EDUARDO ANDRÉ MOSSIN

Diagnóstico automático de redes Profibus

Tese apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para a obtenção do Título de Doutor em Ciências do Programa de Engenharia Elétrica.¹

Área de Concentração: Sistemas Dinâmicos
Orientador: Prof. Dennis Brandão

São Carlos
2012

¹ Trata-se da versão corrigida da tese. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

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

Mossin, Eduardo André
M913d Diagnóstico automático de redes Profibus / Eduardo André Mossin ; orientador Dennis Brandão. – São Carlos, 2012.

Tese (Doutorado - Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos) – Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Profibus. 2. Diagnóstico. 3. Sistemas inteligentes. 4. Redes neurais artificiais. 5. Sistemas especialistas. 6. *Fuzzy*. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **EDUARDO ANDRÉ MOSSIN**.

Título da tese: "Diagnóstico automático de redes Profibus".

Data da defesa: 19/09/2012

Comissão Julgadora:

Resultado:

Prof. Dr. **Dennis Brandão (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **José Alfredo Covolan Ulson**
(Universidade Estadual Paulista "Júlio de Mesquita Filho"/UNESP-Bauru)

APROVADO

Prof. Dr. **Karl Heinz Kienitz**
(Instituto Tecnológico de Aeronáutica/ITA)

APROVADO

Prof. Dr. **Rogério Andrade Flauzino**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Associado **Ivan Nunes da Silva**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e Presidente da
Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

AGRADECIMENTOS

Agradeço primeiramente ao amigo, professor e orientador Dr. Dennis Brandão, pela orientação, dedicação. Não fosse sua ajuda e orientação, este trabalho não se realizaria;

Aos amigos Renato Veiga Torres, Rafaela Castelhana, Francismar Silva e Cesar Cassiolato pela ajuda técnica;

Ao amigo Guilherme Serpa Sestito pela grande ajuda nos testes e coleta de informações realizadas no Laboratório de Automação Industrial.

Ao professor Dr. Ivan Nunes da Silva pelos ensinamentos transmitidos.

À Universidade de São Paulo, por colocar à disposição sua estrutura;

Aos engenheiros e amigos da SMAR, pela cooperação sempre muito útil;

Ao Rodrigo Palucci que sempre deu forças para eu continuar a batalha, principalmente nas horas de desgaste;

Sinceramente, aos meus pais e a minha irmã, por tudo, e a Deus, pela vida, por esta oportunidade e por estes irmãos;

RESUMO

MOSSIN, E. A. (2012). **Diagnóstico Automático de Redes Profibus**. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012.

Esta tese propõe a utilização de sistemas inteligentes para, automaticamente, realizar diagnósticos e localizar falhas na instalação e na operação de redes de comunicação industrial que utilizam o protocolo Profibus DP. Para tais tarefas, uma série de análises é realizada a partir dos sinais transmitidos pela camada física, de telegramas transmitidos pela camada de enlace e de funções da camada de usuário do protocolo Profibus DP. Para a análise da camada física, amostras dos sinais elétricos transmitidos são processadas e apresentadas a algumas Redes Neurais Artificiais para que sejam classificadas de acordo com a sua forma de onda. Caso estes sinais apresentem alguma deformação, o sistema indica uma provável causa para o problema, afinal, os problemas das redes Profibus originam padrões específicos e característicos impressos nas formas de onda do sinal digital. Ainda através da análise das amostras dos sinais oriundos da camada física, algumas fontes de problemas são detectadas a partir da análise do nível médio de tensão do sinal que um determinado dispositivo está transmitindo. Tal análise é realizada a partir de um Sistema Especialista. Também utilizando Sistemas Especialistas, os telegramas transmitidos pela camada de enlace deste protocolo são analisados e a partir destes, falhas de configuração são detectadas. Por fim, é proposto um sistema nebuloso responsável por indicar ao usuário um valor próximo ao ideal para a variável de tempo denominada *target rotation time*. A proposta foi testada e validada a partir de dados obtidos de redes Profibus estabelecidas em laboratório e de alguns dados sintéticos originados por software. Os resultados obtidos foram suficientes para a comprovação da tese de que sistemas computacionais inteligentes podem contribuir de maneira efetiva no diagnóstico de problemas em redes Profibus DP e até mesmo em outros tipos de rede.

Palavras-chave: Profibus, diagnóstico, sistemas inteligentes, redes neurais artificiais, sistemas especialistas, fuzzy.

ABSTRACT

MOSSIN, E.A. (2012). **Automatic Diagnosis for Profibus Networks**. Dr Thesis - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012.

This thesis proposes the use of intelligent systems to automatically perform diagnostics and locate faults during the installation and operation of industrial communication networks that use the Profibus DP protocol. For such tasks, some analyzes are performed from the signals transmitted by the physical layer, from telegrams transmitted by the data link layer and from some user layer functions of the Profibus DP protocol. For physical layer analysis, the transmitted electrical signals samples are processed and submitted for some artificial neural networks that classifies each signal according to its waveshape. If these signals have some deformation, the system indicates a probable cause for the problem, after all, the Profibus problems originate specific and characteristic patterns printed on the digital signal waveform. Still analyzing the physical layer signal samples, some problems sources are detected from the signal voltage analysis. Such analysis is performed from an Expert System. Also using expert systems, the data link layer telegrams are analyzed and configuration faults are detected. Finally, it is proposed a fuzzy system responsible for specify a value close to ideal for the target rotation time variable. The proposal has been tested and validated with data from Profibus networks established in laboratory. Besides, some synthetic data were generated by software. The results were sufficient to prove the thesis that intelligent computational systems can contribute effectively to diagnose problems in Profibus DP networks and even in other types of networks.

Keywords: Profibus, diagnosis, intelligent systems, artificial neural networks, expert systems, fuzzy.

LISTA DE FIGURAS

Figura 2.1 – Camadas do protocolo Profibus-DP	9
Figura 2.2 – Caractere Profibus DP	10
Figura 2.3 – Telegrama contendo os bytes 68H e 27H em sequência	12
Figura 2.4 – Campos de endereço e uso do bit mais significativo como extensão.	14
Figura 2.5 – Uso dos primeiros bytes do DATA_UNIT para indicação do SAP.....	14
Figura 2.6 – Descrição dos bits do campo FC (Frame Control).....	15
Figura 2.7 – Esquema geral de conexão das ferramentas de análise à rede Profibus.....	23
Figura 2.8 – Interface Osciloscópio - Profibus Tester Pb-T4.....	24
Figura 2.9 – Interface Osciloscópio – Profitrace	25
Figura 2.10 – Interface de análise de telegramas da ferramenta Profibus Scope	26
Figura 2.11 – Interface de análise de telegramas da ferramenta Profibus Scope	26
Figura 2.12 – Analisador de Bits	27
Figura 2.13 – Distribuição de eventos obtida a partir da ferramenta BT-Tool	28
Figura 3.1 – Arquitetura da Rede MLP utilizada	37
Figura 4.1 – Visão geral do projeto (Hardware, Simulador e Software de análise).....	50
Figura 4.2 – Telegrama ProfiDoctor.....	51
Figura 4.3 – Janela principal do ProfiDoctor (MDI) com três janelas internas abertas	54
Figura 4.4 – Visão geral dos componentes do software	55
Figura 4.5 – Mecanismos de amostragem dos sinais Profibus	57
Figura 4.6 – Diagrama dos módulos que utilizam sistemas inteligentes para diagnóstico	58
Figura 4.7 – Esquema para classificação dos sinais	60
Figura 4.8 – Sinal com EMI à esquerda e sem EMI à direita.....	62
Figura 4.9 – Sinais com EMI.....	65
Figura 4.10 – Sinais causados por excesso de terminadores	65
Figura 4.11 – Diagrama geral do segmento do sistema de análise utilizando RNAs.....	68
Figura 4.12 – Arquivos de amostras e classificação para RNA (EMI e cabo longo).....	69
Figura 4.13 – Arquivos de amostras e de classificação para RNAs (sinais do tipo IDLE)	70
Figura 4.14 – Arquivos de amostras e de classificação para RNAs do terceiro grupo	71
Figura 4.15 – Janela utilizada para treinamento e teste da RNA.....	72
Figura 4.16 – Nível de tensão pico a pico	73
Figura 4.17 – Árvore de decisão baseada em níveis de tensão de um bit Profibus.....	74
Figura 4.18 – Interfáce gráfica de análise de níveis de tensão	76

Figura 4.19 – Processo de sincronização (SYNC) do mestre com um escravo.....	78
Figura 4.20 – Árvore de decisão para análise baseada em histórico de mensagens Profibus..	79
Figura 4.21 – Formato dos dados de diagnóstico	83
Figura 4.22 – Descrição do byte 1 do diagnóstico padrão	84
Figura 4.23 – Descrição do byte 2 do diagnóstico padrão	84
Figura 4.24 – Descrição dos bytes 3 a 6 do diagnóstico padrão	85
Figura 4.25 – Etapas do sistema para definição do valor do TTR.....	97
Figura 4.26 – Funções trapezoidal e triangular.....	99
Figura 4.27 – Sistema fuzzy de saída com o TTR configurado em 0,9ms	99
Figura 4.28 – Sistema fuzzy da entrada TC com o TTR de 0,9ms	101
Figura 4.29 – Sistema fuzzy da entrada TA com o valor do TTR de 0,9ms.....	102
Figura 5.1 – Fotografia da bancada de testes utilizada na coleta dos resultados	112
Figura 5.2 – Arquitetura interna do Hardware.....	113
Figura 5.3 – Painel para modificar atuar sobre os escravos simulados	117
Figura 5.4 – Arquivo master.properties	118
Figura 6.1 – Topologia da rede Profibus real utilizada maioria dos testes	121
Figura 6.2 – Variância das amostras simuladas	128
Figura 6.3 – Quantidade de amostras acima e abaixo do valor limite da variância.....	128
Figura 6.4 – Exemplo de diagnóstico fornecido pelo ProfiDoctor	132
Figura 6.5 – Níveis de tensão para rede operando de maneira correta	133
Figura 6.6 – Níveis de tensão com rede com curto circuito entre A e B	134
Figura 6.7 – Níveis de tensão para escravos com cabeamento longo	135
Figura 6.8 – Níveis de tensão acima do recomendado pela norma Profibus	136
Figura 6.9 – Níveis de tensão abaixo do recomendado pela norma Profibus	136
Figura 6.10 – Níveis de tensão para cabeamento quebrado.....	137
Figura 6.11 – Mensagem de setParameters enviada ao escravo DP-B-FDN-M.....	138
Figura 6.12 – getDiagnosticResponse com o identNumber de outro escravo	139
Figura 6.13 – Mensagens de requisição de diagnóstico sem respostas.....	140
Figura 6.14 – RequestFDL para endereço 59 sem e com resposta para endereço 69.....	141
Figura 6.15 – Mensagem de retry enviada pelo mestre e respondida pelo escravo 34.....	142
Figura 6.16 – Mensagens de retry e de sincronização	143
Figura 6.17 – Escravo 59 sendo retirado da rede.....	144
Figura 6.18 – Telegramas do escravo DLG XM-210DP com watchdog baixo.....	145
Figura 6.19 – Janela de configuração de mensagens de diagnóstico do Simulador	147

Figura 6.20 – Telegramas com valores inválidos	150
Figura 6.21 – Mensagens de um ciclo de configuração com um mestre e 3 escravos	156
Figura 6.22 – Sistema para T_{TR} obtido a partir da equação 5 ($T_{TR} = 1,0237$)	157
Figura 6.23 – Resultado do operador mínimo nas regras ativas de T_C e T_A	158
Figura 6.24 – Sistema para T_{TR} obtido a partir do AnyBus NetTool ($T_{TR} = 10,10ms$)	159
Figura 6.25 – Resultado do operador mínimo (AnyBus NetTool)	159
Figura 6.26 – Sistema para T_{TR} obtido a partir do Profitrace ($T_{TR} = 13,30ms$).....	161
Figura 6.27 – Resultado do operador mínimo (Profitrace).....	161
Figura 6.28 – Sistema para T_{TR} obtido a partir do Sycon (Hielscher) ($T_{TR} = 2,2920$)	162
Figura 6.29 – Resultado do operador mínimo (Hielscher)	163
Figura 6.30 – Sistema para T_{TR} obtido a partir do Step 7 ($T_{TR} = 23,3$)	164
Figura 6.31 – Resultado do operador mínimo (Step 7)	164
Figura 6.32 – Mensagens de um ciclo de configuração com um mestre e 3 escravos	166
Figura 6.33 – Sistemas para T_{TR} obtidos a partir da equação 5 ($T_{TR} = 1,2877$ e $0,8017$)	168
Figura 6.34 – Resultado do operador.....	169
Figura 6.35 – Sistema para T_{TR} obtido a partir soma ($T_{TR} = 2,0894$).....	170
Figura 6.36 – Resultado do operador (soma dos T_{TR})	171
Figura 6.37 – Mensagens de um ciclo de configuração com um mestre e 3 escravos	172
Figura 6.38 – Sistema para T_{TR} obtido a partir da equação 5 ($T_{TR} = 1,7390$)	173
Figura 6.39 – Resultado do operador mínimo ($T_{TR} = 1,7390$)	174
Figura 6.40 – Região obtida a partir da agregação ($T_{TR} = 1,7390$)	174
Figura 6.41 – Sistema para T_{TR} obtido a partir do AnyBus NetTool ($T_{TR} = 11,5500ms$)	175
Figura 6.42 – Resultado do operador mínimo (AnyBus NetTool)	176
Figura 6.43 – Sistema para T_{TR} obtido a partir do Sycon ($T_{TR} = 2,8827$)	177
Figura 6.44 – Resultado do operador mínimo ($T_{TR} = 2,8827$)	178
Figura 6.45 – Região obtida a partir da agregação utilizando o operador máximo.....	178
Figura 6.46 – Sistema para T_{TR} obtido a partir do Step 7 ($T_{TR} = 23,3ms$)	179
Figura 6.47 – Resultado do operador mínimo (Step 7)	180
Figura 6.48 – Telegramas transmitidos com T_{TR} baixo.....	181
Figura 6.49 – Sistemas <i>fuzzy</i> para T_{TR} baixo.....	182
Figura 6.50 – Resultado do operador mínimo (AnyBus NetTool)	183
Figura 6.51 – Sistemas <i>Fuzzy</i> para o T_{TR} configurado em 2ms.....	184
Figura 6.52 – Análise com base em sinais e nos campos do telegrama Profibus.....	186
Figura 6.53 – Módulo de RNA com módulo de análise de níveis de tensão.....	187

LISTA DE TABELAS

Tabela 2.1 – Comprimento Máximo de Cabo por Segmento	10
Tabela 2.2 – Tipos de Telegramas Profibus DP.	13
Tabela 2.3 – Serviços utilizados pela camada de usuário.....	15
Tabela 2.4 – Valores e significados para os bit 6 e 5.	16
Tabela 2.5 – Principais características das ferramentas de análise Profibus	29
Tabela 4.1 – Variâncias das amostras de cada tipo de sinal	63
Tabela 4.2 – Campos do telegrama da camada de Enlace Profibus e seus possíveis valores ..	89
Tabela 4.3 – Regras de inferência fuzzy.....	105
Tabela 4.4 – Erros possíveis do sistema relacionados aos módulos.....	108
Tabela 5.1 – Ferramentas utilizadas	111
Tabela 5.2 – Softwares utilizados.....	112
Tabela 6.1 – Ondas sintéticas (cor azul) e reais (cor vermelha) para cada tipo de sinal	122
Tabela 6.2 – Resultados utilizando uma RNA para a classificação de todos os sinais	125
Tabela 6.3 – Resultados utilizando oito RNAs para todos os sinais	125
Tabela 6.4 – Sete RNAs para a classificar todos os sinais (exceção EMI)	126
Tabela 6.5 – Resultados da RNA para classificar EMI e cabeamento longo	129
Tabela 6.6 – Resultados utilizando três RNAs para a sinais IDLE	130
Tabela 6.7 – Resultados utilizando quatro RNAs.....	130
Tabela 6.8 – Treinamento com dados sintéticos e testes com dados reais	131
Tabela 6.9 – Resumo dos cenários utilizados nas análises deste módulo	146
Tabela 6.10 – Combinação de testes para as mensagens de diagnóstico	148
Tabela 6.11 – Testes e resultados para a análise dos campos de um telegrama Profibus	151
Tabela 6.12 – T_{TR} de acordo com o número de escravos e de entradas/saídas	152
Tabela 6.13 – Cenários para análise do sistema <i>fuzzy</i>	155
Tabela 6.14 – T_{TR} com um mestre e 3 escravos	156
Tabela 6.15 – Resultados a partir de uma configuração com 1 mestre e 3 escravos.....	165
Tabela 6.16 – T_{TR} com um mestre e 5 escravos	172
Tabela 6.17 – Resultados (1 mestre, 5 escravos e telegramas assíncronos).....	180
Tabela 6.18 – Análise detalhada de cada cenário possível do sistema <i>fuzzy</i>	184
Tabela 6.19 – Resultados da análise através de RNA	188
Tabela 6.20 – Resultados da análise através do nível de tensão.....	189

Tabela 6.21 – Resultados da análise através dos históricos de mensagens 190

Tabela 6.22 – Resultados da análise do T_{TR} através de sistema *Fuzzy* 191

LISTA DE SIGLAS

AI	<i>Analog Input</i>
ADC	Conversor analógico-digital
ANFIS	Sistema de Inferência Neuro-Fuzzy Adaptativo
CSTR	<i>Continuously well-Stirred Tank Reactor</i>
CCP	Competência Profibus do Brasil
DA	<i>Destination Address</i>
DAE	<i>Destination Address Extension</i>
DDLML	<i>Direct Data Link Mapper</i>
DFT	Transformada Discreta de Fourier
DP	<i>Distributed peripheral</i>
DWT	Transformada de Wavelet Discreta
ED	<i>End Delimiter</i>
EEG	Eleetroencefalograma
EIA	<i>Eletronic Industry Association</i>
EMC	<i>Eletromagnetic compatibility</i>
EMI	<i>Electromagnetic Interference</i>
FC	<i>Frame Control</i>
FCB	<i>Frame Count Bit</i>
FCS	<i>Frame Check Sequence</i>
FCV	<i>Frame Count Valid</i>
FDL	<i>Fieldbus Data Link</i>
FFT	Transformada Rápida de Fourier
FMS	<i>Field Message Specification</i>
FPGA	<i>Field-programmable gate array</i>
GSD	General Station Description
HSE	<i>High Speed Ethernet</i>
IDE	<i>Integrated Development Environment</i>
ISSO	<i>International Organization for Standardization</i>
OSI	<i>Open Systems Interconnection</i>
LE	<i>Lenght (Tamanho do campo de informação)</i>
LEr	<i>Lenght redundant (Tamanho do campo de informação)</i>
LVQ	Quantização Vetorial por Amostragem

