Campus de São Carlos

**BLOCO FUNCIONAL PARA CONTROLE
FIELDBUS POR VARIÁVEIS DE ESTADO**

DENNIS BRANDÃO

Orientador: Prof. Tit. Mário Pinotti Júnior

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

Serviço de Pós-Graduação EESC/USP
EXEMPLAR REVISADO
Data de entrada no Serviço: 26/04/00
Ass:

BLOCO FUNCIONAL PARA CONTROLE FIELD BUS POR VARIÁVEIS DE ESTADO

Dennis Brandão

DEDALUS - Acervo - EESC

31100013642

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

ORIENTADOR: Prof. Tit. Mário Pinotti Jr.

São Carlos

2000

Class. TESE
Gutt. 0935
Tombo T 0159/00

st 1097637

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca - EESC/USP

B817b Brandão, Dennis
Bloco funcional para controle fieldbus por
variáveis de estado / Dennis Brandão. -- São Carlos,
2000.

Dissertação (Mestrado) -- Escola de Engenharia de
São Carlos-Universidade de São Paulo, 2000.

Área: Engenharia Mecânica.

Orientador: Prof. Dr. Mário Pinotti Jr.

1. Bloco funcional. 2. Fieldbus. 3. Variáveis de
estado. 4. Controle moderno. I. Título.

Ao povo brasileiro,
que com muito suor viabilizou
a realização desta pesquisa.

AGRADECIMENTOS

Agradeço primeiramente ao professor e amigo Dr. Mário Pinotti Júnior, pela orientação, dedicação e amizade;
aos companheiros do Laboratório de Eletrônica Embarcada Jaime, Xina, Suzete, George, Celso, Elenilton, Júnior, Madson e Álvaro pelo trabalho e estudo em conjunto;
a todos os engenheiros do Departamento de Desenvolvimento Eletrônico da SMAR, em especial aos dos grupos *Field Devices* e Aplicação, pela amizade, cooperação e orientação;
à Adriana pela sua força e presença nos momentos difíceis;
a todos os amigos que sempre estiveram presentes;
sinceramente, aos meus pais e ao meu irmão, por tudo;
e a Deus, pela vida, por esta oportunidade e por estes irmãos.

SUMÁRIO

LISTA DE FIGURAS.....	IV
LISTA DE TABELAS	VII
LISTA DE TABELAS	VII
LISTA DE ABREVIATURAS E SIGLAS	VIII
LISTA DE ABREVIATURAS E SIGLAS	VIII
LISTA DE SÍMBOLOS.....	X
RESUMO.....	XI
ABSTRACT.....	XII
1. INTRODUÇÃO	1
2. REVISÃO BIBLIOGRÁFICA	4
2.1. TRANSMISSÃO DE SINAIS NO CHÃO DE FÁBRICA.....	4
2.1.1. <i>Transmissão Analógica</i>	4
2.1.2. <i>Transmissão Digital</i>	6
2.2. SISTEMAS DIGITAIS DE CONTROLE DISTRIBUÍDO	7
2.3. TRANSMISSORES INTELIGENTES	8
2.3.1. <i>Primeira Geração</i>	9
2.3.2. <i>Segunda Geração</i>	10
2.4. MODELO DE REFERÊNCIA ISO/OSI	12
2.5. O MODELO ISO/OSI REDUZIDO DE TRÊS CAMADAS	13
2.6. PROJETO PROWAY	14
2.7. PROJETO MAP	15
2.8. BARRAMENTO DE CAMPO	17
2.9. O IMPACTO DA INTRODUÇÃO DO FIELDBUS.....	18
2.10. O PROFIBUS.....	22
2.10.1. <i>Profibus DP</i>	23
2.10.2. <i>Profibus PA</i>	25
2.10.3. <i>Profibus FMS</i>	25
2.11. O FIELDBUS FOUNDATION.....	27
2.11.1. <i>Topologia de Rede</i>	29

2.11.2.	<i>Processo de Aplicação (AP)</i>	31
2.11.3.	<i>Function Blocks AP</i>	32
2.11.4.	<i>System Management Kernel</i>	37
2.11.5.	<i>System Management Kernel Protocol Communications - SMKP</i>	38
2.11.6.	<i>Network Management Agent</i>	38
2.11.7.	<i>Camada de Aplicação - FMS</i>	39
2.11.8.	<i>SubCamada de Aplicação - FAS</i>	40
2.11.9.	<i>Camada de Enlace - DLL</i>	41
2.11.10.	<i>Camada Física - PHY</i>	43
2.11.11.	<i>Virtual Communication Relationship - VCR</i>	44
2.11.12.	<i>Device Descriptions - DD</i>	45
3.	PROCESSO DE APLICAÇÃO DOS BLOCOS FUNCIONAIS	48
3.1.	INTRODUÇÃO	48
3.1.1.	<i>Resource</i>	49
3.1.2.	<i>Function Block (Bloco Funcional)</i>	50
3.1.3.	<i>Transducer Block (Bloco Transdutor - TRD)</i>	50
3.1.4.	<i>Link Object (Objeto Link)</i>	50
3.1.5.	<i>Resource Block (Bloco Resource - RES)</i>	50
3.1.6.	<i>View e Trend Objects (Objetos view e trend)</i>	51
3.1.7.	<i>Alert Objects (Objetos alerta)</i>	51
3.1.8.	<i>Domain e Program Invocation Objects (Objetos domínio e program invocation)</i>	51
3.1.9.	<i>Action Objects (Objetos ação)</i>	51
3.1.10.	<i>System Time (Tempo do sistema)</i>	52
3.1.11.	<i>Function Block Schedule (Escalonamento dos blocos funcionais)</i>	52
3.1.12.	<i>Network Traffic (Tráfego de rede)</i>	52
3.2.	OPERAÇÕES CARACTERÍSTICAS DE SISTEMAS DE CONTROLE DISTRIBUÍDOS	53
3.2.1.	<i>Configuração e definição dos function blocks</i>	53
3.2.2.	<i>Configuração das ligações ou links entre os function blocks</i>	54
3.2.3.	<i>Processamento dos alarmes</i>	54
3.2.4.	<i>Acesso a informações para a interface do operador, diagnóstico e configuração</i>	55
3.2.5.	<i>Coordenação do funcionamento dos blocos e sua operação</i>	55
3.3.	INTERAÇÕES EXTERNAS	57
3.3.1.	<i>Interação com Aplicações de Configuração</i>	57
3.3.2.	<i>Interação com Interfaces Homem-Máquina</i>	58
3.3.3.	<i>Interação com Outras Aplicações de Configuração</i>	58
3.3.4.	<i>Interação com outros Dispositivos de Campo</i>	59
3.4.	ESTRUTURA DO FBAP	59
3.4.1.	<i>Recursos em um Transmissor</i>	59

3.4.2.	<i>Objeto Bloco</i>	60
3.4.2.1.	Eventos.....	60
3.4.2.2.	Parâmetros	61
3.4.2.3.	<i>Resource Block</i>	64
3.4.2.4.	<i>Transducer Block</i>	65
3.4.2.5.	<i>Function Block</i>	66
3.4.3.	<i>Function Block Links</i>	66
3.4.4.	<i>Alert Objects</i>	67
3.4.5.	<i>Trend Objects</i>	67
3.4.6.	<i>View Objects</i>	68
3.4.7.	<i>Program Invocation e Domain Objects</i>	68
3.4.8.	<i>Action Objects</i>	68
3.5.	MAPEAMENTO DOS RECURSOS DO FBAP	69
3.5.1.	<i>Function Block Application Model</i>	69
3.5.2.	<i>Data Type e Data Structures</i>	69
3.5.3.	<i>Objet Extension</i>	70
3.5.4.	<i>O objeto Diretório do OD</i>	70
3.5.4.1.	Definição do Diretório.....	71
4.	ESTIMAÇÃO DE ESTADOS NA TEORIA MODERNA DE CONTROLE	74
4.1.	REALIMENTAÇÃO DE ESTADOS.....	76
4.2.	ESTIMAÇÃO DE ESTADOS	78
4.3.	INTRODUÇÃO DE SINAL DE REFERÊNCIA	81
5.	BLOCO FUNCIONAL ESTIMADOR DE ESTADOS	86
5.1.	INTEGRAÇÃO.....	86
5.2.	ESTRUTURA E FUNÇÕES	88
5.3.	MALHAS DE CONTROLE POR ESTIMAÇÃO DE ESTADO	92
6.	TESTES E RESULTADOS	99
6.1.	MODELAGEM DA PLANTA	101
6.2.	CONFIGURAÇÃO DO ESTIMADOR DE ESTADOS	102
6.3.	RESULTADOS PRÁTICOS	106
7.	CONCLUSÕES	109

LISTA DE FIGURAS

FIGURA 1 - SISTEMA DE CONTROLE CENTRALIZADO	6
FIGURA 2 - SISTEMA DIGITAL DE CONTROLE DISTRIBUÍDO (SDCD)	8
FIGURA 3 - MODELO DE ISO/OSI DE SETE CAMADAS	13
FIGURA 4 - MODELO ISO/OSI REDUZIDO DE TRÊS CAMADAS.....	14
FIGURA 5 - ESTRUTURA MAP E MAP/EPA	16
FIGURA 6 - ESTRUTURA MINI-MAP	16
FIGURA 7 - BARRAMENTO DE CAMPO	17
FIGURA 8 - COMPARAÇÃO DA ESTRUTURA FÍSICA DO DCS COM O FCS.....	19
FIGURA 9 - DISTRIBUIÇÃO DA INTELIGÊNCIA ENTRE OS TRANSMISSORES DE CAMPO	21
FIGURA 10 - ESTRUTURAS DAS CAMADAS DE REDE DA FMS, DP E PA.....	23
FIGURA 11 - MÉTODO HÍBRIDO DE ACESSO AO MEIO FÍSICO NO PROFIBUS.....	24
FIGURA 12 - ALOCAÇÃO DAS CAMADAS NO MODELO PROFIBUS FMS	26
FIGURA 13 - ESTRUTURA DA CAMADAS DE REDE DA FIELDBUS FOUNDATION.....	28
FIGURA 14 - OPERAÇÃO DAS ESTRUTURAS DE REDE EM UM TRANSMISSOR INDUSTRIAL FOUNDATION.....	28
FIGURA 15 - TOPOLOGIA SINGLE LINK.....	30
FIGURA 16 - TOPOLOGIA <i>MULTILINK</i>	31
FIGURA 17 - VISUALIZAÇÃO DO PROCESS APPLICATION PELA REDE.....	32
FIGURA 18 - MALHA DE CONTROLE COM PID.....	34
FIGURA 19 - ESCALONAMENTO DE BLOCOS FUNCIONAIS.....	35
FIGURA 20 - MACROCICLO COM FASE SÍNCRONA E ASSÍNCRONA	36

FIGURA 21 - PASSAGEM DE TOKEN.....	42
FIGURA 22 - CODIFICAÇÃO <i>MANCHESTER</i>	44
FIGURA 23 - FUNCIONAMENTO DA <i>DEVICE DESCRIPTION</i>	46
FIGURA 24 - APLICAÇÕES DE BLOCOS FUNCIONAIS EM UM SISTEMA.....	48
FIGURA 25 - COMPONENTES CHAVES DO FBAP	49
FIGURA 26 - ESTRUTURA BÁSICA DO OD.....	71
FIGURA 27 - ESTRUTURA DO DIRETÓRIO DO OD.....	72
FIGURA 28 - REPRESENTAÇÃO DE UM SISTEMA DINÂMICO EM ESPAÇO DE ESTADOS.....	75
FIGURA 29 - REALIMENTAÇÃO DE ESTADOS	76
FIGURA 30 - OBSERVAÇÃO DE ESTADOS	79
FIGURA 31 - REALIMENTAÇÃO DE ESTADOS ESTIMADOS.....	81
FIGURA 32 - ESTIMADOR DE ESTADOS NO CAMINHO DIRETO E DE REALIMENTAÇÃO.....	82
FIGURA 33 - ESTIMADOR EM ALOCAÇÃO DE ZERO.....	83
FIGURA 34 - MALHA COM ESTIMADOR AUTÔNOMO.....	84
FIGURA 35 - ESTIMADOR COM REALIMENTAÇÃO DE SAÍDA.....	85
FIGURA 36 - BLOCO FUNCIONAL ESTIMADOR DE ESTADOS	89
FIGURA 37 - MALHA COM ALOCAÇÃO DE ZEROS EM DIAGRAMA DE BLOCOS FUNCIONAIS.....	93
FIGURA 38 - MALHA COM ESTIMADOR AUTÔNOMO OU SEGUIDOR DE ERRO BASEADA EM BLOCOS FUNCIONAIS.....	94
FIGURA 39 - MALHA COM ESTIMADOR AUTÔNOMO E REALIMENTAÇÃO DE SAÍDA	96
FIGURA 40 - MALHA SEGUIDORA DE ERRO COM SENSOR CONVENCIONAL.....	98

FIGURA 41 - FORNO ELÉTRICO INSTRUMENTADO COM <i>FIELD</i> BUS	99
FIGURA 42 - CANAL <i>FIELD</i> BUS NO CONFIGURADOR SYSCON.....	100
FIGURA 43 - MALHA COM ESTIMADOR AUTÔNOMO.....	103
FIGURA 44 - MALHA COM ESTIMADOR AUTÔNOMO EM BLOCOS FUNCIONAIS....	103
FIGURA 45 - BLOCO ESTIMADOR DE ESTADOS <i>ON LINE</i> NO CONFIGURADOR <i>FIELD</i> BUS.....	105
FIGURA 46 - SIMULAÇÃO DO ESTIMADOR AUTÔNOMO NA CONFIGURAÇÃO SIMULADA.....	106
FIGURA 47 - SIMULAÇÃO DO ESTIMADOR AUTÔNOMO NA CONFIGURAÇÃO IMPLEMENTADA.....	107
FIGURA 48 - ENSAIO DO ESTIMADOR AUTÔNOMO NA CONFIGURAÇÃO IMPLEMENTADA.....	107

LISTA DE TABELAS

TABELA 1 - PRECISÃO COMPARATIVA DOS AMBIENTES ANALÓGICO E DIGITAL .	10
TABELA 2 - BLOCOS FUNCIONAIS PADRÃO	34
TABELA 3 - SERVIÇOS DA FMS	40
TABELA 4 - CARACTERIZAÇÃO DOS VCR ENDPOINTS.....	41
TABELA 5 - TIPOS DE VCRS	42
TABELA 6 - COMPARAÇÃO ENTRE AS VCRS.....	45
TABELA 7 - ESTRUTURA DE INDEXAÇÃO NO DIRETÓRIO DO OD	73
TABELA 8 - PROJETO DE PARÂMETROS DE BLOCOS FUNCIONAIS	87
TABELA 9 - PARAMETRIZAÇÃO DO ESTIMADOR DE ESTADOS	91
TABELA 10 - CONFIGURAÇÃO BÁSICA DA MALHA COM ALOCAÇÃO DE ZEROS.....	94
TABELA 11 - CONFIGURAÇÃO BÁSICA PARA MALHA COM ESTIMADOR AUTÔNOMO	95
TABELA 12 - CONFIGURAÇÃO DA MALHA SEGUIDORA DE ERRO.....	95
TABELA 13 - CONFIGURAÇÃO DA MALHA COM ESTIMADOR AUTÔNOMO E REALIMENTAÇÃO DE SAÍDA	97
TABELA 14 - CONFIGURAÇÃO DA MALHA SEGUIDORA DE ERRO COM SENSOR CONVENCIONAL.....	98

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AP	<i>application process</i> (processo de aplicação)
CLP	controlador lógico programável
DCS	<i>distributed control system</i> (sistema de controle distribuído)
DD	<i>device description</i> (descrição de dispositivo)
DDE	<i>dynamic data exchange</i> (troca dinâmica de dados)
DDL	<i>device description language</i> (linguagem de descrição de dispositivo)
DIN	Instituto Alemão de Normalização
DLL	<i>data link layer</i> (camada de enlace)
DLSAP	<i>data link service access point</i> (ponto de acesso a camada de enlace)
DP-OD	<i>dynamic list of program invocation object dictionary</i> (lista dinâmica de <i>program invocation</i> do dicionário de objetos)
DV-OD	<i>dynamic list of variable lists object dictionary</i> (lista dinâmica de lista de variáveis do dicionário de objetos)
FAS	<i>fieldbus access sublayer</i> (sub camada de acesso)
FBAP	<i>function block application process</i> (processo de aplicação de blocos funcionais)
FCS	<i>fieldbus control system</i> (sistema de controle <i>fieldbus</i>)
FF	<i>Fieldbus Foundation</i>
FIP	Instituto Francês de Normalização
FMS	<i>fieldbus message layer</i> (camada de aplicação)
IEC	<i>International Electrotechnic Commission</i>
IEEE	<i>The Institute of Electrical and Electronics Engineers</i>
ISA	<i>International Society for Measurement and Control</i>
ISO/OSI	<i>International Standards Organization / Open System Interconnection</i>
LAS	<i>link active scheduler</i> (escalonador ativo de conexões)
LME	<i>layer management entity</i> (gerente de camada)
MAC	<i>medium access control</i> (controle de acesso ao meio)
MAU	<i>medium access unit</i> (unidade de acesso ao meio)
MIMO	<i>multiple input, multiple output</i>

NMA	<i>network management agent</i> (gerente de rede)
OD	dicionário de objetos
ODES	<i>object dictionary description</i> (descrição do dicionário de objetos)
OPC	<i>OLE for process control</i>
SDCD	sistema digital de controle distribuído
SISO	<i>single input, single output</i>
SMIB	<i>system management information base</i> (base de dados do gerente do sistema)
SMK	<i>system management kernel</i> (núcleo do gerente do sistema)
SMKP	<i>system management kernel protocol</i> (protocolo do núcleo do gerente do sistema)
S-OD	<i>static object dictionary</i> (dicionário de objetos estáticos)
ST-OD	<i>static types object dictionary</i> (dicionário de objetos de tipos estáticos)
VCR	<i>virtual communication relationship</i> (relação de comunicação virtual)
VFD	<i>virtual field device</i> (transmissor virtual)

LISTA DE SÍMBOLOS

λ	pólos de um sistema dinâmico
ξ	fator de amortecimento
ω_n	frequência natural

RESUMO

A evolução das técnicas de controle em sistemas distribuídos alcançou um nível onde os *loops* de comunicação analógica foram finalmente substituídos por protocolos de comunicação digital. Isto foi realizado através do desenvolvimento de uma nova tecnologia de comunicação de chão de fábrica conhecida como *fieldbus*. Entre os diferentes protocolos de *fieldbus*, está o *Fieldbus Foundation*.

O desenvolvimento do protocolo *Fieldbus Foundation* dá-se, inclusive, com o desenvolvimento dos denominados Blocos Funcionais, estruturas que viabilizam a implementação de diferentes técnicas de controle ao sistema, entre elas técnicas de controle moderno. Este trabalho apresenta um Bloco Funcional para o protocolo *Fieldbus Foundation*, que realiza a estimação de estados de um sistema dinâmico SISO de terceira ordem. O bloco desenvolvido foi utilizado no controle de temperatura em um sistema térmico.

ABSTRACT

The evolution of the control techniques in distributed systems reached a level where analogical control loops technique were finally substituted by digital communication protocols. That was accomplished through the development of a new technology of factory floor communication well-known as fieldbus. Among the different fieldbus protocols, there is Fieldbus Foundation.

Fieldbus Foundation evolution goes by the development of structures called Function Blocks that make the implementation of different control techniques possible, such as techniques of modern control. This work presents a Fieldbus Foundation Function Block that is a state estimator for a third order SISO dynamic system. The block is used in the temperature control of a termic system.

1. INTRODUÇÃO

A Automação da Manufatura e o Controle de Processos podem ser vistos como as duas grandes áreas de automação industrial. A primeira envolve teorias e sistemas ligados a Indústria de autopeças, eletro-eletrônicas, do setor têxtil, enfim, processos com linhas de montagem e de manufatura de uma maneira em geral. A segunda trata principalmente dos processos contínuos como os encontrados em indústrias petroquímicas, farmacêuticas, de bebidas, açúcar e álcool, de geração de energia, etc.

Nos últimos anos contudo, tem sido observado o rápido desenvolvimento de uma nova classe de sistemas de automação, voltados para a automação residencial, predial e hospitalar.

No que diz respeito aos sistemas industriais de instrumentação e controle de processos, a evolução tecnológica nos últimos 20 anos imprimiu fortes mudanças nos equipamentos e sistemas, sendo que a integração e a máxima disponibilização de dados passaram a ser primordiais ao lado da precisão e do desempenho.

Todavia, desde os transmissores pneumáticos analógicos de 3 a 15 psi até os modernos sistemas *fieldbus*, uma das questões chave para o sucesso de determinada tecnologia sempre foi a padronização. Padronização esta que consagrou a transmissão por *loop* de corrente em 4 a 20 mA, que apesar de tecnologicamente ultrapassada, ainda hoje tem forte presença no parque industrial brasileiro e no mundial.

Na década de 80 os primeiros sistemas atualmente conhecidos por *fieldbus* foram concebidos e testados mas somente na década de 90 a questão da padronização para o *fieldbus* começou a ser solucionada com o lançamento das primeiras normas

regulamentadas por institutos de normalização nacionais, entre estes pode-se citar o PROFIBUS pela DIN alemã, o FIP francês e o SP-50 pela ISA americana.

Atualmente existem alguns tipos de sistemas *fieldbus* para controle de processos, sendo que dois deles se consolidaram como padrões internacionais, contando com significativa base instalada e grande número de empresas fabricantes: o PROFIBUS e o *Foundation Fieldbus*. O PROFIBUS-PA (*Process Automation*) é o protocolo padronizado pela Comunidade Européia, no Brasil sua penetração é pequena e recente. O protocolo *Foundation Fieldbus* foi adotado como padrão pela IEC, ISA e recentemente pela ABNT sendo o Brasil é um dos países que mais se destaca na utilização e no desenvolvimento deste sistema, motivo este que justifica a escolha do *Foundation Fieldbus* como objeto de estudo desta pesquisa.

