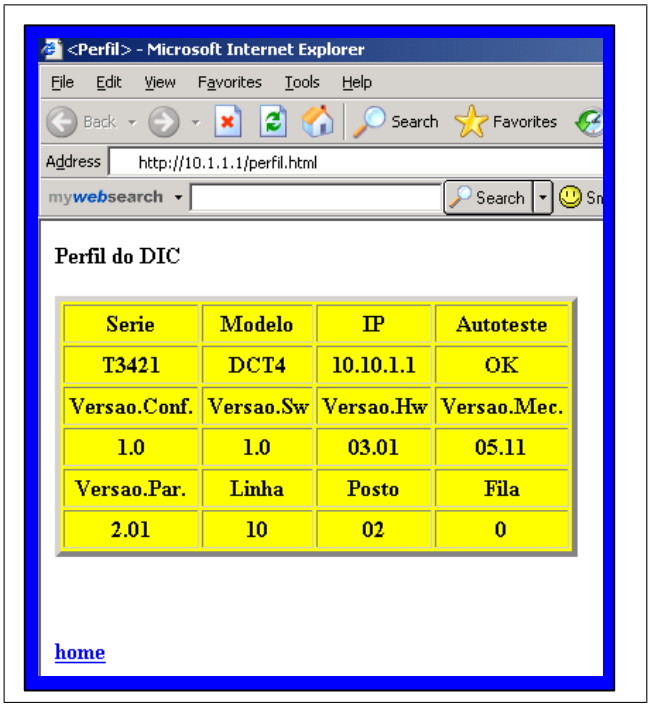


Figura 9.22: *Web Service* - perfil.