MAU	<i>Media Access Unit</i>
MDI	<i>Multiple Document Interface</i>
MLP	<i>Multilayer Perceptron</i>
MSE	<i>Mean Square Error</i>
MSPS	<i>Mega Samples Per Second</i>
PA	<i>Process Automation</i>
PC	<i>Personal Computer</i>
PCA	Análise de Componentes Principais (Principal Components Analysis)
PMC	Perceptron Multi Camadas
RBF	Função de Base Radial
RNA	Redes Neurais Artificiais
SA	<i>Source Address</i>
SAE	<i>Source Address Extension</i>
SAP	<i>Service Access Points</i>
SE	Sistemas Especialistas
SVN	Sistema de controle de revisões
TBit	Tempo para transmissão de 1 bit
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
TTR	<i>Target Rotation Time</i>
TWD	<i>WatchDog Time</i>
UART	<i>Universal Asynchronous Receiver/Transmitter</i>

SUMÁRIO

1	INTRODUÇÃO	1
1.1	OBJETIVOS DA TESE	3
1.2	CONTRIBUIÇÕES DA TESE.....	4
1.3	ORGANIZAÇÃO DO TRABALHO	5
2	PROTOCOLO PROFIBUS E DIAGNÓSTICO	7
2.1	PROTOCOLO PROFIBUS	7
2.1.1	CAMADA FÍSICA	9
2.1.2	CAMADA DE ENLACE.....	11
2.1.3	TAXA DE COMUNICAÇÃO, <i>TIMERS</i> E TEMPORIZAÇÃO ENTRE ESTAÇÕES.....	19
2.2	MÉTODOS E SISTEMAS DE ANÁLISE DE REDES PROFIBUS	20
3	SISTEMAS INTELIGENTES	31
3.1	SISTEMAS ESPECIALISTAS.....	31
3.2	REDES NEURAIIS ARTIFICIAIS	35
3.3	LÓGICA NEBULOSA (<i>FUZZY</i>).....	43
4	PROPOSTA DE PLATAFORMA PARA DIAGNÓSTICO PROFIBUS.....	47
4.1	ANÁLISE DOS FORMATOS DOS SINAIS PROFIBUS	59
4.1.1	PRÉ-ANÁLISE	60
4.1.2	RNAS.....	66
4.2	ANÁLISE DE NÍVEIS DE TENSÃO	73
4.3	ANÁLISE BASEADA EM HISTÓRICO DE MENSAGENS	77
4.4	ANÁLISE DAS MENSAGENS DE DIAGNÓSTICO PROFIBUS	83
4.5	ANÁLISE DOS CAMPOS DE UM TELEGRAMA PROFIBUS	88
4.6	ANÁLISE DA TEMPORIZAÇÃO DA REDE COM SISTEMAS NEBULOSOS (<i>FUZZY</i>).....	90
4.7	ANÁLISE BASEADA EM INFORMAÇÕES DOS SISTEMAS INTELIGENTES.....	106

5	DESCRIÇÃO DA PLATAFORMA EXPERIMENTAL.....	111
5.1	HARDWARE PARA AQUISIÇÃO DO SINAL E DOS TELEGRAMAS PROFIBUS	113
5.2	SIMULADOR.....	115
6	RESULTADOS	121
6.1	RESULTADOS RELACIONADOS ÀS ANÁLISES DOS SINAIS PROFIBUS	121
6.2	RESULTADOS RELACIONADOS À ANÁLISE DE NÍVEIS DE TENSÃO	133
6.3	RESULTADOS RELACIONADOS À ANÁLISE BASEADA EM HISTÓRICO DE MENSAGENS	138
6.4	RESULTADOS RELACIONADOS À ANÁLISE BASEADA EM MENSAGENS DE DIAGNÓSTICO....	147
6.5	RESULTADOS RELACIONADOS À ANÁLISE DOS CAMPOS DOS TELEGRAMAS PROFIBUS	150
6.6	RESULTADOS DA ANÁLISE DA TEMPORIZAÇÃO DA REDE COM SISTEMAS <i>FUZZY</i>	151
6.7	RESULTADOS DA ANÁLISE BASEADA EM RESULTADOS DOS SISTEMAS ESPECIALISTAS	185
6.8	CONCLUSÕES OBTIDAS A PARTIR DOS RESULTADOS.....	188
7	CONCLUSÕES.....	193
	REFERÊNCIAS BIBLIOGRÁFICAS	197
	APÊNDICE A.....	207
	APÊNDICE B.....	217

1 Introdução

A área de automação industrial possui posição de vital importância para o incremento da competitividade e da eficiência do setor industrial. No Brasil e em demais mercados, o uso de arquiteturas e sistemas de automação baseados em grande número de dispositivos automáticos interligados por protocolos de comunicação digital, bem como o uso de mecanismos abertos e padronizados em tais protocolos, é amplamente difundido.

Tal tendência tecnológica nos últimos 20 anos resultou no desenvolvimento de diversos tipos de redes digitais de comunicação industrial (*fieldbus*), aplicáveis a uma gama de tipos de indústrias e de processos de fabricação. Entre os protocolos *fieldbus* amplamente difundidos no mercado mundial, estão os protocolos da família Profibus, tais como os protocolos Profinet, Profibus PA e Profibus DP, objeto de estudo desta tese.

Destaca-se que as redes *fieldbus* apresentam diversas vantagens quando comparadas a sistemas de automação com arquitetura centralizada e ligações “ponto a ponto” entre sensores, controladores, atuadores e demais dispositivos de campo. Pode-se destacar como vantagens a maior facilidade de manutenção, maior flexibilidade no *layout* e no projeto do sistema de automação, facilidade de modificação e de expansões futuras, entre outras.

Porém, estas redes industriais apresentam sensibilidade a falhas relacionadas à sua instalação física. Inobservância a requisitos técnicos como o uso de cabeamentos excessivamente longos, aterramento e blindagem inadequados do cabeamento, excesso ou falta de terminadores ativos de barramento, por exemplo, podem afetar o desempenho da comunicação e alterar características elétricas do sistema de transmissão, degradando assim o sinal a ser transmitido. Além destes tipos de problemas, erros cometidos durante a configuração dos equipamentos podem prejudicar o funcionamento destas redes. Como efeito destes descuidos, problemas de comunicação acontecem dificultando muito o processo de instalação ou manutenção de uma rede, principalmente quando estes problemas aparecem na rede de maneira esporádica (FELSER, 2006).

Com o objetivo de auxiliar os especialistas na detecção destes problemas, existem algumas ferramentas dedicadas ao monitoramento da rede. Estas ferramentas fazem o registro dos sinais elétricos transmitidos pela camada física da rede e dos telegramas que transmitem as mensagens do protocolo. A partir dessas informações, especialistas nestes protocolos

identificam problemas, apontam os motivos dos mesmos e sugerem soluções. Isto é possível, pois com estas informações, o especialista pode observar e qualificar o desempenho da comunicação entre as estações Profibus em todas as camadas de comunicação do protocolo (STEMMER, 2001).

Para análises relacionadas à camada de enlace, o conteúdo de cada telegrama transmitido é interpretado e apresentado ao especialista que, a partir de sequências típicas e de dados, identifica um determinado problema na rede. Para as relacionadas à camada física, avalia-se o aspecto das formas de onda dos sinais digitais no meio físico de comunicação. Para tal análise, os sinais devem ser adquiridos por um osciloscópio ou por circuitos de conversão analógico-digital presentes nas ferramentas de análise. Tanto a operação normal (sem problemas) da rede, como a operação frente a possíveis problemas de comunicação e de instalação, possuem, cada uma delas, padrões específicos e característicos impressos nas formas de onda do sinal digital. Tais formas de onda devem, portanto, ser analisadas por um profissional especialista responsável por interpretar e diagnosticar qual o estado ou qualidade de operação da rede e, caso exista, qual o tipo de erro de projeto ou de instalação é o causador de eventuais falhas de comunicação. Vale enfatizar que para a realização dessas análises é necessária a atuação de um especialista que tenha capacidade em identificar um problema através das informações apresentadas neste parágrafo. Felser (2006) destaca ainda que ferramentas de análise de protocolos fornecem informações estatísticas sobre o tempo de ciclo da comunicação, o número de bytes transmitidos em um ciclo, a quantidade de telegramas com erros, o número de tentativas de envio de mensagens (*retries*), entre outros dados, mas não oferece nenhuma interpretação para estes dados. Segundo Amaya et al. (2010), geralmente, é muito difícil para profissionais responsáveis pela operação e manutenção de plantas industriais analisarem e diagnosticarem de maneira imediata a causa de situações anormais de uma rede. Já em 1994, El Ayeb (1994) afirmou que as plantas industriais estão cada vez maiores e mais complexas e isto aumenta a dificuldade para operadores seguirem de maneira detalhada o processo industrial das indústrias, limitando assim sua contribuição na tarefa de realizar diagnósticos. Reforçando ainda mais tal afirmação, Zheng et al. (2006) destacou que, devido a grande quantidade de dados gerados pelo sistema, mesmo operadores experientes tem dificuldades em perceber se as condições de operação da rede estão normais.

Baseando-se neste contexto, nota-se que não existe disponível um sistema que realiza de forma automática a identificação de possíveis fontes de problemas em redes Profibus. Este tipo de sistema supriria uma necessidade atual e inerente de ferramentas que auxiliam os

usuários de redes Profibus na busca de problemas de instalação ou configuração, afinal, as ferramentas existentes atualmente apenas fornecem informações que deverão ser analisadas e interpretadas por especialistas neste protocolo. Para a implementação de um sistema deste tipo, Angeli (2008) destaca o uso efetivo de sistemas inteligentes e, para comprovar esta afirmação, apresenta uma série de pesquisas bem sucedidas realizadas por diferentes autores.

A aplicação de sistemas inteligentes neste tipo de problema reside no fato de que estes sistemas empregam conhecimentos e/ou exemplos para resolver problemas. Normalmente, uma boa indicação favorável ao uso desta tecnologia em determinado problema é a existência de um especialista humano capaz de auxiliar no desenvolvimento e elaboração de um sistema (REZENDE, 2003). A partir dessas considerações, tem-se que o problema de se obter diagnósticos de redes Profibus de forma automática, instantânea e inteligente, poderia ser investigado através de uma abordagem baseada em sistemas inteligentes como Sistemas Especialistas, Redes Neurais Artificiais (RNA) e Lógica Nebulosa (*Fuzzy*).

1.1 Objetivos da Tese

Esta tese defende que é possível analisar automaticamente as informações trafegadas em redes Profibus e, utilizando sistemas computacionais inteligentes, identificar problemas, apontar os motivos dos mesmos e sugerir soluções, utilizando para isso Sistemas Especialistas, Redes Neurais Artificiais (RNA) e Lógica Nebulosa (*Fuzzy*).

Planeja-se projetar, treinar e validar um conjunto de sistemas inteligentes computacionais para a obtenção do diagnóstico de um problema que esteja acontecendo em uma rede Profibus DP. Tal análise é feita através da classificação das formas de ondas de comunicação digital da camada física do protocolo e da análise dos dados transmitidos pelas camadas de enlace e de usuário do mesmo protocolo.

Para tal objetivo, optou-se pela elaboração de um sistema que deverá:

- Aplicar tecnologias de aprendizado computacional utilizando-se de informações obtidas a partir de uma rede Profibus e de especialistas;
- Diminuir a dependência de operadores e especialistas na resolução de problemas de instalação e operação dessas redes;
- Reduzir o número de intervenções no funcionamento de uma planta industrial;
- Possuir interface simples e amigável a fim de facilitar sua operação;

- Fornecer um analisador de telegramas transmitidos pelo protocolo, que analisará informações de diagnóstico e de processo a fim de indicar ao usuário alguma anomalia em um dado dispositivo da rede;
- Fornecer um mecanismo de *Live List* que indica quais os dispositivos estão *on-line*;
- Fazer uso e estar integrado a um simulador de redes Profibus desenvolvido especialmente para este projeto por Torres (2011). Este recurso permite a observação de uma ampla gama de potenciais problemas em redes Profibus que dificilmente seriam capturados em redes reais.

De maneira resumida, o objetivo do projeto proposto nesta tese é capturar os sinais e telegramas obtidos a partir de redes Profibus DP e, automaticamente, indicar ao usuário qual é o eventual problema que está prejudicando o funcionamento da rede.

1.2 Contribuições da Tese

A partir das informações introduzidas neste documento, nota-se uma grande necessidade de uma ferramenta que apoiará os usuários das redes Profibus na descoberta de eventuais falhas. Neste contexto, é feita uma proposta de uma ferramenta inteligente capaz de identificar problemas nestas redes. Para a implementação da mesma, foram utilizados SE, RNAs e Fuzzy. Neste contexto, pode-se destacar algumas contribuições que esta tese trouxe.

- Através de uma análise geral do trabalho, tem-se que a primeira contribuição é uma ferramenta capaz de diagnosticar de maneira automática problemas em redes Profibus-DP.
- Quanto à utilização de RNAs, uma série de contribuições pode ser destacada. A principal delas é a comprovação da tese de que RNAs são ótimas ferramentas para análise de sinais amostrados. O sucesso nos treinamentos dessas redes deve-se a capacidade de generalização que as mesmas possuem. Outra contribuição ainda relacionada à RNAs é a proposta de uma arquitetura que utiliza uma série de RNAs trabalhando de maneira integrada.
- Além disso, a criação das árvores de decisão propostas neste trabalho também merecem destaque. Este tipo de informação organizada em forma de árvores colabora para a detecção de problemas de maneira mais rápida e intuitiva.

- Por fim, um estudo relacionado à temporização das redes Profibus é feito. Este estudo destaca a importância de uma boa configuração dos parâmetros Profibus relacionados a tempo. Além deste estudo, a tese contribuiu com uma proposta de uma ferramenta para o cálculo automático de um dos parâmetros de tempo das redes Profibus.

1.3 Organização do trabalho

Considerando que o principal objetivo do trabalho é a criação de um sistema computacional inteligente de reconhecimento de padrões e de classificação de possíveis fontes de problemas em redes Profibus, uma introdução aos conceitos sobre este protocolo e suas camadas é apresentada no capítulo 2. Além disso, este capítulo apresenta algumas ferramentas que se propõem a analisar o protocolo Profibus DP.

Devido à utilização de alguns sistemas inteligentes no desenvolvimento do sistema proposto nesta tese, conceitos relacionados a sistemas especialistas, redes neurais artificiais e sistemas *fuzzy* são apresentados no capítulo 3. Além disso, o capítulo apresenta uma pesquisa bibliográfica com alguns trabalhos relevantes realizados através deste tipo de tecnologia.

Após isso, o capítulo 4 apresenta detalhes da elaboração do sistema de diagnóstico inteligente desenvolvido. Considerando que as análises realizadas dão-se a partir da observação das informações trafegadas na rede Profibus, este capítulo também apresenta alguns trabalhos acadêmicos que realizaram análises em redes de comunicação utilizando mecanismos correlatos com os utilizados nesta tese.

Para a obtenção dos resultados desta tese, uma série de ferramentas e equipamentos foram utilizadas. Os detalhes dos mesmos são apresentados no capítulo 5. Informações mais detalhadas de cada equipamento podem ser encontradas no APÊNDICE B.

O capítulo 6 apresenta todos os resultados obtidos no trabalho e para finalizar tem-se o capítulo 7 em que a conclusão e sugestões de trabalhos futuros são apresentadas.

Além dos capítulos apresentados, têm-se o APÊNDICE A contendo informações sobre a ferramenta utilizada para o controle de versões do software implementado no trabalho, as bibliotecas utilizadas no desenvolvimento do software e por fim, detalhes do software implementado.

2 Protocolo Profibus e diagnóstico

O presente trabalho baseia-se no diagnóstico automático de problemas que acontecem em redes Profibus DP utilizando para isso sistemas inteligentes. Tal diagnóstico está focado na análise de informações transmitidas pela camada física, de enlace e de usuário deste protocolo. Desta maneira, a fim de esclarecer detalhes necessários para o entendimento desta tese, este capítulo apresenta uma breve explicação sobre o funcionamento deste protocolo e suas camadas. É importante notar que o protocolo Profibus possui muitos conceitos em sua especificação, mas este documento foca apenas nos conceitos necessários para o entendimento do trabalho. Com os conceitos necessários explanados, tem-se, ainda neste capítulo, uma análise das ferramentas e métodos existentes para o diagnóstico do protocolo em questão.

2.1 Protocolo Profibus

O Profibus é um dos protocolos que fazem parte do grupo dos *fieldbuses*. É um padrão de rede de campo aberto e independente de fornecedores, que permite a integração de equipamentos de diversos fabricantes.

O padrão Profibus nasceu da necessidade de padronização das redes industriais que inicialmente não eram padronizadas trazendo diversas limitações, como por exemplo, a não compatibilidade entre equipamentos de fabricantes diferentes. O protocolo conta com três tipos de tecnologias: Profibus DP, Profibus PA e Profibus FMS (que foi substituído pela tecnologia Profinet) (NORMATIVE PARTS OF PROFIBUS FMS, DP AND PA, 1998).

- **Profibus DP (*Decentrallised Periphery*)**: é o perfil mais utilizado. Foi desenvolvido para operar com alta velocidade e conexão de baixo custo. É utilizado na comunicação entre sistemas de controle de automação e suas respectivas entradas/saídas distribuídas no nível de dispositivo. O Profibus DP pode ser utilizado para substituir a transmissão de sinal em 24V em sistemas de automação de manufatura assim como para a transmissão de sinais de 4 a 20 mA ou HART® em sistemas de automação de processo.

- **Profibus FMS:** é o perfil de comunicação universal para tarefas de comunicação complexas, por exemplo, comunicação entre mestres. O FMS oferece muitas funções sofisticadas de comunicação entre dispositivos inteligentes, porém já foi descontinuado e atualmente vêm gradativamente sendo substituído por redes Profinet, baseadas nos protocolos Ethernet e UDP, entre outros.
- **Profibus PA (*Process Automation*):** solução Profibus para automação de processos. No entanto, este termo também se refere ao universo de componentes de tecnologia que são utilizados predominantemente para conectar dispositivos de campo inteligentes no barramento Profibus em indústrias de processos, de fabricação farmacêutica, sistemas de tratamento de águas residuais, etc. O PA contempla uma tecnologia de transmissão que utiliza o mesmo meio físico para enviar os sinais de dados e a alimentação de energia para os dispositivos. Além disso, este protocolo possui uma característica muito importante que é a transmissão intrínseca segura, o que faz com que ele possa ser utilizado em áreas classificadas, ou seja, ambientes onde existe o perigo de explosão caso ocorra uma centelha (DIEDRICH et al., 2007).

Considerando que o foco do trabalho está no Profibus DP, a partir deste ponto, as considerações sobre o protocolo serão direcionadas apenas para esta vertente.

O Profibus DP permite a comunicação entre diversos dispositivos e os diferencia entre mestres e escravos. Os dispositivos mestres determinam a comunicação de dados em um barramento e são chamados de estações ativas no barramento. Já os dispositivos escravos são dispositivos como, válvulas, módulos de entrada/saída, posicionadores, transmissores etc. Esses periféricos não possuem direito de acesso ao barramento, e somente enviam ou reconhecem alguma informação do mestre quando for solicitado.