Com a substituição dos tradicionais sistemas DCS para os FCS (*Fieldbus Control System*), constatou-se que a distribuição das malhas de controle entre os equipamentos de campo introduziu uma série de problemas que até então não existiam nos controladores centralizados, e as soluções para tal questão foram resolvidas no *Foundation Fieldbus* em se tratando de sistemas e malhas de controle clássicas, porém as técnicas e algoritmos da moderna teoria de controle não foram devidamente integradas ao *Foundation Fieldbus*. Tais técnicas de controle, englobam entre outros, algoritmos de controle multivariável, adaptativos, preditivos e são exclusivas aos sistemas de processamento digital, ou seja, não podem ser implementadas em sistemas analógicos.

A razão pela qual não existe controle moderno em redes *Foundation Fieldbus*, é a inexistência de malhas distribuídas e de blocos funcionais no mercado que possam realizar os processamentos adequados.

Este trabalho **propõe**, como parte da solução para tal carência, um bloco funcional *Foundation Fieldbus* Estimador de Estados pleno de terceira ordem com uma entrada e uma saída, ou seja, para ser utilizado em sistemas SISO (*Single Input, Single Output*), visto que o estimador de estado é um elemento fundamental para a implementação de sistemas de controle modernos. O projeto do bloco, desenvolvimento e testes de campo do bloco e das malhas foram realizados no Laboratório de Eletrônica Embarcada do Departamento de Engenharia de Materiais, Mobilidade e Aeronáutica da Escola de Engenharia de São Carlos, em um forno,

planta instrumentada e controlada com o sistema *Foundation Fieldbus* de fabricação da Industria de equipamentos para automação SMAR.

O **objetivo** deste projeto é portanto, também, estudar as implicações teóricas e práticas inerentes a integração e a operação de algoritmos da moderna teoria de controle às estruturas e camadas normalizadas do protocolo *Foundation Fieldbus*, projetando um bloco funcional e sua malha de controle e implementando-os em um transmissor industrial *Fieldbus Foundation* certificado. A longo prazo prevê-se que a integração e o uso de tais algoritmos caracterizarão uma nova geração de controladores, na qual se utiliza a tecnologia de controle moderno por variáveis de estado em rede (controles multivariáveis, adaptativos, preditivos, entre outros) para solução dos problemas de controle mais complexos, onde somente tais técnicas de controle podem alcançar resultados da qualidade desejada. Tudo isto no sistema *Foundation Fieldbus*, dispondo assim, de todas as suas vantagens.

Complementarmente, são **propostas**, como outros exemplos de aplicação, cinco malhas de controle baseadas em realimentação de estados estimados.

Além desta introdução, está apresentada nesta dissertação uma revisão bibliográfica (Capítulo 2) sobre a evolução dos transmissores e dos sistemas de controle industriais, que é finalizada com a apresentação do protocolo *Foundation Fieldbus* e suas camadas e estruturas.

O capítulo 3 detalha o processo de aplicação dos blocos funcionais, dentro da camada de rede do usuário, pois dentre as estruturas do *Foundation Fieldbus*, tal processo de aplicação é a de maior importância para o contexto deste projeto.

Em seguida (Capítulo 4) está o desenvolvimento teórico sobre estimadores de estados, necessário para o entendimento do projeto do bloco funcional Estimador de Estados, que é apresentado no capítulo seguinte (Capítulo 5), onde são descritos os aspectos de implementação e de execução.

No Capítulo 6 são descritos os ensaios de campo e apresentados seus respectivos resultados. E, por fim, no Capítulo 7, as conclusões e discussões sobre o trabalho realizado, seus resultados e perspectivas, e sugestões de novos desenvolvimentos na área.

2. REVISÃO BIBLIOGRÁFICA

2.1. TRANSMISSÃO DE SINAIS NO CHÃO DE FÁBRICA

No campo da instrumentação e do controle de processos, o sensor de grandezas físicas é um dos elementos fundamentais assim como o controlador e o atuador. Em sistemas de controle automáticos, desde os primeiros sistemas mecânicos como o conhecido regulador de esferas de James Watt aos modernos sistemas digitais distribuídos, existe a necessidade da transmissão da grandeza física medida do sensor ao controlador e também do sinal de atuação do controlador ao atuador sem a participação de um operador, para que se possa fechar a malha de controle.

O desenvolvimento dos sensores industriais independentes foi realizado com a integração aos transdutores em questão, de unidades especiais capazes de transmitir variáveis de processo para um local remoto, resultando no que se conhece hoje por transmissores industriais.

Transmitindo grandezas físicas de maneira mecânica, pneumática, elétrica ou digital, tais instrumentos possibilitaram o desenvolvimento de sistemas de controle a distância e, desde então, este tipo de comunicação no chão de fábrica passa por um processo de evolução contínua, na qual se visa maximizar a qualidade e quantidade de informações trocadas entre sensor, controlador e atuador.

2.1.1. TRANSMISSÃO ANALÓGICA

O primeiro padrão de comunicação de chão de fábrica efetivamente difundido foi a transmissão analógica por sinal pneumático, modulado de 3 a 15 psi (20.7 a 103.4 Kpa). Os transmissores desta classe geram um sinal pneumático piloto proporcional a variável medida (p. ex. temperatura, pressão e vazão), relacionando a faixa 3 a 15 psi com 0% a 100% da faixa total de medição do sensor. Estes transmissores são conectados ponto a ponto com o controlador, onde se estabelece um *loop* de pressão com o sinal piloto. Os principais problemas encontrados neste tipo de comunicação é a relação da constante de tempo da transmissão do sinal em relação a do processo controlado, e a baixa imunidade do sinal as variações do ambiente.

Segundo SANTOS (1996), na década de 60 o avanço da eletrônica industrial possibilitou o surgimento dos transmissores eletrônicos, que geram sinais elétricos para representar a grandeza medida. Dentre as formas adotadas de transmissão de sinal elétrico, as mais comuns são modulação de tensão, corrente e deslocamento de fase e de frequência.

Gradativamente este padrão substituiu o pneumático, porém a transmissão eletrônica também apresenta problemas, como a atenuação do sinal na linha e a suscetibilidade a ruídos eletromagnéticos.

Em 1975, através da norma SP-50, a ISA (*International Society for Measurement and Control*) adotou como padrão industrial a comunicação por *loop* de corrente modulada em 4 a 20 mA, devido a vantagens apresentadas neste sistema. Este padrão foi amplamente difundido atualmente existe uma grande base de sistemas industriais controlados por *loop* de 4 a 20 mA. Segundo ZECH & SANTORI (1996), este é ainda hoje o único padrão amplamente aceito para a implementação de sistemas de controle de processos.

Os transmissores analógicos, tanto pneumáticos quanto eletrônicos, basicamente são compostos por três partes, completa FRANCO (1998):

- Transdutor: é o elemento que converte a grandeza física em um sinal de medida, geralmente elétrico.
- Condicionador: filtra o sinal do transdutor e gera um sinal na faixa normalizada adequada.

- Transmissor: envia o sinal condicionado para o controlador remoto.

Os sistemas de controle remoto da década de 70 eram compostos por controladores centrais ou estações de controle (EC). Estes controladores podem processar de uma a várias malhas de controle, conectando-se diretamente aos transmissores e aos atuadores, por *loop* de pressão ou corrente, além de serem também utilizados no supervisionamento da planta:

Figura 1 - Sistema de Controle Centralizado

2.1.2. TRANSMISSÃO DIGITAL

Na década de 70, o desenvolvimento no campo de microprocessadores viabilizou o seu uso em transmissores e em controladores industriais permitindo assim, entre outras coisas, o aparecimento de sistemas de comunicação digital no chão de fábrica. Tal forma de transmissão de dados tem algumas características superiores a analógica, como as citadas abaixo:

- Grande imunidade a ruídos externos, pois alterações no sinal causadas por ruídos, geralmente não causam alteração na informação codificada.
- Transmissão de dados sem distorção, ou seja, menor porcentagem de erro.
- São obtidas maiores taxas de transmissão.
- Permite conexões multi-ponto e multiplexagem de dados transmitidos em uma mesma linha.

- Possibilita a comunicação bidirecional.

A comunicação digital foi baseada fundamentalmente na codificação binária, onde os valores transmitidos podem ter somente dois valores, "0" ou "1", e são denominados *bits*. As formas básicas de transmissão de *bits* são o chaveamento de frequência (*frequency-shift keying*), chaveamento de amplitude (*amplitude-shift keying*) e chaveamento de fase (*phase-shift keying*). Deste três formas, o chaveamento de amplitude é o mais utilizado.

Os diferentes padrões de transmissão digital tem características que os tornam particularmente adequados as diversas aplicações, entre estas características pode-se citar a capacidade de sincronização, potência média de sinal, taxa de transmissão, sistemas de checagem de dados e adequação do sinal e dos circuitos a áreas classificadas.

2.2. SISTEMAS DIGITAIS DE CONTROLE DISTRIBUÍDO

Se por um lado a microeletrônica foi aplicada aos transmissores, por outro lado ela também foi de grande valia na evolução dos controladores centrais. Em 1975, iniciou-se o desenvolvimento e a implantação dos chamados Sistemas Digitais de Controle Distribuídos (SDCD) em substituição aos sistemas de controle centralizados.

Nos sistemas SDCCDs, os transmissores são conectados diretamente as estações de controle digitais locais (ECL), também por loop de corrente, e os controladores locais por sua vez, comunicam-se com as estações de controle através de redes de comunicação digital.

A principal proposta dos SDCCD, segundo PRZYBYLSKI (1996), é descarregar o computador central distribuindo as funções de controle básicas, que necessitam de processamento rápido e repetitivo, para um controladores distribuídos e conectados por uma rede. Como resultado obteve-se maior confiabilidade, maior capacidade de atuação corretiva no caso de falha e execução das malhas de controle de modo mais rápido.

Figura 2 - Sistema Digital de Controle Distribuído (SDCD)

2.3. TRANSMISSORES INTELIGENTES

Além da possibilidade de implementação de comunicação digital, o uso de microprocessadores nos transmissores tornou viável uma nova gama de funcionalidades a estes. Os primeiros transmissores microprocessados foram lançados em meados de 1983 e são conhecidos por *smart transmitters* ou transmissores inteligentes.

Um transmissor inteligente típico possui uma unidade de processamento e uma interface de comunicação, além do transdutor ou dos transdutores e da unidade condicionadora de sinais. Nos *smart transmitters* o sinal é adquirido pelo transdutor e convertido por um conversor A/D, só então o valor digital é processado e posteriormente transmitido, diretamente na forma digital ou então convertido por um conversor D/A e enviado na forma analógica.

Entre as funcionalidades de um transmissor inteligente, pode-se destacar:

- possibilidade de aquisição, linearização e compensação de grandezas secundárias (p. ex. compensação de temperatura em uma medida de pressão) além da grandeza primária.

- combinação de grandezas primárias e secundárias para o cálculo de grandezas derivadas (p. ex. aquisição de pressão diferencial e temperatura para a transmissão de concentração)
- capacidade de registro histórico de eventos, tais como mal funcionamento do sensor e situações de alarme, muito úteis para diagnóstico.
- integração de bases de dados aos transmissores, com informação de processo e de manejo da planta.
- possibilidade de adaptação de um transmissor a diferentes tipo de transdutor, apenas com mudanças no software de controle (p. ex. um transmissor de temperatura compatível com termopares e com sensores ôhmicos).
- configuração de faixa de medida, *offset* e *cutoff* e outras características de forma remota.
- conversão das grandezas em diferentes unidades de engenharia.

A evolução dos transmissores inteligentes se dá, entre outros fatores, com o aumento da capacidade de memória e de processamento interno. Esta evolução permite a classificação dos transmissores inteligentes em duas gerações:

2.3.1. PRIMEIRA GERAÇÃO

A primeira geração teve como característica a sobreposição do sinal de comunicação digital, se este for implementado, ao sinal de 4 a 20 mA, característica esta que permitiu a perfeita compatibilidade destes transmissores aos tradicionais sistemas de controle por *loop* de corrente. Apesar de possuírem algumas vantagens do ambiente digital, como aumento da precisão e menor susceptibilidade a variação de temperatura de trabalho, os transmissores da primeira geração eram essencialmente analógicos, pois a transmissão assim o era.

Entre estes primeiros protocolos de comunicação digital sobreposta destaca-se o HART (*Highway Address Remote Transducer*) baseado na codificação binária

por chamamento de frequência . No final dos anos 80 a grande difusão e aceitação do HART tornou este um padrão "de facto".

Segundo GARRONE (1997), o grande fator limitante do HART é a baixa velocidade de transmissão de 1200 bits/s, o que torna o protocolo muitas vezes indevido para o loop de controle, mas bastante utilizado como ferramenta de manutenção e diagnóstico. Uma pesquisa realizada pela ARC Advisory Group em 1999 concluiu que o HART provavelmente registrará crescimento até o ano 2001, considerando a base instalada, só então será gradativamente substituído por padrões de comunicação digital mais recentes (possivelmente PROFIBUS-PA e Fieldbus Foundation).

2.3.2. SEGUNDA GERAÇÃO

A segunda e atual geração de transmissores inteligentes se caracteriza pela eliminação definitiva da transmissão analógica, são portanto puramente digitais. Com a eliminação da grande maioria dos componentes analógicos, estes transmissores possuem todas as vantagens do ambiente e da transmissão digital citadas anteriormente.

A precisão da malha de controle foi dobrada com a eliminação das conversões D/A e A/D necessárias para a transmissão de dados, conforme registrou BROWN na tabela 1: (Isa conferences Advances in Instrumentation 89 parte 4 p.1625)

Transmissor	Modo de Transmissão	Erro (%) Transmissão	erro (%) malha
analógico	analógico	0,150	0,225
digital	Analógico	0,075	0,150
digital	digital	0,075	0,075

Tabela 1 - Precisão comparativa dos ambientes analógico e digital

A queda do preço de componentes eletrônicos registrada na última década e a diminuição do custo de manutenção proporcionada pelos avanços nos transmissores

inteligentes foram fatores decisivos na disseminação destes instrumentos principalmente os da segunda geração.

Outra vantagem importante foi observada como consequência da gradativa migração de parte do processamento de sinais para os transmissores, aliada com a comunicação digital bidirecional transmissor-controlador: a distribuição das funções de controle e assim um aumento na capacidade e no desempenho dos controladores e dos SDCDs.

Apesar de toda a sofisticação dos transmissores e dos sistemas de controle, como constata WREYFORD (1996) a configuração, operação e documentação dos sistemas SDCD é relativamente complexa. Além disto, não houve um esforço de normalização que unificasse as diversas soluções existentes. Os usuários portanto ficam restritos ao fabricante que forneceu o sistema, não podendo integrar novos transmissores ou controladores caso estes não sejam do mesmo fabricante, fato que é possível em sistemas que utilizam somente comunicação 4 a 20mA.

Esta situação desfavorável aos usuários e fabricantes, aliada ao contínuo avanço dos microprocessadores que tornava os transmissores industriais cada vez mais compactos e com mais recursos, e a necessidade crescente por sistemas de controle mais rápidos e completos motivou grupos de empresas e usuários, a buscarem a criação de padrões e normas para o projeto de sistemas de comunicação no ambiente fabril.

Neste esforço de normalização foi considerada a necessidade de (além do controle de processos) monitorar e diagnosticar toda a planta assim como de tornar o sistema de controle flexível - evolução natural dos sistemas de automação industrial. A partir desta necessidade, indústrias e organismos de normalização nacionais e internacionais iniciaram o desenvolvimento de um novo conceito em redes de comunicação de chão de fábrica. Estas pesquisas criaram as bases do que se conhece atualmente por *fieldbus*.

2.4. MODELO DE REFERÊNCIA ISO/OSI

O modelo de referência ISO/OSI (*International Standards Organization / Open System Interconnection*) de 7 camadas de rede é atualmente uma das bases para o desenvolvimento redes de computadores. Apresentada entre o final dos anos 70 e o início dos 80, a norma internacional ISO 7498 propõe uma série de regras para a comunicação entre sistemas. O modelo é basicamente constituído por um conjunto de camadas independentes e hierarquicamente sobrepostas, onde cada camada é responsável por uma classe de serviços e se comunica com as camadas vizinhas através da prestação destes serviços. O modelo ISO/OSI não é por si só um protocolo de rede, ele apenas propõe um modelo que visa reduzir a complexidade do sistema total, segundo o princípio de substituir o estudo do todo pelo estudo das partes. As 7 camadas se comunicam de maneira transparente e horizontalmente entre duas máquinas camada a camada, embora localmente a comunicação se faça de maneira vertical.

Um dos requisitos do modelo é que as sete camadas propostas sejam independentes e possuam limites bem definidos para que se possa minimizar o volume de informações trocadas entre elas. As sete camadas: 1- Camada Física, 2- Camada de Enlace, 3- Camada de Rede, 4- Camada de Transporte, 5- Camada de Sessão, 6- Camada de Apresentação e 7- Camada de Aplicação.

Uma explicação das funções de cada camada é apresentada por SANTOS (1996), NAZAR (1993) e SILVA (1996).

A figura seguinte representa o mecanismo de comunicação vertical entre as camadas vizinhas de um mesmo ponto de rede e o mecanismo de comunicação horizontal entre as camadas semelhantes de dois pontos de rede distintos:

Figura 3 - Modelo de ISO/OSI de sete camadas

2.5. O MODELO ISO/OSI REDUZIDO DE TRÊS CAMADAS

O modelo reduzido ISO/OSI serviu de referência para o desenvolvimento de diferentes protocolos de redes industriais *fieldbus*. Para se adaptar as necessidades de um *fieldbus*, a estrutura de 7 camadas foi reduzida para 3 camadas, pois os serviços das camadas 3 a 6 não são necessários em um *fieldbus* ou em qualquer rede local, serviços como por exemplo os de correio eletrônico, mapeamento de servidores,

roteamento de mensagens e outros. A eliminação destas camadas torna também o processo de comunicação mais rápido.

O modelo ISO/OSI reduzido de 3 camadas foi especificado da seguinte maneira: Camada 1 - Física, Camada 2 - de Enlace e Camada 7 - de Aplicação.

Figura 4 - Modelo ISO/OSI reduzido de três camadas

2.6. PROJETO PROWAY

O desenvolvimento dos protocolos para ambientes de chão de fábrica iniciou-se em meados dos anos 70. Em 1976 o grupo de estudos IEC TC 65C/WG6, em parceria com a ISA SP-72 iniciou os trabalhos de um projeto denominado PROWAY. Estes trabalhos resultaram nos protocolos PROWAY-A e PROWAY-B. Em 1980 a IEEE iniciou o projeto 802 e, futuramente em 1984, em parceria com membros da ISA, foi definido o IEEE 802.4 e o ISA SP-72.01 em 1985. Este último ficou conhecido como PROWAY-C. O projeto PROWAY apresentou algumas deficiências graves quanto a tempos de resposta e confiabilidade, embora fosse um protocolo adaptado para os níveis de rede administrativa e de sistema, porém foi importante pois lançou as bases para o desenvolvimento de futuros projetos.

2.7. PROJETO MAP

No início de 1980 a General Motors Co. propôs a arquitetura MAP (*Manufacturing Automation Protocol*) como padrão para comunicação no ambiente industrial. Projetado para atender as necessidades do CIM (*Computer Aided Manufacturing*), tais como controle da manufatura, gerenciamento e controle de material, este protocolo visa um chão de fábrica automatizado com CLPs e computadores de área.

As camadas do MAP seguiram as especificações descritas nas seguintes normas:

- 1) Camada Física: IEEE 802.4.
- 2) Camada de Enlace: composta pelas subcamadas LLC definida pela IEEE802.2 e pela MAC definida em IEEE802.4.
- 3) Camada de Rede: padrão ISO 8473 (protocolo semelhante ao IP).
- 4) Camada de Transporte: ISO 8072.
- 5) Camada de Sessão: ISO 8326 e 8327.
- 6) Camada de Apresentação: ISO 8822 e 8823.
- 7) Camada de Aplicação: foi adotado um protocolo genérico OSI denominado MMS (*Manufacturing Message Specification*), adequado a mensagens curtas com requisitos de respostas rápidas.

Esta arquitetura de 7 camadas se mostra bastante abrangente mas o tempo de resposta fica prejudicado. Desta limitação temporal do MAP original, surgiram alguns protocolos derivados que tem um maior desempenho porem são mais restritos quanto a aplicação do que o MAP original (Full-MAP). Estes protocolos são o EPA-MAP (*Enhanced Performance Architecture MAP*) e o Mini-MAP. Vide figuras 5 e 6:

Figura 5 - Estrutura MAP e MAP/EPA

Figura 6 - Estrutura Mini-MAP

As estruturas Mini-MAP e EPA-MAP não apresentaram entretanto a eficiência e o baixo custo necessários para a utilização no nível de chão de fábrica.

Como consequência desta lacuna, e do contínuo desenvolvimento dos transmissores inteligentes, teve início o desenvolvimento de um tipo de rede que se adequasse totalmente e somente ao ambiente de chão de fábrica - o *fieldbus* ou barramento de campo.

2.8. BARRAMENTO DE CAMPO

O *fieldbus* é definido como o elo de comunicação serial entre os dispositivos de automação primários localizados junto ao processo e o nível de controle imediatamente superior na sala de controle com mais capacidade de processamento.

Figura 7 - Barramento de Campo

Ao contrário das iniciativas anteriores onde protocolos foram projetados para ter ampla atuação e depois modificados a fim de obter maior desempenho nas aplicações críticas, os *fieldbuses* já foram concebidos para o nível de controladores e transmissores.

Organismos de padronização pioneiros nos trabalhos com *fieldbus*, como o IEEE, o IEC e a ISA, iniciaram os trabalhos por elaborar um conjunto de requisitos iniciais que o novo protocolo deveria suportar, no que se diz respeito a tempos de resposta, topologia física, redundância, confiabilidade, número de dispositivos por ramo de rede, método de acesso ao meio físico, alimentação dos transmissores, comprimento do barramento e métodos de configuração do sistema.