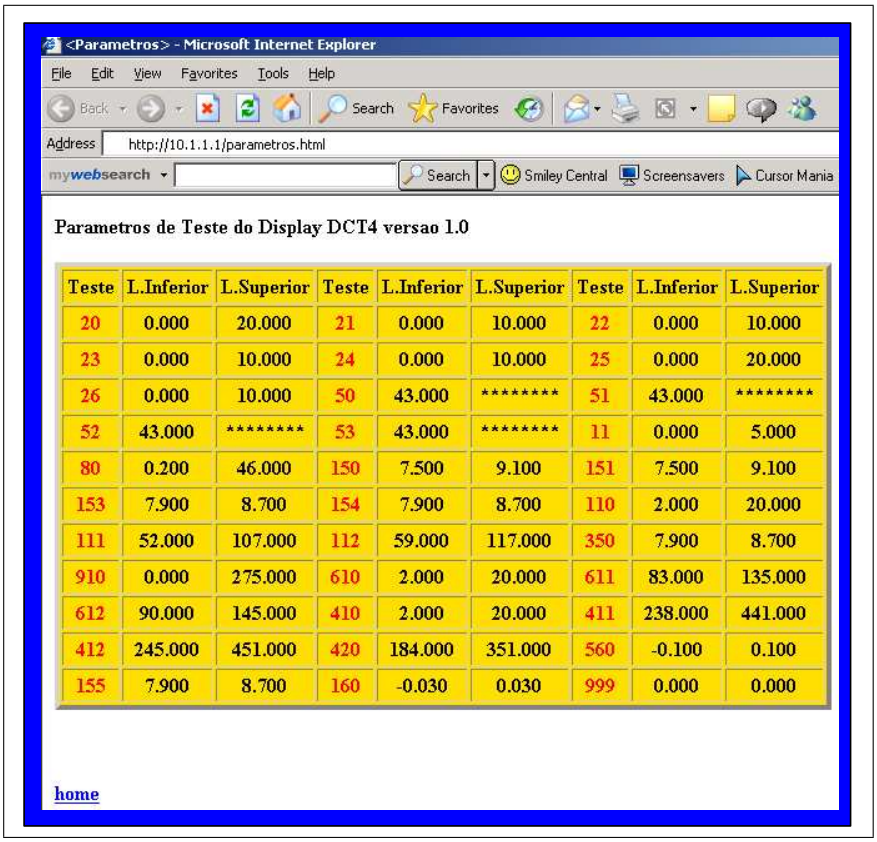
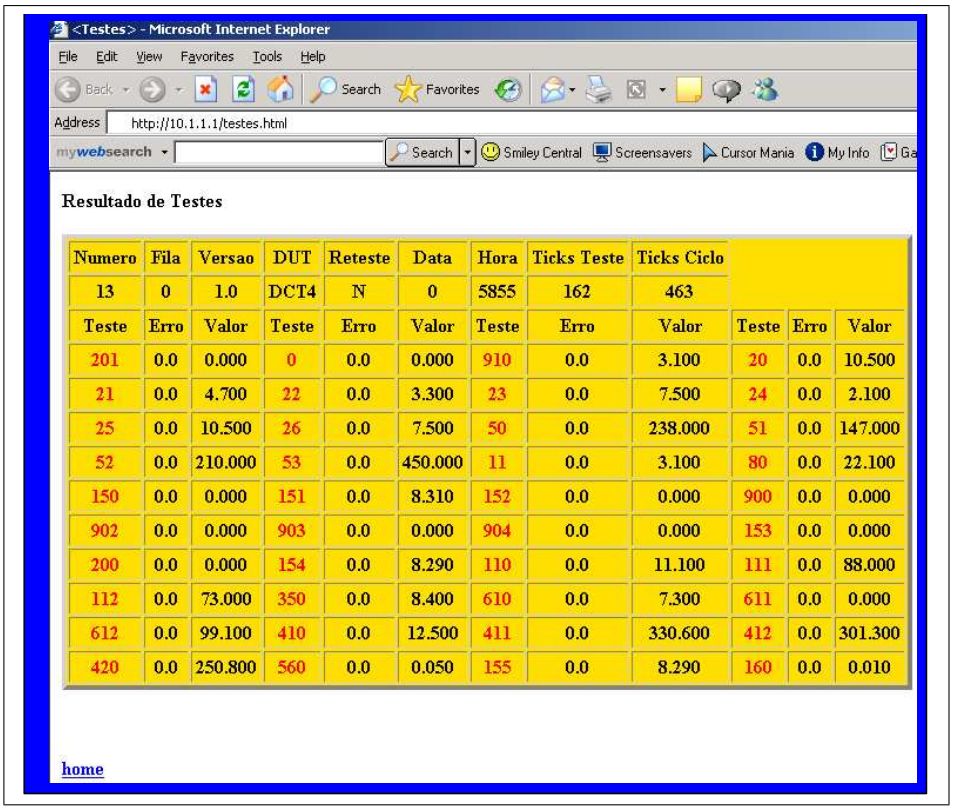
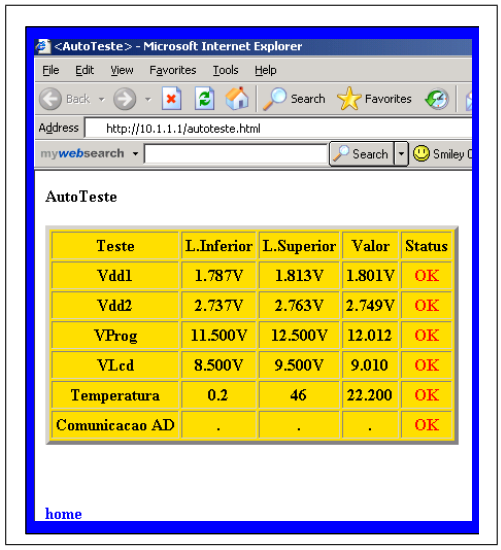


Figura 9.23: *Web Service* - parâmetros.