A implementação do protocolo está baseada no modelo ISO/OSI em que apenas duas camadas são implementadas: a camada de enlace (2) e a camada física (1). Além disso, tem-se a implementação da camada de usuário (não contemplada pelo modelo ISO/OSI) que acessa a camada de enlace através do *Direct Data Link Mapper* (DDLML). Através do DDLML o usuário ou as ferramentas de configuração e manutenção fornecem ou obtêm informações de parametrização, diagnóstico, configuração, entre outras de cada dispositivo da rede. A Figura 2.1 apresenta as camadas do protocolo.

ISO/OSI		PROFIBUS DP	
		Camada Usuário DDL Direct Data Link Mapper	
Camadas 3 a 7		Não Implementadas	
Camada 2 (Enlace)		FDL - Fieldbus Data Link	
Camada 1 (Física)		RS-485 ou EIA-485 Fibra Ótica	

Figura 2.1 – Camadas do protocolo Profibus-DP

Segundo Silva (2008), as redes que utilizam o Profibus DP apresentam certa sensibilidade relacionada à baixa qualidade em suas instalações que alteram as características elétricas da rede e conseqüentemente degradam o sinal que está sendo transmitido até o ponto de provocar falhas na comunicação. Além disso, problemas de comunicação também podem afetar os telegramas transmitidos pelo protocolo e, focando na detecção e diagnóstico de todos estes problemas, surge a necessidade de uma ferramenta que faça este tipo de tarefa de maneira automática. Assim, o próximo conceito a ser explorado neste capítulo é o funcionamento da camada física, onde o sistema proposto nesta tese analisa o sinal transmitido e identifica um eventual problema e sua causa. Tal conceito está apresentado no subitem 2.1.1. Após isso, no subitem 2.1.2, conceitos relacionados ao funcionamento da camada de enlace são apresentados, afinal, uma das funções do sistema é apresentar detalhes dos telegramas que trafegam nesta camada, além de diagnosticar alguns problemas ocorridos nos mesmos. No mesmo subitem utilizado para descrever a camada de enlace, algumas informações relacionadas à camada de usuário são apresentadas. Por fim, explicações sobre alguns parâmetros relacionados à temporização deste protocolo são apresentados no item 2.1.3.

2.1.1 Camada Física

A camada física do Profibus DP descreve a tecnologia de transmissão dos dados, a posição e as funções de todas as conexões dos conectores do protocolo e os parâmetros técnicos que devem ser cumpridos (PROFIBUS GUIDELINE, 1998). É nesta camada que

ocorre o transporte dos dados via EIA-485 (RS-485)² ou fibra ótica. Vale destacar que o projeto implementado para a obtenção dos resultados desta tese tem foco na resolução de problemas do Profibus DP no meio de transmissão EIA-485. Este meio de transmissão foi escolhido devido ao fato de ser o meio físico mais utilizado, pois apresenta como características principais altas taxas de transmissão e instalação simples e barata.

O EIA-485 utiliza como transporte dos dados um cabo de par trançado e blindado e permite que até 32 estações sejam conectadas ao barramento. Porém, o uso de repetidores é permitido, fazendo com que o barramento se estenda a 126 estações.

Além da limitação do número de estações, outra limitação está relacionada ao tamanho do cabo a ser utilizado. Este tamanho está diretamente relacionado com a velocidade de transmissão dos dados. A Tabela 2.1 mostra as distâncias baseadas em velocidade de transmissão para cabos do Tipo A.

Tabela 2.1 – Comprimento Máximo de Cabo por Segmento
FONTE: (PROFIBUS GUIDELINE, 1998)

Taxa (Kbps)	9,6	19,2	45,45	93,7	187,5	500	1500	3000	6000	12000
Comprimento Máx.(m)	1200	1200	1200	1200	1000	400	200	100	100	100

No EIA-485 os dados são transmitidos por dois canais, denominados A e B, que transmitem níveis de tensão iguais, porém com polaridades opostas (V_A e V_B). Por esta razão, é importante que a rede seja ligada com a polaridade correta. O gráfico em azul da Figura 2.2 apresenta um exemplo de um sinal transmitido nesta camada.

Figura 2.2 – Caractere Profibus DP

² - O termo RS-485 é um termo desatualizado. Um termo atual para referenciar este padrão é EIA-485. (EIA – Eletronic Industry Association).

Outro ponto importante a ser destacado é a maneira de como o cabo é conectado aos equipamentos. Tal conexão é feita através de conectores Profibus que são disponibilizados com algumas variedades de classes de proteção e projetos mecânicos. A escolha do melhor tipo de conector varia de acordo com a necessidade da instalação, porém o mais recomendado pela Normative Parts of Profibus FMS, DP and PA (1998) é o conector Sub-D com 9 pinos. Os pinos 3, 5, 6 e 8 são sempre utilizados. Os demais pinos são opcionais.

Os conectores Sub-D com 9 pinos apresentam em sua estrutura os terminadores de barramento. Esses terminadores são indispensáveis no projeto da rede Profibus. É necessário que sejam habilitados dois terminadores no barramento, um no início e outro no fim de cada segmento. A terminação ativa na posição incorreta faz com que, tanto o nível quanto a forma de onda sejam degradados. A ativação incorreta dos terminadores causa descasamento de impedância e reflexões do sinal. A ausência de terminadores pode causar erros na transferência de dados, pois ocorre uma reflexão no sinal que chega ao fim do barramento, fazendo com que este sinal volte pela linha de dados e se sobreponha ao sinal que está sendo enviado. Já o excesso de terminadores habilitados pode causar intermitência nos dados transmitidos fazendo com que ocorram interrupções na comunicação. A classificação de um problema no sinal causado por excesso ou falta de terminadores é um dos objetivos desta tese.

2.1.2 Camada de Enlace

Neste item são descritos os aspectos da camada de enlace importantes para o entendimento desta tese. Outros detalhes poderão ser encontrados em Normative Parts of Profibus FMS, DP and PA (1998).

A rede Profibus DP é uma rede do tipo *multidrop*, assíncrona, *half duplex* e utiliza a comunicação do tipo passagem de *token* (*Token Passing*) e mestre-escravo. O mecanismo de passagem de *token* permite a aplicação de múltiplos mestres em uma mesma rede compartilhando o acesso.

Somente o mestre pode iniciar a comunicação na rede. Os escravos comunicam somente para responder requisições do mestre. A rede Profibus DP permite a operação permanente com mais de um mestre, desde que configurados individualmente e de maneira adequada nas restrições da norma.

O número máximo de estações em uma rede Profibus DP é 126. Assim, a faixa de endereços disponível para uso vai de 0 a 125. Os endereços 126 e 127 são de uso especial,

sendo o 126 utilizado como valor padrão para estações não endereçadas entrarem na comunicação e o endereço 127 (0x7F) reservado para comandos de *broadcast*.

Para a comunicação entre cada estação, o protocolo define alguns telegramas. Cada telegrama é formado por um conjunto de caracteres, no qual cada caractere é formado por 11 bits, sendo apenas 8 deles utilizados como dado. Este é o padrão UART e os três bits extras são utilizados para fornecer uma sinalização de início e fim de transmissão de cada caractere (2 bits) e um bit de paridade par utilizado para conferição da integridade da comunicação no receptor. A Figura 2.2 apresenta um exemplo deste caractere juntamente com o sinal transmitido pela camada física deste protocolo.

Nota-se que antes do bit de início, tem-se o estado de linha desocupada (*IDLE*) da comunicação que é representado pelo nível de tensão 1. Antes de terminar a transmissão do caractere (bit de fim), tem-se o bit de paridade. O receptor avalia a paridade a cada byte recebido. Caso a paridade avaliada não tenha o mesmo valor que o bit de paridade, o telegrama inteiro será descartado (não somente o caractere). Um telegrama é constituído por um ou mais caracteres e não são permitidos períodos de linha desocupada dentro da transmissão de um telegrama. Assim, o início de um telegrama com mais de um caractere é exemplificado na Figura 2.3:

Figura 2.3 – Telegrama contendo os bytes 68H e 27H em sequência.
FONTE: TORRES (2011)

Um telegrama é composto por 1 a 255 caracteres. Existem alguns tipos de telegramas definidos pela Normative Parts of Profibus FMS, DP and PA (1998). A diferenciação entre os tipos é realizada pelo cabeçalho do telegrama (primeiro caractere), onde cada um dos tipos possui um valor diferente.

Os telegramas são especificados de acordo com a natureza do campo de dados:

- Telegramas de tamanho fixo sem campo de dados.
- Telegrama de resposta curta ou reconhecimento.
- Telegrama com campo de dados de tamanho variável.
- Telegrama de *token*.

A Tabela 2.2 apresenta um resumo dos tipos de telegramas e aplicações:

Tabela 2.2 – Tipos de Telegramas Profibus DP.
FONTE: TORRES (2011)

TELEGRAMA	FORMATO DO TELEGRAMAS
Telegrama de tamanho fixo sem campo de dados	<p>SD1 – Delimitador de início do tipo 1 (10H).</p> <p>Serviço de manutenção da rede (<i>Request FDL Status</i>) e troca de dados quando não há valores de escrita.</p>
Telegrama de resposta curta ou reconhecimento	<p>SC – <i>Single Character</i> (E5H).</p> <p>Resposta curta a requisições do mestre onde o escravo não precisa ou não pode responder imediatamente</p>
Telegrama com campo de dados de tamanho variável	<p>SD2 – Delimitador de início do tipo 2 (68H).</p> <p>A grande maioria dos serviços utiliza esse tipo de comando devido à flexibilidade no tamanho do campo de dados.</p>
Telegrama de token	<p>SD4 – Delimitador de início do tipo 4 (DCH)</p> <p>Utilizado somente para transmissão do <i>token</i>.</p> <p>No caso de apenas um mestre na rede: DA=SA.</p>

DA – *Destination Address* – Endereço de destino.

FCS – *Frame Check Sequence*

ED – *End Delimiter*, valor 16H

DATA_UNIT – Campo de dados com tamanho variável (1 a 246)

SA – *Source Address* – Endereço de origem.

DATA_UNIT8 – Campo de dados com tamanho fixo igual a 8.

LE – Tamanho do campo de informação.

FC – *Frame Control*

O tamanho máximo de um telegrama são 255 caracteres. O valor do LE (ou LEr) varia de 0 a 249 e compreende a quantidade de bytes do campo DATA_UNIT além do DA, SA e o FC. Portanto o tamanho máximo do campo DATA_UNIT é de 246 bytes. O DATA_UNIT é a porção do telegrama destinada à carga útil de dados (*payload*).

Embora os campos de endereço suportem valores entre 0 e 255 (1 byte), somente os 7 bits menos significativos são utilizados efetivamente para o endereçamento das estações (Figura 2.4), permitindo o endereçamento de até 127 estações (0 a 126). O endereço 127 conforme explicado anteriormente é reservado ao *broadcast*.

Figura 2.4 – Campos de endereço e uso do bit mais significativo como extensão.
FONTE: TORRES (2011)

Em um telegrama de resposta os endereços DA e SA são simplesmente invertidos em relação ao telegrama de requisição. O bit mais significativo do byte de endereço é utilizado para sinalizar que o telegrama que está sendo enviado possui uma extensão de endereço. Isso indica que no campo DATA_UNIT os dois primeiros bytes estarão ainda relacionados com o protocolo (Figura 2.5).

Figura 2.5 – Uso dos primeiros bytes do DATA_UNIT para indicação do SAP.
FONTE: TORRES (2011)

Essa extensão de endereço é utilizada pela camada de usuário para implementar diferentes serviços de parametrização e configuração. Os códigos dos serviços são chamados *Service Access Points* (SAP) e serão detalhados posteriormente. *Destination Address Extension* (DAE) é a extensão do endereço de destino ou SAP de destino e *Source Address Extension* (SAE) é a extensão do endereço de origem ou SAP de origem.

Quando uma estação entra no estado de troca de dados, que é o objetivo final do processo de configuração, não há o uso das extensões de endereço, destinando a totalidade do DATA_UNIT para os dados específicos dos equipamentos (dados de usuário).

As extensões aplicadas ao Profibus DP (DPV0) estão na Tabela 2.3 e seus valores são apresentados em formato hexadecimal e decimal (entre parênteses).

Tabela 2.3 – Serviços utilizados pela camada de usuário

SERVIÇO	SAP MESTRE	SAP ESCRAVO
<i>Data Exchange</i>	Não possui	Não possui
<i>Set Slave Address</i>	3E (62)	37(55)
<i>Read Inputs</i>	3E (62)	38(56)
<i>Read Outputs</i>	3E (62)	39(57)
<i>Global Control</i>	3E (62)	3A(58)
<i>Get Configuration</i>	3E (62)	3B(59)
<i>Slave Diagnostics</i>	3E (62)	3C(60)
<i>Set Parameters</i>	3E (62)	3D(61)
<i>Check Configuration</i>	3E (62)	3E(62)

O campo FC é utilizado para indicar o tipo de telegrama (requisição ou resposta), controle de sequência que evita a perda de mensagens e contém o tipo da estação e o status da FDL (*Fieldbus Data Link*) desta estação (escravo ou mestre). O significado dos bits do campo FC é apresentado na Figura 2.6:

Figura 2.6 – Descrição dos bits do campo FC (Frame Control)
FONTE:TORRES (2011)

Caso o bit b7 seja 1 então o telegrama é uma requisição. Neste caso os bits b6 e b5 são chamados FCB (Frame Count Bit) e FCV (Frame Count Valid) e são utilizados para controle de sequenciamento. Caso o b7 seja 0 então se trata de uma resposta. Neste caso, o b6 e b5 identificam o tipo (mestre ou escravo) e o estado da estação que está respondendo. O código da função indica o tipo de requisição ou resposta (como requisição com confirmação ou sem confirmação e resposta positiva ou negativa).

O bit FCB evita recepção de mensagens duplicadas e reenvio da última mensagem quando é detectado que o escravo não respondeu a última requisição. Após um escravo de endereço específico se tornar operacional, na primeira mensagem o mestre envia o FCB=1 e FCV=0. O escravo guarda o valor do FCB=1 e marca essa mensagem como a primeira recebida do mestre. Após receber a resposta deste escravo, o mestre sempre enviará FCV=1 para este escravo (sinalizando que o mestre iniciará a indicação do controle de sequência) e ficará alternando o valor do FCB entre 0 e 1 (começando com FCB=0). Deste modo, caso o mestre não receba alguma resposta desta estação, ele enviará novamente a última requisição sem alternar o valor do FCB, indicando ao escravo que foi um pedido de *retry*, isto é, o mestre não recebeu a última resposta deste equipamento.

Caso o b7 seja 0 então trata-se de uma resposta. Neste caso os bits b6 e b5 são utilizados para identificar o tipo da estação que esta respondendo. A Tabela 2.4 apresenta os possíveis valores para estes bits.

Tabela 2.4 – Valores e significados para os bit 6 e 5.

B6	B5	Tipo de estação
0	0	Estação do tipo escravo
0	1	Estação mestre. Mestre não está pronto para entrar no ciclo de <i>token</i> .
1	0	Estação do tipo mestre. Mestre pronto para entrar no ciclo de <i>token</i> .
1	1	Estação do tipo mestre já operando no ciclo de <i>token</i> .

O último campo do FC representa o código da função. Os valores deste campo possuem significados diferentes dependendo se o telegrama é de requisição (*request*) ou resposta (*reply*). A função principal deste campo é definir prioridade de atendimento de requisições e de sinalizar se há um novo diagnóstico a ser enviado ao mestre. Mais detalhes sobre cada valor possível para este campo podem ser encontrados em Normative Parts of Profibus FMS, DP and PA (1998).

Outro campo de um telegrama Profibus é o FCS, que é utilizado para elevar a integridade da comunicação e permitir ao receptor descartar telegramas inválidos ou corrompidos. O cálculo do FCS é a soma aritmética do valor dos bytes contidos no DATA_UNIT e dos valores dos bytes do DA, SA e FC. No caso dos telegramas sem campo de dados, somente o DA, SA e FC são somados.

Além de definir os telegramas utilizados no protocolo, o Profibus define o procedimento de transmissão dos mesmos. Esta transmissão dá-se através de ciclos de comunicação entre mestres e escravos. Um ciclo de comunicação de um mestre com um

conjunto de escravos é chamado ciclo de mensagens. Este ciclo é interrompido somente para a transmissão de *token* e mensagens de *broadcast*. Todas as estações devem monitorar todas as requisições. Uma estação deve responder somente quando uma requisição estiver endereçada a ela. A resposta deve ocorrer dentro de um tempo pré-definido (*SlotTime*) – este é o tempo que o mestre aguarda a resposta, antes de enviar novamente a requisição para a mesma estação.

Existem quatro modos de operação. Estes definem o comportamento em relação à temporização e prioridade dentro de um ciclo de mensagens. As operações são as seguintes:

1. Recepção e envio do *Token*.
2. Comunicação acíclica.
3. Comunicação cíclica ou *polling*.
4. Registro das estações

O *token* é um tipo de mensagem que é transmitida entre os mestres e é utilizado com o objetivo de um mestre transferir o acesso ao meio para outro mestre. Cada mestre acessa o canal de comunicação no modo mestre-escravo. Ao fim do acesso, transmite uma mensagem de *token* ao próximo mestre. Então, este comanda o canal e posteriormente passa o *token* ao próximo mestre. Quando o último mestre termina de realizar os comandos necessários aos escravos e aos outros mestres, ele passa o *token* novamente ao primeiro mestre e o processo reinicia. A esse anel lógico dá-se o nome de *logical token ring*.

Uma rede que tem somente um único mestre resume sua operação na comunicação tipo mestre-escravo e ao fim de cada ciclo o mestre passa o *token* para ele mesmo (pois é único).

A manutenção da rede é uma obrigação de cada mestre. Assim, cada mestre deve ter o conhecimento de todos os equipamentos (endereços) que estão presentes na rede e qual sua natureza (mestre ou escravo). Após um conjunto de ciclos de comunicação, o mestre consulta um novo endereço através de um comando dedicado à manutenção da rede. O mestre aguarda o retorno da resposta do endereço consultado. Se houve resposta então esse endereço é armazenado em uma lista para não ser mais consultado. Caso contrário, considera-se que esse endereço é vago e o mestre após consultar os demais endereços ainda continuará a testar este último. Um equipamento presente na rede (que responde a comandos) é denominado operacional.

A lista de equipamentos da configuração que pertencem ao modo de comunicação cíclica é passada ao controlador FDL. As estações que não responderem durante a comunicação cíclica são classificadas como não operacionais. Cada mestre mantém sua própria lista (configuração) de equipamentos pertencentes à comunicação cíclica. Ao fim da comunicação cíclica, prioritária, é realizada a comunicação acíclica e pelo menos um endereço não operacional é consultado.