As vantagens que devem ser observadas neste tipo de barramento são:

- possibilidade da migração do processamento do controlador para os transmissores,
- redução do custo de cablagem,
- maior facilidade e economia na manutenção e na instalação, devido a possibilidade de configuração remota dos transmissores,
- maior desempenho na comunicação, totalmente digital.
- disponibilização rápida de informações em qualquer ponto da rede, além de uma maior gama de informações disponíveis,
- detecção automática de falhas nos transmissores e no barramento,
- expansão e evolução flexíveis.
- diminuição do tempo dos *loops* de controle
- diminuição do tamanho e complexidade das salas de controle.

2.9. O IMPACTO DA INTRODUÇÃO DO FIELDBUS

A redução do custo de cablagem do *fieldbus* sobre a tecnologia SDCCD, foi avaliada por MANGANO & DUMDIE (1998) em 40% a 60%, resultando, juntamente com outros fatores, em uma redução do custo de projeto (hardware, software, engenharia e instalação) de 55% sobre o SDCCD. Tal economia em cablagem se deu principalmente pela substituição da topologia ponto a ponto pela topologia de barramento multiponto.

Figura 8 - Comparação da estrutura física do DCS com o FCS

O impacto da introdução dos sistemas baseados em *fieldbus* no cenário atual de controle de processos, a integração dos SDCDs nos *fieldbuses* ou até mesmo a substituição daqueles tem sido objeto de estudos nos últimos anos.

No que diz respeito a integração dos sistemas SDCD em uma rede *fieldbus*, HODSON (1996) considera três níveis de integração:

- **ELEMENTAR:** apenas a variável primária é passada do *fieldbus* para o SDCD e vice versa. O SDCD não explora as funcionalidades do *fieldbus* como alarmes, status e Descrição de Dispositivos.
- **HETEROGÊNEA:** existe um *gateway* que mapeia algumas implementações do *fieldbus* no SDCD, existem entretanto funcionalidades do *fieldbus* que não são exploradas pelo SDCD.
- **HOMOGÊNEA:** a integração é dita homogênea se o SDCD incorpora 100% do protocolo *fieldbus*. Neste caso todas as funcionalidades do *fieldbus* podem ser encontradas no SDCD, uma vez que o protocolo *fieldbus* possui estruturas que possibilitam o acesso completo de informações do SDCD em tempo real aos transmissores sem prejudicar o processamento de controle distribuído nestes, completa BLEVINS et alli (1996).

Baseado nesta classificação, o mesmo autor propõe ainda um índice de compatibilidade de integração SDCCD-*Fieldbus* denominado FFCI, que varia de 0 a 10, para uma integração perfeita.

A integração física de uma rede *fieldbus* a um sistema SDCCD pode ser realizada de três modos de acordo com BLEVINS et alli 2 (1996) tendo como objetivo o mínimo impacto no sistema existente:

- através de um *gateway* SDCCD / barramento *fieldbus*
- através de um cartão *fieldbus* no controlador SDCCD ou
- através de um cartão *gateway* diretamente no controlador.

Como defende MORI et alli (1996), a migração de um sistema baseado em SDCCD para o *fieldbus* passa por três estágios. Em um estágio de introdução do *fieldbus*, apenas processamento I/O dos transmissores *fieldbus* são utilizados, sendo que as funções de controle ficam a cargo do SDCCD. Num segundo estágio, elementos de controle secundários passam a ser processados nos transmissores *fieldbus*, como mostra a figura 9, enquanto as malhas de controle primários continuam centralizadas no SDCCD pois estas tem relações com várias malhas e processamento de funções sequenciais que são realizadas mais eficientemente num controlador centralizado do que em transmissores distribuídos. O último estágio da migração é caracterizado pela aceitação dos elementos de controle nos transmissores *fieldbus*. Nesta fase todos as malhas de controle são distribuídas entre os transmissores, e o SDCCD fica encarregado apenas de funções críticas de máxima confiabilidade. A tendência portanto é que o SDCCD realize o controle a um nível mais alto, ou seja, entre-áreas e para a manutenção da otimização do sistema concorda HODSON (1996).

Figura 9 - Distribuição da inteligência entre os transmissores de campo

Como desvantagens do *fieldbus* tem-se a concentração da comunicação em poucos canais de rede, o que compromete o desempenho do tráfego de informação e torna todo o sistema mais suscetível a um rompimento do barramento, a concentração de malhas por canal deve portanto ser estudada caso a caso conforme comenta PRZYBYLSKI (1996).

Várias iniciativas de normalização de *fieldbus* foram observadas e existe atualmente um número considerável de tipos de *fieldbus* que atendem aos requisitos de diferentes tipo de chão de fábrica e também de outros ambientes, como por exemplo ambientes prediais e automotivos.

A padronização internacional do barramento de campo, tornará fácil a elaboração de interfaces para os diversos tipos de aplicações industriais, baseados em um mesmo protocolo, além de tornar o mercado de sistemas aberto, ou seja, garantir a interoperabilidade de equipamentos de diferentes fabricantes.

Este objetivo porém, é difícil de ser alcançado, pois os processos industriais são muito diversos e assim tem de ser os sistemas de controle e de informação, além do que, estão envolvidos muitos interesses econômicos da parte de fabricantes de instrumentos e de sistemas.

No cenário atual entretanto, algumas normas se destacam e possuem grande aceitação no mercado.

A nível de sistemas para processo industriais contínuos, os protocolos de maior destaque e base instalada atualmente são o PROFIBUS (*Process for Fieldbus*)

alemão e o *Fieldbus* Foundation. A seguir serão apresentadas algumas características destes protocolos.

2.10. O PROFIBUS

O PROFIBUS foi desenvolvido a partir de 1987 na Alemanha, por um conjunto de 15 empresas e 5 institutos de pesquisa financiados pelo governo alemão, por iniciativa das indústrias Bosch, Klockner-Moeller e Siemens. Como resultado destes trabalhos, foi publicada em 1991 a norma DIN 19254

As especificações do PROFIBUS baseiam-se no consagrado modelo reduzido ISO/OSI de 3 camadas, e nas normas internacionais IEC-955 para o gerenciamento das camadas, IEEE 802.2 para a LLI (*Lower Layer Interface*) e o ISO IS 9506 para a camada de aplicação FMS (*Fieldbus Message Specification*).

Desde 1996 o PROFIBUS é adotado como parte da norma europeia EN 50170 e atende a uma vasta gama de aplicação pois o protocolo é composto por 3 versões totalmente compatíveis: o PROFIBUS-DP (*Distributed Control*), otimizado para comunicação de alta velocidade e especialmente projetado para sistemas de controle discreto e de I/O distribuído sendo que das três versões é a mais difundida; o PROFIBUS-FMS, dedicado a comunicação no nível de células de manufatura para tanto, tem estrutura mais complexa e possui serviços mais elaborados e não determinísticos; e o PROFIBUS-PA (*Process Automation*) projetado para controle de processos contínuos e adaptado a áreas classificadas.

O acesso ao meio físico do PROFIBUS segue um modelo híbrido, onde dois tipos de comunicação são permitidas: a comunicação entre estações mestre pelo método *token-passing* e a comunicação mestre-escravo do tipo cíclica determinística, para garantir transferência de dados de forma mais simples e rápida possível. A topologia física é do tipo barramento linear com possibilidade de ramos secundários, garantindo flexibilidade de instalação da rede. Uma descrição mais detalhada sobre cada versão PROFIBUS será apresentada abaixo:

2.10.1. PROFIBUS DP

Como mencionado anteriormente, esta é uma versão da norma destinada a aplicações a nível de dispositivos de automação e controle, dispositivos de I/O e outros sem propósitos de controle de processos contínuos. A versão DP se destaca pela alta taxa de comunicação (9,6 Kbits/s a 12 Mbits/s) e baixo tempo de resposta. Permite operações de sincronização de operações de I/O, diagnósticos de equipamentos, implementação de alarmes além de operações de configuração. Os dispositivos DP (e também os PA) não possuem a camada FMS, e a comunicação do processo aplicativo com a estrutura de rede é feita através do mapeamento direto na camada de enlace (DDL – *Direct Data Link Mapper*), que oferece serviços mais simples que os disponíveis através da FMS, porém com maior facilidade de operação e processamento e portanto menor tempo de resposta. A figura 10 apresenta a estrutura das camadas de rede de dispositivos DP e PA com DDLM e a estrutura correspondente de um dispositivo no padrão FMS:

Figura 10 - Estruturas das camadas de rede da FMS, DP e PA

Quanto a comunicação física, a DP utiliza o padrão serial RS-485 com barramento de par trançado ou então fibra ótica, isto significa que a DP é totalmente compatível com a FMS, podendo até coexistirem em um mesmo barramento. Esta característica é alcançada uma vez que todas as 3 versões da PROFIBUS são

compatíveis a nível de protocolo, e utilizam o mesmo método de acesso ao meio (MAC – *Medium Access Control*), regras que determinam quando um dispositivo pode enviar uma mensagem na rede, não apresentando portanto problemas de conflito de comunicação. O MAC no PROFIBUS como foi citado é do tipo acesso híbrido ao meio: por *token* entre *masters* e comunicação *master - slave* como mostra a figura 11:

Figura 11 - Método híbrido de acesso ao meio físico no PROFIBUS

O *profile* (classificação básica prevista por norma para as classes de instrumentos disponíveis e suas características classes de parâmetros e variáveis, ferramenta para garantir interoperabilidade entre dispositivos de diferentes fabricantes) para equipamentos DP prevê as seguintes classes:

- NC/RC: manipuladores ou robôs,
- *Encoder*: *encoders* angulares, lineares e absolutos (*resolvers*),
- *Variable*: driver para controle de velocidade e posicionamento,
- HMI: sistemas de interface com operador e monitoramento.

2.10.2. PROFIBUS PA

Versão da norma destinada a aplicações em controle de processos, sua estrutura é direcionada para o tratamento de variáveis do tipo temperatura, pressão, vazão e nível. A PA pode ser aplicada em áreas de segurança intrínseca como por exemplo em indústrias com atmosferas explosivas. O meio físico segue a norma IEC 1158-2 no padrão H1 com codificação *manchester* em par trançado a 31,25 Kbits/s. Os dispositivos neste caso podem ser alimentados diretamente do barramento, observando-se a limitação de potência em áreas classificadas.

Por ter a camada física diferente da DP e da FMS, a comunicação com estas deve utilizar apenas um *segment coupler* RS-485 – IEC 1158-2, pois como foi dito, as 3 classificações da norma utilizam o mesmo MAC.

A estrutura do protocolo PA é semelhante ao DP no que diz respeito a ausência da FMS e a adoção do DDML, porém a PA tem funções de tratamento de comportamento e *status* dos dispositivos.

A PA suporta a nível de aplicação o modelo de Function Blocks, com Dicionário de Objetos e Descrição de Dispositivo (*Device Description - DD*). O *Profile* da PA prevê as seguintes classes de instrumentos, definindo suas variáveis, estruturas, *Function Blocks* e parâmetros:

- *Measuring transmitters*: transmissores de pressão, nível, temperatura e vazão,
- *Digital Input, Digital Output*: módulos de I/O digitais,
- *Analog Input, Analog Output*: módulos de I/O analógicos,
- *Valves*: válvulas,
- *Positioners*: posicionadores em geral.

2.10.3. PROFIBUS FMS

A Profibus FMS foi elaborada com o propósito de ser uma rede de aplicação a nível de células de fabricação, ou seja, uma nível mais alto que a PA e a DP e sua aplicação portanto é mais geral, com serviços mais complexos e poderosos.

Um dos motivos de tal flexibilidade é porque na versão FMS, ao invés da DDLM, existem a FMS, camada de aplicação do modelo ISO/OSI com a sub-camada LLI (*Lower Layer Interface*) e todas as estruturas a estas ligadas. Esta camada oferece serviços de comunicação que não são suportados na DDLM. A figura 12 mostra a estrutura detalhada do protocolo FMS com todas as camadas:

Figura 12 - Alocação das camadas no modelo PROFIBUS FMS

Existe a possibilidade de se ter as versões PA e FMS implementadas em um mesmo dispositivo, denominado *combi device*. Neste caso a FDL é capaz de acionar funções diretamente por DDML ou a FMS em função do destino da mensagem.

O *profile* para a FMS prevê os seguintes tipos de dispositivos:

- *Communication between Controllers*: para ser usado em comunicação de controladores como PLCs, PCs e outros,
- *Profile for building automation*: exclusivo para aplicações em automação predial,
- *Low voltage switch gear*: especifica aplicações para chaves de baixa voltagem utilizando os serviços da FMS

2.11. O FIELDBUS FOUNDATION

O protocolo *Fieldbus Foundation* foi proposto inicialmente em 1994 por uma fundação internacional de mesmo nome sediada nos Estados Unidos composta pela união de duas organizações de *fieldbus* existentes, a ISP (*Interoperable System Protocol*) e uma divisão da WorldFIP francesa, mais um grupo de 85 empresas. Aliados aos organismos internacionais de normalização IEC e ISA, a fundação propõe uma arquitetura também baseada no modelo ISO/OSI reduzido de 3 camadas. O protocolo da camada física é o proposto atualmente na norma IEC 61158-2. A camada de Enlace baseia-se em um subconjunto da norma IEC FDIS 61158-3 e 4 e a camada de Aplicação na norma IEC FDIS 61158-5 e 6.

Existe definida no Fieldbus Foundation uma camada superior a camada de aplicação denominada camada de usuário, a *User Layer*. Nesta camada estão definidas estruturas que realizam funções do controle e supervisão da planta em si, configuradas em uma arquitetura que muito se assemelha àquela encontrada no PROFIBUS PA.

Baseada nas normas IEC 65C e ISO 1131-3, a camada de usuário do Fieldbus Foundation possui entre suas estruturas, algumas derivadas do PROFIBUS e do HART.

O protocolo Fieldbus Foundation, ao contrário do PROFIBUS, não possui versões diferentes e foi concebido para a aplicação em controle de processos contínuos. A estrutura das suas camadas, presentes em todos os transmissores, está representada na figura 13:

Figura 13 - Estrutura das camadas de rede da Fieldbus Foundation

Referência: FIELDBUS FOUNDATION (1998). *FF 800*

Estas camadas podem ser representadas com enfoque na estrutura de operação no transmissor, de um ponto de vista computacional como na figura 14:

Figura 14 - Operação das estruturas de rede em um transmissor industrial Foundation

As especificações *foundation* não estão finalizadas, fabricantes e usuários portanto atualmente trabalham para definir e aprovar os capítulos inacabados da

norma. Por outro lado, é natural que mesmo as especificações que já foram aprovadas passem por um contínuo processo de evolução, observa-se que periodicamente novas revisões e novos capítulos são publicados pela *Foundation*, na medida em que os requisitos dos sistemas de automação vão se tornando cada vez mais complexos, e que os fabricantes e usuários vão acumulando experiência na operação e instalação de sistemas.

Atualmente existem vários softwares configuradores e de supervisão para sistemas *fieldbus foundation*. Estes softwares via de regra utilizam o padrão de comunicação de interface Windows denominado OPC (*OLE for Process Control*). FAYAD (1996) observa que a fácil transferência de dados entre aplicativos Windows através de DDE (*Dynamic Data Exchange*) torna informações do *fieldbus* acessíveis a redes de comunicação corporativas dentro ou fora da fábrica, ou seja, o computador realiza a função de *bridge*.

Uma descrição de cada estrutura que compõe o protocolo será dada a seguir, com o propósito de se entender o funcionamento geral do sistema.

2.11.1. TOPOLOGIA DE REDE

A norma prevê uma topologia de rede bastante aberta, sendo que o usuário tem liberdade para definir sua estrutura. Um ramo básico de rede (*link*) tem que ter no mínimo um dispositivo (*device*) do tipo *Link Master*, ou seja, um *device* cuja camada de enlace pode se tornar a LAS - *Link Active Scheduler*, ou o árbitro da rede, aquele que envia mensagens de *token* e requisições de dados aos outros *devices*. O LAS organiza também a *live list* que é uma lista atualizada de todos os *devices* presentes naquele *link* da rede. Pode-se ter dois ou mais *link masters* em um *link*, sendo que neste caso apenas um terá o LAS, e os outros serão mestre *backup*, *devices* que poderão se tornar o LAS caso o mestre ativo sai da rede ou apresente alguma falha, este é um dos mecanismos de redundância da rede. Vide figura 15:

Figura 15 - Topologia Single Link

Outro tipo de topologia é a denominada *multi-link*, neste caso vários links estão em comunicação, e a "ponte" entre elas é um *device* cuja camada de enlace é do tipo *bridge*. Este *device* tem vários MAUs ou *Medium Attachment Units* que são as portas que realizam as conexões ao meio físico, no caso o par traçado.

Um *device bridge* tem uma porta para o *link upstream* que é o link primário onde aquela *bridge* é configurada como *device* simples e no máximo 4 portas para 4 *links* secundários ou ramificações, para os quais a *bridge* tem que ser a *Link Master*.

Uma *bridge* tem algumas funções especiais para estabelecer a conexão entre os diferentes *link*. Entre as funções principais temos: encaminhamento de mensagens entre *devices* de *links* distintos, republicação de mensagens para os links secundários e sincronização da temporização da camada de enlace e da aplicação entre os *links*.

Neste caso a topologia é do tipo árvore.

O usuário no caso deve estudar a melhor maneira de configurar a topologia: ligando os *devices* que estão participando de uma mesma malha de controle em um mesmo *link* e assim aumentando o desempenho do controle mas perdendo em cablagem ou então ligando os *devices* de maneira a otimizar a cablagem mas assim poderá ter de utilizar várias *bridges* e conseqüentemente perder em desempenho de controle, e em segurança.

Figura 16 - Topologia Multilink

2.11.2. PROCESSO DE APLICAÇÃO (AP)

Um processo de aplicação é definido como as informações mais os componentes de processamento de uma aplicação distribuída que pode ser acessada ou endereçada em uma rede. A norma FF prevê dois tipos de aplicação: as aplicações determinísticas ou *time critical* e as não determinísticas. Estas aplicações estão definidas nos *devices* como um conjunto de funções denominado *Application Process* (AP). Pode-se ter em um *device* mais de uma AP. A parte visível de uma AP para a rede é composta por objetos, que podem ser simples ou compostos, estes últimos são um conjunto de objetos simples, como um exemplo de um objeto composto temos um *Function Block*. Os objetos de uma AP estão listados em um estrutura denominada *AP Directory*, que contem referências para todos os seus objetos visíveis pela rede com seus respectivos índices no Dicionário de Objetos (OD), além de um cabeçalho definindo sua estrutura. No caso de um objeto composto, a primeira linha deste objeto na *AP directory* possui um cabeçalho definindo a sua estrutura. A figura 17 ilustra estas relações:

Figura 17 - Visualização do Process Application pela rede

Referência: FIELDBUS FOUNDATION (1998). *FF 800*

2.11.3. FUNCTION BLOCKS AP

Os *Function blocks* APs (FBAP) são APs construídas de *function blocks*. Os *functions blocks* são estruturas que realizam as funções básicas da malha de controle, como algoritmos de compensação, aquisição e atuação e são realizados repetitivamente fechando assim as malhas de controle da aplicação.

Os *functions blocks* possuem parâmetros de configuração, de entrada e de saída além do algoritmo em si e são identificados e acessados nos *devices* por um nome ou *tag* exclusivo, e estão integrados aos *devices*.

Dependendo das características do bloco, este pode ser definido com *transducer*, *resource* ou *function block*.

Um *resource block* é utilizado para especificar as características gerais do *device* onde este se encontra, assim só é permitido um *resource* por *device* e como é um bloco de especificações, não possui parâmetros de entrada ou de saída.

O bloco *transducer* serve para desacoplar os outros blocos de operações específicas de I/O nos diferentes sensores e atuadores, é ele quem opera o hardware do dispositivo físico.

O *function block* em si é o que realiza as funções algébricas e algoritmos de controle, possui parâmetros de configuração e de entrada e saída. Sua execução é disparada repetidamente em intervalos de tempo fixo, e também são realizados ciclicamente pedidos para a publicação dos seus parâmetros de saída.

A *foundation* normalizou um conjunto de *function blocks* básicos e outro de *function blocks* avançados, que realizam as funções dos elementos básicos de uma malha de controle. A tabela seguinte mostra lista todos os blocos normalizados pela Foundation:

Conjunto	Bloco Funcional	Função
Básico	AI - Entrada analógica	Aquisição analógica
Básico	DI - Entrada discreta	Aquisição digital
Básico	AO - Saída analógica	Atuação analógica
Básico	DO - Saída discreta	Atuação digital
Básico	ML - Carregador manual	Saída definida pelo usuário
Básico	B/G - Bias / Ganho	Adiciona bias e amplifica
Básico	CS – Seletor de controle	
Básico	PD - Controlador proporcional derivativo	Controlador PD
Básico	PID - Controlador proporcional integral derivativo	Controlador PID
Básico	R - Controle de relação	Estação de relação
Avançado	DC - Controle de dispositivo	Controle de referência de dispositivos discretos
Avançado	SPLT - Divisor de saída	Permite controlar dois dispositivos com uma só referência
Avançado	CHAR - Caracterizador de sinais	Geração de sinal baseado em tabela
Avançado	LLAG - Lead lag	Compensador avanço – atraso
Avançado	Deadtime	Tempo morto
Avançado	INTG - Integrador (Totalizador)	Integrador / Totalizador
Avançado	SPG - Gerador de rampa de referência	Gerador de rampa de sinal analógico
Avançado	ISEL - Seletor de entradas	Seletor de canais
Avançado	ARTH - Aritmético	Cálculos aritméticos
Avançado	TMR - Timer	Temporizador
Avançado	AALM - Alarme analógico	Gerador de alarme
Múltiplos I/O	MDI - Entrada digitais múltiplas	Para PLCs
Múltiplos I/O	MDO - Saídas digitais múltiplas	Para PLCs

Múltiplos I/O	MAI - Entradas analógicas múltiplas	Para PLCs
Múltiplos I/O	MAO - Saídas analógicas múltiplas	Para PLCs

Tabela 2 - Blocos funcionais padrão

Estima-se que de 60% a 80% das malhas de controle passarão para os transmissores, e com a utilização dos blocos avançados acredita BLEVINS et alli (1996), este número cresce ainda mais. O desenvolvimento de novos blocos todavia, por decisão da *foundation*, é aberto aos usuários e fabricantes.