Numero	Fila	Versao	DUT	Reteste	Data	Hora	Ticks Teste	Ticks Ciclo			
13	0	1.0	DCT4	N	0	5855	162	463			
Teste	Erro	Valor	Teste	Erro	Valor	Teste	Erro	Valor	Teste	Erro	Valor
201	0.0	0.000	0	0.0	0.000	910	0.0	3.100	20	0.0	10.500
21	0.0	4.700	22	0.0	3.300	23	0.0	7.500	24	0.0	2.100
25	0.0	10.500	26	0.0	7.500	50	0.0	238.000	51	0.0	147.000
52	0.0	210.000	53	0.0	450.000	11	0.0	3.100	80	0.0	22.100
150	0.0	0.000	151	0.0	8.310	152	0.0	0.000	900	0.0	0.000
902	0.0	0.000	903	0.0	0.000	904	0.0	0.000	153	0.0	0.000
200	0.0	0.000	154	0.0	8.290	110	0.0	11.100	111	0.0	88.000
112	0.0	73.000	350	0.0	8.400	610	0.0	7.300	611	0.0	0.000
612	0.0	99.100	410	0.0	12.500	411	0.0	330.600	412	0.0	301.300
420	0.0	250.800	560	0.0	0.050	155	0.0	8.290	160	0.0	0.010

Figura 9.24: *Web Service* - testes.



Teste	L.Inferior	L.Superior	Valor	Status
Vdd1	1.787V	1.813V	1.801V	OK
Vdd2	2.737V	2.763V	2.749V	OK
VProg	11.500V	12.500V	12.012	OK
VLcd	8.500V	9.500V	9.010	OK
Temperatura	0.2	46	22.200	OK
Comunicacao AD	.	.	.	OK

Figura 9.25: *Web Service* - auto-teste.

9.9 Benefícios alcançados com as informações dos DICs

Com as informações provenientes dos DICs, o sistema de execução de manufatura (MES) poderá desempenhar as seguintes funções:

- **Centralização dos dados da produção**. Os parâmetros elétricos dos *displays* em cada ciclo de teste são enviados ao MES. Assim, calculam-se as médias e desvios padrões de cada medida que realimentarão a Engenharia de Projeto dos circuitos eletrônicos.

- **Acompanhamento da produção**. Em cada ciclo de teste o DIC envia a data e a hora da produção, o tempo de teste de cada *display* de celular e o tempo de operação de inserção do *display* de celular no DIC e posterior retirada. Mesmo não havendo produção a cada período de dois ciclos de testes os dados são enviados como teste não realizado (parada automática). Deste modo, o MES pode calcular a eficiência de cada linha de produção, indicando se a mesma está acima ou abaixo do planejado.

- **Controle das paralisações**. Caso ocorra alguma parada de produção, o operador poderá digitar no DIC o código do motivo da parada. Deste modo fica fácil para o MES informar quais as causas das paradas mais freqüentes.

- **Disponibilização de dados do controle de qualidade**. Em cada ciclo de teste são enviado o status do DIC, assim como o valor de tensão das fontes de alimentação, *off-set* dos circuitos condicionadores de sinal, temperatura e tempo de abertura da válvula do dispositivo eletro-mecânico.

- **Relatórios gerenciais**. Com as funções descritas anteriormente pode-se obter inúmeros relatórios por totalização, tendências, eficiência, etc.

Capítulo 10

Conclusão

O projeto e implementação do sistema embarcado em tempo real com banco de dados em chão de fábrica baseado no conceito *Web* foi uma experiência bem sucedida. O tempo de execução dos testes apesar do uso de sistema operacional, conexão de rede, banco de dados, uso de IHM e overload causado pelo uso de arquivos XML a serem enviados pela *Ethernet* foi reduzido de $30s$ para menos de $20s$ em uma dos *displays* sob teste, tempo bem menor do que os fornecidos pelos dispositivos de testes usados anteriormente que não possuíam qualquer dos recursos citados anteriormente.