A comunicação cíclica é baseada em respostas imediatas dos escravos e mestres, mantém prioridade sobre as mensagens acíclicas e são utilizadas para comunicação de dados de controle do processo. As mensagens acíclicas são baseadas em consulta, isto é, o mestre inicia a requisição e no próximo ciclo pergunta novamente para verificar se a estação já tem a resposta. Em geral, são utilizadas para supervisão ou parametrização de dados internos das estações.

Mensagens cíclicas e acíclicas, além da manutenção da rede devem ocorrer dentro de um período configurado pela aplicação (*Target Rotation Time* - T_{TR}).

Como explicado anteriormente, o registro das estações é de responsabilidade de cada mestre da rede. O registro é realizado através do comando *Request FDL Status* que é um telegrama do tipo SD1, destinado à manutenção da rede.

Mais detalhes relacionados a outras mensagens Profibus, como por exemplo, mensagens de *getDiagnostic*, *setParameter*, *checkConfiguration* e *dataExchange* serão apresentados no momento oportuno no decorrer deste documento.

É importante observar que os serviços apresentados neste capítulo estão relacionados ao Profibus DP V0. Os serviços relacionados ao Profibus DP V1 e V2 não são tratados nesta tese. Em suma, tem-se que:

- DP V0 engloba as funções de parametrização, configuração, troca de dados e leitura das informações de diagnóstico dos escravos.
- DP V1 engloba as funções do DP V0 além de serviços para troca de dados acíclica para configuração, operação, visualização e controle.
- DP V2 engloba as funções do DP V0 e V1 além de permitir operação em modo determinístico, comunicação entre escravos, upload e download.

Apesar das diferenças entre o Profibus DP V0, V1 e V2, o sistema proposto deve fornecer bons resultados em todas as versões do protocolo, afinal, os tipos de telegramas transmitidos em todas as versões são os mesmos.

2.1.3 Taxa de Comunicação, *Timers* e Temporização entre estações

A taxa de comunicação de uma rede Profibus DP é geralmente atribuída pelo mestre. Em uma rede com múltiplos mestres, cada mestre deve ser configurado separadamente, mas todos com a mesma taxa de comunicação. O mestre utiliza parâmetros de tempo para verificar e garantir que a resposta dos escravos está sendo realizada dentro do previsto, assim como evitar que ocorra colisão ou transmissão simultânea de mais de uma estação. Existem muitos parâmetros de tempo a serem configurados, mas apenas dois deles serão considerados aqui. Estes são o T_{TR} e o *WatchDog*. Mais detalhes sobre os outros parâmetros de tempo podem ser encontrados em TORRES (2011).

Estes parâmetros de tempo são apresentados a seguir na forma de T_{Bit} que permite manter os mesmos valores em função da taxa de comunicação.

Um T_{Bit} é o tempo decorrido durante a transmissão de 1 bit. O valor em segundos do T_{Bit} é inversamente proporcional à taxa de comunicação (ou taxa de transmissão) em bits por segundo.

Target Rotation Time (T_{TR})

O *Target Rotation Time* é um valor de tempo configurado pelo usuário com o objetivo de estipular o tempo entre a estação mestre receber duas vezes o *token*, ou seja, o tempo necessário para que um ciclo de comunicação ocorra entre todos os mestres e escravos configurados na rede. Este tempo é configurado antes da rede entrar em regime de operação. Com a rede em regime de operação, o tempo necessário para todos os mestres e escravos se comunicarem é chamado de T_{RR} (*Real Rotation Time*).

Quando um mestre recebe o *token* ele guarda o valor de tempo e após a operação do *Logical token ring*, o mestre recebe novamente o *token* e calcula a diferença de tempo. Este tempo é o T_{RR} . Cada vez que o mestre recebe o *token*, ele realiza pelo menos um ciclo de mensagens prioritárias e, após este ciclo, ele verifica se $T_{RR} < T_{TR}$. Caso o T_{TR} seja maior do que o T_{RR} , o mestre conclui que pode ocupar o barramento por mais tempo e transmite suas mensagens acíclicas.

WatchDog

Este parâmetro é utilizado para garantir a operação segura de um sistema no caso de uma falha no mestre ou no cabo de comunicação. Caso o escravo pare de receber comandos

do mestre por qualquer motivo, ele aguardará este período (*WatchDog Time* - T_{WD}) antes de ir para o modo de segurança.

Durante o processo de configuração dos escravos, o mestre envia num telegrama de parametrização o valor do T_{WD} . Este tempo é armazenado no escravo e após o início da troca de dados, o escravo fica monitorando continuamente se está recebendo novas requisições. Caso não receba uma nova requisição do mestre dentro deste período de tempo, ele irá para uma condição segura e aguardará por uma nova configuração.

2.2 Métodos e sistemas de análise de redes Profibus

Este capítulo apresenta os métodos e os sistemas mais utilizados para a análise do protocolo Profibus.

Como já explanado anteriormente, tem-se que o projeto em questão apresenta técnicas de inteligência computacional que auxiliará os especialistas na detecção de problemas fornecendo sugestões e diagnósticos automáticos. Esta automatização difere esta proposta das apresentadas nos itens que seguem.

Os métodos mais utilizados abrangem a validação dos critérios de projeto, configuração e instalação da rede, versões de *firmware* e dos arquivos GSD³ e a verificação do meio físico (SILVA, 2008). Dessa maneira, tem-se abaixo a lista dos principais métodos de diagnóstico:

- Inspeção Visual da Rede;
- Testes com multímetro;
- Testes com testadores de barramento;
- Testes com osciloscópio;
- Testes utilizando repetidores com diagnóstico;
- Ferramentas de monitoração da rede.

a) Inspeção Visual

Existem algumas condições geradoras de falhas que podem ser observadas:

- curvaturas no cabo devem estar dentro de um raio mínimo recomendado pelo fabricante;

³ O arquivo GSD contém detalhes de revisão de hardware e software, *bus timing* do equipamento e informações sobre a troca de dados cíclicos, entre outras informações (NORMATIVE PARTS OF PROFIBUS FMS, DP AND PA, 1998).

- cabeamento utilizado na interconexão dos dispositivos deve ter comprimento mínimo de 1m;
- cabeamento deve possuir comprimentos específicos de acordo com a velocidade de transmissão;
- blindagem do cabo não deve estar para fora do conector e deve estar em contato com a parte metálica do conector;
- blindagem e os dispositivos devem estar aterrados nas extremidades do segmento de rede;
- todos os dispositivos devem suportar a taxa de transmissão escolhida para a rede;
- todos os dispositivos devem estar endereçados corretamente.

b) Testes utilizando Multímetro

Com o uso do multímetro pode-se detectar as seguintes falhas na rede:

- Curto-circuito entre as linhas de dados A e B;
- Curto-circuito entre as linhas de dados A e B e a blindagem do cabo;
- Inversão simples das linhas de dados A e B;
- Interrupção de uma das linhas de dados A e B;
- Interrupção na blindagem do cabo.

c) Teste utilizando testadores de barramento portáteis (*Bus Tester*)

Este método é utilizado após a montagem física da rede e dos conectores. Além dos problemas citados no tópico que fala dos testes utilizando o multímetro, os *Bus Testers* podem:

- identificar a perda de terminação ou terminação excessiva;
- identificar a existência de reflexões e localização do problema no barramento;
- realizar a medição dos níveis de sinal nas linhas de dados A e B com o barramento em operação;
- testar os drivers EIA-485 de um dispositivo Profibus;
- listar todos os equipamentos conectados ao barramento.

Atualmente existem disponíveis no mercado algumas opções destes dispositivos que são similares em questão de funcionalidade e escopo, porém, se diferem na forma como são

operados. Dentre estes, pode-se destacar o BT200 - PROFIBUS DP *Cable Tester*, da fabricante Siemens e o NETTEST II - PROFIBUS *Analysis and Test Tool* da fabricante Comsoft.

d) Teste utilizando o Osciloscópio

O osciloscópio é muito utilizado no diagnóstico de redes Profibus, porém exigem o mínimo de experiência do usuário para trabalhar com o equipamento e um grande conhecimento para interpretar as formas de ondas apresentadas por ele. Essas formas de onda são obtidas através da medição entre as linhas de dados A e B. Esta ferramenta permite a detecção de problemas através da análise do formato do sinal transmitido, porém não é possível correlacionar o sinal a um determinado escravo da rede.

e) Testes utilizando Repetidores com Diagnósticos

Esses repetidores são similares aos repetidores comuns, porém possuem a capacidade de monitorar os segmentos de rede a fim de detectar defeitos. De acordo com Mitchell (2004), com este repetidor é possível encontrar os seguintes problemas na rede:

- Curto-circuito entre as linhas de dados A e B;
- Interrupção de uma das linhas de dados A e B;
- Ausência das terminações;
- Perda de conexões;
- Reflexões excessivas, mostrando a distância a partir do repetidor;
- Número de estações acima do permitido em um segmento;
- Distância muito grande entre as estações e o repetidor;
- Telegramas com mensagens de diagnósticos.

f) Softwares para diagnósticos

Atualmente, além de todos os métodos de diagnósticos apresentados anteriormente, é possível também realizar análise de uma rede Profibus utilizando-se softwares avançados de monitoração da rede. Diferentemente das ferramentas apresentadas nos itens anteriores, que tinham o foco totalmente voltado para a camada física do sistema, os sistemas apresentados aqui, além da análise da camada física, possuem a possibilidade da análise da camada de enlace e da camada de usuário. Essas ferramentas são capazes de mostrar e registrar o tráfego de dados na rede e fornecer um caminho para que se possa observar o desempenho da

comunicação entre as estações Profibus. Porém, a análise das informações registradas exige muita experiência e conhecimento detalhado do protocolo Profibus e das interações entre mestres e escravos.

De maneira geral estes softwares obtêm as informações da rede conectando-se à mesma de maneira passiva, ou seja, sem interferir no funcionamento do protocolo. Para realizar esta conexão, as ferramentas disponibilizam um hardware para a coleta do sinal e dos telegramas transmitidos. A Figura 2.7 apresenta um exemplo deste tipo de sistema.

Figura 2.7 – Esquema geral de conexão das ferramentas de análise à rede Profibus

As ferramentas mais utilizadas para esse processo de monitoração e diagnóstico da rede serão apresentadas nos próximos subitens.

- ***Profibus Tester (PB-T4) e PROFITRACE***

As ferramentas de monitoração Profibus *Tester* PB-T4 e PROFITRACE são fornecidas pelas fabricantes Softing e Procentec, respectivamente.

Ambas as ferramentas, ao serem conectadas à rede, atuam como dispositivos passivos e capturam o tráfego de mensagens no barramento. Uma característica importante é que elas não geram tráfego adicional e também não alteram os tempos que foram inicialmente configurados na rede (SOFTING,2012 e PROCENTEC,2012).

Segundo Silva (2008), esses monitores de barramento possuem basicamente as mesmas funcionalidades:

- Capturar e mostrar mensagens, permitindo a definição de triggers para monitoração de telegramas. Por exemplo, quando um mestre requisita uma informação de diagnóstico ou quando uma mensagem de parametrização é enviada pelo mestre;
- Operação em tempo real na velocidade de comunicação do sistema;
- Filtro para mensagens, permitindo a visualização apenas dos telegramas selecionados;
- “Live List”, ou seja, uma lista mostrando os dispositivos ativos no barramento;
- Decodificação dos telegramas capturados;
- Geração de *trigger* para osciloscópio, facilitando a captura de formas de onda transmitidas por um Escravo em particular;
- Estatísticas da comunicação, mostrando o número de telegramas corrompidos ou repetidos no sistema, ou por estação.

Essas ferramentas possuem várias interfaces que disponibilizam os dados para a análise da rede. Exemplos das ferramentas de monitoração mostrando a interface com osciloscópio podem ser visualizados na Figura 2.8 e Figura 2.9.

Figura 2.8 – Interface Osciloscópio - Profibus Tester Pb-T4
FONTE: SOFTING (2012)

Figura 2.9 – Interface Osciloscópio – Profitrace

É importante notar que estas ferramentas, apesar de apresentarem ao usuário a forma de onda capturada da rede, não apresentam um diagnóstico baseado nessas formas de onda. Nos manuais de usuário dessas ferramentas, existem explicações sobre os formatos de sinais com problemas e como os mesmos devem ser classificados, mas tal classificação depende da análise do usuário. O sistema proposto para a obtenção dos resultados desta tese utiliza sistemas inteligentes para classificar estes sinais e apoiar o usuário em suas decisões. O mesmo pode-se afirmar quanto à análise dos telegramas e da variável de tempo T_{TR} .

- **Profibus Scope**

O *Profibus Scope* é fabricado pela Trebing + Himstedt e seu funcionamento é baseado em uma conexão com a rede Profibus através de um hardware que atua como dispositivo passivo e captura o tráfego de mensagens no barramento.

Esta ferramenta fornece ao usuário a possibilidade de analisar os telegramas transmitidos e para tornar isso viável, oferece ferramentas para filtrar mensagens facilitando a busca de determinados problemas na rede. A Figura 2.10 apresenta um exemplo desta interface com alguns telegramas Profibus.

Figura 2.10 – Interface de análise de telegramas da ferramenta Profibus Scope
 FONTE: PROFIBUS SCOPE (2010).

Além disso, a ferramenta apresenta uma interface que agrega informações gerais sobre a rede e seus diagnósticos. A Figura 2.11 apresenta tal interface.

Figura 2.11 – Interface de análise de telegramas da ferramenta Profibus Scope
 FONTE: PROFIBUS SCOPE (2010).

- **PBMaster**

Outra opção é a ferramenta PBMaster. Esta ferramenta, diferentemente das apresentadas até então, é resultado de um trabalho acadêmico desenvolvido por Khanh

(2009). Esta ferramenta tem como objetivo oferecer uma solução para a ligação de computadores pessoais ao barramento industrial, além de disponibilizar para o usuário uma interface gráfica e um programa de monitoramento da rede. Os problemas podem ser diagnosticados e localizados analisando-se a comunicação no barramento. Ao analisar os dados capturados, é possível detectar problemas como quadros com erros (muitas vezes causado pelo sinal de reflexão), inatividade da estação, colisões de endereços, *slot time* muito pequenos nas estações mestre, etc. A Figura 2.12 apresenta um exemplo da interface gráfica desta ferramenta. É importante observar que o sinal mostrado na Figura 2.12 não é um sinal real da rede Profibus, mas sim o sinal interpretado.

Figura 2.12 – Analisador de Bits
FONTE: KHANH, 2009

- **BR-TOOL**

Assim como o PBMaster, esta ferramenta é resultado de um trabalho acadêmico. Desenvolvida por Husemann (2003), esta ferramenta tem como objetivo validar o comportamento temporal de sistemas de automação industrial baseados em barramentos de campo e comparar o desempenho de diferentes protocolos. Uma diferença importante desta

ferramenta para as apresentadas até então e a de que esta última foi projetada para analisar diferentes tipos de protocolo, como por exemplo, o Foundation Fieldbus.

Este sistema captura cada evento que ocorre no barramento de transmissão dos protocolos de comunicação industrial através de uma placa de aquisição e salva cada evento juntamente com uma informação de tempo do momento em que o evento ocorreu. A partir dessas informações, um software trata estes dados a fim de identificar eventos relevantes. Esta ferramenta então valida os eventos identificados e, baseando-se nas especificações temporais definidas para a aplicação, determina se os requisitos temporais exigidos pela mesma foram ou não atendidos. Além disso, estes eventos podem ser visualizados de forma gráfica, permitindo assim que o usuário faça as análises necessárias e obtenha as suas conclusões. Como exemplo, pode-se apresentar um estudo de caso encontrado em Husemann (2003). Neste caso, uma configuração Profibus é analisada e a partir da contagem de cada evento capturado pela ferramenta BR-Tool, tem-se a estatística de ocorrência de cada tipo de mensagem transmitida. A Figura 2.13 apresenta estes dados.

Figura 2.13 – Distribuição de eventos obtida a partir da ferramenta BT-Tool
 FONTE: Husemann (2003)

Baseando-se nas características principais das ferramentas descritas neste item, tem-se a Tabela 2.5.

Tabela 2.5 – Principais características das ferramentas de análise Profibus

Nome	Fabricante	Osciloscópio	Telegramas	Live List	Trigger	Filtros
PB-Tester	Softing	Sim	Sim	Sim	Sim	Sim
Profitrace	Procentec	Sim	Sim	Sim	Sim	Sim
Scope	T+Himsted	Não	Não	Sim	Não	Não
PB-master	Acadêmico	Sim	Não	Sim	Não	Não
BR-Tool	Acadêmico	Não	Não	Não	Não	Sim

É importante notar que nenhuma das ferramentas apresentadas neste capítulo apresenta ao usuário o diagnóstico automático de problemas. Em cada uma dessas, o usuário precisa analisar os telegramas ou o formato dos sinais e, a partir disso, entender qual é a origem do problema. Sendo assim, no próximo capítulo, os conceitos necessários para o entendimento desta tese são apresentados.

3 Sistemas Inteligentes

Segundo Tatai (2003) o termo sistemas inteligentes engloba a inteligência artificial clássica baseada em lógica de primeira ordem, sistemas especialistas e inteligência computacional, que incorpora sistemas nebulosos (*fuzzy*), redes neurais artificiais e sistemas evolutivos. Considerando o fato de que o projeto implementado nesta tese utiliza-se de sistemas computacionais inteligentes para a realização do diagnóstico em redes Profibus, os subitens 3.1, 3.2 e 3.3 apresentam, respectivamente, alguns conceitos sobre sistemas especialistas, RNA e lógica *fuzzy*. Além disso, estes subitens apresentam uma pesquisa bibliográfica que destaca uma série de trabalhos que utilizaram tais tecnologias para desenvolverem sistemas que tenham alguma relação com o projeto apresentado neste documento.

3.1 Sistemas especialistas

Um Sistema Especialista (SE) é uma forma de sistema baseado no conhecimento especialmente projetado para emular a especialização humana de algum domínio específico. Um SE irá possuir uma base de conhecimento formada de fatos, regras e heurísticas sobre o domínio, tal como um especialista humano faria, e deve ser capaz de oferecer sugestões e conselhos aos usuários e, também, adquirir novos conhecimentos e heurísticas com essa interação (FLORES, 2003).

Segundo Mendes (1997), existem características que indicam se determinado problema deve ou não ser resolvido a partir da utilização de um SE. Por exemplo, se um determinado problema exige o conhecimento da tecnologia de forma geral e especialistas com este tipo de conhecimento sejam raros, uma boa abordagem seria o uso de SE. Outra característica importante na opção pelo uso de um SE destacada por Mendes (1997) é o fato de que a tarefa de diagnosticar algum problema requer conhecimento de detalhes que, se esquecidos, provocam a degradação do desempenho do sistema. No caso dos problemas a serem resolvidos nesta tese, tais considerações encaixam-se perfeitamente, pois a análise dos

telegramas Profibus e dos níveis de tensão da camada física do mesmo protocolo exigem que um especialista conheça uma série de detalhes da tecnologia. Ao mesmo tempo, poucos especialistas neste tipo de análise são encontrados no mercado. Mendes (1997) afirmou também que a construção deste tipo de sistema só é possível com a existência de especialistas para apoiar o desenvolvedor na construção das regras do SE.