Uma malha de controle com compensador PID é configurada em blocos funcionais utilizando-se um bloco de entrada analógica (AI), um bloco controlador PID (PID) e um bloco de saída analógica (AO) distribuídos entre dois transmissores, de acordo com a figura 18:

Figura 18 - Malha de controle com PID

Uma ferramenta de construção de novos blocos foi proposta em 1997 por KALOGERAS et alli (1997), tal ferramenta além de construir o algoritmo do bloco a partir de informações do projetista, classifica-o e faz testes de operação através de um ambiente que simula uma aplicação.

Segundo BERGE (1996), praticamente qualquer malha de controle na teoria clássica pode ser configurada com os blocos funcionais *foundation*, e a evolução da norma está no desenvolvimento de técnicas de sensoriamento que aumente a acuracidade, confiabilidade, estabilidade a efeitos do ambiente e no desenvolvimento de transmissores multivariáveis.

A sequência de operação dos *function blocks* é determinada no *function block schedule*, esta é uma tabela que registra quando cada *function block* deve ser disparado. Esta sequência é repetitiva e cada repetição é realizada dentro de um período de tempo denominado macrociclo. Portanto os *function blocks* são disparados pelo menos um vez por macrociclo.

A figura 19 mostra o escalonamento de execução de blocos funcionais distribuídos entre dois transmissores:

Figura 19 - Escalonamento de blocos funcionais

Este macrociclo, portanto é que determina a taxa de execução da malha de controle, pois esta é executada uma vez a cada macrociclo. Quanto maior o número de blocos na estratégia de controle, mais demorado é o macrociclo.

Atualmente a otimização temporal do macrociclo e da estratégia de controle vem sendo estudada e algoritmos de otimização de escalonamento foram propostos por FRANCO (1998), PACHECO (1997) e GARCIA et alli (1997).

A viabilidade do controle de um processo dinâmico por *fieldbus*, assim como o de ferramenta de análise, está portanto diretamente ligada a taxa de aquisição e de atuação oferecida pela rede, assim como o tipo de filtragem de dados executada na aquisição, como afirma BLEVINS & WOJSZNIS (1996).

Além dos *function blocks*, o FBAP possui mais dois tipos de objetos: os *link objects* e os *alert objects*.

Os *link objects* são responsáveis por fazer a comunicação entre parâmetros de entrada e de saída dos blocos no mesmo transmissor (*links* locais) ou em transmissores diferentes (*links* remotos), e de tornar estes disponíveis para a FMS via VCRs, caso tenham que ser publicados na rede.

Os *alert objects* são objetos que monitoram os valores dos parâmetros dos blocos e a partir destes, podem disparar um alarme ou então um evento. São subdivididos em *alarm objects* e *event objects*.

Conforme observa DOVE et alli (1996) os *function blocks* e suas malhas e estruturas podem ser melhor analisados através de uma filosofia de orientação a objeto. Estes objetos serão detalhados a diante.

O macrociclo da malha é dividido em duas bandas sendo que uma delas é reservada para o envio e resposta de comandos acíclicos não referentes ao controle em si (por exemplo a sincronização da DLL) e a outra banda destinada às chamadas de alta prioridade que são as mensagens cíclicas de requisição e publicação de dados referentes ao controle (mensagens relacionadas aos *function blocks*). A relação entre estes dois períodos é sugerida por norma de modo que a banda assíncrona tenha a duração equivalente a 60% da duração da banda síncrona. A figura 20 representa a execução e macrociclos:

Figura 20 - Macrociclo com fase síncrona e assíncrona

WILD & PEREIRA (1999) propuseram uma ferramenta de avaliação temporal das mensagens no barramento, com ferramentas deste tipo é possível medir com precisão todos os tempos envolvidos nos ciclos, e assim ter informações sobre o desempenho da rede e da estratégia de controle adotada. Já HIROOKA et alli,

desenvolveram um ambiente virtual para testar o algoritmo e o comportamento dos parâmetros de um bloco em operação.

2.11.4. SYSTEM MANAGEMENT KERNEL

Esta é uma entidade destinada a garantir a interoperabilidade entre os *devices* em uma rede. Possui uma base de dados que armazena parâmetros tanto estáticos quanto dinâmicos de configuração da rede. Suas funções básicas para garantir tal tarefa são:

- No Link *master*, sincronizar todos os nós da rede através da publicação periódica de uma *clock message*, garantindo assim um senso de comum de tempo em toda a rede.
- Identificar cada *device* com o seu *tag*, identificação dada pelo usuário do sistema, ou então adotado como *default* e garantir que não existam *devices* com *tags* repetidos.
- No Link *master*, garantir a sincronização da aplicação e realizar o *schedule* da aplicação, ou seja, emitir mensagens de requisição de dados para os *devices* certos nos momentos certos, garantindo a realização dos macrociclos.
- No Link *master*, identificar a entrada e a saída de *devices* na rede e assim atualizar uma lista de *devices* vivos naquele momento, a *live list*.

A base de dados do SMK é denominada SMIB ou *System Management Information Base*. DOVE et alli (1998) propuseram uma base de dados orientada a objeto (ODBMS) em substituição a base de dados relacional para conter informação referentes a configuração do sistema, e afirmam que a filosofia de orientação a objetos aplicada a base de dados é a mais adequada para este tipo de aplicação.

Os *Systems Management* em uma rede podem se comunicar acessando diretamente a camada de enlace, e assim enviar suas mensagens sem utilizar VCRs para tal. Estas mensagens geralmente são de *assign tag*, escrita de *tag* no *device*;

locate tag, localização do *device* com o referido *tag* ou *identify device*, onde o *link master* pede o *tag* do *device* para atualização da sua base de dados.

Esta comunicação direta entre a camada de enlace e o SMK é feita por uma "linguagem" ou protocolo de comunicação denominado SMKP - *System Management Kernel Protocol*.

2.11.5. SYSTEM MANAGEMENT KERNEL PROTOCOL COMMUNICATIONS - SMKP

Para garantir o correto funcionamento do SMK, comunicações entre SMKs são feitas usando-se o SMKP. Esta comunicação ao invés de VCRs, utiliza-se diretamente de serviços da DLL, por meio de acesso direto a DLL. Este pontos de comunicação DLL- SMK são denominados DLSAP ou *Data Link Service Access Point*.

Este tipo de comunicação pode ser ponto a ponto ou distribuída a um grupo de SMKs e é usada para mensagens de *assign addresses*, *locate tags*, *distribute time*, *identify device* e outras do contexto do SMK.

2.11.6. NETWORK MANAGEMENT AGENT

Este agente faz parte da estrutura de comunicação da norma denominada *Communications Entity*. Sua função é de garantir, para a rede o acesso as informações estatísticas e de configuração do denominado *communication stack* (conjunto formado pela FMS, FAS, DLL e PHY). O agente possui também uma base de dados composta por informações estáticas e dinâmicas que dizem respeito a basicamente 3 domínios:

- informações referentes a entidade de comunicação como um todo,
- informações referentes as VCRs em uso e,
- informações referentes aos protocolos de cada camada separadamente.

O NMA acessa as informações das camadas diretamente através de LMEs - *Layer Management Entity*. Como esta comunicação é realizada de modo local, a norma deixa aberta a cada fabricante a arquitetura e as funções dos LMEs. Uma importante função da NMA é armazenar o *LAS schedule* no *Link master*, informação da FMS utilizada para fazer a operação de *polling* dos *devices*.

2.11.7. CAMADA DE APLICAÇÃO - FMS

A camada de aplicação é responsável pelo acesso as informações das APs, ou seja, é ela quem tem acesso aos objetos das APs e as suas descrições. Do ponto de vista da FMS, os objetos visíveis pela rede (e descrições) de cada AP são modelados como um VFD - *Virtual Field Device*, estrutura inicialmente proposta no protocolo MMS do projeto MAP. É função da FMS também codificar toda as informações do Processo de Aplicação para serem enviadas em uma mensagem.

A FMS proporciona para os APs os VCR *Endpoints*, que são os pontos de comunicação entre as VCRs e os APs e podem ser basicamente de 3 tipos dependendo do tipo da respectiva VCR: *client/server*, *report distribution* ou *publisher/subscriber*.

Alguns exemplos de serviços da FMS estão apresentados na tabela 3:

Objeto da FMS	Serviço	Tipo do Serviço	Significado	Tipo de VCR Endpoint
<i>VCR Endpoint</i>	<i>Initiate</i>	Confirmado	Iniciar uma conexão	Todos
	<i>Abort</i>	Não Confirmado	Terminar uma conexão	Todos
<i>VFD</i>	<i>Status</i>	Confirmado	Ler status de um <i>device</i>	Cliente/Servidor
	<i>Identify</i>	Confirmado	Ler nome do fabricante, tipo e versão	Cliente/Servidor
<i>OD</i>	<i>GetOD</i>	Confirmado	Ler a descrição de um objeto	Cliente/Servidor
	<i>PutOD</i>	Confirmado	Carregar a descrição de um objeto	Cliente/Servidor
<i>Domain</i>	<i>Upload Segment</i>	Confirmado	Pedir um bloco de dados	Cliente/Servidor
	<i>Download Segment</i>	Confirmado	Enviar um bloco de dados	Cliente/Servidor
<i>Program Invocation</i>	<i>Start</i>	Confirmado	Iniciar um programa	Cliente/Servidor
	<i>Reset</i>	Confirmado	Resetar um programa	Cliente/Servidor
<i>Variable List</i>	<i>Write</i>	Confirmado	Escrever uma variável	Cliente/Servidor
	<i>Physwrite</i>	Confirmado	Escrever em uma posição de memória	Cliente/Servidor
<i>Event</i>	<i>Event Notification</i>	Não Confirmado	Comunicar um evento	<i>Report Distribution</i>
	<i>Alter Event Condition Monitoring</i>	Confirmado	Habilitar/Desabilitar um evento	Cliente/Servidor

Tabela 3 - Serviços da FMS

2.11.8. SUBCAMADA DE APLICAÇÃO - FAS

A FAS permite o acesso da camada de enlace aos *VCR Endpoints*, e cada um destes é definido pela FAS de acordo com as capacidades da camada de enlace. A FAS para tal tem acesso aos parâmetros da camada de enlace.

Alguns das características e tipos de *VCR Endpoints* estão na próxima tabela:

Características do VCR Endpoint	Tipo da VCR		
	Publisher/ Subscriber	Report Distribution	Cliente/Servidor
Número de Endpoints por VCR	1 publisher N subscribers	1 source N sinks	1 cliente 1 servidor
Endpoint livre	Não	Sim	Sim
Confirmação de transferência de dados	Não	Não	Sim
Cíclico / Acíclico	Ambos	Acíclico	Acíclico
Unidirecional / Bidirecional	Unidirecional	Unidirecional	Bidirecional
Modo de Operação	Orientado a conexão	Não orientado a conexão	Não orientado a conexão
Queued/Buffer	Buffer	Queued	Queued

Tabela 4 - Caracterização dos VCR Endpoints

2.11.9. CAMADA DE ENLACE - DLL

A Camada de Enlace é a que faz a ligação do meio físico com as estruturas de aplicação.

Ela pode ser configurada como *Basic* para *devices* comuns sem nenhuma função de *schedule* ou de *bridge*. Este tipo de DLL é capaz de receber e responder mensagens de requisição de dado. O acesso ao meio físico (tanto cíclico como acíclico) é organizado por uma DLL especial, a LAS, que envia *tokens* (operação de *polling*) e requisições de dado a todas as outras estações do *link*. Além disso, a LAS ainda envia mensagens de sincronização e mantém atualizada a *live list*, ou seja, monitora quais *devices* saem ou entram na comunicação.

Uma DLL pode ser configurada como Mestre *Backup*, assim ela pode se tornar a LAS quando a LAS ativa perder a comunicação. A figura 21 é uma representação do mecanismo de passagem de *token* representando a LAS e as estações do tipo *basic*:

Figura 21 - Passagem de token

Nota: DeT = Delegate token; CiT = Circulate token

Outro tipo de DLL é a *bridge*, esta tem a capacidade de interligar até quatro *link* secundários a um *link* primário. Assim a *bridge* redistribui mensagens entre *devices* de diferentes *links* respeitando os endereços de destino e de origem e re-publica mensagens do *link master* aos *links* secundários. No caso da *clock message*, a *bridge* recalcula o tempo de sincronismo da mensagem antes de enviá-la para compensar o atraso da operação de republicação.

A norma prevê três mecanismos da camada de enlace para distribuição de dados (VCRs) , um orientado a conexão e dois não orientados: *report distribution*, *publisher/subscriber* e cliente/servidor. As aplicações destes três tipo de comunicação estão listadas na tabela 5:

Tipo da VCR	VCR Endpoint	Aplicação
Report Distribution	QUU ¹	Notificação de eventos e <i>trend reports</i>
Publisher/Subscriber	BNU ²	Output de parâmetros do FBAP (uma VCR para cada parâmetro a ser publicado)
Client/Server	QUB ³	Acesso a objetos visíveis pela rede, Acesso dos NMA e SMK aos respectivos MIBs

Tabela 5 - Tipos de VCRs

Nota: 1) QUU = Queued, User Triggered, Unidirecional multicast

2) BNU = *Buffered, Network Scheduled, Unidirecional broadcast*

3) QUB = *Queued, User Triggered, Bidirecional peer to peer*

Um ambiente virtual para teste da camada DLL foi desenvolvido por [HIROOKA et alli, 1997], onde interrupções simulando chamadas da PHY e da FMS são realizadas em tempo real em duas DLL operando como que em rede.

2.11.10. CAMADA FÍSICA - PHY

A camada física pode ser de diferentes tipos de acordo com as características e ambiente de aplicação da rede, estes tipos são H1 e H2. O padrão H1 provê baixa velocidade, e é intrinsecamente seguro. Em alguns casos pode-se ter no próprio barramento H1 de par trançado superposto ao sinal de comunicação a tensão de alimentação, para que os *devices* sejam alimentados pelo par trançado ou não. Por outro lado se a limitação de potência e segurança não são requeridas, pode ser usada a camada física de alta velocidade, H2. O barramento H2 também é utilizados para unir segmentos isolados de H1. A norma para o H2 está em fase de finalização, pois só começou a ser estruturada em abril de 1998, quando a *foundation* decidiu que seria baseada no padrão *high speed ethernet* de 100 Mbits/s, onde a conexão entre os ramos H1 e H2 é realizada por um dispositivo do tipo *bridge*.

PIMENTA & LAZARI (1996) propõem o conceito de FDC - *Fieldbus Discrete Controller* - um controlador dotado de módulos I/O e *bridge* H1/H2. O pesquisador afirma que um FDC tem menos restrições de consumo do que um transmissor, o que pode ser um fator de aumento de desempenho no processamento, que aliado ao contato direto ou indireto a múltiplas variáveis de processo, torna o FDC um dispositivo potencial para conter blocos funcionais de processamento de algoritmos multivariáveis.

Como foi constatado na transição *SDCD-fieldbus*, a evolução de software e hardware favorece a migração das tarefas de controle para os transmissores, mas por outro lado, avanços na teoria de controle irão introduzir nos algoritmos pesados que possivelmente não poderão ser processados em transmissores. Assim, estes

dispositivos mais poderosos como o FDC irão servir de plataforma para os algoritmos de controle pesados.

A codificação do sinal H1 é do tipo *manchester*, e a comunicação é do tipo *half duplex* em linha balanceada com uma taxa de comunicação de 31.25 Kbits/s.

A norma deixa em aberto a utilização ou não de blindagem com aterramento no par trançado, como proteção a interferências eletromagnéticas

Para estes ambientes, utiliza-se ainda barreiras de proteção, que são limitadores de potência e também geradores de impedância. O número de dispositivos por link e o comprimento deste também é limitado por motivos de qualidade e potência de comunicação. Caso a aplicação exija maiores distâncias, utiliza-se estações repetidoras de sinal.

Figura 22 - Codificação Manchester

Como alternativa ao meio de transmissão por cabos, GERON(1999) propôs técnicas de transmissão por fibra ótica, através de repetidores óticos e por rádio frequência. Tais técnicas visam a aplicação do *fieldbus* a ambientes onde a transmissão por cabo se faz inadequada.

2.11.11. VIRTUAL COMMUNICATION RELATIONSHIP - VCR

Os APs remotos se comunicam via VCRs, que essencialmente são linhas de comunicação dedicadas e estabelecidas entre os seus respectivos *devices*. O ponto de ligação entre um AP e uma VCR é o VCR *Endpoint* como dito anteriormente. Este

VCR *Endpoint* é semelhante a um *buffer* associado à VCR e ao AP. Cada VCR tem um contexto particular pois seus parâmetros são relacionados as camadas FMS, FAS e DLL. A definição do tipo de VCR é baseada no tipo de comunicação a ser estabelecida, é importante lembrar que as comunicações entre SMKs não utilizam VCR, e sim são estabelecidas pelo SMK *Protocol*. Abaixo está um quadro comparativo entre as VCR definidas por norma:

Tipo de VCR Endpoint	Tipo de Aplicação	Modo de operação da DLL	Papel de VCR Endpoint
BNU	Publicação	Orientado a conexão	<i>Publisher</i> <i>Subscriber</i>
QUB	Requisição/Resposta	Orientado a conexão	Cliente Servidor
QUU	<i>Report Distribution</i>	Não orientado a conexão	<i>Source</i> <i>Sink</i>
Nenhuma	Acesso da SMK pelo SMKP	Não orientado a conexão	Acesso ao SMK

Tabela 6 - Comparação entre as VCRs

2.11.12. *DEVICE DESCRIPTIONS - DD*

Associada aos *Function Blocks*, esta ferramenta serve de apoio ao OD na descrição dos objetos da aplicação visíveis pela rede. A DD carrega descrições mais detalhadas dos objetos e de procedimentos e métodos relativos ao FBAP que são úteis na fase de configuração da rede e da aplicação, associando os objetos aos seus índices no OD do transmissor. Para a elaboração da DD, foi criada uma linguagem dedicada e muito semelhante ao C chamada de DDL ou *Device Description Language*.

Cada *device* deve ter uma *device description* relativa aos seus FBAPs e *function Blocks*.

Uma vez escrita a DD, o código gerado em DDL é compilado para uma linguagem legível pela máquina por uma ferramenta denominada *tokenizer*. Este código pode então ser carregado no próprio *device* ou então em mídia externa como por exemplo disquete, e então ser lido diretamente desta.

A DD é utilizada para facilitar a interoperabilidade de dispositivos de fabricantes diferentes, uma vez que todas as capacidades e objetos de cada dispositivo estão detalhadas em uma linguagem comum.

Figura 23 - Funcionamento da *Device Description*

A norma prevê cinco tipos básicos de *devices* no seu *profile*, sendo que esta classificação é uma das informações que devem estar contidas na DD. Os cinco tipos de *devices* previstos por norma são os seguintes:

- 1- *Device* Passivo: é aquele *device* que só "escuta" a linha. Ele não tem permissão para transmitir mensagem alguma, portanto os outros *devices* não conseguem detectar sua presença.
- 2- *Device* de I/O remoto: age apenas como *publishers* ou *subscribers*, e tem capacidades limitadas na DLL. Geralmente suas aplicações acessam diretamente a FAS e eles não possuem nenhuma entidade de gerência (*management*).
- 3- *Device* Temporário: seu endereço (*node address*) é um dos quatro endereços reservados a visitantes. Geralmente são usados para configurações ou para a detecção de problemas através de mudanças nos NMA ou SMK dos outros *devices*.

- 4- *Device de Campo*: São os *devices* que contém os *Function Blocks*, e tem endereço definitivo.
- 5- *Device de Display* ou de Controle: São *devices* cuja função é fazer a interface homem máquina ou a supervisão do sistema e também podem atuar como *gateways*. Contém todas as estruturas de rede e podem conter também *Function Blocks*, sendo que seus endereços são permanentes.

Além dos arquivos de DD a norma prevê, em um capítulo denominado *common file format*, o uso de outros tipo de arquivos de dados, a serem utilizados para troca de informações entre fabricante, desenvolvedor de sistemas e usuário. Esta troca de informações é particularmente necessária na fase de instalação e *start up* do sistema como observa TOMITA et alli (1996).

3. PROCESSO DE APLICAÇÃO DOS BLOCOS FUNCIONAIS

3.1. INTRODUÇÃO

De acordo com a Fieldbus Foundation, as especificações da Processo de aplicação dos blocos funcionais (FBAP) visam normalizar:

1. O projeto e a implementação de aplicações de blocos funcionais de modo interoperável,
2. A configuração e a implementação de *function blocks* em diferentes tipos de transmissores de campo
3. A interoperação em rede das aplicações de blocos durante a operação do sistema, como mostra a figura 24:

Figura 24 - Aplicações de blocos funcionais em um sistema

A FBAP possui para tal, estruturas e definições chaves que formam a base para a operação do sistema e para o seu entendimento. Estas estruturas e definições estão na figura seguinte e descritas em diante:

Figura 25 - Componentes chaves do FBAP

Referência: FIELDBUS FOUNDATION (1998). *FF 800*

3.1.1. *RESOURCE*

Um *resource* é considerado uma subdivisão lógica do software ou do hardware do transmissor. Tem como característica o controle independente da sua própria operação, assim, um *resource* é independente e pode ser modificado sem prejuízo de outras estruturas. Os processamentos, se utilizados, são feitos baseados em dados ou processos externos, acessados via estruturas de interfaceamento. Do ponto de vista da rede, uma aplicação é um conjunto de *resources*, e cada *resource* define quais os aspectos da aplicação que são visíveis pela rede.

3.1.2. *FUNCTION BLOCK (BLOCO FUNCIONAL)*

O *function block* é uma função de automação básica realizada por uma aplicação, sendo o mais independente possível do meio físico e da estrutura da rede. Os *function block* processam parâmetros de entrada em função da sua configuração e assim geram saídas que serão utilizadas por outros blocos.