O circuito de instrumentação e controle satisfez os requisitos do SQC, as medidas ficaram muito mais acuradas. O desenvolvimento do *software* com uso de um *kernel* preemptivo possibilitou gerenciar as tarefas concorrentes de testes dos parâmetros dos displays de celulares, teclado, banco de dados, formatação XML e servidor Web. Essas tarefas foram programadas e configuradas no módulo microcontrolado RCM3200 com conexão *Ethernet* de modo que usuários remotos podem supervisionar o status de funcionamento do dispositivo de teste enquanto este testa os *displays* e envia os dados de produção e engenharia para o sistema de execução de manufatura.

O *download* automático do programa quando da troca do módulo eletro-mecânico facilitou bastante a configuração das linhas de produção.

A *Internet* possibilitou a disseminação e compartilhamento dos dados num grau de integração jamais visto entre a fábrica da Philips no DI da ZFM, laboratório de desenvolvimento dos drivers de LCD na Europa e fabricantes dos mesmos na Ásia.

O monitoramento *on-line* das linhas de produção permitiu melhorar a eficiência das mesmas, através do fornecimento das principais causas de paradas que revelaram os gargalos de produção.

Apesar dos objetivos terem sido alcançados, a existência de várias tarefas no programa, configuração dos displays a serem testados, auto-diagnose, etc. tornou a codificação do software um pouco complexa. Esta complexidade sugere um estudo no uso de linguagens orientadas a objeto para microcontroladores, ferramentas de desenvolvimento de *software* ou implementação de uma linguagem visual para tornar o desenvolvimento de aplicativos mais rápido para novos modelos de *displays*.

Embora a *Ethernet* TCP/IP com uso de linguagens como XML seja bastante promissora em sistemas embarcados no chão de fábrica, principalmente tratando-se da padronização de protocolos, há ainda alguns problemas a serem resolvidos, tais como segurança relacionada a *Web*, assim como o *overhead* causado pela transformação de estruturas em C para arquivos XML precisa ter um impacto menor sobre a carga de processamento do sistema. Assim como também o tempo entre abrir e fechar um soquete para transmissão do arquivo XML. Deste modo, não recomendamos o uso da *Ethernet* TCP/IP em processos que necessitem rápida transmissão de dados, que não é o nosso caso. Por outro lado, com o contínuo desenvolvimento dos microcontroladores e compiladores acreditamos que controles baseados na *Web* e aplicações similares serão cada vez mais usados.

Bibliografia

AS-I (1998), *Actuator Sensor Interface - Technical Overview*, AS-I Trade Organization. http://www.as-interface.com.

AS-I (2001*a*), *Automation Networking - The Simple System Solution*, AS-I Trade Organization. http://www.as-interface.com.

AS-I (2001*b*), *Put Your Line on Track with AS-i*, rev1101 edn, AS-I Trade Organization. http://www.as-interface.com.

AXELSON, J. (2003), *Embedded Ethernet and Internet Complete*, 1 edn, Lakeview Research LLC.

BENITO, M. D. R., FUERTES, J. M., KAHORAHO, E. & ARZOZ, N. P. (1999), 'Performance evaluation of four fieldbuses', In: *Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation,* pp. 881-890, Barcelona, Spain .

BERGE, J. (2002), *Fieldbuses for Process Control: Engineering, Operation and Maintenance*, 1 edn, ISA - The Instrumentation, Systems, and Automation Society, NC.

BOLLELLA, G. & GOSLING, J. (2000), *The Real-time Specification for Java*, 1 edn, Pearson Education Corporate Sales Division, NJ.

BROOKS, P. (2001), 'Ethernet/ip - industrial protocol', In: *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation,* pp.505-514 .

CAMARGO, L. A., LOPEZ, L. C., BUENO, N. M. & CELANTE, S. R. (2003), 'Gerenciamento das informações dos processos industriais na cosipa utilizando a web'. VII Seminário de Automação do Processo - ABM, Santos-SP.