Além do embasamento teórico considerado na escolha de um SE para a implementação do sistema proposto nesta tese, tal decisão apoia-se também em afirmações de que pesquisas realizadas mostraram que sistemas especialistas são boas opções para resolverem uma série de problemas de diagnóstico de falhas e processamento de alarmes (TAN, 2001).

Sistemas Especialistas podem ser aplicados em vários tipos de sistemas, cada um contendo suas características. No caso do sistema proposto nesta tese, o SE implementado tem o objetivo de diagnosticar problemas em uma rede Profibus. O uso de um SE para a finalidade de diagnosticar diferentes tipos de problemas em sistemas já foi feito por outros pesquisadores.

Na área industrial, Kaszkurewicz et al. (1997) desenvolveu um sistema especialista para supervisionar e diagnosticar problemas em uma planta industrial para produção de óleo. A ferramenta tinha como objetivo apoiar o operador informando o mau funcionamento do processo, melhorar a segurança e reduzir o número de paradas da planta. Este sistema foi implementado na plataforma Gensym G2, que é um *framework* para desenvolvimento de sistemas baseados em conhecimentos (*knowledge-based systems*) (MOORE, 1992).

Andrade (2007) desenvolveu um SE para operar em tempo real com sistemas supervisórios no auxílio da tomada de decisão dos operadores do sistema. Em seu trabalho o autor destacou que ao longo do tempo de operação de uma planta automatizada ocorrem novas situações que passam a compor novos cenários do sistema e também contribuem para o aumento do conhecimento e da experiência dos operadores humanos. Dessa maneira, destaca-se que os SE constantemente devem ser atualizados com novas regras para atender às novas demandas da planta automatizada.

Jain et al. (2008) dizem que sistemas especialistas têm provado ser um mecanismo efetivo no diagnóstico e no controle nas indústrias de energia. O autor afirma que sistemas sendo executados em apenas um computador ou até mesmo em uma intranet apresentam limitações se comparados a sistemas voltados para a internet, os quais permitem acesso de qualquer lugar a qualquer hora. Neste contexto, seu trabalho propõe o desenvolvimento de um

sistema especialista com acesso via *Web* com a finalidade de diagnosticar problemas e de controlar sistemas e equipamentos de empresas de energia.

Outro sistema voltado para sistemas de energia foi desenvolvido por Fergany et al. (2001). Em seu trabalho, o autor aplicou uma RNA e um SE para o diagnóstico de falhas em sistemas de energia utilizando para isso informações de relés e disjuntores. Sherwali (1997) também realizou um trabalho nesta área. Em seu trabalho, o autor apresentou um sistema especialista designado a analisar relés utilizados para a proteção de linhas de transmissão.

Zalis (2004) criou um sistema especialista para avaliar o status de máquinas e equipamentos elétricos de alta tensão. Para tal trabalho, vários sistemas especialistas baseados em regras foram utilizados (ALTONEX, CVEXON, CVEX, ISOLEX).

Outro SE foi utilizado para diagnosticar falhas em sistemas denominados *fire control system*⁴. Wang (2009) utilizou este tipo de sistema combinando uma árvore de falhas com uma rede neural.

Zheng et al. (2006) desenvolveram um sistema especialista para diagnosticar problemas em um processo químico. Para isso, os dados do processo são obtidos em tempo real e armazenados em uma base de dados. Após isso, a máquina de inferência do sistema processa estes dados em conjunto com o conhecimento inserido na base de conhecimentos. Por fim, o módulo de explicação descreve como e porque uma determinada conclusão foi obtida. Neste sistema, uma interface gráfica com o usuário foi desenvolvida a fim de colaborar com a comunicação entre o sistema e o usuário.

Amaya et al. (2010) desenvolveram um sistema especialista para detectar problemas em equipamentos hidro-eletrônicos. Este sistema inteligente foi implementado internamente a um sistema de manutenção inteligente chamado SIMPREBAL. Assim como no sistema proposto nesta tese, a base de conhecimentos do SIMPREBAL foi desenvolvida com base no conhecimento empírico e técnico de especialistas, a partir de manuais de usuários, de documentos técnicos e de procedimentos de instalação e de operação.

Huang Yi (2009) propôs um sistema para diagnosticar falhas em computadores de maneira remota, através da internet. Utilizando este sistema, quando alguma falha acontece no computador de um usuário, o sistema de diagnóstico se conecta a este computador de maneira remota e através de um software, coleta todas as informações necessárias. Assim, estas informações são processadas pelo sistema especialista e um diagnóstico é enviado como

resposta ao usuário. Por fim, Hori et al. (1991) desenvolveu um sistema especialista chamado VGM (*Visual Guide Maintenance System*) que apoia operadores no reparo de pequenos problemas que apesar de mínimos, são muito frequentes nos dispositivos das indústrias.

Os sistemas apresentados nos parágrafos iniciais deste item foram apenas alguns exemplos do uso de SE para diagnóstico. Além desses sistemas, muitos outros exemplos de SE utilizados na área de diagnóstico podem ser encontrados.

Voltando o foco da explicação para o funcionamento geral de um SE, tem-se que este tipo de sistema apresenta uma arquitetura com os seguintes componentes: base de conhecimentos, mecanismos de inferência, explanação e aquisição de conhecimento (FLORES, 2003):

- A base do conhecimento reúne o conhecimento do especialista modelado conforme a representação do conhecimento escolhida para modelar o domínio em questão. A implementação da base de conhecimentos normalmente é feita pelo desenvolvedor do sistema em conjunto com um especialista que fornece o conhecimento profundo sobre o assunto abordado. O desenvolvedor é o especialista em linguagens de inteligência artificial e em representação. A sua tarefa principal é selecionar as ferramentas de software e hardware para projeto, ajudar o especialista no domínio a articular o conhecimento necessário e implementar este conhecimento numa base correta e eficiente. O desenvolvedor ainda tem a missão de extrair a informação do especialista tanto através de discussões gerais da resolução do problema como através da junção de características do conjunto de problemas e sua solução (RAMOS, 2006). No caso deste trabalho, a base de conhecimento foi criada com a ajuda do engenheiro especialista em redes Profibus Cesar Cassiolato, que foi presidente regional da associação Profibus de 2006 até 2010 e é diretor da empresa Smar Equipamentos Industriais Ltda. Outro especialista que colaborou muito nesta tese foi o engenheiro Renato Veiga Torres, responsável pelo desenvolvimento dos Controladores HSE/Profibus-DP desenvolvidos pela empresa Smar Equipamentos Industriais Ltda. Além desses dois engenheiros, contou-se com o conhecimento do professor doutor Dennis Brandão e o Engenheiro Certificado Profibus Guilherme Serpa Sestito, ambos instrutores do curso de Engenheiro e Instalador Profibus

⁴ *Fire control System* é o nome dado à um conjunto de componentes trabalhando em conjunto utilizados geralmente para o controle de armas de fogo

Certificado com formação acreditada internacionalmente pela PROFIBUS *Internacional* e fornecido pela ASSOCIAÇÃO PROFIBUS BRASIL - AMÉRICA LATINA. Além disso, informações coletadas na literatura (SOFTING, 2012 e PROCENTEC, 2012) colaboraram para o aprimoramento desta base.

- O mecanismo de inferência aplica o conhecimento à solução de problemas reais. Ele é o interpretador para a base de conhecimento (RAMOS, 2006). Neste sistema, o mecanismo de inferência foi criado através de árvores de decisão, afinal, esta ferramenta oferece um alto grau de interpretabilidade. Sendo assim, as ramificações da árvore possuem todas as regras de produção que foram construídas com base no conhecimento dos especialistas já apresentados.
- O módulo de aquisição de conhecimentos é responsável pela atualização da base de conhecimentos através de um mecanismo de interação cooperativa, gerado a partir do módulo de explicação.
- O módulo de explicação é responsável pela descrição do raciocínio do sistema para o usuário, sendo ativado tanto pelo mecanismo de inferência como pelo módulo de aquisição de conhecimentos. O mecanismo de inferência evoca o módulo de explicação, visando a um processo de transformação do conhecimento representado na base de conhecimentos do SE ou deduzido no processo de busca de uma solução. No caso do sistema em questão, este mecanismo procura mostrar ao usuário o motivo exato de um determinado diagnóstico.

Os SEs não são os únicos sistemas inteligentes utilizados nesta tese. Sendo assim, o subitem 3.2 apresenta conceitos e trabalhos relacionados às RNAs.

3.2 Redes Neurais Artificiais

As RNAs, muito importantes no desenvolvimento do projeto proposto nesta tese, são definidas por Silva et al. (2010) como:

[...] modelos computacionais inspirados no sistema nervoso dos seres vivos. Possuem a capacidade de aquisição e manutenção do conhecimento (baseado em informações) e podem ser definidas como um conjunto de unidades de processamento, caracterizadas por neurônios artificiais, que são interligados por um grande número de

interconexões (sinapses artificiais), sendo as mesmas representadas aqui por vetores/matrizes de pesos sinápticos.

Em suma, as RNAs consistem em modelos computacionais que apresentam seu funcionamento baseado no comportamento do cérebro humano.

Para o entendimento desta tese, são destacados nos próximos parágrafos, alguns fatores importantes a serem considerados no desenvolvimento de uma RNA. Estes fatores são a arquitetura, a topologia e o algoritmo de aprendizagem ou treinamento da RNA.

Segundo Silva et al. (2010), a arquitetura de uma rede neural define a forma como os seus neurônios estão arranjados uns em relação aos outros. Estes arranjos são essencialmente estruturados através do direcionamento das conexões sinápticas dos neurônios. Já a topologia de uma RNA corresponde às diferentes formas de composições estruturais que a rede pode assumir. Por fim, tem-se o treinamento de uma arquitetura específica que consiste na aplicação de um conjunto de passos ordenados com o intuito de ajustar os pesos e os limiares de seus neurônios. Tal processo de ajuste também é conhecido como algoritmo de aprendizagem.

Nesta tese, o tipo de RNA utilizada é a Perceptron Múltiplas Camadas (PMC) (do inglês, *Multilayer Perceptron* - MLP). A arquitetura utilizada é a *feedforward* de camadas múltiplas e o algoritmo de treinamento é o *backpropagation* tradicional. O valor da precisão é $\varepsilon = 10^{-6}$ e a taxa de aprendizagem foi de 0,01. Outras redes como a Função de Base Radial (RBF) e a Quantização Vetorial por Amostragem (LVQ) poderiam ter sido adotadas, afinal, estas são bastante utilizadas para resolver problemas de classificação, mas optou-se pela utilização da PMC baseando-se para isso nos ótimos resultados obtidos em testes preliminares desenvolvidos por Souza (2012). Outro fator importante relacionado ao treinamento é o fato do mesmo ser supervisionado. A estratégia de treinamento supervisionado consiste em ter-se disponível, considerando cada amostra dos sinais de entrada, as respectivas saídas desejadas (SILVA et al., 2010). Mais detalhes sobre o funcionamento de uma PMC pode ser encontrado em Silva et al. (2010).

Definido o tipo de rede a ser utilizado, resta definir a topologia da mesma. Sendo assim, a PMC criada possui vinte neurônios de entrada, 20 neurônios em sua única camada escondida e dois neurônios de saída. Com apenas dois neurônios, apenas dois tipos de saídas poderão ser obtidas. O uso de apenas dois neurônios na camada de saída é possível, pois o sistema é composto por várias RNAs e cada uma é responsável apenas por informar se um sinal Profibus possui um determinado tipo de problema ou não possui, ou seja, cada RNA

possuirá apenas dois tipos de saída. Para finalizar, tem-se que a função de ativação utilizada é do tipo *Sigmoidal*.

Uma série de análises foram realizadas até a obtenção desta arquitetura. Detalhes dessas análises podem ser encontrados em Souza (2012). A Figura 3.1 mostra uma imagem com a MLP utilizada.

Figura 3.1 – Arquitetura da Rede MLP utilizada

Como pode ser observado na Figura 3.1, nas redes do tipo PMC os neurônios de cada camada apenas são conectados aos neurônios da camada imediatamente superior, não existindo qualquer outro tipo de conexão entre eles. Todo o conhecimento obtido pelo processamento dos neurônios da camada intermediária será armazenado na forma de pesos sinápticos (SILVA et al. 2010).

A escolha de uma RNA para a classificação dos sinais da camada física do protocolo Profibus justifica-se por uma série de vantagens oferecidas por esta tecnologia e que são destacadas por Ambrósio (2002). Primeiramente, pode-se destacar que as RNAs se baseiam nos exemplos históricos que lhes são fornecidos. Para o caso da análise dos sinais Profibus, esta característica é muito importante, afinal, grande parte deste trabalho foi realizado no laboratório do Centro de Competência Profibus do Brasil (CCP), localizado na Escola de Engenharia de São Carlos, USP, onde foi possível a coleta de uma série de amostras que seriam então utilizadas no treinamento das RNAs. Outra vantagem citada por Ambrósio (2002) é o nível de dificuldade relativamente baixo oferecido por estas redes em sua implementação, afinal, o tempo necessário para implementar uma rede é menor do que o utilizado para a construção de um sistema especialista equivalente, além do menor custo envolvido. Por fim, tem-se a questão da capacidade de generalização dessas redes,

considerada a característica mais importante para esta tese, afinal, os sinais Profibus com ou sem deformações causadas por inadequações de instalação não possuem um formato exato. Estes sinais possuem certo padrão em sua forma dependendo do problema na instalação da rede Profibus, mas estes não são exatamente os mesmos para cada tipo de problema. Reforçando ainda mais a vantagem da RNA de ser capaz de se generalizar, Cavani (2007) afirmou que estas redes tem tolerância a ruídos e alta velocidade de processamento. Ambrósio (2002) também cita a adaptabilidade que as RNAs possuem. Segundo o autor, uma vez construída uma RNA eficiente em dada aplicação, o surgimento de novos padrões de classificação não exigem que uma nova arquitetura seja criada. Na maioria das vezes, basta que a RNA seja retreinada com base nos novos dados históricos que surgiram. Para o caso do treinamento com sinais Profibus, esta característica é muito importante, afinal, o formato de um sinal para determinados tipos de problemas de instalação podem sofrer deformações suficientes para que a RNA não seja capaz de classifica-lo corretamente, mesmo estas redes apresentando uma boa capacidade de generalização. Nestes casos, amostras de novos sinais são inseridas na base de exemplos de treinamentos e um novo treinamento é realizado.

Vale lembrar que os dados de entrada e de treinamento de uma RNA necessitam de tratamento prévio. Estes devem ser normalizados e cuidadosamente selecionados para que a rede seja corretamente ensinada a agir. Dados de má qualidade produzem resultados falhos (AMBRÓSIO, 2002). Um exemplo desta situação pode ser encontrado em Souza (2012). Em seu trabalho, foram realizados testes com 10, 15, 20 e 30 neurônios. A RNA utilizada foi do tipo *Multi Layer Perceptron*. Esta RNA foi treinada por 10 vezes a partir do *toolbox* de Redes Neurais do Matlab (*Neural Network Toolbox*) e obteve-se uma margem muito baixa de acerto de em média 44%. Após isso, as amostras de treinamento foram analisadas e notou-se que existiam muitas amostras de má qualidade. Por exemplo, algumas amostras de uma determinada classe eram muito parecidas com algumas amostras de outra classe. Isso gerava estes resultados ruins. Por fim, foram selecionadas manualmente apenas as amostras que realmente caracterizavam uma determinada classe e a rede foi treinada e testada novamente com o excelente resultado de 100% de acerto.

Da mesma maneira que uma RNA foi utilizada aqui, outros autores que realizaram trabalhos na área de processos industriais já utilizaram deste artifício para reconhecer variados tipos de padrões. Como é o caso de Naidu et al. (1990) que discute o uso de uma RNA para detecção de falhas em sensores em um sistema de controle. Este autor também utiliza o algoritmo *backpropagation* para análise do sistema e detecção das falhas. A rede foi composta

de uma camada de entrada, uma de saída e uma camada neural intermediária. O estudo realizado com essa arquitetura mostrou que as principais vantagens da metodologia proposta são a capacidade deste método de capturar as características não lineares, a possibilidade de treinamento *on-line* e a possibilidade de implementação rápida, reforçando ainda mais a escolha das RNAs para a análise dos sinais Profibus. No mesmo ano Narendra et al. (1990), afirmou que as RNAs poderiam ser utilizadas efetivamente na identificação e controle de sistemas dinâmicos não lineares. Foi discutido um modelo de RNA onde duas arquiteturas eram interconectadas: uma rede PMC e uma rede recorrente, ou rede *Hopfield*. Em cinco dos seis testes realizados, foi utilizado um modelo que tinha a função de identificar os dados da planta. O algoritmo de *backpropagation* foi utilizado para ajustar os parâmetros da RNA. O sexto exemplo foi utilizado para indicar como o algoritmo *backpropagation* deve ser utilizado na identificação dos problemas. Ao final do experimento, os resultados da simulação mostraram que essa combinação era possível e este método era aplicável.

Ainda tratando-se de processos industriais, mais especificamente de processos químicos, Becraft (1993) realizou em um mesmo trabalho dois estudos de casos para realizar diagnósticos utilizando RNAs do tipo PMC com algoritmo de treinamento *backpropagation*. No primeiro caso, o objeto de estudo foi um forno industrial e no segundo caso foi estudada uma usina de destilação de multi-colunas. Em ambos os estudos, são analisados os dados de entrada e seus efeitos sobre as possíveis falhas que ocorrem nesses processos. Neste primeiro caso obteve-se 97% de acerto na classificação das amostras. Já no segundo estudo de caso o número de variáveis analisadas é muito maior que no primeiro caso devido ao fato desta planta ser dividida em três colunas, o que fez com que o uso de uma RNA simples não fosse viável. Portanto, foram utilizados dois níveis de redes PMC para implementar a estratégia de diagnóstico. O primeiro nível é implementado na planta principal e o segundo é implementado nas três colunas de baterias, sendo uma RNA para cada bateria. Após os testes, foi constatado que este esquema oferece um bom grau de robustez, afinal, a porcentagem de acerto no melhor caso chegou a 94%. Além disso, a estratégia hierárquica foi capaz de diagnosticar cenários com novas falhas. Esta técnica de dividir os dados para serem analisados por diferentes RNAs também foi utilizada nesta tese para analisar os diferentes sinais Profibus.

Outro trabalho que merece destaque aqui foi realizado por Wu et al. (2000). Neste, uma RNA do tipo PMC é utilizada para realizar diagnóstico de falhas em linha mecânica de motor de indução. A RNA foi implementada com quatro camadas, ou seja, uma camada de

entrada, duas camadas intermediárias e uma camada de saída. Como resultado obteve-se 100% de reconhecimento dos dados com falhas e 97,6% de reconhecimento dos dados considerados sem nenhum tipo de falha.