3.1.3. *TRANSDUCER BLOCK (BLOCO TRANSDUTOR - TRD)*

Os *transducer block* isolam os *function blocks* dos dispositivos específicos de I/O, como sensores, atuadores e chaves. Os *transducer* controlam os dispositivos de I/O e geram uma interface padrão para os *function blocks*, realizam também funções como calibração, filtragem de sinais e conversão de dados. Os *transducer* seguem definições que se dividem em grupos comuns ou *profiles*, embora possam ser de um tipo particular definido pelo fabricante.

3.1.4. *LINK OBJECT (OBJETO LINK)*

Os *link objects* promovem a troca de dados entre os *resources* e a rede de comunicação. Todos os dados ou eventos que são enviados na rede devem ser definidos por *link objects*. Estas estruturas geram *trend reports*, e notificações de alerta.

3.1.5. *RESOURCE BLOCK (BLOCO RESOURCE - RES)*

As características específicas do *hardware* que são associadas a qualquer *resource* são mapeadas e se tornam visíveis através do *resource block*. Assim como o *transducer*, o *resource* isola características do *hardware* além de conter algumas funções relacionadas ao *hardware*.

3.1.6. VIEW E TREND OBJECTS (OBJETOS VIEW E TREND)

Estes objetos tornam o acesso a parâmetros dos blocos mais eficiente. O *view object* permite o pedido de um conjunto de parâmetros com uma só requisição e o *trend object* permite o acesso a um conjunto de amostras de um mesmo parâmetro de uma só vez.

3.1.7. ALERT OBJECTS (OBJETOS ALERTA)

Permite a notificação de eventos entre transmissores. Isto torna possível iniciar um procedimento de alarme em caso de um evento significativo.

3.1.8. DOMAIN E PROGRAM INVOCATION OBJECTS (OBJETOS DOMÍNIO E PROGRAM INVOCATION)

Programas ou códigos podem ser definidos como *domain objects*, assim serviços de *download* ou *upload* tornam-se possíveis. O objeto *program invocation* pode associar a um *domain* um código executável.

3.1.9. ACTION OBJECTS (OBJETOS AÇÃO)

Um resource pode ou não conter um *action object*. Este objeto instancia ou apaga um objeto ou um bloco dentro do resource.

3.1.10. SYSTEM TIME (TEMPO DO SISTEMA)

É uma sincronização gerada pelo *system management* para o FBAP, para sincronizar a operação dos *function blocks*. O *system management* mantém um padrão de tempo comum a todas as aplicações da rede e todos os registros de tempo dos alarmes, eventos e *trends* são baseados nesta sincronização.

3.1.11. FUNCTION BLOCK SCHEDULE (ESCALONAMENTO DOS BLOCOS FUNCIONAIS)

O *system management kernel* estrutura um agendamento, ou *function block schedule*, para a execução dos blocos de cada transmissor da rede, este agendamento está baseado no *system time*. A execução de um bloco pode ser também requisitada pela conclusão de outro bloco. Os métodos de montagem do sequenciamento dos blocos podem ser definidos pelo fabricante.

3.1.12. NETWORK TRAFFIC (TRÁFEGO DE REDE)

É todo o tráfego de informações do *function block application* via rede. Estas informações podem ser agendadas ou não. Através das comunicações agendadas, os blocos enviam e recebem parâmetros sincronizadamente para que possam executar suas funções dentro do *schedule*. As mensagens não agendadas ou *background traffic* não interferem nas comunicações agendadas que são críticas.

Alguns serviços da FMS organizam o trânsito de mensagens na rede e o FMS em conjunto com o *function block application* definem o formato das mensagens.

3.2. OPERAÇÕES CARACTERÍSTICAS DE SISTEMAS DE CONTROLE DISTRIBUÍDOS

Estes componentes e definições chaves apresentados acima realizam funções, procedimentos e interações durante todo o processo de operação do sistema. Basicamente existem 5 tipos primários de operações características no sistema *Foundation Fieldbus*, e em geral, nos sistemas de controle distribuído. São eles:

3.2.1. CONFIGURAÇÃO E DEFINIÇÃO DOS *FUNCTION BLOCKS*

Os *function blocks* são definidos pelos seus parâmetros de entrada, de saída, parâmetros de configuração e os algoritmos que utilizam estes parâmetros. A identificação dos blocos é dada por um *tag* e por um índice no dicionário de objetos (OD). O *tag* é uma identificação simbólica do bloco, única em todo o sistema e o índice no OD é um número associado a bloco que otimiza as rotina de acesso ao bloco. O índice no OD ao contrario do *tag* só tem validade no recurso a qual o bloco está inserido.

Os parâmetros são visíveis e acessíveis pela rede, sendo que os parâmetros de entrada e de saída são usados para a troca de dados entre os blocos através dos links e os parâmetros de configuração apenas para operações dentro do bloco, portanto não podem ser “linkados”.

Todos os parâmetros estão associados a números de identificação no OD denominados *DD member Ids*, estes *Ids* são relativos apenas ao parâmetro, assim, a informação do bloco é necessária para diferenciar parâmetros semelhantes em blocos distintos.

A descrição de cada parâmetros de cada bloco do transmissor está presente também na respectiva DD.

3.2.2. CONFIGURAÇÃO DAS LIGAÇÕES OU *LINKS* ENTRE OS *FUNCTION BLOCKS*

Os parâmetros dos blocos necessitam ser acessados tanto por outros blocos, como por elementos de interface como configuradores e gerenciadores de *status* (ferramentas de diagnóstico). Cada elemento de comunicação é portanto configurado e associado a um objeto de *link*.

A comunicação entre dois blocos se dá por um *link* (local ou remoto) entre um parâmetro de saída de um bloco e um parâmetro de entrada de outro bloco.

A transferência de dados em um *link* remoto, ou seja entre transmissores, se dá através das camadas de rede por um canal de comunicação virtual conhecido denominado VCR (*virtual communication relationship*), como já apresentado anteriormente. A VCR utilizada em um *link* remoto deve ser identificada por um número (*VCR number*) e é configurada ou pré determinada pelo fabricante.

Para suprir as necessidades do FBAP, dois tipos de VCR são utilizadas: Cliente/Servidor para comunicações entre aplicações e *Publisher/Subscriber* onde uma aplicação envia um dado na rede para uma ou mais aplicações.

3.2.3. PROCESSAMENTO DOS ALARMES

Os alarmes e eventos, ou simplesmente alertas são mudanças nos estados do FBAP. Cada aplicação possui um objeto responsáveis por notificar os outro FBAP sobre a ocorrência de um alerta, estes elementos são os *alert objects*.

Os objetos de alerta executam todo o procedimento necessário em um situação de alerta, aliviando assim os blocos *transducers*, *resource* e até mesmo o *function block* de procedimentos de emergência, permitindo que estes continuem nas suas respectivas funções usuais.

Os objetos de alerta monitoram constantemente a operação dos blocos, assim que seja identificada uma situação previamente definida como de alerta, ou algum alarme, o objeto publica na rede uma notificação do evento ocorrido associado a um registro de tempo. Uma confirmação é então esperado por parte do *link master*

indicando que as devidas providencias por parte da interface operacional já foram tomadas, caso esta confirmação não seja recebida, a mensagem de alerta é novamente publicada na linha.

3.2.4. ACESSO A INFORMAÇÕES PARA A INTERFACE DO OPERADOR, DIAGNÓSTICO E CONFIGURAÇÃO

Existem dois tipos de comunicação relativas a operação dos function blocks: as comunicações cíclicas e as acíclicas. As comunicações acíclicas são de baixa prioridade, para que não interfiram nas comunicações cíclicas (conforme Figura 19) e são utilizadas para informações de configuração, interface com operador e diagnóstico do sistema. Os parâmetros de um bloco são agrupados em quatro conjuntos para otimizar este tipo de comunicação acíclica, facilitando o acesso aos parâmetros por parte do configurador. Estes conjuntos são denominados *views*. Das quatro *views*, duas são dinâmicas e duas estáticas. A definição das *views* é de responsabilidade do usuário.

Outro tipo de estrutura de dados para otimização de comunicação acíclica é o *trend object*, onde são enviados 16 registros consecutivos de uma mesma variável.

3.2.5. COORDENAÇÃO DO FUNCIONAMENTO DOS BLOCOS E SUA OPERAÇÃO.

A execução de um bloco é sempre disparada por um *input event*. A geração deste evento depende da configuração do sistema, e as fonte de *input event* são as seguintes:

1. *Schedule*, a taxa com que um bloco é executado é definido por um *schedule*, este *schedule* determina o instante em que o bloco inicia sua execução.

2. Final da execução de outro bloco, o final da execução de um bloco pode disparar outro desde que estes esteja no mesmo *resource*.
3. Especificação do fabricante, o fabricante pode especificar o evento de disparo do bloco.

O escalonamento da execução dos blocos de uma aplicação é influenciado pelas seguintes características (requisitos) de um sistema de controle distribuído:

- Todos os blocos devem compartilhar um senso comum de tempo, denominado tempo do sistema
- As aquisições devem ser realizadas em um período preciso para que possam ser disponibilizadas aos blocos de entrada e então aos parâmetros de entradas dos blocos de controle.
- Os parâmetros de entrada e de configuração devem ser atualizados corretamente de modo a garantir a presença de dados corretos durante a execução dos blocos.
- O período entre a geração de um parâmetro de saída e o início da execução de um bloco subsequente deve ser muito pequeno comparado ao período de execução dos blocos. Este período é composto pelo tempo de transferência do parâmetro e o tempo que o bloco leva para começar a iniciar após o parâmetro estar pronto.

Estas características tornam necessário o controle de três aspectos na execução dos blocos:

1. Deve ser possível sincronizar os blocos utilizando um único sistema de referência temporal em todo o sistema,
2. Deve ser possível gerar um *schedule* que garanta que os blocos gerem saídas antes do início da execução de outros blocos que utilizem estas saídas,
3. O *schedule* deve permitir o trafico de outras mensagem independentemente da publicação escalonada de parâmetros

O *system management agent* realiza este controle de sincronização dos blocos e sincroniza o senso comum de tempo nos transmissores para todas as aplicações, atendendo assim todos os requisitos acima apresentados.

3.3. INTERAÇÕES EXTERNAS

Todos os dispositivos que contem *function blocks* são chamados dispositivos de campo. Existem 3 outros tipos de dispositivos que podem estar conectados a uma rede, e não que possuem *function blocks*:

1. Dispositivos temporários: são associados a um conjunto de endereços temporários, a aplicação presente neste tipo de equipamento geralmente é capaz de realizar tarefas de diagnóstico e ajuste em outros dispositivos.
2. Dispositivos de Interface: possuem endereço definitivo no barramento, são utilizados como interface homem-máquina e dispositivos de configuração e diagnóstico da aplicação.
3. Dispositivos Monitores: não possuem endereço e são utilizados para monitorar a comunicação.

3.3.1. INTERAÇÃO COM APLICAÇÕES DE CONFIGURAÇÃO

Uma aplicação de configuração, ou configurador, pode estar em um dispositivo temporário ou de interface e é utilizada para os modificar os blocos existentes assim como para instanciar ou apagar blocos.

Quando o configurador não tem informações sobre a disposição dos blocos, e portanto sobre o acesso aos parâmetros, este deve utilizar funções da *System Management* para determinar onde os blocos estão instanciados e qual VCR deve ser utilizada para acessar cada parâmetro. A partir daí são utilizados serviços da FMS para estabelecer as respectivas VCRs.

Para obter as informações completas dos parâmetros a serem acessados, o configurador utiliza os registros do DD, e assim a FMS pode ler e modificar os parâmetros remotamente de maneira adequada.

3.3.2. INTERAÇÃO COM INTERFACES HOMEM-MÁQUINA

A FMS utiliza números de identificação associados aos recursos do sistema, por motivo de otimização. Caso a interface homem-máquina (IHM) não possui registros deste identificadores, ou então o endereço do recurso, deve utilizar serviços de procura de *tag* da FMS para obtê-los.

Quando um alerta é gerado, o recurso associado a este alerta utiliza serviços de notificação de eventos da FMS para acionar a IHM ativa.

3.3.3. INTERAÇÃO COM OUTRAS APLICAÇÕES DE CONFIGURAÇÃO

Na rede podem existir dispositivos tais como estações *host*, controladores DCS, PLCs ou outros elementos que não possuem *function blocks* mas possuem aplicações que executam tarefas de controle. Este tipo de aplicação geralmente necessita acessar ou modificar parâmetros em *function blocks*. Para tais ações, devem ser comunicados o parâmetro e o seu *status*.

O operador pode configurar o modo do bloco de modo a aceitar tais ações de controle remotas, assim certos parâmetros podem ser atualizados periodicamente por tais aplicações.

Caso o parâmetro a ser recebido não o foi, dentro do período de tempo esperado, ou então foi recebido com status *bad*, o bloco muda de modo de operação para que este parâmetro possa ser atualizado por outro bloco ou pelo usuário, e assim que a comunicação com o sistema original for restabelecida, o bloco volta ao modo de operação anterior.

3.3.4. INTERAÇÃO COM OUTROS DISPOSITIVOS DE CAMPO

Um parâmetro de saída de um bloco pode ser ligado a um ou mais parâmetros de entrada em outros blocos, juntamente com o dado é transmitido seu *status*. Estes parâmetros estão sempre associados a uma unidade de engenharia. Caso um parâmetro não seja recebido, entende-se que ocorreu uma falha de comunicação e esta informação será utilizada pelo processamento de modo a evitar descontinuidades nas ações de controle.

Existem dois sentidos de fluxo de dados entre *function blocks*:

- Fluxo de dados para frente: quando os dados vão de um bloco de entrada para um de controle e deste para um de saída.
- Fluxo de dados para trás: quando os dados vão de um bloco de saída para um bloco de controle e deste para um outro bloco de controle. O *status* desta informação diz se a informação foi possível ou não de ser enviada pelo bloco posterior.

3.4. ESTRUTURA DO FBAP

3.4.1. RECURSOS EM UM TRANSMISSOR

Cada recurso em um transmissor tem um nome que, junto com o *tag* do respectivo transmissor tornam a referência única, e estes recursos trocam dados entre si ou com interfaces de comunicação. Cada recurso tem um diretório onde estão listados seus componentes. A interface com os recursos remotos da mesma aplicação se dá pela FMS, e esta interface é realizada através do dicionário de objetos (OD), que é acessado tanto pela FMS quanto pelos blocos. Todas as informações referentes aos blocos são mantidas no OD e utilizadas na comunicação.

Devido haverem vários recursos em uma aplicação, distribuídos em um ou mais transmissores, uma lista de ponteiros de recursos é mantida atualizada no *Management Information Base* (MIB) pelo *System Management*.

Os componentes de uma aplicação de blocos funcionais podem ser acessados através dos seus respectivos recursos, estes recursos serão descritos mais detalhadamente abaixo:

3.4.2. OBJETO BLOCO

Neste tópico será descrito o modelo formal de um bloco, que pode ser um *transducer*, *resource* ou um bloco funcional.

Um bloco é uma unidade de processamento individual. Os eventos de entrada e os parâmetros de configuração podem afetar o algoritmo interno, assim como o algoritmo influencia na geração de eventos e modifica os parâmetros. Parâmetros de entrada e de saída são alocados em *buffers* especiais que evitam que tais dados sejam modificados durante a execução do bloco, assim pode-se garantir que os valores de entrada e saída não serão corrompidos.

Além dos parâmetros especificados, existem variáveis internas que não possuem *member Id* portanto não podem ser vistas por nenhuma estrutura ou função, assim como o algoritmo interno não pode ser identificado, ficando a cargo do fabricante suas especificações.

Um bloco é identificado por seu tipo e pelo nível de revisão, com estas especificações é possível identificar os parâmetros e as características de execução do algoritmo. A informação do tipo do bloco identifica todos os parâmetros que devem ser universais para todas as instanciações daquele bloco.

A DD de cada bloco é identificada pelo fabricante, tipo do bloco e o número de revisão.

3.4.2.1. Eventos

Os blocos são executados em resposta a um evento. Dependendo do tipo do bloco, uma ou mais ações podem gerar estes eventos, como foi descrito anteriormente. Após a execução de um bloco, também podem ser gerados eventos:

- Evento de Execução de Bloco: inicia a execução de um bloco no mesmo transmissor.
- Evento de Notificação de Alerta: dependendo das condições de execução o notificador de alerta do seu recurso pode ser acionado.
- Evento Específico do Fabricante: a critério do fabricante.

3.4.2.2. Parâmetros

Cada parâmetro é caracterizado pelo seu identificador, modo de armazenagem, modo de uso, e relação com os outros parâmetros.

Identificadores:

Cada parâmetro é identificado pelo conjunto “*Tag.Parameter*”. Onde o *Parameter* diz o *DD member ID* do parâmetro e o *tag* informa qual o bloco. O conjunto “*Tag.Parameter*” é utilizado para se obter o índice do parâmetro no OD, através de serviços de procura de *tag* do *System Management*.

O tipo de dado do parâmetro é especificado pelo *data type index*, referência no OD a um tipo de dado padronizado.

Outra maneira de identificar um parâmetro é através do *Item ID*, este *Item ID* especifica a semântica do dado, ou seja, os valores permitidos e as regras de uso do dado. Caso dois parâmetros tenham a mesma semântica, eles tem o mesmo *Item ID*. A semântica de cada parâmetro fica documentada na DD.

Modo de Armazenagem:

Os parâmetros podem ser classificados como dinâmicos, estáticos ou não voláteis.

Os dinâmicos são os que são calculados pelo algoritmo do bloco, assim não precisam ser recuperados após por exemplo uma falha por queda de energia.

Os estáticos são os dados configurados que precisam ser salvos no caso de uma falha por queda de energia. Normalmente os parâmetros estáticos são periodicamente verificados pelo configurador e eventualmente atualizados por este. Para auxiliar na tarefa de monitorar os parâmetros estáticos, sempre que um parâmetro estático for modificado, a *static revision* do transmissor é incrementada e um evento é gerado.

Os parâmetros não voláteis são os que estão constantemente sendo atualizados e devem ser salvos no caso de queda de energia do equipamento. Como estes parâmetros são constantemente modificados, não são verificados por configuradores como os dinâmicos.

Modo de Uso:

Como já foi descrito, quanto ao modo de uso, os parâmetros podem ser de entrada, saída ou de configuração, existem parâmetros de configuração chamados de universais, que devem estar presentes em todos os blocos, são eles:

- *Static Revision*
- *Block Description*
- *Stratery*
- *Alert Key*
- *Mode Block*
- *Block Error*

Além destes parâmetros universais, outros parâmetros de configuração podem ser adicionados a critério do fabricante.

Em parâmetros de entrada e saída, existe um status associado que indica a qualidade da informação presente. Este status é composto por 3 partes: qualidade, *sub-status* e limites. A informação de qualidade pode ser:

- *Good (Cascade)* - o dado é confiável e pode ser utilizado em uma estrutura de controle do tipo cascata.

- *Good (Non-Cascade)* - a qualidade do dado é boa, porem o bloco não pode fazer parte de uma estrutura cascata.
- *Uncertain* - a qualidade deste dado é suspeita, pode ter acontecido um erro na aquisição ou em cálculo.
- *Bad* - o dado não é válido e não deve ser utilizado.

O *sub-status* é uma informação complementar da qualidade, é utilizado na inicialização ou finalização de estruturas tipo cascata e alarmes. O limite informa se o valor é limitado ou não.

Os parâmetros de entrada possuem *status* semelhantes aos de saída. No caso de uma estratégia de controle onde um parâmetros de entrada não seja "linkado", este possuirá comportamento semelhante a um parâmetros de configuração sendo que o seu valor será uma constante a ser modificada pelo configurador.

Relação com os outros parâmetros:

A execução do algoritmo do bloco depende de vários fatores, entre eles, dos valores e *status* dos parâmetros, do *hardware* e da implementação do *software*. Em função desta dependência, o bloco pode executar seu algoritmo em modos diferentes. Cada modo de execução do bloco impõe características ao algoritmo que influem nas saídas e nos *status*. A seguir estão os oito modos possíveis de operação:

- *Out of Service (OOS)*: o bloco não está sendo executado.
- *Initialization Manual (Iman)*: O algoritmo não está sendo executado e as saídas do bloco seguem um parâmetro externo vindo do bloco que está ligado em sequência no fluxo direto.
- *Local Override (LO)*: as saídas do bloco de controle apenas segue um parâmetro externo. Este modo é executado durante uma situação de falha.
- *Manual (Man)*: As saídas do bloco não estão sendo calculadas, o operador configura diretamente as saídas.
- *Automatic (Auto)*: O algoritmo está sendo executado normalmente e está calculando as saídas.

- *Cascade (Cas)*: O *setpoint* está vindo de outro bloco por um *link*, assim o operador não pode atuar diretamente no *setpoint*.
- *Remote Cascade (Rcas)*: O *setpoint* está vindo por um link de outra aplicação de controle remota em um dispositivo de interface, assim o operador não pode atuar diretamente no *setpoint*.
- *Remote-Output (Rout)*: As saídas estão sendo modificadas por uma outra aplicação de controle remota em um dispositivo de interface.

O parâmetro *Mode Block* indica o modo de operação do bloco. Este parâmetro é composto por quatro elementos:

- *Target*: é o modo desejado em que o bloco execute. Normalmente o *target* é definido pelo operador.
- *Actual*: é o modo em que o bloco está sendo executado efetivamente.
- *Permitted*: são os possíveis modos que o bloco pode apresentar.
- *Normal*: lista os modos normais de operação do bloco dentre aqueles listados no *permitted*. Serve para orientar o operador o momento da seleção do *target mode*.

3.4.2.3. *Resource Block*

As características do *hardware* associado ou *resource* estão descritas nos parâmetros de configuração do *resource block*. Também o *resource* possui os parâmetros universais, e através destes parâmetros de configuração, o *resource* isola os *function blocks* das características de *hardware* dos dispositivos.

O algoritmo do *resource* é utilizado para monitorar o estado de operação do hardware, e indicar possíveis alarmes neste aspecto.