CI-86 (2003), 'Realcafé cede às exigências da globalização: Automatizar é preciso.', Controle e Instrumentação, 86, Editora Valete.

CI-94 (2004), 'Protocolos de comunicação', Controle e Instrumentação, 94, Editora Valete.

CIA (2003), *CANopen, an overview*, CAN in Automation (CIA). http://www. can-cia.org/can/protocol.

DEITEL, H., DEITEL, P. & NIETO, R. (2003), *XML - como programar*, 1 edn, Artmed Editora S.A., Porto Alegre.

DOUGLASS, B. P. (2000), *Designing Real-time Systems with UML - Part I.* http://www.embedded.com/98/9803fe2.htm.

DOUGLASS, B. P. (2001), *Designing Real-time Systems with UML - Part II.* http://www.embedded.com/98/9804fe3.htm.

DOUGLASS, B. P. (2002), *Designing Real-time Systems with UML - Part III.* http://www.embedded.com/98/9805fe3.htm.

EFEI (1998), 'E-947 tópicos especiais em automação industrial', Escola Federal de Engenharia de Itajubá. http://www.iee.efei.br/ gaii/can/hp.can.htm.

FELSER, M. & SAUTER, T. (2002), 'The fieldbus war: History or short break between battles?', In: *Proceedings of the 4th IEEE International Workshop on Factory Communication Systems,* pp. 73-80, Aug .

FF (1996), *Foundation Fieldbus Technical Overview FD-043 revision 3.0*, Fieldbus Foundation. http://www.fieldbus.org.

FRANCO, L. R. (2001), 'Uma visão geral sobre fieldbuses', Escola Federal de Engenharia de Itajubá - Comitê Fieldbus do COBEI/ABNT. http://www.iee.efei.br.

GHAHRAMANI, B. & PAULEY, M. A. (2003), 'Java in high-performance environments', *Computer* , v.36, n.9, pp.109–111.

GOH, H. & R., D. (2002), 'Fieldbus for control and diagnostics', 7*h International Conference on Control, Automation, Robotics and Vision,* v.3, pp.1377-1382 .

GRAVES, M. (2003), *Projeto de Banco de Dados com XML*, 1 edn, Pearson Education do Brasil, São Paulo.

HALLBERG, B. (2003), *Networking Rede de Computadores*, 1 edn, Editora Alta Books, Rio de Janeiro.

HARDIN, D. S. (2000), 'Embedded and real-time java', *Instrumentation and Measurement Magazine, IEEE* ,v.3, n.2, pp.49-50 .

HELPS, C. & HAWKS, V. (1999), 'Instrumentation technologies for improving manufacturing quality', In: *Proceedings of the Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Conference,* pp. 567-571, Cincinnati, OH , Oct .

HIGUERA-TOELDANO, M. & ISSARNY, V. (2000), 'Java embedded real-time systems: An overview of existing solutions', In: *Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing,* pp. 567-571, March, Newport, CA .

HOLZNER, S. (2002), *C++ Black Book*, 1 edn, MAKRON Books, São Paulo.

HUGHES, T. A. (2001), *Introduction to Programmable Controllers*, 3 edn, ISA - The Instrumentation Systems and Automation Society, NC.

KOULAMAS, C., ORPHANOS, G., KOUBIAS, S. & PAPADOPOULOS, G. (1999), 'Java and jini techologies as a vehicle to construct a unified control engineering tool and extended the real time aspects of a control network', In: *Proceedings of the* 25*th Annual Conference of the IEEE On Industrial Electronics Society,* pp. 831-836, Nov, San Jose, CA .

LABROSSE, J. (2002), *Micro C/OS II: The Real Time Kernel*, 1 edn, Hardcover.

LEE, Y., TAK, B. C., MAENG, H. S. & KIM, S. D. (2000), 'Real-time java virtual machine for information appliances', *IEEE Transactions on Consumers Electronics,* v. 46, n.4, pp.949-957 .