Em Wu et al. (2004) é utilizada uma RNA do tipo Função Base Radial (RBF) para realizar análise e classificação de falhas em máquinas de indução. Neste caso os dados de entrada eram compostos por dados de falhas elétricas, mecânicas e dados de situação normal da rede. Os resultados obtidos foram: 99,8% de reconhecimento dos dados da rede em condições normais, 96,5% de reconhecimento dos dados com falhas elétricas e 100% de reconhecimento dos dados de falhas mecânicas.

O uso das RNAs em processos industriais também foi feito no trabalho de Maki et al. (1997), onde foi realizado um estudo utilizando-se uma RNA do tipo PMC para detectar e diagnosticar falhas em um processo produtivo. Este processo trata-se de um reator de tanque continuamente agitado (*Continuously well-Stirred Tank Reactor - CSTR*), onde situações envolvendo perdas de calor desprezíveis e densidades constantes eram assumidas. O trabalho tratou sobre as falhas que ocorriam em períodos transitórios de operação do tanque.

Apesar de não possuir relação com processos industriais, alguns trabalhos, como o trabalho de D'Emery (2007) merece destaque, pois neste, foi realizada a análise de um determinado tipo de sinal. Em seu trabalho, o autor utilizou uma RNA para o reconhecimento de sinais de odores utilizando nariz artificial, com o objetivo de se detectar um determinado tipo de diabetes, conhecido como *Diabetes Mellitus*. Os dados foram obtidos através da leitura de um sensor de aroma. Neste caso, as redes neurais PMC e RBF foram utilizadas na classificação dos padrões e obtiveram a porcentagem de acerto de 100%.

Também analisando-se sinais, Lira (2004) propõe o uso de RNA para a classificação de sinais com distúrbio em sistemas de potência. Foi utilizada uma rede neural do tipo PMC a fim de que esta indique qual é o tipo de distúrbio presente no sinal. A classificação média para todos os testes realizados foi de 94,4% de reconhecimento em todos os sinais apresentados.

No trabalho de Shamsuddin (2005), foram realizados experimentos com o objetivo de classificar, através de uma RNA PMC, os possíveis sons que vem do coração e, através dessa análise, identificar o significado de cada uma das possíveis situações. Neste caso, os sons do coração foram coletados utilizando-se válvulas cardíacas e após a coleta esses dados foram digitalmente filtrados, segmentados e em seguida, reduzidos a uma estrutura arbitrária de 64 pontos. Por fim, foram pré-processados utilizando-se FFT (Transformada Rápida de *Fourier*) e carregados na RNA para que fosse feita a classificação. Para testar a robustez da rede,

foram utilizados sons com diferentes tipos de ruídos. O resultado dos testes foi ótimo, com a RNA classificando corretamente 100% dos sinais apresentados.

No trabalho de AMINIAN (et al. 2000) foi proposto um método de diagnóstico de falha em circuitos analógicos baseados em RNA, cujo método de pré-processamento utilizado foi a transformada *Wavelet*. A RNA utilizada foi do tipo *backpropagation*. A RNA trabalha analisando as respostas de impulso do circuito eletrônico analógico e, através disso, consegue detectar e identificar falhas no sistema. Além disso, obteve-se uma classificação correta de 100% dos dados analisados.

No estudo de GÜLER (et al. 2005), utilizou-se uma rede neural para detectar mudanças eletrocardiográficas em pacientes com epilepsia parcial. O trabalho foi dividido em duas etapas. Na primeira foi realizada uma extração das características dos sinais cardíacos utilizando-se transformada *wavelet* e, na segunda, foi feita a classificação desses sinais utilizando-se uma RNA do tipo PMC, com algoritmo de treinamento *backpropagation*. Foram utilizados como dados de entrada, dois tipos de situações: sinais cardíacos considerados normais e sinais representando o problema de epilepsia parcial. Após o treinamento foi realizada uma comparação entre os resultados da classificação, os valores dos parâmetros estatísticos e os parâmetros de avaliação de desempenho da RNA e, como resultado, obteve-se uma variação de 90% a 97,5% de reconhecimento dos dados apresentados, o que indica que a RNA atingiu os objetivos esperados.

Ainda se tratando de classificação de sinais, foi realizado um estudo por ORHAN (et al. 2010), onde um sistema para classificação de sinal de eletroencefalograma (EEG) utilizando-se uma RNA do tipo PMC foi proposto. Foram analisadas cinco classes de dados, cada uma delas contendo 100 amostras de um exame de EEG. Foram utilizados como entrada da RNA a média, o desvio padrão e a entropia desses coeficientes. Foram realizadas duas diferentes classificações utilizando-se a mesma RNA. Na primeira foram classificados os dados dos exames de EEG de três tipos de pacientes: os saudáveis, os pacientes com epilepsia durante uma crise de convulsão e os pacientes com epilepsia em um intervalo livre de crises. Na segunda foram analisados os dados dos exames referentes apenas aos pacientes com epilepsia, em momentos com e sem crise. No primeiro teste obteve-se um resultado de 99,60% de reconhecimento dos dados analisados e, no segundo a rede conseguiu classificar 100% dos dados apresentados a ela.

Já no trabalho de Angrisani (2001), foi realizado um estudo para desenvolver uma arquitetura de processamento de sinal digital capaz de classificar e detectar, automática e

simultaneamente, sinais transitórios. A unidade básica desta arquitetura é a rede *Wavelet*, que combina a capacidade da transformada *wavelet* na análise de sinais estacionários com a capacidade de classificação das redes neurais.

A partir das pesquisas destacas aqui, pode-se notar que os dados analisados sofrem um pré-processamento do sinal antes que eles sejam carregados na RNA. Alguns estudos mostram o uso de técnicas como PCA (Análise de Componentes Principais), Transformada de *Fourier* e Transformada *Wavelet*, entre outros, que são utilizadas para extrair dos dados suas principais características e diminuir a dimensionalidade desses dados. Na implementação do sistema proposto por esta tese não foi utilizada nenhuma dessas técnicas de pré-processamento do sinal, mas alguns mecanismos para eliminar das amostras algumas características prejudiciais ao treinamento das RNAs foram criados. O primeiro deles foi eliminar dos dados de treinamento os sinais com valores negativos. Após isso, os dados foram todos normalizados e por fim, um filtro para eliminar do treinamento os sinais com interferência eletromagnética foi criado. O uso de apenas este tipo de pré-processamento foi possível devido ao fato dos sinais colhidos possuírem características que os diferenciam entre si, fazendo com que, na hora da classificação, a rede conseguisse identificar corretamente cada uma das classes apresentadas. De qualquer forma, pré-processamentos como PCA, Transformada de *Fourier* e Transformada *Wavelet* normalmente deixam uma RNA mais robusta para novas formas de sinais, fazendo com que análises baseadas nessas técnicas possam ser estudadas em trabalhos futuros.

Além dos trabalhos que utilizaram exclusivamente RNAs, existe uma série de trabalhos que utiliza-se de mais de um sistema inteligente para construir uma boa solução para algum problema. Um exemplo desses trabalhos é apresentado por Awadallah et al. (2003) que propõe a aplicação de ferramentas de inteligência artificial no diagnóstico de falhas em máquinas elétricas e drivers. O trabalho teve como objetivo realizar o monitoramento de condições que levam à falha. O autor afirma que o diagnóstico correto e precoce de falhas resulta em manutenção programada e rápida resolução do problema em questão e também evita consequências nocivas para a máquina e possíveis prejuízos no processo. Os autores analisaram o uso de sistemas inteligentes, como RNA, Sistemas Fuzzy, Sistemas Fuzzy Adaptativo e Sistemas Especialistas para realizar automação de procedimentos de diagnósticos. Esses sistemas foram utilizados principalmente com a finalidade de realizar a previsão de operação anormal do sistema, localizar elementos defeituosos, classificar as

falhas, fornecer prognósticos e por fim, tomar decisões ou sugerir soluções em um nível de qualidade comparável a especialistas humanos.

Também utilizando alguns sistemas inteligentes como RNA e Lógica Fuzzy (denominado Sistema de Inferência Neuro-Fuzzy Adaptativo - ANFIS), o trabalho de Kamel et al. (2009) mostra um estudo baseado em inteligência artificial com o objetivo de proteger linhas de transmissão em sistemas elétricos de potência. Este projeto foi dividido em três partes. A primeira etapa consistiu em detectar a ocorrência da falha em um curto período de tempo e isolar a seção defeituosa da linha de transmissão. Na segunda etapa, foi realizada a classificação do tipo de falha. Por fim, na última etapa foi realizada a localização da falha. Esse esquema proposto apresentou um alto grau de satisfação, visto que, no pior caso, a rede obteve apenas 8% de erro na classificação dos sinais apresentados.

Assim como descrito nestes trabalhos, alguns sistemas inteligentes são utilizados no sistema proposto nesta tese a fim de atingir os objetivos desejados. Sendo assim, o subitem 3.3 descreve conceitos de lógica *fuzzy* e apresenta alguns trabalhos que utilizaram-se da mesma.

3.3 Lógica Nebulosa (*Fuzzy*)

Um sistema *fuzzy* foi utilizado para auxiliar os usuários nas definições de algumas variáveis de tempo utilizados em uma rede Profibus DP.

A ideia principal deste tipo de sistema é aproximar a decisão computacional da decisão humana (ROSS, 2005). Em vez da resposta do sistema ser um valor 1 ou 0, significando respectivamente, verdadeiro e falso, um sistema nebuloso pode dar como resposta valores como Baixo, Muito Baixo, Médio, Alto, Muito Alto, entre outros. Este tipo de característica destaca a natureza imprecisa e qualitativa do conhecimento humano. Tal característica dificulta o tratamento matemático preciso exigido pelos modelos tradicionais, mas, por outro lado, a capacidade humana de lidar com sistemas complexos e imprecisos tem estimulado a busca por mecanismos que sejam capazes de trabalhar com este tipo de informação (BABUSKA, 2001). Um sistema *fuzzy* constitui, portanto, uma técnica que representa elementos ou quantidades imprecisos ou definidos de modo vago (BARDOZZY, 1995).

Segundo Zadeh (1973), o pensamento humano não é elaborado em forma de números, mas sim, por classes de objetos onde a transição de pertinência para não pertinência é gradual

e não abrupta. Esta maneira de organizar o pensamento é diferente em relação à lógica clássica, em que um determinado objeto pertence ou não a uma determinada categoria ou a um determinado conjunto. Esta ideia é flexibilizada quando trata-se de conjuntos fuzzy. Neste caso, um determinado elemento pode pertencer parcialmente a um ou mais conjuntos. O termo utilizado para definir o quanto um elemento pertence ou não a um determinado conjunto é grau de pertinência. O grau de pertinência de um elemento varia entre 0 e 1. Se o grau de pertinência é 1, significa que o mesmo está totalmente incluso no conjunto. Grau zero significa totalmente exclusivo. Qualquer valor entre 0 e 1 indica um grau de pertinência parcial ao conjunto. Dessa maneira, existem infinitos graus de incerteza, e uma resposta no formato de um “sim” ou de um “não”, na maioria das vezes é incompleta (ZIMMERMANN, 1987).

Em particular, as fronteiras entre os conjuntos não são claramente definidas, fazendo com que o uso desses sistemas para classificação de informações inexatas torna-se muito indicado (CHIANG E HSU, 2002).

A curva que define como cada ponto no espaço de entrada é mapeado com um valor do grau de pertinência de 0 a 1, é chamada de função de pertinência. Esta função pode ser trapezoidal, triangular, guassiana, sigmoidal ou baseada em algum polinômio (BUENO, 2006). No caso desta tese, as funções de pertinência triangular e trapezoidal são utilizadas.

A aplicação de um sistema fuzzy inicia-se com a fuzzificação das entradas, ou seja, a transformação dos valores CRISP em valores nebulosos. Esta transformação dá-se através dos conjuntos fuzzy definidos, ou seja, o valor CRISP é analisado e o grau de pertinência deste valor é definido com base nas funções de pertinência. A partir desses valores, um modelo de inferência é aplicado com base em um conjunto de regras e fornece como saída outro conjunto nebuloso. Este conjunto então poderá ser novamente transformado em um valor CRISP, ou seja, defuzzificado.

O modelo de inferência utilizado nesta tese é o modelo de Mamdani. Neste modelo os valores das variáveis de entrada fuzzificados são analisados e a partir das regras criadas pelos especialistas do sistema são obtidas as regras que estão ativas. As regras ativas irão colaborar com o cálculo da saída do sistema de inferência. Para cada regra ativa, um conjunto de saída é obtido. A obtenção do mesmo é realizada a partir do operador MIN ou MAX. Por fim, os conjuntos resultantes das regras ativas são agregados e a saída final do sistema é fornecida. Tal saída é um valor nebuloso e caso seja necessário, um processo de defuzzificação pode ser feito. Para esta tese, o método do centroide foi utilizado. Mais detalhes sobre sistemas nebulosos podem ser encontrados em Ross (2005).

Analisando-se as definições apresentadas, tem-se que este tipo de sistema apresenta-se como uma ótima alternativa no momento de definir a variável de tempo T_{TR} , que representa o tempo de um ciclo de comunicação completo de uma rede Profibus. Para o cálculo do valor desta variável, muitos parâmetros são necessários e a maioria deles é estimada de maneira aproximada. Sendo assim, justifica-se o uso do sistema *fuzzy* para que o T_{TR} de uma rede Profibus seja estipulado. Uma explicação mais detalhada sobre o cálculo do T_{TR} , o porquê de suas imprecisões e sobre o sistema fuzzy criado nesta tese podem ser encontrados no item 4.6.

Sistemas deste tipo são utilizados em diversas aplicações. Nedeljkovic (2004), por exemplo, utilizou sistemas fuzzy na classificação de imagens. O autor realizou os experimentos a partir da ferramenta *Matlab's Fuzzy Logic Toolbox*. Kavdir (2003) e Guyer (2005) utilizaram lógica fuzzy como suporte à tomada de decisão para determinar o nível de qualidade de maçãs. Na área da saúde, a classificação através de sistemas fuzzy também é bastante explorada, como é o caso do trabalho de Harikumar (2003), que desenvolveu um modelo de classificação fuzzy para análise do nível de risco de epilepsia baseando-se para isso em sinais de eletroencefalograma.

Na área de automação industrial, assunto desta tese, o grande uso de sistemas nebulosos acontece nas operações de controle, como é o caso de Spatti (2006), que apresenta uma estratégia para realizar o controle de tensão em um sistema de distribuição de energia elétrica através da aplicação de sistemas nebulosos para tornar os relés reguladores de tensão mais flexíveis. Na mesma área, Usida (2007) propõe um relé regulador de tensão baseado em sistemas fuzzy que irá atuar em transformadores de potência instalados em subestações de distribuição. Tal relé foi implementado por meio de um algoritmo computacional no *Matlab*. Os resultados foram obtidos por meio de simulações do real comportamento de carga indicando a viabilidade do sistema proposto. Bolños (2007) propõe em seu trabalho técnicas de controle para veículos submarinos, através de um sistema de controle inteligente e adaptativo a ser aplicado na navegação de veículos submarinos semiautônomos. Utiliza-se uma técnica baseada no controle nebuloso, visando gerenciar o veículo submarino no controle de velocidade, profundidade, orientação e na evasão de obstáculos.

Lee K.C. (2003b) criou um sistema fuzzy para realizar um mecanismo para o controle de um motor. Este estava distribuído em uma rede Profibus-DP. A ideia do sistema foi a criação de inferências a partir de variáveis linguísticas relacionadas ao valor de entrada do

sistema de controle, o valor desejado, o erro e a variação do erro entre uma iteração e outra do sistema.

Ainda na área industrial e envolvendo análise de variáveis de tempo, Chen (2003), criou um mecanismo que utiliza o método *fuzzy* de maneira conjunta com uma rede *backpropagation* para estimar o tempo de produção de circuitos eletrônicos (*wafer*). Neste sistema, os especialistas no controle da produção são convidados a expressar suas opiniões sobre a importância de cada parâmetro de entrada na previsão do tempo de saída com termos linguísticos, o que pode ser convertido em números nebulosos, que são processados para que um valor de saída seja obtido.

Além desses artigos citados, Altrock et al. (1996) escreveram um artigo que apresenta e discute oito aplicações da lógica fuzzy na área da automação industrial. Ainda na área de automação industrial, envolvendo também a área da educação, Mossin (2008) publicou um trabalho que classifica alunos como “Reprovado”, “Aprovado Com Muitas Restrições”, “Aprovado com Poucas Restrições” e “Aprovado”, em vez de apenas aprovar ou reprovar o mesmo. Nota-se então, a vasta possibilidade de aplicação dos sistemas fuzzy para classificar algum tipo de informação.

Considerando que os conceitos necessários para o entendimento desta tese foram explanados, tem-se no capítulo 4 a proposta de uma plataforma para a realização de diagnósticos em uma rede Profibus.

4 Proposta de plataforma para diagnóstico Profibus

A manutenção de sistemas é um dos principais custos para indústrias fazendo com que trabalhos para diminuir estes gastos sempre estão sendo desenvolvidos (FORNATO, 2004). Nos últimos anos, pesquisas têm sido feitas na área de supervisão de processos, detecção de falhas e diagnóstico com o intuito de melhorar a confiabilidade e a disponibilidade dos sistemas de controle (HUANG et al., 2006). Quanto antes um problema é observado e diagnosticado, menor será o prejuízo de um processo. Para que problemas sejam detectados, é necessário que informações dos dispositivos e das redes de comunicação dos mesmos sejam coletadas e analisadas constantemente. Este tipo de análise é amplamente estudada e uma série de trabalhos deste tipo, em diferentes redes e protocolos de comunicação, colabora para o aprimoramento do diagnóstico de problemas. Além destes trabalhos, existem também algumas patentes que tratam deste assunto. Um exemplo é o caso da patente desenvolvida por Rosemount Inc. (2009) que sugere a identificação de problemas em sistemas para controle de processos industriais através da análise de telegramas transmitidos com defeitos, pedidos de retransmissão de dados, entre outros. A obtenção destes telegramas dá-se a partir de um dispositivo conectado ao segmento *fieldbus*. De maneira similar, Softing AG (2011) propõe uma patente para analisar os campos de dados dos telegramas transmitidos em redes fieldbus com o objetivo de detectar mudanças no número de dispositivos de rede ativos, a ocorrência de telegramas repetidos, a repetição de processos de reinicialização ou reparametrização de dispositivos e o acúmulo de telegramas de alarmes. Com o objetivo de obter diagnóstico a partir da análise dos sinais transmitidos por redes de controle de processos industriais, Rosemount Inc. (2008) patenteou a idéia de detectar problemas em um dispositivo através da análise da amplitude e da forma de onda dos sinais transmitidos por este tipo de rede.