A execução do *resource* não entra no escalonamento dos blocos, portanto este bloco não possui parâmetros de entrada ou de saída. A execução do *resource* segue regras que são definidas pelo fabricante. Um dos parâmetros de configuração

presentes no *resource* é o *resource state*. Este parâmetro indica o estado do *hardware*:

- *Start/Restart*
- Inicialização
- *On-line*
- *On-line Linking*
- *Failure*
- *Standby*

Em alguns casos outros blocos podem utilizar os parâmetros do *resource* por acesso local.

3.4.2.4. Transducer Block

O bloco *transducer*, ou transdutor, tem a função de isolar os *function blocks* das funções de I/O específicas assim como acionamento de sensores, atuadores ou chaves.

O algoritmo do *transducer* realiza a aquisição, linearização e filtragem do dado, disponibilizando para os blocos de entrada um valor em formato padrão independente do sensor. A comunicação do *transducer* com um *function block* de entrada ou saída é feita através de canais, sendo que um *transducer* pode ter vários canais. Uma das vantagens deste isolamento do *function block* com o *hardware* é a possibilidade da execução do *transducer* quantas vezes forem necessárias para gerar um dado de boa qualidade sem sobrecarregar os *function blocks*.

Os blocos *transducer* são classificados em três tipos:

- *Input* - realizam a interface com sensores
- *Output* - realizam a interface com atuadores ou dispositivos de saída
- *Display* - operam dispositivos de interface locais

Assim como o *resource block*, o *transducer* não possui parâmetros de entrada e de saída e sua execução não é comandada pelo *schedule*, mas definida pelo fabricante.

3.4.2.5. *Function Block*

Os *function blocks* realizam as ações e funções básicas presentes em um sistema de automação e controle, independentemente da plataforma de *hardware*.

Como foi citado, os *function blocks* possuem parâmetros de entrada, saída e de configuração e a partir deste parâmetros realizam um algoritmo específico.

Os function blocos são classificados em:

- Blocos de Entrada: acessam medidas físicas através da comunicação com *transducer blocks* de entrada por meio de canais. Estas medidas então são disponibilizadas para outro blocos através dos parâmetros de saída.
- Blocos de Saída: acionam *transducers* de saída pelos canais a partir de um valor recebido pelos *links* com outros blocos nos seus parâmetros de entrada. Realizam também a *back-calculation*.
- Blocos de controle: Realizam cálculos com parâmetros de blocos de entrada e enviam parâmetros par outros blocos de controle ou para blocos de saída. Realizam também a *back-calculation* e possuem processamentos que evitam efeito de *bump* e de *wind-up*.
- Blocos de Cálculo: Realizam cálculos matemáticos sobre parâmetros de entrada, gerando parâmetros de saída.

3.4.3. *FUNCTION BLOCK LINKS*

Como foi descrito, os *link objects* fazem o mapeamento entre os recursos, possibilitando a troca de informações via rede.

Os *link objects* podem ser configurados para comunicar tanto variáveis como *trend objects* e notificações de alarme e eventos. Através dos *link objects*, as

características da comunicação podem ser configuradas de modo a possibilitar a troca de dados adequada.

3.4.4. *ALERT OBJECTS*

Alert objects são utilizados na comunicação da ocorrência de um alerta assim que este for detectado.

Os alertas são classificados da seguinte maneira:

- *Analog Alert*: utilizados para comunicar eventos relativos a parâmetros do tipo ponto flutuante.
- *Discret Alert*: utilizados para comunicar eventos relativos a parâmetros discretos.
- *Update Alert*: utilizados para comunicar eventos relativos mudanças nos parâmetros estáticos do bloco.

3.4.5. *TREND OBJECTS*

Os *trend object* fornecem informação sobre o comportamento de um certo parâmetros em um espaço de tempo contínuo através de registros consecutivos deste parâmetro. Os *trend object* podem ser classificados como:

- *Trend Float*: usado para coletar valores e status de pontos flutuantes em parâmetros de entrada ou de saída.
- *Trend Discrete*: usado para coletar valores e status de parâmetros de entrada ou de saída discretos.
- *Trend Bit String*: usado para coletar valores e status de *bit strings* em parâmetros de entrada ou de saída.

3.4.6. *VIEW OBJECTS*

Os *view objects* permitem a configuração e monitoramento dos blocos, tornando possível o acesso aos parâmetros de forma otimizada. As *view* dão acesso a conjuntos de parâmetros estáticos e dinâmicos, de forma rápida.

Para cada bloco, pelo menos 4 *views* devem ser definidas:

- *view* 1: acesso aos parâmetros dinâmicos de operação;
- *view* 2: acesso aos parâmetros estáticos de operação;
- *view* 3: acesso a todos os parâmetros dinâmicos;
- *view* 4: acesso aos outros parâmetros estáticos.

3.4.7. *PROGRAM INVOCATION E DOMAIN OBJECTS*

Programas ou códigos relacionados ao bloco podem ser associados a *domain objects*, assim serviços de download ou *upload* deste códigos tornam-se possíveis. O objeto *program invocation* pode associar a um *domain* um código executável. O objeto de *program invocation* pode executar serviços de execução, suspensão da execução, e pode também apagar o código. As especificações para estes serviços estão na FMS.

3.4.8. *ACTION OBJECTS*

Este objeto é opcional no *resource*. Através dele, é possível instanciar ou apagar um bloco ou um objeto.

3.5. MAPEAMENTO DOS RECURSOS DO FBAP

Para ser inserido na estrutura do VFD, todos os objetos do *Function Block Application Resource* são mapeados no OD, assim como todos os outros objetos do VFD.

Os objetos do VFD que descrevem o FBAP são:

1. *Variable Objects*: todos os parâmetros dos blocos, sejam variáveis simples, vetores, matrizes são definidos no OD. *Trend*, *Action* e *Link objects* são definidos como *record objects* e o acesso a informações por grupos pode ser feito via *variable list objects*.
2. *Event Management Objects*: utilizados para notificar alarmes e eventos.
3. *Domain Objects*: definidos como anteriormente.
4. *Program Invocation Objects*: definidos como anteriormente.

Todo o *function block applicaion* é mapeado por estes tipos objetos de comunicação.

Em conjunto com estes objetos, a DD deve ser utilizada pelo configurador ou pela estação de IHM como base de dados descritiva de cada item especificado na aplicação. Assim é importante garantir a coerência dos dados da DD com os objetos implementados.

3.5.1. FUNCTION BLOCK APPLICATION MODEL

O *Function Block Application Model* mapeia diretamente os objetos do FBAP no OD. O mapeamento do FBAP é apresentado a seguir.

3.5.2. DATA TYPE E DATA STRUCTURES

O tipo de dado de uma variável simples ou de uma matriz é definido no OD através do *data type index*. A estrutura de objetos tipo *record* é definida na descrição de objeto através da referência para *Data Structure Object*. As regras de endereçamento para estes objetos são definidas pela FMS.

No *Function Block Application*, os índices de objeto 1 a 255 são reservados para os *data type* e *data structures* padrões da *Fieldbus Foundation*. Os índices 1 a 14 e 21 são os *data types* e o índices 64 a 86 são as estruturas padrões definidas pelo *Function Block Application*.

3.5.3. OBJET EXTENSION

Todos os objetos do OD além dos *data type* e *data structure* possuem uma extensão, que é utilizada pelo FBA para serviços de nomeação, referenciamento na DD, e verificação de *link*.

A extensão de cada objeto possui 3 elementos:

- *Usage*: indica se este objeto pode ser modificado por outro bloco (I), modifica outro bloco (O), ou tem papel na comunicação outros com dispositivos (C).
- *Member ID*: identifica a função do objeto. Cada parâmetro possui um único *member ID* dentro do bloco. O *member ID* é identificado quando se utiliza o serviço de localização em conjunto com a estrutura *Tag.Parameter*.
- *Item ID*: identifica a semântica do objeto. Utilizado para referenciar o objeto na DD.

3.5.4. O OBJETO DIRETÓRIO DO OD

O diretório do OD é um *array* que tem a função de servir como um guia para a localização dos objetos do FBAP.

O diretório é definido a partir do primeiro índice do S-OD. O ponto inicial do S-OD é definido no OD *Object Description* (ODES), que tem índice 0. Além do ponto inicial do S-OD, o OD *Object Description* possui o número da versão do OD, o ponto de início e o tamanho do DV-OD e do DP-OD associados com *variable lists* e *program invocations* respectivamente.

Figura 26 - Estrutura básica do OD

3.5.4.1. Definição do Diretório

Os seguintes atributos devem ser definidos para o objeto diretório:

- *Data Type Index* dos Elementos: 6 (*Unsigned* 16)
- Tamanho do elemento: 2 octetos por elemento
- Número de Elementos: variável, limitado a 50 elementos por *array*

O diretório consiste de um cabeçalho, seguido de objetos que contém índices consecutivos. A estrutura física do OD pode ser descrito como um vetor simples composto por todos as instâncias do diretório. A implementação do OD não é definida pela *Foundation*, ficando a cargo do fabricante. A estrutura do diretório segue o modelo representado na figura 27:

Figura 27 - Estrutura do diretório do OD

A indexação no OD pode ser resumida como na representação contida na tabela 7:

Segmento do OD	Índice	Elemento
Cabeçalho	1	Reservado
	2	Número da Revisão do Diretório
	3	Número de Objetos no Diretórios
	4	Número Total de Entradas do diretório
	5	Índice da Primeira Lista de Referências Composta
	6	Número de Listas de Referências Compostas
Entradas Compostas do Diretório	7	Índice do Primeiro <i>action objects</i>
	8	Número de <i>action objects</i>
	9	Índice do Primeiro <i>link object</i> do VFD
	10	Número de <i>link objects</i> no VFD
	11	Índice do Primeiro <i>alert object</i> do VFD
	12	Número de <i>alert objects</i> no VFD
	13	Índice do Primeiro <i>trend object</i> do VFD
	14	Número de <i>trend objects</i> no VFD
	15	Índice do Primeiro <i>domain object</i> do VFD
	16	Número máximo de <i>domain objects</i> permitidos no VFD
Entradas de Listas Compostas	Índice da Lista + 1)	Número de <i>resource blocks</i> no VFD = 1
	Índice da Lista + 2)	Índice no Diretório para o ponteiro do primeiro <i>transducer block</i>
	Índice da Lista + 3)	Número de <i>transducer blocks</i> no VFD
	Índice da Lista + 4)	Índice no Diretório para o ponteiro do primeiro <i>function block</i>
	Índice da Lista + 5)	Número de <i>function blocks</i> no VFD
Entradas de Listas Compostas no Diretório de Entradas Compostas	Índice no Diretório para <i>Resource Blocks</i> ;	Primeiro índice no OD e número de índices para o <i>resource block</i>
	Índice no Diretório para <i>Transducer Blocks</i>	Primeiro índice no OD e número de índices para cada <i>transducer block</i>
	Índice no Diretório para <i>Function Blocks</i>	Primeiro índice no OD e número de índices para cada <i>function block</i>

Tabela 7 - Estrutura de indexação no diretório do OD

4. ESTIMAÇÃO DE ESTADOS NA TEORIA MODERNA DE CONTROLE

A representação de um sistema dinâmico por espaço de estados, diferentemente da representação por função transferência, é feita por meio de 2 equações matriciais que relacionam as entradas do sistema aos estados e estes com as saídas do sistema. As k entradas do sistema são denominadas u_k e formam a matriz $u_{k \times 1}$, as saídas m do sistema são denominadas y_m formam a matriz $y_{m \times 1}$. Os estados do sistema são denominados x_n e o número n de estados é o mesmo da ordem do sistema. O espaço de estados do sistema é representado por um vetor $x_{n \times 1}$. O vetor $\dot{x}$ representa a derivada no tempo de x .

A equação dinâmica e a equação de saída de um sistema estão respectivamente representadas abaixo:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}\tag{1} \text{ e } (2)$$

Nesta representação, definem-se 4 matrizes: matriz dinâmica A , matriz de controle B , matriz de saída C e matriz direta D .

A matriz A tem dimensão $n \times n$, B tem dimensão $n \times k$, C tem dimensão $m \times n$ e a dimensão de D é $m \times k$.

$$A_{n \times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}; B_{n \times k} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nk} \end{bmatrix};$$

$$C_{m \times n} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}; C_{m \times k} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1k} \\ d_{21} & d_{22} & \cdots & d_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mk} \end{bmatrix}$$

Para sistemas SISO, a matriz B é do tipo coluna e C do tipo linha.

A representação por diagrama de blocos do sistema descrito acima está na figura:

Figura 28 - Representação de um sistema dinâmico em espaço de estados

Na maioria dos sistemas a matriz direta D é nula, portanto a equação de saída pode ser simplificada por:

$$y = Cx \quad (3)$$

As características dinâmicas do sistema são determinadas pela localização dos pólos e zeros no plano $s - j\omega$. A determinação dos pólos λ do sistema é feita através dos autovalores da matriz A . Assim, o polinômio característico $Q(\lambda)$ do sistema é dado pela equação abaixo:

$$Q(\lambda) = |\lambda I - A| = 0 \quad (4)$$

Onde I é a matriz identidade de dimensão n . E a solução de $|\lambda I - A| = 0$ é o conjunto de autovalores de A .

4.1. REALIMENTAÇÃO DE ESTADOS

O sistema original pode ter seu comportamento dinâmico modificado, ou seja, a localização dos pólos alterada, por um sistema de controle regulador. A finalidade de um sistema regulador é imprimir ao sistema um determinado padrão de comportamento dinâmico ideal de projeto caso a dinâmica do sistema original não seja adequada.

Uma das maneiras de se regular um sistema é através de uma técnica de alocação de pólos denominada técnica de Realimentação de Estados. Esta técnica consiste em realizar a aquisição das grandezas adotadas como estados, multiplicá-los por ganhos específicos e somá-los com as entradas do sistema, conforme a figura 29:

Figura 29 - Realimentação de estados

Temos assim que adotar uma matriz de ganhos K de dimensão $n \times k$. Assim, o vetor de entradas u pode ser escrito como:

$$u = r + K \cdot x \quad (5)$$

Onde r é denominado vetor de entradas de referência e tem a mesma dimensão de u .

Substituindo-se a equação (5) em (1), temos:

$$\dot{x} = Ax + B(r + K \cdot x) \quad (6)$$

Portanto:

$$\dot{x} = (A + B.K)x + B.r \quad (7)$$

Temos assim por analogia a fórmula (1), uma nova matriz dinâmica de malha de realimentação de estados fechada A_f , onde:

$$A_f = A + B.K \quad (8)$$

Isto resulta em uma nova equação característica de malha fechada:

$$Q(\lambda) = |\lambda.I - A - B.K| = |\lambda.I - A_f| = 0 \quad (9)$$

Os novos autovalores do sistema são portanto função dos elementos da matriz K , assim, o projeto de alocação de pólos por realimentação de estados se resume ao cálculo dos elementos da matriz K . Para sistemas SISO, definidos os autovalores desejados, a matriz K é única. Já para sistemas MIMO existe um número infinito de matrizes de K que são correspondente a um certo conjunto de autovalores.

Existe uma condição necessária e suficiente para a seleção de K , o sistema deve ser completamente controlável.

Um sistema é dito completamente controlável se existir uma única ação de controle ou excitação u capaz de levar o sistema de um estado inicial $x(t)$ para um estado final $x(t+\Delta t)$. Ou seja, um sistema é completamente controlável se através de ações de controle seja possível modificar todos os estados.

Numericamente, a controlabilidade é uma propriedade da matriz de controlabilidade. Esta matriz é definida como:

$$M_c = \begin{bmatrix} B & A.B & A^2.B & \dots & A^{n-1}.B \end{bmatrix} \quad (10)$$

A matriz de controlabilidade depende somente das matrizes dinâmica A e de entrada B . O posto desta matriz deve ser numericamente igual ao número de estados n do sistema para que a controlabilidade seja completa. A diferença de módulo entre

o número de estados e o posto de M_c será igual ao número de estados insensíveis a ação de controle, ou seja estados incontroláveis.

Condição de controlabilidade:

$$\text{posto}(M_c) = \text{posto} \begin{bmatrix} B & A.B & A^2.B & \dots & A^{n-1}.B \end{bmatrix} = n \quad (11)$$

4.2. ESTIMAÇÃO DE ESTADOS

A técnica de alocação de pólos por realimentação de estados assume que todos os estados estejam disponíveis para aquisição. Em muitos sistemas de controle é fisicamente ou economicamente inviável instalar os transdutores necessários para medir todos os estados.

Para contornar tal problema, desenvolveu-se uma técnica denominada observador de estados. O observador de estados é um dispositivo computacional que avalia os estados do sistema baseado em informações nas da entrada u e da saída y do sistema. Esta reconstrução de todos os estados só será possível se o sistema for completamente observável.

A observabilidade é a propriedade de se reconstruir os estados a partir da entrada e da saída do sistema. Para calcular a observabilidade é necessário montar a matriz de observabilidade:

$$M_o = \begin{bmatrix} C & A.C & A^2.C & \dots & A^{n-1}.C \end{bmatrix} \quad (12)$$

De maneira similar ao conceito de controlabilidade, a diferença do posto de M_o frente ao número de estados do sistema é equivalente ao número de estados não observáveis do sistema.

Condição de observabilidade:

$$\text{posto}(M_o) = \text{posto} \begin{bmatrix} C & A.C & A^2.C & \dots & A^{n-1}.C \end{bmatrix} = n \quad (13)$$

O propósito de se utilizar um observador é utilizar os estados reconstruídos denominados $\hat{x}$ na realimentação.

$$u = r + K.\hat{x} \quad (14)$$

O método utilizado para reconstruir os estados é simular em tempo real as equações dinâmicas e de saída com sob as mesmas condições de entrada e saída do sistema real. Entretanto o sistema real está sujeito a perturbações e distúrbios que o modelo simulado não está, assim a diferença entre a saída real y e a simulada $\hat{y}$ também é uma entrada do observador. A figura 30 é uma representação por diagrama de blocos do observador:

Figura 30 - Observação de estados

A equação dinâmica e de saída do observador são:

$$\begin{aligned} \dot{\hat{x}} &= A.\hat{x} + B.u + L(y - \hat{y}) \\ \hat{y} &= C.\hat{x} \end{aligned} \quad (15) \text{ e } (16)$$

A matriz L é denominada matriz do observador e deve ser projetada de modo que $\hat{x}$ convirja para x dentro de um espaço de tempo aceitável. Define-se:

$$e \equiv x - \hat{x} \quad \text{e} \quad \dot{e} \equiv \dot{x} - \dot{\hat{x}}$$

Utilizando-se as equações (1), (2) nas equações (15) e (16), temos que:

$$\dot{e} = (A - LC)e \quad (17)$$

Esta é uma equação diferencial homogênea que representa um sistema dinâmico, no qual define-se L de modo que os pólos deste sistema dinâmico estejam no semi-plano esquerdo em uma posição adequada para que a velocidade de convergência do erro seja expressivamente maior que a velocidade de resposta do sistema real, assim, para qualquer condição inicial temos:

$$\lim_{t \rightarrow \infty} e(t) = 0$$

Um sistema regulador por realimentação de estados reconstruídos segue portanto a seguinte lei:

$$u = r + K.\hat{x} = r + K.(x - e) \quad (18)$$

Assim, a equação dinâmica geral fica:

$$\dot{x} = (A + B.K)x - B.K.e + B.r \quad (19)$$

O sistema em malha fechada com observador pode ser representado pela seguinte equação matricial:

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A + B.K & -B.K \\ 0 & A - L.C \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} . r \quad (20)$$

A correspondente equação característica é:

$$Q(\lambda) = |\lambda.I - (A + B.K)| |\lambda.I - (A - L.C)| = 0 \quad (21)$$

Desta forma pode-se notar que os pólos do observador e da realimentação de estados podem ser determinados de forma independente, ou seja, o projeto da matriz K e da matriz L são independentes. Assim o comportamento dinâmico do sistema será uma composição da dinâmica do sistema realimentado com a dinâmica do observador. A figura 31 representa um sistema regulado com observador:

Figura 31 - Realimentação de estados estimados

Através da equação (21) nota-se que se um processo original tem ordem n , o processo regulado e dotado de observador terá ordem $2n$. Neste caso é conveniente que os pólos do observador sejam projetados para que não tenham influência na resposta transitória do sistema.

4.3. INTRODUÇÃO DE SINAL DE REFERÊNCIA

Quando se leva em conta a entrada de um sinal de referência na malha, como na figura acima, o estimador pode ser utilizado com algumas variações quanto a sua disposição na malha. Pode-se colocar o estimador no caminho direto da malha ou no caminho de realimentação, conforme a figura 32:

Figura 32 - Estimador de estados no caminho direto e de realimentação

Analizando a dinâmica de malha fechada do sistema com o estimador no caminho de realimentação, no caso da aplicação de uma entrada referência, esta entrada afetará a planta e o estimador no mesmo instante, o que mantém o erro do estimador nulo durante a toda a fase transitória, não excitando assim os seus pólos. A resposta dinâmica global do sistema, portanto, corresponde a caracterizada pela Equação (9).

Isto significa que os zeros do sistema global se localizam sobre os pólos do estimador, o que caracteriza uma grande vantagem desta configuração.

No caso do estimador no caminho direto, qualquer variação na referência afetará primeiramente o estimador, causando a excitação de todos os seus pólos, só então a planta será afetada, causando assim uma resposta dinâmica global composta de duas dinâmicas em sobreposição: a da planta e a do estimador, conforme a Equação (21).

Entre os dois casos, o estimador no caminho de realimentação é o mais atrativo do ponto de vista prático.

O estimador no caminho direto, entretanto, deve ser utilizado no caso em que se dispõe de um sensor que mede diretamente o erro da medida, como por exemplo um termostato. Neste caso, a presença do estimador pode ocasionar um sobresinal elevado na resposta do sistema, pois o estimador causará um efeito semelhante ao que se observaria com um compensador de Avanço na mesma posição. Este tipo de estimador é conhecido como seguidor de erro.