MANTOVANI, E. (1998), 'Aplicações e limitações da tecnologia lonworks na automação', Escola Politécnica da USP.

MAURER, S. S. (2002), 'A survey of embedded systems programming languages', *Potentials, IEEE,*v.21, n.2, pp.30-34 .

MOKARZEL, M. P. & CARNEIRO, K. P. M. (2004), *Internet Embedded TCP/IP para Microcontroladores*, 1 edn, Editora Érica Ltda, São Paulo.

MORAES, A. F. (2004), *Rede de Computadores*, 1 edn, Érica, São Paulo.

MOREIRA, J. E., MIDKIFF, S. P. & GUPTA, M. (1998), 'A comparison of java, c/c++, and fortran for numerical computing', *Antennas and Propagation Magazine, IEEE,*v.40, n.5, pp.102-105 .

MORTON, T. D. (2001), *Embedded Microcontrollers*, 1 edn, Prentice Hall, Ohio.

MOTOROLA (2004), *MCF5282 ColdFire Microcontroller User's Manual*, 2nd edn, Motorola Semiconductors, Inc., Denver, Colorado, USA.

MULLER, J., EPPLE, U., WOLLSCHLAEGER, M. & DIEDRICH, C. (2003), 'An efficient information server for advanced plant asset management', In: *Proceedings of the* 3*th IEEE Conference on Emerging Technologies and Factory Automation. ETFA '03. IEEE Conference,* pp. 66-73, Sept .

MUSKINJA, N., TOVORNIK, B. & TERBUC, M. (2003), 'Use of tcp/ip protocol in industrial environment', In: *Proceedings of the IEEE International Conference on Industrial Technology,* pp. 896-900, Dec .

NAIDITCH, R. (1998), 'Selecting a programming language for your project', In: *Proceedings of the* 17*th Digital Avionics Systems Conference,* pp. C34/1-C34/6 .

ODA, S. & TORIOGE, K. (2002), 'New i/o architecture in process control systems', In: *Proceedings of the* 41*st SICE Annual Conference,* pp. 3251 - 3254, 5-7 Aug., Tokyo, Japan .

OLSON, M. A. (2000), 'Selecting and implementing an embedded database system', *Computer* , v.33, n.9, pp.27–34.

PADRÃO Jr, F., FILHO, M. A. & LANNA, V. M. (2002), 'Implantação de mes em indústrias de bebidas', Controle e Instrumentação, 71, Editora Valete.

PINCETI, P. (2004), 'Fieldbus more than a communication link', *Instrumentation & Measurement Magazine, IEEE,*v.7, n.1, pp.17-23 .

POSTOLACHE, O., GIRÃO, P., PEREIRA, M. & RAMOS, H. (2002), 'An internet and microcontroler-based remote operation multi-sensor system for water quality monitoring', In: *Proceedings of the IEEE Sensors,* pp.1532-1536, June .

POZZUOLI, M. (2003), 'Industrial ethernet - issues and requirements'. http://www.toronto.ieee.ca/events/IndustrialEthernetIssuesandRequirementsv2a.pdf.

PROFIBUS (2002), *Profibus Technology and Application*, october 2002 edn, PROFIBUS International Support Center. http://www.profibus.com.

RAJA, P., VIJAYANANDA, K. & DECOTIGNIE, J. D. (1993), 'Polling algorithms and their properties for fieldbus networks', In: *Proceedings of the IECON'93 International Conference on Industrial Electronics, Control and Instrumentation,* pp. 530-534, Maui, HI .

RAJBHARTI, N. (2002), *AN833 - The Microchip TC/IP Stack.* http://www.microchip.com.

RODRIGUES, L. H. (2004), 'Adaptação da automação de uma planta industrial tradicional para tecnologia fieldbus'. Anais dos Trabalhos de Diploma, Instituto de Engenharia Elétrica, Escola Federal de Engenharia de Itajubá.