Sabe-se que o projeto proposto por esta tese realiza a coleta de dados de redes Profibus e a partir disso faz uma análise fornecendo, como saída, um diagnóstico de algum tipo de problema. Para tal análise, sistemas inteligentes são utilizados. Assim como nesta tese, outros autores criaram sistemas que coletam dados trafegados em redes de comunicação, realizam uma análise e fornecem o diagnóstico de problemas com base em sistemas inteligentes. Este é

o caso de Mataix et al. (2003) que utilizou sistemas inteligentes para detectar problemas em uma rede de comunicação TCP/IP responsável pelo controle de dispositivos que supervisionam ferrovias. Este tipo de rede é composta por sensores que monitoram o estado das linhas de trem, a temperatura dos eixos do trem, pessoas atravessando a linha em pontos proibidos, etc. Como qualquer rede de comunicação, problemas como defeitos no barramento de comunicação, nós sem comunicação, etc. podem acontecer. A dificuldade para este tipo de rede é que a mesma cobre centenas de quilômetros e por isso o monitoramento da comunicação torna-se necessário. Este sistema trabalha enviando telegramas para sondar os nós da rede e a partir das respostas um diagnóstico automático é fornecido. Sendo assim, o sistema intervém na comunicação da rede para obter informações, algo que não acontece com o sistema proposto nesta tese, que coleta as informações trafegadas sem interferir na comunicação, ou seja, de forma passiva.

Um trabalho que utilizou uma técnica muito parecida com o sistema proposto nesta tese foi realizado por Fong et al. (1994), que criou um sistema para realizar diagnóstico automático em redes de computadores. O sistema analisa telegramas do protocolo de comunicação, como por exemplo, os cabeçalhos destes telegramas, os mecanismos de reconhecimento dos dispositivos conectados na rede (*handshake*), telegramas para controle de erro, controle de fluxo além dos procedimentos de gerenciamento da rede. A partir destes dados, um sistema especialista deduz quais os componentes da rede estão funcionando de maneira incorreta.

Também analisando telegramas transmitidos em uma rede de comunicação, Lai (1989) desenvolveu um sistema que possui como entrada mensagens de erro geradas pelos elementos conectados. A rede analisada era denominada *Intelligent Network*. Estas mensagens eram analisadas por um sistema especialista híbrido que fornecia, como saída, o nome do elemento onde o problema aconteceu. O autor destacou que o conhecimento para a elaboração do sistema foi adquirido a partir das especificações que haviam sido criadas para este tipo de rede.

A maioria das redes de comunicação possuem telegramas de diagnóstico nativos do protocolo. A rede Profibus não é diferente. Muitas ferramentas analisam estes telegramas e apresentam ao usuário informações de possíveis problemas que estão acontecendo em uma rede. O sistema proposto por esta tese possui um módulo que realiza este tipo de análise. De maneira semelhante, Fornaro (2004) analisa os dados de diagnóstico para informar algum tipo de problema em algum dispositivo da rede. Além disso, o autor analisa algumas informações

sobre o dispositivo e a partir disso busca detectar problemas no mesmo. Um exemplo é a análise que o mesmo faz em um transmissor de pressão diferencial. Nesta análise, dados como a quantidade de vezes que um determinado transmissor recebeu uma pressão muito alta ou casos em que a pressão variou bruscamente são quantificados. A partir disso, o sistema prediz a condição de “saúde” do dispositivo, afinal, estas variações e picos de pressão são um dos fatores que contribuem para diminuir a vida útil de um dispositivo. Outro trabalho que procura detectar problemas em um dispositivo de maneira preditiva foi realizado por Simeón (2008). Neste, o autor analisa dados oriundos de um servidor OPC e apoia a decisão dos operadores a partir de sugestões de ações de manutenção. As sugestões são baseadas nos diagnósticos e prognósticos do estado de funcionamento de máquinas e equipamentos. Para tal, o autor utilizou-se dos mesmos sistemas inteligentes utilizados no trabalho descrito por este documento, ou seja, lógica *fuzzy*, SEs e RNAs.

Outro sistema que realiza diagnósticos baseando-se em dados dos sensores, atuadores e do processo em si foi desenvolvido por Huang et al., (2006). Estes autores criaram um sistema de detecção de falhas e diagnóstico em tempo real que analisa três níveis de um processo industrial. O primeiro é o nível de componentes em que sensores e atuadores são analisados individualmente. Depois a análise é realizada nas malhas de controle individualmente e, por fim, uma análise generalista, onde todos os laços de controles são analisados em conjunto com outras informações é executada. Para tal, uma série de algoritmos foi utilizada, destacando-se entre estes um sistema *fuzzy*.

Assim como os trabalhos mencionados focaram na análise de dados de redes de comunicação para realizarem diagnósticos, o projeto desenvolvido nesta tese analisa dados de cada camada do protocolo Profibus e fornece como saída o diagnóstico de algum problema que possa estar acontecendo. Para a validação desta proposta, uma plataforma contendo um hardware, um simulador e um sistema de análise inteligente foi criada.

A Figura 4.1 apresenta uma visão superficial de como o sistema está estruturado.

Figura 4.1 – Visão geral do projeto (Hardware, Simulador e Software de análise)

Na Figura 4.1 pode-se destacar dois módulos do sistema. O primeiro módulo, foco deste trabalho, é o software de análise inteligente, denominado ProfiDoctor. Este receberá os dados e utilizando-se de algoritmos baseados em Redes Neurais Artificiais, Sistemas Especialistas e Lógica *Fuzzy* fornecerá alguns diagnósticos.

O segundo módulo é um hardware responsável pela leitura e amostragem do sinal e dos telegramas do Profibus DP. Este módulo conecta-se à rede Profibus DP e a partir daí, coleta amostras dos sinais da camada física além de capturar os telegramas transmitidos na rede. A partir disso, através de uma rede Ethernet e de uma conexão TCP/IP, envia estes dados ao ProfiDoctor. O desenvolvimento deste hardware está sendo realizado em um projeto de pesquisa na EESC-USP e considerando o fato de que a elaboração de um hardware funcional para a coleta de dados exige um prazo mais prolongado, este módulo não será utilizado na obtenção dos resultados desta tese. Para substituir isso, um terceiro módulo foi criado por Torres (2011), tendo como objetivo principal simular uma rede Profibus para auxiliar na obtenção dos resultados deste projeto. Este simulador conecta-se ao ProfiDoctor e envia informações através da mesma interface de comunicação Ethernet utilizada pelo hardware. Assim, os mesmos dados que o hardware obtém da rede Profibus e envia ao ProfiDoctor são criados sinteticamente e enviados a partir do simulador, eliminando a necessidade de uma rede Profibus DP real. Além de ser muito importante na execução de

testes e na obtenção dos resultados deste trabalho, o simulador pode ser útil em treinamentos, afinal, o grande crescimento deste tipo de protocolo e a consequente necessidade de profissionais capazes de lidar com este tipo de rede impulsionam uma grande demanda de ambientes para capacitação. É importante destacar que o simulador e o hardware não se conectarão ao software de maneira concomitante.

Considerando que o ProfiDoctor precisa receber os mesmos tipos de dados tanto do hardware quanto do simulador, um padrão para o envio dessas informações foi criado.

Primeiramente, vale lembrar que a conexão entre estes módulos dá-se a partir do protocolo TCP/IP. Esta conexão baseia-se em um número de IP e de uma porta que são pré-definidas com um valor padrão e podem ser modificadas a partir de um arquivo de configuração que armazena essas informações. Quando a conexão é estabelecida, telegramas de dados iniciam a troca de informações. Da mesma maneira que os outros protocolos possuem um determinado formato para trocar informações, é necessário que os dados a serem enviados ao ProfiDoctor pelo hardware ou pelo simulador sejam organizados para possibilitar uma interpretação eficiente. Neste contexto, a Figura 4.2 apresenta a estrutura dessas informações.

Figura 4.2 – Telegrama ProfiDoctor

O padrão apresentado na Figura 4.2 foi denominado telegrama ProfiDoctor. Abaixo segue uma explicação do significado e da utilização de cada um dos campos desta figura:

- **Timestamp:** representa o momento em que a mensagem Profibus é lida pelo hardware. É importante notar que o momento em que o telegrama de dados é recebido pelo ProfiDoctor não é o mesmo momento que o mesmo telegrama é lido pelo hardware. O hardware recebe o telegrama e registra o momento do recebimento. Após isso, o telegrama ProfiDoctor é montado e enviado. Sendo assim o valor do *timestamp* é o valor registrado no momento da leitura do dado feita pelo hardware. No caso dos telegramas criados pelo simulador, o *timestamp* utilizado é emulado e calculado de acordo com as equações de temporização da rede Profibus. Isso garante que o

simulador seja coerente em relação às taxas de comunicação reais (mais detalhes em TORRES, 2011).

- **Timestamp (μs):** considerando a necessidade de uma precisão de tempo de microssegundos, um segundo campo de *timestamp* é utilizado. Assim, o campo *Timestamp* (μs) envia apenas a informação de microssegundos e o campo *Timestamp* envia o tempo com a precisão de milissegundos.
- **Tipo de pacote:** o objetivo deste campo é saber que tipo de sinal o telegrama ProfiDoctor carrega (campo *BITs amostrados*). Caso o sinal possua uma deformação grande o suficiente para corrompê-lo, este campo possuirá o valor zero. Caso o sinal tenha sido interpretado corretamente, o campo possuirá o valor 1. Este bit é necessário, pois quando o telegrama não é interpretado, é necessário que o ProfiDoctor seja informado disso. Sabendo isso, o sistema analisará o histórico das mensagens para verificar a mensagem anterior. Pelo campo *Destination Address* da mensagem anterior (*request* do mestre), é possível saber qual escravo está respondendo com uma interferência que prejudique a interpretação do sinal. Outro valor para este campo é o valor 2, que representará o sinal de IDLE, ou seja, quando o tipo do telegrama for 2, o sinal carregado no campo *BITs amostrados* será o sinal de IDLE (aproximadamente 1V quando a rede funcionando corretamente). Para obter este tipo de informação, o hardware irá amostrar um sinal IDLE a cada ciclo. No caso do simulador, o mesmo criará um telegrama deste tipo a cada ciclo.
- **Endereço do dispositivo fonte:** endereço do dispositivo que enviou o telegrama. Caso o tipo de telegrama seja SC (*Short Acknowledgement Frame*), este campo possuirá valor -1;
- **Endereço do dispositivo destino:** endereço do dispositivo de destino do telegrama enviado. Caso o tipo de telegrama seja SC (*Short Acknowledgement Frame*), este campo possuirá valor -1;
- **Mensagem Profibus:** esta é a própria mensagem Profibus oriunda da camada de enlace deste protocolo;
- **Tamanho da mensagem Profibus:** este campo possui o tamanho da mensagem Profibus propriamente dita, oriunda da camada de enlace deste protocolo;
- **Classe do sinal:** este campo não será utilizado nesta primeira versão. O uso do mesmo seria para que o hardware possa enviar ao ProfiDoctor uma classificação do tipo de erro do sinal. Para obter tal classificação, o hardware iria utilizar um método de

análise de sinais diferente da análise realizada pelo ProfiDoctor (RNA). O melhor método de análise ainda está em avaliação. Com esta informação “extra”, uma comparação entre os resultados da análise feita pela RNA e por esta análise pode ser feita, aumentando assim, a chance de acerto no momento da classificação do tipo de erro do sinal amostrado.

- **Número de bits amostrados:** O usuário poderá configurar no hardware a quantidade de bits amostrados por telegrama Profibus que serão enviados ao ProfiDoctor. O valor poderá variar entre 0 e 5. Este intervalo foi definido, pois o sinal Profibus de um escravo com problemas possuirá boa parte dos bits enviados com o formato característico de um determinado problema. Sendo assim, por motivos de desempenho, apenas 5 bits (no máximo) são analisados para cada telegrama Profibus trafegado na rede. Caso o valor seja 0, significa que o software ProfiDoctor não está utilizando as amostras em nenhum momento.
- **Bits amostrados:** Amostras de cada bit. Para cada bit, serão coletadas 20 amostras. Cada amostra possui 4 bytes (1 float). O número máximo de amostras é 5 e mínimo é zero. Assim, o tamanho mínimo deste campo será zero e o máximo é 400 bytes (5 x 20 x 4).

Com o padrão de comunicação entre cada componente do sistema definido, foca-se a explicação no software de análise que é o principal componente do sistema.

O software em questão foi desenvolvido com a linguagem de programação Java e utilizou-se como IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) de programação o *Netbeans 7.0*. Além disso, algumas bibliotecas foram utilizadas e detalhes sobre as mesmas podem ser encontrados no APÊNDICE A.

Ao abrir o software o mesmo oferece uma interface gráfica do tipo MDI⁵ (*multiple document interface*). A Figura 4.3 mostra uma imagem da janela principal do ProfiDoctor e, internamente a esta, três outras janelas.

⁵ MDI é uma maneira de organização de aplicações gráficas em janelas, onde todas essas janelas estarão contidas em outra janela principal da aplicação

Figura 4.3 – Janela principal do ProfiDoctor (MDI) com três janelas internas abertas

A partir do momento em que o sistema é aberto, o usuário tem três opções:

- Carregar informações armazenadas em arquivos;
- Iniciar uma conexão com o hardware que por sua vez estará coletando dados da rede Profibus;
- Iniciar uma conexão com o simulador da rede Profibus.

Considerando estas três opções, nota-se que o desenvolvimento e a manutenção do sistema poderiam atingir um grau de complexidade alto. A fim de diminuir a complexidade do software, foi criado um mecanismo onde a entrada dos dados no sistema acontece de uma única maneira padrão.

Para que este mecanismo funcione, o software possui um *buffer* de armazenamento onde cada telegrama ProfiDoctor obtido de uma rede real ou do simulador é armazenado. É importante notar que devido à velocidade da rede Profibus e o tempo de processamento desses telegramas pelo software, este *buffer* é preenchido para posterior análise. No momento em que o buffer fica cheio, as novas mensagens vindas da rede são descartadas. Neste período, as análises necessárias são realizadas e ao final deste processo, o *buffer* é esvaziado e uma nova coleta de mensagens é feita.

Continuando a descrição do sistema, tem-se que, quando o usuário escolhe a opção para salvar os dados que está trabalhando, o software percorre este *buffer* e armazena cada um

dos telegramas em um arquivo no formato binário. Este formato do arquivo é o mesmo formato dos dados oriundos da conexão do software com o hardware ou com o simulador. Assim, ao carregar o arquivo, o sistema simplesmente transforma os dados binários em telegramas ProfiDoctor e os armazena os mesmos no *buffer*. Trabalhando dessa maneira, garante-se um padrão no funcionamento do sistema facilitando o desenvolvimento e a manutenção do mesmo. A Figura 4.4 mostra este mecanismo.

Figura 4.4 – Visão geral dos componentes do software

Ainda analisando a Figura 4.4, nota-se que quando um telegrama entra no sistema, o mesmo é encapsulado em um objeto Java que foi denominado ProfiDoctorPackage. Este encapsulamento facilita a interpretação dos dados simplificando para as outras classes do software, a tarefa de obter as informações relevantes do telegrama sem a necessidade de conhecer a posição de cada byte do mesmo.

A partir deste encapsulamento, estes telegramas são armazenados no *buffer* já mencionado anteriormente e, além disso, eventos são enviados para a atualização das interfaces gráficas e dos módulos que analisam os dados utilizando sistemas inteligentes.

Como exemplo das interfaces gráficas, tem-se a interface gráfica para visão geral de tudo que acontece na comunicação Profibus (telegramas, sinais, *Live List*, etc.), a interface que oferece um osciloscópio que permite a análise dos sinais da camada física, a interface com um gráfico de barras que apresenta o nível médio de tensão transmitido por cada dispositivo da rede Profibus e por fim, uma interface que mostra a *Live List* de uma determinada rede. Detalhes dessas interfaces e da funcionalidade das mesmas podem ser encontrados no APÊNDICE A.

Ainda com base na Figura 4.4, outra parte do sistema que recebe eventos de atualização são os módulos de diagnóstico inteligente. Estes são responsáveis por receber os telegramas de dados e as amostras de sinais Profibus e, a partir disso, utilizar sistemas inteligentes para diagnosticar problemas na rede. Desta maneira, cada módulo fornece como saída um diagnóstico parcial e, com base nessas informações, um último sistema especialista gera a saída final. Esta saída é o diagnóstico final da análise.

Para que este tipo de análise obtenha os melhores resultados possíveis, uma série de parâmetros é analisada ao mesmo tempo. Uma dessas análises foi feita na camada física da rede em que amostras de sinais transmitidos são classificadas. Para tal classificação alguns mecanismos foram utilizados, sendo que o principal deles foram as RNAs. Ainda através da análise dos sinais da camada física Profibus, outra maneira de diagnosticar um problema é através da análise do nível de tensão do sinal de um ou mais dispositivos conectados à rede. Para esta análise, um sistema especialista baseado em regras foi criado.

Como já citado, os sinais analisados nestes módulos são amostrados a partir de uma rede real Profibus ou a partir do simulador de rede. O sistema recebe como entrada, um bit Profibus discretizado em 20 amostras. É muito importante entender que estes vinte valores representam apenas um bit Profibus. Cada um desses valores amostrados são utilizados como uma entrada para o sistema, sendo assim, o conjunto de sinais de entrada é composto por 20 amostras que representam um bit Profibus. A Figura 4.5 apresenta detalhes de como um sinal é amostrado.

Figura 4.5 – Mecanismos de amostragem dos sinais Profibus

Saindo do âmbito dos sinais e analisando-se os dados da camada de usuário do protocolo, pode-se diagnosticar problemas através da análise do telegrama de diagnóstico (nativo do próprio protocolo) enviado por um escravo ao mestre. Para esta última análise, um sistema que analisa cada bit do campo de dados dos telegramas de diagnóstico foi criado. Ainda através da análise dos telegramas Profibus, outra maneira de diagnóstico foi implementada. Neste caso, uma análise baseada no histórico dos telegramas recebidos é feita e através da implementação de outro sistema especialista, alguns problemas podem ser identificados. Por fim, uma última análise feita nos telegramas Profibus está relacionada com os valores permitidos em cada um dos campos de um determinado telegrama Profibus.

Os módulos citados no parágrafo anterior realizam as análises baseando-se em informações de sinais da camada física e de telegramas da camada de enlace. Sabe-se que o protocolo Profibus exige a configuração de alguns parâmetros de tempo. Neste contexto, uma análise baseada em sistemas *Fuzzy* é feita a fim de fornecer ao operador do sistema um valor adequado para estes parâmetros.

Além de todos os módulos apresentados acima, existe ainda um módulo que realiza uma análise final a partir de informações das saídas dos módulos citados anteriormente. Este último possui um mecanismo responsável por unir todos os diagnósticos obtidos em cada um dos módulos e a partir dessas informações, fornecer uma saída sumarizada e completa ao usuário. Além disso, outras conclusões podem ser obtidas a partir da análise conjunta das saídas dos outros módulos. Por exemplo, a classificação de um sinal oriunda de uma RNA

pode ser de que o sinal está comprometido devido ao cabeamento excessivamente longo. Esta mesma conclusão poderá ser obtida a partir da análise do nível de tensão do sinal dos escravos da rede. Considerando tal fato, pode-se unir estas informações para fornecer uma classificação com um grau de certeza maior. A Figura 4.6 mostra um diagrama que apresenta os módulos citados neste parágrafo e suas relações.