Quando se utiliza o estimador na realimentação, deve-se imprimir ganhos à entrada referência, de modo a adequar o sistema a uma certa dinâmica transitória e ausência de erro em regime.

Neste caso os ganhos para a referência devem ser diferenciados para o estimador e para a planta, este tipo de técnica é conhecida como alocação de zeros, e está indicada pelo esquema da figura 33:

Figura 33 - Estimador em alocação de zero

As equações dinâmicas que regem o sistema passam a incorporar os ganho da referência e podem ser escritas assim:

$$\begin{aligned}\dot{\hat{x}} &= (A - B.K - L.C)\hat{x} + L.y + M.r \\ u &= -K.\hat{x} + N.r\end{aligned}\tag{22}$$

Selecionando-se adequadamente M e N , garante-se a máxima flexibilidade de projeto, com a possibilidade de alocar todos os zeros do sistema em quaisquer posições. É importante notar que a presença dos ganhos de referência não afetam os pólos do sistema, portanto não podem mudar a estabilidade deste, apenas modificar o comportamento da resposta transitória.

Um caso particular da técnica de alocação de zeros é a chamada malha com estimador autônomo. Neste caso implementa-se $M = B.N$ e a malha se modifica para a indicada na figura 34:

Figura 34 - Malha com estimador autônomo

Este caso denomina-se autônomo pois o erro do estimador é independente da entrada referência, em consequência da incontrolabilidade de $\hat{x}$ pela referência. A seleção de N neste caso deve ser tal o ganho estático do sistema em malha fechada seja unitário. As equações dinâmicas para este tipo de estimador são:

$$\begin{aligned}\dot{\hat{x}} &= (A - L.C)\hat{x} + L.y + B.u \\ u &= -K.\hat{x} + N.r\end{aligned}\tag{23}$$

Porém, este fator pode caracterizar um problema pois existem plantas nas quais o ganho estático varia com o tempo, e portanto a técnica acima descrita não garante erro zero em regime. Como alternativa a este problema, pode-se utilizar uma malha de controle com realimentação de saída e um integrador com ganho no erro. Neste caso, N pode ser desconsiderado e igualado a um, resultando no esquema da figura 35:

Figura 35 - Estimador com realimentação de saída

O caso da malha com estimador seguidor de erro também é um caso particular da malha por alocação de zeros. Para configurar o seguidor de erros basta fazer $N = 0$ e $M = -L$, Com esta configuração, a Equação (22) passa a:

$$\begin{aligned}\dot{\hat{x}} &= (A - L.C - B.K)\hat{x} + L.(y - r) \\ u &= -K.\hat{x}\end{aligned}\tag{24}$$

5. BLOCO FUNCIONAL ESTIMADOR DE ESTADOS

O projeto do Estimador de Estados atende aos requisitos obrigatórios da norma *Foundation* e aos fundamentos teóricos da teoria de controle moderna. O seu algoritmo realiza a estimação plena dos estados de um sistema dinâmico SISO (*single input single output*) de terceira ordem, portanto com 3 estados. Tem a tarefa de realizar a estimação dos estados e aplicar os ganhos, relativos a matriz K , baseado no modelo do sistema descrito pelas matrizes dinâmica A , de entrada B e de saída C , e na matriz L do observador e em informações da entrada e da saída do sistema.

5.1. INTEGRAÇÃO

A norma *Foundation* sugere uma metodologia para a implementação de novos Function Blocks, apresentando os passos necessários. A implementação do bloco Estimador de Estados seguiu o modelo proposto pela norma, o qual que está descrito adiante:

• Passo Um	Inicialmente deve-se modelar o funcionamento do bloco e a partir deste modelo, deve-se identificar as variáveis que serão acessadas por procedimentos de fábrica e aquelas que serão acessadas pelo usuário com o objetivo de diagnóstico, configuração e operação. Pode-se então montar uma lista de variáveis com estas características.
• Passo Dois	<p>A lista de Variáveis deve ser dividida em três partes de acordo com o uso das variáveis:</p> <ol style="list-style-type: none"> 1. Variáveis que descrevem o recurso, como espaço de memória e mínimo tempo de execução. Estas variáveis não são editadas pelo usuário 2. Variáveis configuradas ou verificadas em fábricas ou no lugar de instalação, apenas uma vês. Exemplo: unidade de Engenharia, offset e span. 3. Variáveis verificadas periodicamente pelo usuário por procedimentos de manutenção ou operação. Exemplo: alarmes, setpoints, sintonias e modo de operação. <p>Neste caso, as variáveis são definidas no casos 1 e 2, e os parâmetros são definidos como da classe 3.</p>
• Passo três	As variáveis descrevendo os recursos são parâmetros para o resource block do transmissor, além daqueles parâmetros do resource block definidos pela norma. As variáveis definidas como sendo de configuração de fábrica ou de uso na instalação do sistema são parâmetros a serem adicionados ao transducer block.
• Passo Quatro	Os parâmetros identificados como sendo de uso periódico, em operação ou manutenção são parâmetros do futuro <i>function block</i> .

Tabela 8 - Projeto de parâmetros de blocos funcionais

Referência: FIELDBUS FOUNDATION (1998). *FF 890*

A norma recomenda que os parâmetros do novo bloco sejam comparados como os parâmetros dos blocos padrões, a nível de funcionalidade. Caso seja identificada alguma semelhança nos parâmetros do bloco novo com os parâmetros de um bloco padrão, então este deverá servir como base para o projeto, assim bastará adicionar os parâmetros inéditos no bloco padrão. Estes parâmetros são variáveis que devem ser mapeadas como variáveis simples, vetores ou records, de acordo como os tipos de dados padrões e então listados em seguida aos parâmetros padrões na definição do bloco.

Os parâmetros de um bloco não padrão são classificados de acordo com a regra acima em parâmetros de *device* normalizados e parâmetros específicos do fabricante.

Caso nenhum bloco padrão sirva de modelo para o novo bloco, os parâmetros deste, caso seja possível, deverão ter as mesmas características dos parâmetros padrões, como nome, função, status e tipo de operação.

A Fieldbus Foundation prevê a possibilidade de normalização tanto de parâmetros como de blocos específicos de fabricante caso estes tenham utilidade abrangente.

Na implementação computacional do bloco deve-se criar estruturas que disponibilizem para a camada de aplicação os serviços de inicialização, execução e de leitura e escrita para cada parâmetro disponível. Após esta implementação, faz-se a integração destas estruturas, ou métodos, nas estruturas complementares ao FBAP: a *Device Description* e o *Object Dictionary*.

A DD do transmissor deve incluir a descrição do bloco e dos seus parâmetros. Na caracterização dos parâmetros temos: tipo de dado, permissão de acesso, *label*, texto de *help* e a classe do parâmetro (estático, de entrada ou de saída). Já a descrição do bloco na DD inclui o fabricante, a revisão, todos o parâmetros, as views e os grupos de unidades relacionadas.

Quando compilada, a DD fornece dois números inteiros denominados *Member ID* e *Item ID* para cada parâmetro, os quais são utilizados na inclusão dos parâmetros do bloco no OD, juntamente com outras informações que caracterizam o parâmetro, entre as quais as mostradas adiante na Tabela PARAMETRIZAÇÃO DO ESTIMADOR DE ESTADOS.

Informações sobre o bloco devem estar presente também no arquivo *Capability file*, utilizado pelo configurador juntamente com o arquivo de DD para uma correta identificação e configuração de cada diferente transmissor.

No arquivo de *Capability* do transmissor estão as informação de tempo de execução do bloco e quantidade de memória RAM necessária para armazenar todos os seus parâmetros, assim como o *Item ID* do bloco e o número de parâmetros deste.

5.2. ESTRUTURA E FUNÇÕES

O bloco funcional possui um sistema de equações dinâmicas matriciais (Equações (15) e (16)) que são executadas em *loop* num processo de convergência. A cada execução do bloco, estas equações são executadas um número constante de vezes, no caso 5 vezes, de modo que o tempo de execução do bloco por definição

seja aproximadamente o tempo de execução de um bloco de controle PID. Deste modo, a convergência do bloco pode se dar em algumas execuções, dependendo das características dinâmicas da planta.

A seguir está uma representação esquemática do bloco, com seu algoritmo, entradas e saídas:

Figura 36 - Bloco Funcional Estimador de Estados

Ao usuário cabe configurar, através dos 34 parâmetros de configuração disponíveis, as matrizes do modelo dinâmico da planta controlada (A, B e C), a matriz do observador (L), as unidades das grandezas de entrada e os ganhos (K) para os estados estimados, estes que são as saídas do bloco. Existem ainda os parâmetros de visualização da operação do bloco BLOCK_ERR, CONVERGENCE e OUTPUT_ERR. O conjunto completo de parâmetros do bloco estão listados na tabela descritiva a seguir, com todas as informações necessárias para a completa caracterização de cada parâmetro:

Índi- ce	Parâmetro	Tipo de Dado (tamanho)	Faixa Válida / Opções	Valor Default	Unid.	M. A.	Descrição
1	St_rev	Unsigned16		0	Na	S	Revisão estática do bloco
2	Tag_desc	OctString(32)		Espaços	Na	S	Rótulo descritivo
3	Strategy	Unsigned16		0	Na	S	Identificador de grupo de blocos
4	Alert_key	Unsigned8	1 a 255	0	Na	S	Identificador da unidade da planta
5	Mode_blk	DS-69	Auto / OOS/ Man	OOS	E	S	Relação dos modos permitidos e atual

6	Block_err	BitString(2)			E	D / RO	Status de erro do bloco
7	IN_1	DS-65			IN	D	Entrada da variável de atuação no sistema
8	IN_2	DS-65			IN	D	Entrada da variável de saída do sistema
9	Output_err	Float			Na	D / RO	Diferença entre a saída do sistema real e a saída do modelo no observador
10	L11	Float		1	Na	S	Elemento da matriz L do observador
11	L12	Float		1	Na	S	Elemento da matriz L do observador
12	L13	Float		1	Na	S	Elemento da matriz L do observador
13	A11	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
14	A12	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
15	A13	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
16	A21	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
17	A22	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
18	A23	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
19	A31	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
20	A32	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
21	A33	Float		1	Na	S	Elemento da matriz dinâmica A do sistema
22	B11	Float		1	Na	S	Elemento da matriz de entrada B do sistema
23	B21	Float		1	Na	S	Elemento da matriz de entrada B do sistema
24	B31	Float		1	Na	S	Elemento da matriz de entrada B do sistema
25	C11	Float		1	Na	S	Elemento da matriz de saída C do sistema
26	C12	Float		1	Na	S	Elemento da matriz de saída C do sistema
27	C13	Float		1	Na	S	Elemento da matriz de saída C do sistema
28	Pre_X1	Float		0	Na	D / RO	Valor do estado 1
29	Pre_X2	Float		0	Na	D / RO	Valor do estado 2
30	Pre_X3	Float		0	Na	D / RO	Valor do estado 3
31	OUT_1	DS-65			Na	D / Man	Valor do estado 1 multiplicado por Gain_1
32	OUT_2	DS-65			Na	D / Man	Valor do estado 2 multiplicado por Gain_2
33	OUT_3	DS-65			Na	D /	Valor do estado 3

						Man	multiplicado por Gain_3
34	Gain_1	Float		1	Na	S	Ganho aplicado ao estado 1
35	Gain_2	Float		1	Na	S	Ganho aplicado ao estado 2
36	Gain_3	Float		1	Na	S	Ganho aplicado ao estado 3
37	Convergence	Unsigned8	1 = Not Converged 2 = Converged	1	E	S	O identificador de convergência do estimador
38	IN_1_UNITS	Unsigned16		0	IN	S	Unidade de engenharia da entrada IN_i para o display
39	IN_2_UNITS	Unsigned16		0	IN	S	Unidade de engenharia da entrada IN_o para o display
40	Block_Alm	DS-72			Na	D	Status de alarme do bloco
41	Update_Evt	DS-73			Na	D	Alerta gerado pela alteração de parâmetros estáticos
42	Grant_Deny	DS-70		0	Na	D	Controle de acesso para aplicações remotas

Legenda: M. A. - Modo de Armazenamento; E - Parâmetro *enumerated*; Na - Parâmetro adimensional; RO - Somente leitura; D - dinâmico; N - não-volátil; S - estático

Tabela 9 - Parametrização do Estimador de Estados

Além do modo OOS (*Out of Service*), no qual o bloco fica "desligado", este pode ser executado em dois modos: em automático onde a estimação dos estados é feita normalmente e as saídas são atualizadas com os estados estimados ou em manual, onde a estimação também é feita porém as saídas são diretamente estipuladas pelo usuário, sendo que o valor *default* destas saídas em modo manual é zero.

As condições de execução do algoritmo não mudam o modo do bloco, nem com as entrada em status *Bad*. Caso ocorreram divisões por zero ou outra operação indefinida, a execução do bloco não é abortada.

Os status das entradas IN_1 e IN_2 não afetam o algoritmo de execução, sendo que os *status* de X1, X2 e X3 são semelhantes ao pior *status* entre IN_1 e IN_2. Este tipo de implementação garante o mecanismo da norma *Foundation* de propagação de *status* na malha, como ferramenta para a garantia de segurança do sistema em caso de falhas ou emergências.

No caso de perda de energia no transmissor, todos os parâmetros de configuração são armazenados em memória não volátil, garantindo assim uma correta reinicialização da operação normal do bloco assim que o transmissor é novamente ligado, sem a necessidade de reconfiguração.

O sistema de alarme do Estimador de Estados segue os requisitos básicos da *Foundation*, portanto o bloco pode ser perfeitamente integrado a qualquer tipo de malha de controle.

Em operação Man ou Auto, o bloco deve recuperar os valores de Pre_X1, Pre_X2 e Pre_X3 da execução anterior para serem utilizados na execução seguinte.

O parâmetro CONVERGENCE indica se o sistema de equações dinâmicas do bloco está respondendo da mesma forma que a planta, ou seja, se a saída do modelo convergiu ou não para IN_2. Para determinar esta convergência foi utilizado uma margem de 5% de erro.

5.3. MALHAS DE CONTROLE POR ESTIMAÇÃO DE ESTADO

As malhas de controle por estimação de estados descritas no capítulo anterior foram projetadas e implementadas no Foundation Fieldbus, distribuídas entre dois transmissores. Para a implementação foram necessários, além do bloco funcional Estimador de Estados, os blocos padrões Aritmético (ARTH), Entrada Analógica (AI), Saída Analógica (AO), Controlador PID (PID) e Constante (CT) somente no caso da malha com alocação de zero. O bloco funcional Constante é um bloco não padrão definido pela SMAR, que possui seis saídas analógicas e duas digitais constantes e configuráveis pelo usuário.

A configuração dos blocos de cada uma das malhas é variável, pois para cada tipo de grandeza física, os parâmetros de escalas, limites e alarmes são diferentes. Entretanto alguns parâmetros de configuração dos blocos não mudam quaisquer que sejam as aplicações, nestes parâmetros se definem as operações da malha e sua configuração básica.

A malha de controle com alocação de zeros e estimador (vide Figura 33) está representada na figura 37:

Figura 37 - Malha com alocação de zeros em diagrama de blocos funcionais

Sua configuração básica está na tabela a seguir:

Malha com alocação de zeros		
Bloco Funcional	Parâmetro	Valor
AI	Mode Block.Target	Auto
	Mode Block.Target	Auto
ARTH_1	Arith_type	Traditional summer
	Gain_In_1	M / L
ARTH_2	Mode Block.Target	Auto
	Arith_type	Traditional summer
	Gain	N
	Gain_In_1	K1 / N
	Gain_In_2	K2 / N
EE	Gain_In_3	K3 / N
	Mode Block.Target	Auto
	Gain_X1	1
	Gain_X2	1
CT	Gain_X3	1
	Mode Block.Target	Auto
AO	Out_1	Referência
	Mode Block.Target	Cas
Links		
1	CT.OUT_1 -> ARTH_1.IN_1	

2	AI.OUT -> ARTH_1.IN
3	CT.OUT_1 -> ARTH_2.IN
4	ARTH_1.OUT -> EE.IN_2
5	ARTH_2.OUT -> AO.CAS_IN
6	ARTH_2.OUT -> EE.IN_1
7	EE.OUT_1 -> ARTH_2.IN_1
8	EE.OUT_2 -> ARTH_2.IN_2
9	EE.OUT_3 -> ARTH_2.IN_3

Tabela 10 - Configuração básica da malha com alocação de zeros

A malha com estimador autônomo (vide Figura 34) está na figura 38:

Figura 38 - Malha com estimador autônomo ou seguidor de erro baseada em blocos funcionais

A configuração básica para o estimador autônomo:

Malha com estimador autônomo		
Bloco Funcional	Parâmetro	Valor
AI	Mode Block.Target	Auto
EE	Mode Block.Target	Auto
	Gain_X1	1
	Gain_X2	1
	Gain_X3	1
ARTH	Mode Block.Target	Auto
	Arith_type	Traditional summer
	Gain	N
	Gain_In_1	K1 / N
	Gain_In_2	K2 / N
	Gain_In_3	K3 / N
	In	Referência
AO	Mode Block.Target	Cas
Links		
1	AI.OUT -> EE.IN_2	
2	EE.OUT_1 -> ARTH.IN_1	
3	EE.OUT_2 -> ARTH.IN_2	
4	EE.OUT_3 -> ARTH.IN_3	
5	ARTH.OUT -> EE.IN_1	
6	ARTH.OUT -> AO.CAS_IN	

Tabela 11 - Configuração básica para malha com estimador autônomo

A malha de controle com estimador seguidor de erro também é representada do modo acima, desde que o sensor utilizado tenha como saída a diferença entre o valor de referência e o valor da grandeza física lida da planta, sua configuração está a diante:

Malha seguidora de erro com sensor referencial		
Bloco Funcional	Parâmetro	Valor
AI	Mode Block.Target	Auto
EE	Mode Block.Target	Auto
	Gain_X1	K1
	Gain_X2	K2
	Gain_X3	K3
ARTH	Mode Block.Target	Auto
	Arith_type	Traditional summer
	Gain	1
	Gain_In_1	1
	Gain_In_2	1
AO	Mode Block.Target	Cas
Links		
1	AI.OUT -> EE.IN_2	
2	EE.OUT_1 -> ARTH.IN	
3	EE.OUT_2 -> ARTH.IN_1	
4	EE.OUT_3 -> ARTH.IN_2	
5	ARTH.OUT -> EE.IN_1	
6	ARTH.OUT -> AO.CAS_IN	

Tabela 12 - Configuração da malha seguidora de erro

Para utilizar o estimador autônomo com realimentação de saída, conforme a Figura 35, a malha passa a ser configurada do seguinte modo:

Figura 39 - Malha com estimador autônomo e realimentação de saída

A respectiva configuração:

Malha com estimador autônomo e realimentação de saída		
Bloco Funcional	Parâmetro	Valor
AI	Mode Block.Target	Auto
EE	Mode Block.Target	Auto
	Gain_X1	1
	Gain_X2	1
	Gain_X3	1
ARTH	Mode Block.Target	Auto
	Arith_type	Traditional summer
	Gain	1
	Gain_In_1	K1
	Gain_In_2	K2
	Gain_In_3	K3
PID	Mode Block.Target	Auto
	SP	Referência
	Rate	0
	Gain	1
	Reset	Ti
AO	Mode Block.Target	Cas
Links		
1	AI.OUT -> PID.IN	
2	AO.BKCLA_OUT -> PID.BKCAL_IN	
3	PID.OUT -> ARTH.IN	
4	EE.OUT_1 -> ARTH.IN_1	
5	EE.OUT_2 -> ARTH.IN_2	
6	EE.OUT_3 -> ARTH.IN_3	
7	AI.OUT -> EE.IN_2	
8	ARTH.OUT -> AO.CAS_IN	
9	ARTH.OUT -> EE.IN_1	

Tabela 13 - Configuração da malha com estimador autônomo e realimentação de saída

No caso da malha seguidora de erro mas com o uso de um sensor convencional, ou seja, em que a saída corresponde a grandeza física medida, a malha se configura da maneira representada na figura 40:

Figura 40 - Malha seguidora de erro com sensor convencional

A configuração passa a ser:

Malha seguidora de erro com sensor convencional		
Bloco Funcional	Parâmetro	Valor
AI	Mode Block.Target	Auto
	Mode Block.Target	Auto
	Gain_X1	K1
	Gain_X2	K2
EE	Gain_X3	K3
	Mode Block.Target	Auto
	Arith_type	Traditional summer
	Gain_In_1	1
ARTH	Gain_In_2	1
	Gain_In_3	1
	Mode Block.Target	Auto
	SP	Referência
PID	Rate	0
	Gain	1
	Reset	0
	Mode Block.Target	Auto
AO	Mode Block.Target	Cas
Links		
1	AI.OUT -> PID.IN	
2	AO.BKCLA_OUT -> PID.BKCAL_IN	
3	PID.OUT -> EE.IN_2	
4	EE.OUT_1 -> ARTH.IN_1	
5	EE.OUT_2 -> ARTH.IN_2	
6	EE.OUT_3 -> ARTH.IN_3	
7	ARTH.OUT -> AO.CAS_IN	
8	ARTH.OUT -> EE.IN_1	

Tabela 14 - Configuração da malha seguidora de erro com sensor convencional

6. TESTES E RESULTADOS

Na realização dos testes de campo do Estimador de Estados, foi utilizado um sistema dinâmico térmico, este sistema é composto de um forno elétrico instrumentado com transmissores *Foundation Fieldbus* certificados: um transmissor de temperatura TT302 da SMAR e um conversor *fieldbus* – 4 a 20ma FI302 da SMAR, no qual está integrado o bloco funcional Estimador de Estados, a figura 41 é um esquema da montagem do sistema:

Figura 41 - Forno elétrico instrumentado com *fieldbus*

Os dois transmissores estão em um canal do *fieldbus* ligado a uma placa de interface PCI302 da SMAR, esta placa PCI é o LAS do canal *fieldbus*. Para configuração do sistema foi utilizado o software Syscon da SMAR e para a

supervisão, o TagView também da SMAR. A figura 42 mostra a tela do Syscon com a configuração do canal utilizado:

Figura 42 - Canal *fieldbus* no configurador Syscon

Cabe notar que os blocos funcionais componentes da malha de controle com estimador autônomo estão distribuídos entre os dois transmissores da maneira mostrada pela figura.