SANTORI, M. & ZECH, K. (1996), 'Fieldbus brings protocol to process control', *IEEE Spectrum,*v.33, n.3, pp.60-64 .

SCHNEIDER, K. (2003), 'Intelligent field devices in factory automation - modular structures into manufacturing cells', In: *Proceedings of the Emerging Technologies and Factory Automation,* ,pp. 101-103 .

SCOTT, A. V. & BUCHANA, W. (2000), 'Truly distributed control system using fieldbus technology', In: *Proceedings of the Seventh IEEE International Con-*

ference and Workshop on the Engineering of Computer Based Systems, pp. 165-173, April, Edinburgh .

SCRIMGER, R. (2002), *TCP/IP a Bíblia*, 3 edn, Editora Campus e Negócio.

SCRIMGER, R., LASALLE, P., PARIHAR, M. & GUPTA, M. (2002), *TCP/IP a Bíblia*, 3 edn, Editora Campus, Rio de Janeiro.

SHAHNASSER, H. & WANG, Q. (1998), 'Controlling industrial devices over tcp/ip by using lonworks', *Global Telecommunications Conference,* v.2, pp. 1309-1314 .

SHARP, D., PLA, E., LUECKE, K. R. & HASSAN II, R. J. (2003), 'Evaluating realtime java for mission critical large scale embedded systems', In: *Proceedings of the 9th Real-Time and Embedded Technology and Applications Symposium,* pp.30-36, May .

SNOONIAN, D. (2003), 'Smart buildings', *IEEE Spectrum* , v.40, n.8, pp.18–23.

SOUZA, C. S. & MATA, R. S. (2004), 'Microcontroladores na automação industrial: Evolução e tendências', Controle e Instrumentação, 89, Editora Valete.

TIAN, G. Y. (2001), 'Design and implementation of distributed measurement systems using fieldbus-based intelligent sensors', *IEEE Transactions on Instrumentation and Measurement,*v.50, n.5, pp.1197-1202 .

WBF (2003*a*), *Business to Manufacturing Markup Language (B2MML) - Common Version 0.2*, 2nd edn, World Batch Forum B2MML. http://www.wbf.org.

WBF (2003*b*), *Business to Manufacturing Markup Language (B2MML)ANSI/ISA-95.00.02-2001 Completeness, Compliance and Conformance Statement*, 2nd edn, World Batch Forum B2MML. http://www.wbf.org.

WOLLSCHLAEGER, M. (2000), 'A framework for fieldbus using xml descriptions', In: *Workshop on Factory Communication Systems,* pp. 3-10, 6-8 Sept., Porto .

WOLLSCHLAEGER, M. (2001), 'Framework for web integration of factory communication system', In: *Proceedings of the 8th IEE International Conference On*

Emerging Technologies and Factory Automation, pp. 261-265, Antibes-Juan les Pins .

WOLLSCHLAEGER, M., NEUMANN, P. & BANGEMANN, T. (2002), 'Web services for remote mainetance of fieldbus based automation systems', In: *Proceedings of the* 6*th Africon Conference,* pp. 247-252, Oct., Africa .

WOOD, G. (2000), 'State of play', *IEE Review,*v.46, n.4, pp.26-28 .

YABIN, P. & NISHITANI, H. (2003), 'Evaluation techniques for fieldbuses', *SICE Annual Conference, IEEE,* Fukui , Japan, 4-6 August .

YANG, H. & EAGLESON, R. (2002), 'Design and implementation of a internet-based embedded control system', In: *Proceedings of the 2003 IEEE Conference on Control Applications,* pp. 1181-1185, June, Ont., Canada .

ZUJUN, Y., HONGMEI, S., LI, A. & BAOQING, G. (2002), 'Lonworks-based intelligent train's fire alarming control networks', In: *Proceedings The* 2*nd International Workshop on Autonomous Decentralized System,* pp. 262-266, Nov .