O transmissor de temperatura está ligado a um termopar tipo K, o qual mede a temperatura de dentro do forno. A compensação de junta fria é realizada pelo transmissor através da leitura de um termistor PT100 à temperatura ambiente.

O conversor de *fieldbus* para 4 a 20 mA está acionando uma chave estática a qual alimenta a resistência elétrica que fornece potência ao forno.

O sistema dinâmico utilizado possui entrada em porcentagem de potência (%) (onde vale a relação proporcional direta entre 0 a 100% e 4 a 20 mA) e saída em temperatura (°C), e é composto por cinco elementos dinâmicos em série:

1. Transmissor FI302
2. Chave estática
3. Forno
4. Termopar
5. Transmissor TT302

6.1. MODELAGEM DA PLANTA

A função transferência deste sistema composto foi avaliada via análise gráfica da resposta a entrada degrau. A curva de resposta do sistema ao degrau é característica de sistemas de primeira ordem com tempo morto, e os parâmetros desta curva foram estimados em:

- Ganho estático = $1.9 [^{\circ}\text{C} / \%]$
- Constante de tempo = $196 [\text{s}]$
- Tempo morto = $13 [\text{s}]$

A função transferência do sistema pode ser considerada como a integral de convolução da função de primeira ordem do forno com a função de segunda ordem provinda da linearização da função transferência do tempo morto por expansão em série de Taylor. Neste caso temos:

$$FT_{ma} = (327.8600.s^2 - 100.8800.s + 15.5200) / (33124.s^3 + 10361.s^2 + 1620.s + 8)$$

Na forma de espaço de estados:

$$A = \begin{bmatrix} -0.3128 & -0.0489 & -0.000241 \\ 1.000 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$C = [0.00969387755102 \quad -0.00298273155416 \quad 0.00045888177756]$$

$$D = [0]$$

6.2. CONFIGURAÇÃO DO ESTIMADOR DE ESTADOS

Para a implementação do modelo da planta, as matrizes A, B e C, no bloco funcional Estimador de Estados é necessário converter o modelo do domínio contínuo para o discreto, ou seja é necessário discretizar as matrizes. Esta conversão deve ser realizada com a inclusão de um *holder* de ordem zero. O tempo de amostragem do sistema digital deve ser calculado com base no macrociclo da estratégia de controle utilizada, neste caso 550ms.

O período de 550 ms de macrociclo se adequa perfeitamente ao sistema dinâmico controlada, que apresenta velocidade cerca de 350 mais lenta. Desta maneira não se caracteriza a influência da discretização do controle no desempenho deste, ou seja, o sistema de aquisição e controle pode ser considerado contínuo.

Durante o período do macrociclo, o Estimador de Estados executa uma única vez, porém o modelo dinâmico interno realiza cinco *loops* a cada execução, portanto o valor de tempo do macrociclo deve se dividido por cinco para se achar o correto valor do período de amostragem para a discretização.

Discretizando-se o modelo descrito em A, B e C com *holder* de ordem zero e $(550 \text{ ms}) / 5 = 110 \text{ ms}$ de taxa de amostragem, temos as seguintes matrizes dinâmicas discretas:

$$A = \begin{bmatrix} 0.96588859837722 & -0.00528920818842 & -0.00002611241333 \\ 0.10811844737734 & 0.99970743768638 & -0.00000144448986 \\ 0.00598091026074 & 0.10998924230584 & 0.99999994688281 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.10811844737734 \\ 0.00598091026074 \\ 0.00021993172406 \end{bmatrix}$$

$$C = [0.00969387755102 \quad -0.00298273155416 \quad 0.00045888177756]$$

O cálculo de L leva em conta a velocidade relativa do observador em relação a planta. Como o sistema *fieldbus* é bastante imune a ruídos, pode-se trabalhar com o

estimador seis vezes mais rápido do que a planta, o que seria inviável em ambientes ruidosos. Neste caso, temos:

$$L = \begin{bmatrix} 0.00139991262557 \\ -0.27438287461208 \\ 53.77904342396437 \end{bmatrix}$$

Para a validação do bloco desenvolvido, será implementada a malha de controle com o estimador autônomo, conforme a figura 43:

Figura 43 - Malha com estimador autônomo

No *fieldbus*, esta malha por blocos funcionais será configurada conforme a figura 44:

Figura 44 - Malha com estimador autônomo em blocos funcionais

O sinal de referência na malha acima é gerado por um bloco funcional “Constante”, no qual os valores das saídas disponíveis são configurados manualmente e permanecem constantes.

Para a malha descrita, implementou-se um sistema em malha fechada com resposta característica de um sistema de primeira ordem.

Como o sistema é de terceira ordem, o pólo real deve ser dominante ao par de pólos complexos. Os três pólos originais da planta em malha aberta são:

$$P_{ma} = \begin{bmatrix} -0.15384615384615 + 0.15384615384615i \\ -0.15384615384615 - 0.15384615384615i \\ -0.00510204081633 \end{bmatrix}$$

O requisito de controle deve ser resposta de primeira ordem dominante com constante de tempo $T = 250$ s, deste modo os pólos de malha fechada devem migrar para:

$$P_1 = \begin{bmatrix} -0.15384615384615 + 0.15384615384615i \\ -0.15384615384615 - 0.15384615384615i \\ -0.00400000000000 \end{bmatrix}$$

Esta configuração de pólos acarreta na seleção da seguinte matriz K_1 :

$$K_1 = [-0.00110204081633 \quad -0.00033908948195 \quad -0.00005216761261]$$

A figura 45 mostra a tela de configuração de blocos funcionais do Syscon, com o bloco Estimador de Estados sendo monitorado *on line*:

On Line: FI302EE - State Estimator - EE-FI302

Parameter	Operate	Value	Quality	Changed	Offset	Handling
ST_REV		27	Good Non Specific Not Limi		1	RO
TAG_DESC		Estimador de Estados	Good Non Specific Not Limi V		2	RW
STRATEGY		0	Good Non Specific Not Limi		3	RW
ALERT_KEY		0	Good Non Specific Not Limi		4	RW
MODE_BLK					5	
TARGET		Auto	Good Non Specific Not Limi		1	RW
ACTUAL		Auto	Good Non Specific Not Limi		2	RO
PERMITTED		Auto Man OOS	Good Non Specific Not Limi		3	RW
NORMAL		OOS	Good Non Specific Not Limi		4	RW
BLOCK_ERR		<None>	Good Non Specific Not Limi		6	RO
IN_1					7	
IN_2					8	
OUTUT_ERR		0	Good Non Specific Not Limi		9	RO
L11		0.0013710484	Good Non Specific Not Limi		10	RW
L12		-0.26872548	Good Non Specific Not Limi		11	RW
L13		52.670197	Good Non Specific Not Limi		12	RW
A11		0.96588886	Good Non Specific Not Limi		13	RW
A12		-0.0052892081	Good Non Specific Not Limi		14	RW
A13		-2.6112413E-05	Good Non Specific Not Limi		15	RW
A21		0.10811844	Good Non Specific Not Limi		16	RW
A22		0.99970746	Good Non Specific Not Limi		17	RW
A23		-1.4444898E-06	Good Non Specific Not Limi		18	RW
A31		0.0059309103	Good Non Specific Not Limi		19	RW
A32		0.10998924	Good Non Specific Not Limi		20	RW
A33		0.99999994	Good Non Specific Not Limi		21	RW
B11		0.10811844	Good Non Specific Not Limi		22	RW
B21		0.0059309103	Good Non Specific Not Limi		23	RW
B31		0.00021993172	Good Non Specific Not Limi		24	RW
C11		0.0058979594	Good Non Specific Not Limi		25	RW
C12		-0.0030455253	Good Non Specific Not Limi		26	RW
C13		0.00046854245	Good Non Specific Not Limi		27	RW
FRE_X1		0.026497062	Good Non Specific Not Limi		28	RO
FRE_X2		-5.1930304	Good Non Specific Not Limi		29	RO
FRE_X3		1017.8328	Good Non Specific Not Limi		30	RO
OUT_1					31	

Buttons: Cancel Edit Close Help

Taskbar: Start SmartDIOLE Re SmartDIOLE Server SmartPCIOLE Se ETAGVIEW Ta Syscon - Plant... Syscon - Plant 10:03 PM

Figura 45 - Bloco Estimador de Estados *on line* no configurador *fieldbus*

A título de ilustração, simulou-se uma segunda configuração da malha, onde o sistema em malha fechada tem a resposta característica de um sistema de segunda ordem.

Para a esta situação simulada, o requisito de controle deve ser resposta de segunda ordem dominante com $\xi = 0.707$ e $\omega_n = 0.0218$ rad/s (0.00347 Hz), deste modo os pólos de malha fechada devem migrar para:

$$P_2 = [-0.01538461538462 + 0.01538461538462i \\ -0.01538461538462 - 0.01538461538462i \\ -0.51020408163265]$$

Para a configuração acima, deve ser considerada a saturação no sinal de entrada da planta. Uma vez que a velocidade do sistema em malha fechada é maior que a em malha aberta, ocorrerá a saturação da entrada e a velocidade será limitada. Esta configuração P_2 de pólos acarreta na seleção da matriz K_2 :

$$K_2 = [0,22817896389325 \quad -0,03273517691100 \quad 0,00000000000000]$$

O resultado simulado desta configuração pode ser visto na figura 46:

Figura 46 - Simulação do estimador autônomo na configuração simulada

6.3. RESULTADOS PRÁTICOS

A resposta dinâmica da malha com estimador autônomo implementada em *fieldbus*, com a configuração descrita no capítulo anterior, juntamente com o resultado obtido em simulação computacional de entrada degrau estão nas figuras 47:

Figura 47 - Simulação do estimador autônomo na configuração implementada

Na figura seguinte é possível verificar a dinâmica de cada estado estimado do sistemas, juntamente com a saída da planta em temperatura e a entrada em potência da mesma:

Figura 48 - Ensaio do estimador autônomo na configuração implementada

Pode-se notar a característica de primeira ordem projetada no ensaio prático, desta forma, validando a simulação e o bloco. A adaptação dos estados estimados também fica clara, em dinâmica seis vezes mais rápida que a dinâmica da planta.

7. CONCLUSÕES

O projeto do bloco Estimador de Estados mostrou que a norma *Foundation* é suficientemente aberta a integração de novos blocos funcionais. A flexibilidade da camada de aplicação dos blocos funcionais permitiu que todos os requisitos da norma fossem avaliados e preenchidos no decorrer do projeto e da implementação do Estimador de Estados.

Em termos de funcionalidade do bloco, o Estimador de Estados já se mostrou uma possível alternativa ao controlador PID, sendo mais flexível em termos de configuração do que este. É um bloco que permite modificações e a integração de novas funcionalidades, para servir de base em futuros projetos na linha de controladores modernos.

Existem algoritmos na teoria de controle moderna muito interessantes em termos de desempenho e de flexibilidade, por exemplo os adaptativos, que embora ainda pouco aplicados industrialmente devido a sua complexidade, tendem a se popularizar com o desenvolvimento de interfaces de configuração amigáveis. Estes algoritmos matematicamente complexos requerem grande disponibilidade de recursos (*hardware* e *software*) para serem implementados. O avanço da microeletrônica e consequentemente dos transmissores industriais, entretanto, fará com que progressivamente estes transmissores se tornem aptos a suportar tais algoritmos. Assim a qualidade no projeto e no desenvolvimento de novos blocos funcionais terá um papel de grande importância na disseminação dos sistemas *fieldbus*, bem como na diferenciação entre os diversos fabricantes do *Foundation*.

Os resultados avaliados da malha com estimador autônomo confirmaram as expectativas e as simulações. Novos resultados podem ser conseguidos com a

implementação das malhas propostas no Capítulo 4, pois tais malhas podem ser facilmente configuradas a partir da malha testada.

A integração e o desempenho prático do bloco Estimador de Estados mostrou portanto, a plena viabilidade do desenvolvimento de novos algoritmos de controle integrados ao protocolo *Foundation Fieldbus*.

Esta dissertação veio demonstrar que a compatibilidade entre *fieldbus* e controle moderno é um fato, e que deve ser mais explorado. Para tal, cabe nesta conclusão se fazer sugestões para trabalho futuros nesta linha:

- Aperfeiçoamento da estrutura do bloco Estimador de Estado, com a inclusão de processamentos que viabilizem a redução de blocos nas malhas de controle apresentadas.
- Desenvolvimento de *softwares* amigáveis de configuração e sintonia de malhas, em particular de malhas por estimação de estado, de modo a tornar viável a disseminação deste tipo de controlador no setor industrial.
- Estudo das mudanças no escalonamento dos blocos quando se parte de processamentos clássicos para modernos e/ou alternativos.
- Estudo de algoritmos de otimização destes escalonamentos.
- Estudo da metodologia presente no projeto e implementação de blocos funcionais, visando a otimização do processo.
- Projeto de controladores multivariáveis baseados na teoria de controle moderna.
- Implementação de sistemas de controle modernos avançados, para aplicação em plantas não lineares ou que justifique um desempenho elevado.

O protocolo *Foundation Fieldbus*, ao criar a abertura de desenvolvimento por parte de fabricante e usuários, não só em nível de blocos funcionais, garante a constante evolução e ampliação do leque de aplicações. No momento atual de indefinição quanto a padrões internacionais de *fieldbus*, este é um grande diferencial que o *Foundation* apresenta em relação aos outros protocolos e, em particular, ao PROFIBUS-PA que em sua norma não prevê este tipo de desenvolvimento.

De fato, a implementação de estruturas em transmissores industriais de mercado requer a participação do fabricante do sistema, mas por outro lado, o estudo de novas alternativas em controladores e o projeto da interface destes com *fieldbus* pode ser realizado independentemente. É importante porém que haja um intercâmbio entre a prática e a teoria para que um projeto nesta área seja frutífero.

BIBLIOGRAFIA

REFERÊNCIAS BIBLIOGRÁFICAS

- BERGE, J. (1996). Fieldbus enables innovative measurements. *Advances in Instrumentation and Control*, v.51, n.1, p.587-92.
- BLEVINS, T.; DUFFY, J.; WILLEMS, R. (1996). DCS integration of fieldbus. *Advances in Instrumentation and Control*, v.51, n.1, p.753-62.
- BLEVINS, T.; WOJSZNIS, W. (1996). Fieldbus support for process analysis. *ISA Transactions*, v.36, p.177-83.
- DOVE, A. P.; JUNDT, L. O.; OTT, M. G. (1996). An object oriented approach to control strategies. *Advances in Instrumentation and Control*, v.51, n.1, p.337-46.
- DOVE, A. P.; JUNDT, L.O.; LUCAS, M. (1998). Using an object oriented database as a repository for process configuration data. *ISA Transactions*, v.36, n.4, p.339-43.
- FAYAD, C. A. (1996). Interfacing windows applications with fieldbus. *Advances in Instrumentation and Control*, v.51, n.1, p.593-604.

- FRANCO, L. R. H. R. (1998). *Escalonamento de mensagens para a comunicação no fieldbus*. São Paulo. 124p. Tese (Doutorado) - Escola Politécnica. Universidade de São Paulo.
- GARCIA, R. F.; PAZ, J. V.; MARTINEZ, X. C. P.; COEGO, J. B. (1997). Fieldbus: preliminary design approach to optimal network management. In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS, Piscataway, 1997. *Proceedings*. Piscataway, IEEE. p.321-5.
- GARRONE, P. (1997). Digital fieldbus technology unleashes full potential of intelligent field devices. *Instrumentation and control systems*, p.25-9, Feb.
- GERON, L. C. (1999). Operação remota de rede fieldbus: telemetria via cabo, rádio e fibra ótica. *InTech Brasil*, p.62-72. Ano II, n.15, maio.
- HIROOKA, I.; KITANO, K. I.; AKIYAMA, C.; MORIOKA, Y. (1997). A virtual environment to evaluate fieldbus technology. In: NETWORKING, INDUSTRIAL COMMUNICATIONS AND BUSES ISA TECH EXPO TECHNOLOGY UPDATE CONFERENCE, Anaheim, 1997. *Proceedings*. Anaheim, ISA. v.1, n.3, p.91-100.
- HODSON, W. R. (1996). How fieldbus will affect DCS architecture. *InTech*, v.51, n.1, p.50-3, Nov.
- HODSON, W. R. (1996). Will fieldbus kill the DCS?. *Advances in Instrumentation and Control*, v.51, n.1, p.773-82.
- KALOGERAS, A. P.; PAPADOPOULOS, G.; KOUBIAS, S. (1996). Advanced user layer software tools for industrial fieldbuses. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, Guimarães, 1997. *Proceedings*. Guimarães, IEEE. v.1, p.94-8.

- MANGANO, J. M.; DUMDIE, D. P. (1998). Foundation fieldbus has arrived and is here to stay. *InTech*, v.45, n.2, p.45-9, Feb.
- MORI, H.; KAWANO, T.; SHIRAI, T. (1996). Fieldbus system integrated into distributed control system. *Advances in Instrumentation and Control*, v.51, n.1, p.763-72.
- NAZAR, R. F. (1993). *Uso de rede de campo para aplicação em instrumentação e controle em tempo real*. São Carlos. 255p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos. Universidade de São Paulo.
- PACHECO, A. C. C. (1997). *Escalonamento temporal em rede fieldbus*. São Carlos. 83p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos. Universidade de São Paulo.
- PIMENTA, A.; LAZARI, D. B. (1996). A distributed discrete controller for fieldbus. *Advances in Instrumentation and Control*, v.51, n.1, p.547-53.
- PRZYBYLSKI, F. J. (1996). Foundation fieldbus enables the realization of the DCS' full potential. *Advances in Instrumentation and Control*, v.51, n.1, p.791-802.
- SANTOS, J. L. (1996). *Implementação e uso de rede fieldbus para monitoramento e controle de aplicação em tempo real*. São Carlos. 154p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos. Universidade de São Paulo.
- SILVA, R. D. (1996). *Comunicação de dados em ambiente industrial: um protocolo para automação e controle em tempo real*. São Carlos. 190p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos. Universidade de São Paulo.
- TOMITA, S.; AKIYAMA, C.; HAWKINS, W. (1996). Interoperability for fieldbus engineering - files exchanged during design and maintenance of fieldbus. *Advances in Instrumentation and Control*, v.51, n.1, p.565-72.

WILD, R.; PEREIRA, C. E. (1999). Proposta de ferramenta para avaliação temporal em barramentos de campo. *InTech Brasil*, p.178-86, set. /Apresentado ao Congresso técnico do ISA Show Brasil, São Paulo, 1999/

WREYFORD, J. M. (1996). PC control systems vs traditional DCS's - Are PCs ready to control your plant?. *Advances in Instrumentation and Control*, v.51, n.1, p.507-10.

ZECH, K.; SANTORI, M. (1996). Fieldbus brings protocol to process control. *IEEE Spectrum*, p.60-4, Mar.

BIBLIOGRAFIA COMPLEMENTAR

D'AZZO, J.; HOUPIS, C. H. (1984). *Linear Control System. Analysis & Design*. 3rd Edition.

DULTRA, M. L. (1996). Fieldbus control systems. *Advances in Instrumentation and Control*, v.51, n.1, p.573-86.

FELLOW, E. H. B. (1996). Continuous control standards after field bus; Why only 0, 1 and n make sense. *Advances in Instrumentation and Control*, v.51, n.1, p.783-9.

FIELDBUS FOUNDATION (1997). *FF 891 - Foundation Specification - Function Block Application Process. Part 2*. Austin, Texas.

FIELDBUS FOUNDATION (1998). *FF 800 - Foundation Specification System Architecture*. Austin, Texas.

FIELDBUS FOUNDATION (1999). *FF 892 - Foundation Specification - Function Block Application Process. Part 3*. Austin, Texas.

FIELDBUS FOUNDATION (1999). *FF 893 - Foundation Specification - Function Block Application Process. Part 4*. Austin, Texas.

FIELDBUS FOUNDATION (1999). *FF 894 - Foundation Specification - Function Block Application Process. Part 5*. Austin, Texas.

FISHER-ROSEMOUNT (1998). *Fieldbus Technical Overview*. www.frco.com

FRANKLIN ; POWELL (1994). *Feedback Control of Dynamic Systems*. 3rd Edition. Englewood Cliffs, Prentice Hall.

GLANZER, D. (1996). Foundation fieldbus forcing high-level network standard. *InTech*, v.51, n.1, p.20-2, Nov.

OCHSNER, M. J. L.; SCHRIER, M. (1997). Comparing foundation fieldbus with current digital communications. In: NETWORKING, INDUSTRIAL COMMUNICATIONS AND BUSES ISA TECH EXPO TECHNOLOGY UPDATE CONFERENCE, Anaheim, 1997. *Proceedings*. Anaheim, ISA. v.1, n.3, p.1-10.

OGATA, K. (1997). *Engenharia de Controle Moderno*. Terceira Edição. Rio de Janeiro, Prentice Hall do Brasil

PELUSO, M. (1996). The start up of the first five fieldbus control systems based on the IEC physical layer and on the fieldbus foundation specification. *ISA Transactions*, v.35, p.153-8.

PHILLIPS, C. L.; NAGLE, H. T. (1995). *Digital Control Systems Analysis and Design*. 3rd Edition. Englewood Cliffs, Prentice Hall.

PROFIBUS USER ORGANIZATION (1998). *Tutorial PROFIBUS*.

www.profibus.com

ROCHA, R. R. (1998). Redes de comunicação industriais: solução ou opção?.

InTech Brasil, p.26-30, maio. /Apresentado ao segundo Fieldbus World Congress, Belo Horizonte, 1998/

SANTORI, M. (1997). A tale of three buses: devicenet, profibus-DP, foundation fieldbus. *EDN*, p.149-60, Oct.

SMAR EQUIPAMENTOS INDUSTRIAIS (1999). *Function blocks instruction manual*. Sertãozinho.