Figura 4.6 – Diagrama dos módulos que utilizam sistemas inteligentes para diagnóstico

Detalhes sobre cada um dos módulos apresentados na Figura 4.6 são apresentados nos subitens 4.1, 4.2, 4.3, 4.4, 4.5, 4.7 e 4.6.

4.1 Análise dos formatos dos sinais Profibus

Neste módulo, uma análise relacionada ao sinal da camada física da rede Profibus é realizada. Sabe-se que cada problema oriundo desta camada gera um sinal com uma forma de onda típica. Tal característica permite que as RNAs sejam capazes de indicar o motivo de um determinado problema a partir de amostras destes sinais.

Anteriormente ao desenvolvimento desta tese, Souza (2012) analisou formas de ondas oriundas de cinco tipos de topologia de redes Profibus, sendo estes: redes funcionando normalmente com cabeamento de 1 e 200 metros; rede com cabeamento rompido; rede com curto circuito entre as linhas A e B; rede com inversão das linhas A e B; rede com cabeamento longo. A análise destes sinais a partir de RNAs obteve 100% de acerto. Considerando tal fato, uma abordagem parecida com a utilizada pela autora foi implementada. Tal abordagem será detalhada no decorrer deste capítulo.

Constatada a viabilidade da utilização de RNAs para a classificação de sinais da camada física das redes Profibus, tem-se que nesta tese, outros tipos de sinais foram analisados e classificados. Os sinais analisados representam formas de ondas oriundas de redes Profibus com as seguintes configurações físicas:

- utilizando cabeamento de 1m e 200m sem nenhum problema de instalação;
- utilizando terminador sem alimentação de energia com cabeamento de 200m;
- falta de terminador, cabeamento quebrado ou conector desconectado com medição realizada a 200m da origem do problema;
- cabeamento de 1000m (maior do que o permitido pela norma);
- curto circuito entre as linhas de dados A e B;
- excesso de terminadores de linha;
- sinal com EMI (interferência eletromagnética);

Em uma primeira análise, amostras de todos estes tipos de sinais foram utilizadas no treinamento e teste das RNAs e um bom resultado não foi obtido. Desta maneira, uma série de outros treinamentos seguidos de testes foram realizados a fim de detectar qual das formas de ondas estaria impedindo um bom resultado da RNA, sendo que, para cada uma destas novas análises, um dos tipos de sinais era retirado da base de treinamento/teste. Em um determinado momento, os sinais com EMI foram retirados da base e os resultados das RNAs

atingiram 100% de acerto. Tal fato acontece, pois sinais com EMI são fundamentalmente aleatórios por natureza e por isso não apresentam um padrão típico.

Sendo assim, constatou-se que para classificar este tipo de problema uma pré-análise deveria ser executada para evitar que este tipo de sinal seja enviado para as RNAs classificarem. Tal pré-análise foi realizada a partir da variância das amostras dos sinais (item 4.1.1). Caso o sinal não possua indícios de EMI, as amostras são inseridas como entradas em algumas RNAs que fornecem, como saída, a classificação do sinal (item 4.1.2). Além deste pré-processamento descrito, todas as amostras são normalizadas e são considerados apenas os sinais Profibus positivos nestas análises. A Figura 4.7 apresenta um esquema geral do funcionamento deste módulo.

Figura 4.7 – Esquema para classificação dos sinais

4.1.1 Pré-análise

Sabe-se que o Profibus se comunica através de sinais positivos que representam um valor lógico 1 e sinais negativos que representam o valor lógico 0. Assim sendo, quando um

problema acontece, a deformação do sinal vai acontecer tanto em um sinal positivo quanto em um negativo. Neste caso, nota-se que não há a necessidade de inserir como entrada da RNA ambos os tipos de sinais. Desta maneira, o primeiro passo do processo de classificação do sinal é diminuir a quantidade de amostras diferentes para cada tipo de sinal a ser classificado, eliminando-se os sinais negativos e mantendo-se apenas os positivos. Analisando-se ambos os casos, obteve-se 100% de acerto para treinamento com sinais positivos e quando misturaram-se os dois tipos de bits ao treinamento (positivos e negativos), as RNAs apresentaram tipicamente 10% de erro.

Normalmente, as instalações físicas das redes industriais dividem o espaço com outros equipamentos eletrônicos ou elétricos e cada dispositivo gera diferentes formas de perturbações eletromagnéticas que interferem na comunicação Profibus, mesmo esta última possuindo uma série de mecanismos para impedir este tipo de ruído (blindagem, sinal diferencial, etc.). Para garantir um bom funcionamento da rede, os dispositivos Profibus devem ser certificados em relação à compatibilidade eletromagnética (EMC), ou seja, deve existir alguma garantia de que mesmo nesses ambientes, os dispositivos deverão manter-se em funcionamento dentro de padrões aceitáveis de operação. A questão aqui é que nem sempre estes dispositivos são certificados e nessas condições, poderão prejudicar o funcionamento da rede. Além disso, inadequações das instalações como má segregação do cabeamento, aterramento deficiente, má blindagem, entre outros, fazem com o sinal Profibus sofra deformações originadas pela energia eletromagnética existente nesses ambientes. A EMI é a energia que causa resposta indesejável a um determinado equipamento e que pode ser gerada por centelhamento nas escovas de motores, por chaveamento de circuitos de potência, por acionamentos de cargas indutivas e resistivas, acionamentos de relés, chaves, disjuntores, lâmpadas fluorescentes, aquecedores, ignições automotivas, descargas atmosféricas e mesmo as descargas eletrostáticas entre pessoas e equipamentos, aparelhos de micro-ondas, equipamentos de comunicação móvel, etc. Tudo isto pode provocar alterações causando sobretensão, subtensão, picos, transientes, etc. e que em uma rede de comunicação digital, pode causar impactos. Os mais diversos problemas podem ser gerados pela EMI. Por exemplo, podem-se ter falhas na comunicação entre dispositivos de uma rede de equipamentos e/ou computadores, alarmes gerados sem explicação, atuação em relés que não seguem uma lógica e sem haver comando para tal, etc. A topologia e a distribuição do cabeamento, os tipos de cabos, as técnicas de proteções são fatores que devem ser considerados para a minimização dos efeitos de EMI, afinal, em altas frequências, os cabos se

comportam como um sistema de transmissão com linhas cruzadas e confusas, refletindo energia e espalhando-a de um circuito a outro (CASSIOLATO, 2010).

Sabe-se que sinais com EMI possuem características aleatórias e quando amostras desses sinais são inseridas na lista de sinais a serem classificados pelas RNAs, uma taxa de erros próxima de 30% acontece, afinal, sua característica aleatória não permite que a RNA encontre um padrão para este tipo de sinal. Sendo assim, um sinal com EMI deve ser detectado de maneira antecipada por outro tipo de análise, eliminando das RNAs responsáveis pela classificação dos sinais a tarefa de classificar este tipo de problema. Dessa maneira as amostras do sinal são enviadas para as RNAs somente quando uma EMI não é detectada pela pré-análise. A Figura 4.8 apresenta um sinal com EMI comparado com um sinal normal.

Figura 4.8 – Sinal com EMI à esquerda e sem EMI à direita

Analisando-se visualmente um sinal com EMI, nota-se que o nível de tensão do mesmo possui uma variação grande, ou seja, nota-se uma diferença grande de valor entre uma amostra e outra de um mesmo bit (um bit possui 20 amostras).

A fim de calcular a variabilidade entre o conjunto de amostras, a variância amostral do conjunto de amostras de um determinado bit do sinal recebido deve ser calculada (UPTON, G. et al., 1998). A variância amostral dos bits $x_1, x_2, \dots, x_{20}$ é definida pela equação 1.

$$s^2 = \frac{\sum_{i=1}^{20} (x_i - \bar{x})^2}{20 - 1} \quad (1)$$

Na equação 1, $\bar{x}$ representa a média aritmética entre todas as amostras do conjunto de dados. Com o cálculo da variância realizado, é necessário definir um valor para a mesma que

irá diferenciar um sinal Profibus com EMI de um sinal sem EMI. Tal definição foi realizada de maneira heurística e para isso, vários sinais representando diferentes cenários de uma rede Profibus foram amostrados e a variância de cada um foi calculada. Dentre os sinais analisados, existiam sinais com deformações causadas por erros de instalação, sinais sem problemas e sinais com EMI. A Tabela 4.1 mostra os valores das variâncias calculadas.

Tabela 4.1 – Variâncias das amostras de cada tipo de sinal

Tipo do sinal	Variância de cada bit	Gráfico de bits amostrados
Sinal originado por rede utilizando cabeamento de 1 metro corretamente configurada	0,0053
	0,0823	
	0,0050	
	0,0038	
	0,0265	
Sinal originado por rede utilizando cabeamento de 200 metros corretamente configurada	0,0354
	0,0380	
	0,0053	
	0,0462	
	0,0061	
Sinal <i>IDLE</i> originado por rede utilizando cabeamento de 1 metro corretamente configurada	0,0010
	0,0087	
	0,0023	
	0,0035	
	0,0014	
Curto Circuito em linhas de dados A e B	0,0076
	0,0057	
	0,0016	
	0,0036	
	0,0042	
Sinal originado por cabeamento longo (1000m)	0,3351
	0,5884	
	0,8474	
	0,6514	
	0,8357	

Tipo do sinal	Variância de cada bit	Gráfico de bits amostrados
Sinal originado por cabeamento quebrado ou falta de terminadores com cabeamento de 200m	0,0987
	0,2745	
	0,1124	
	0,2856	
	0,1070	
Sinal originado por uma rede com terminador sem alimentação de energia	0,0577
	0,0096	
	0,0042	
	0,0159	
	0,0042	
Sinal originado por uma rede com excesso de terminadores de linha (VALORES SINTÉTICOS)	0,0189
	0,0301	
	0,0107	
	0,0217	
	0,0135	
Sinal originado por uma rede sofrendo EMI (VALORES SINTÉTICOS)	1,8002
	2,6901	
	0,8662	
	1,2787	
	1,1104	

Os valores calculados para as redes com excesso de terminadores de linha e para redes com EMI são gerados de maneira sintética. Sendo assim, os valores obtidos para as variâncias apresentadas na Tabela 4.1 poderiam ser contestados. Para solucionar isso, uma análise com base em sinais coletados por Procentec (2012) também foi realizada:

- **Sinal com EMI:** a Figura 4.9 apresenta o sinal obtido a partir de Procentec (2012) à esquerda e, à direita, o sinal gerado sinteticamente. Para este sinal, o valor da variância encontrado foi de 2,29.

Figura 4.9 – Sinais com EMI

- **Sinal causado por excesso de terminadores:** a Figura 4.10 apresenta o sinal obtido a partir de PROCENEC (2012) à esquerda e, à direita, o sinal gerado sinteticamente. Para este sinal, o valor da variância encontrado foi de 0,019.

Figura 4.10 – Sinais causados por excesso de terminadores

Analisando o valor da variância de todos os tipos de sinais obtém-se as seguintes conclusões:

- sinais obtidos a partir do uso de cabeamento longo possuem a menor variância com um valor aproximado de 0,53 e a maior de 0,84.
- sinais com EMI possuem a menor variância de 0,85.
- os outros sinais possuem variância inferiores, sendo que o valor da maior delas é de aproximadamente 0,3.

A partir dessas observações, optou-se em dividir os tipos de sinais em dois grupos:

- O primeiro grupo é composto pelos sinais com variância maiores do que 0,4, valor intermediário entre a menor variância dos sinais com cabeamento longo e EMI e os outros sinais. Sendo assim, neste primeiro grupo ficariam os sinais que representam o problema de cabeamento longo e os sinais com EMI.
- O segundo grupo contempla todos os outros sinais, ou seja, os sinais com variância menor do que 0,4.

Ao final desta divisão, através da utilização de RNAs, realizaram-se testes nos dois grupos criados e analisando os resultados, notou-se que, apesar de bons, os resultados do segundo grupo poderiam ser melhorados ainda mais. Observando-se novamente estes dados, notou-se que existiam três classes de sinais que se diferenciavam bastante dos outros pelo fato dos mesmos serem amostrados no momento em que a rede não está enviando nenhuma informação, como por exemplo, o sinal de *IDLE*. Sendo assim, este segundo grupo foi novamente dividido, sendo que os sinais que representam redes com terminadores sem energia, sinais *IDLE* da rede funcionando corretamente e redes com curto circuito entre as linhas de dados A e B foram colocados em um grupo e os outros sinais em outro.

Desta maneira, os sinais foram separados em três grupos e para cada um desses grupos, sistemas utilizando RNAs foram criados. Os detalhes destas RNAs estão descritos no item 4.1.2.

4.1.2 RNAs

Após a pré-análise descrita no item anterior onde grupos de amostras de sinais são criados, tem-se que o sistema é dividido em três partes. A primeira delas é composta por uma única RNA responsável apenas por diferenciar o sinal entre EMI ou cabeamento longo. A segunda parte do sistema é composta por três RNAs, cada uma responsável por classificar um dos tipos de sinais apresentados nos marcadores a seguir:

- Terminador sem alimentação de energia;
- Sinal de *IDLE* com rede configurada corretamente com cabeamento de 1m e 200m;
- Curto circuito em linhas de dados A e B

Por fim, a outra parte do sistema é composta por quatro RNAs, cada uma responsável por classificar um dos tipos de sinais restantes, sendo estes:

- Sinal com rede configurada corretamente com cabeamento de 1m e 200m;
- Falta de terminador, cabeamento quebrado ou conector desconectado com cabeamento de 200m;
- Rede utilizando cabeamento longo de 1000m;
- Excesso de terminadores de linha;

É importante notar que os sinais com problemas de cabeamento longo podem ser inseridos tanto no primeiro quanto no terceiro grupo, pois o valor da variância do mesmo está próximo ao valor responsável por dividir os grupos. Sendo assim, estas duas partes do sistema são treinadas para serem capazes de classificar este tipo de sinal.

Cada uma das RNAs da segunda e da terceira parte do sistema possui como saída a informação de que o sinal possui ou não um determinado problema. A Figura 4.11 mostra um diagrama geral focando neste segmento do sistema. É importante notar que os sinais com problemas de cabeamento longo podem ser inseridos tanto no primeiro quanto no terceiro grupo, pois o valor da variância do mesmo está próximo ao valor responsável por dividir os grupos. Sendo assim, estas duas partes do sistema são treinadas para serem capazes de classificar este tipo de sinal.

Figura 4.11 – Diagrama geral do segmento do sistema de análise utilizando RNAs

Além desses tipos de sinais apresentados, planejou-se inicialmente detectar o problema causado por um curto circuito entre a linha de dados B e o *Shield* do cabo Profibus. Tal detecção foi eliminada do escopo do projeto, pois, segundo PROCENTEC (2012), a detecção deste problema só é possível através da análise das linhas de dados A e B de maneira separada, ou seja, não utilizando-se o sinal diferencial. Considerando que as amostras dos sinais que o software de análise recebe do hardware ou do simulador não possuem os sinais A e B isolados, tal análise tornou-se inviável para o projeto.

Antes de dar continuidade nas explicações do funcionamento deste módulo, é importante destacar que a configuração apresentada na Figura 4.11 foi obtida após a

realização de uma série de tentativas com outras combinações de RNAs. Os resultados dessas tentativas são apresentados no item 6.1.

Sabe-se que uma RNA do tipo PMC depende de um treinamento para que seus pesos sinápticos sejam definidos. Durante a etapa de treinamento a rede recebe um conjunto de dados. Esses dados são utilizados no treinamento da rede, que assim como o cérebro humano, aprende através de exemplos e associações. Para tal, bits representando cada tipo de problema na rede são discretizados e estes dados são armazenados em arquivos com uma série de linhas (cada linha é um bit Profibus) e colunas (cada coluna significa um valor discretizado do bit Profibus). Em suma, para treinar cada uma das redes, uma série de bits da rede Profibus são amostrados e armazenados em arquivos de texto para serem utilizados como entradas para o treinamento das RNAs. Além dos arquivos que armazenam os bits amostrados, têm-se os arquivos onde a classificação de cada amostra é feita, afinal, o treinamento da RNA é supervisionado.

Considerando que uma das redes é responsável apenas por diferenciar um sinal com EMI de um sinal deformado por efeitos de cabeamento longo, tem-se que o conjunto de dados amostrados e armazenados no arquivo de treinamento desta rede possuem amostras apenas destes dois tipos de sinais. Neste contexto, o arquivo utilizado para classificar estes sinais logicamente possuirá apenas dois tipos de informação, ou seja, uma para classificar sinal com EMI e outra para classificar cabeamento longo. A Figura 4.12 apresenta uma pequena parte do arquivo utilizado para o treinamento desta rede.

Arquivo com as amostras (cada linha representa 1 bit Profibus)	Classificação
3.3600, 4.0000, 3.9200, 3.9200, 3.5200, 3.5200, 3.6800, 3.7600, 3.8400	0,1
2.5600, 2.4000, 2.2400, 2.4800, 2.4000, 2.2400, 2.0800, 2.2400, 2.4000	0,1
3.6800, 3.6000, 3.6000, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800	0,1
3.6000, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800, 3.6800	0,1
3.6800, 3.6800, 3.7600, 3.7600, 1.2800, 2.2400, 2.4000, 2.6400, 2.2400	0,1
0.4000, 0.4800, 0.5600, 0.6400, 0.4000, 0.0000, 0.4800, 0.7200, 0.9600	1,0
0.4000, 0.2400, 0.3200, 0.2400, 0.1600, 0.3200, 0.6400, 1.1200, 1.2800	1,0
2.3200, 2.2400, 2.3200, 2.3200, 2.3200, 2.0800, 1.6800, 1.4400, 1.0400	1,0
0.1600, 0.0800, 0.1600, 0.1600, 0.2400, 0.1600, 0.1600, 0.6400, 0.9600	1,0
2.2400, 2.3200, 2.1600, 2.1600, 2.2400, 2.1600, 1.6800, 1.4400, 1.1200	1,0

Figura 4.12 – Arquivos de amostras e classificação para RNA (EMI e cabo longo)

A partir da Figura 4.12 é possível observar que as cinco primeiras amostras foram classificadas como EMI (valor 0,1) e as outras como cabeamento longo (valor 1,0).

Os valores 0,1 e 1,0 representam os valores de saída dos neurônios da RNA, ou seja, após o treinamento, se uma amostra de um sinal com EMI for classificada pela RNA, espera-

se que o neurônio número 1 da camada de saída assuma o valor 0 e o neurônio número 2 assumo valor 1.

Com o mecanismo de treinamento para a primeira RNA explicado, tem-se o treinamento para as três RNAs responsáveis por classificar os sinais do tipo *IDLE*. Estas RNAs têm no conjunto de treinamento amostras dos sinais com rede em curto circuito, rede com terminadores sem energia e amostras do sinal *IDLE* de uma rede funcionando corretamente. No arquivo de classificação, ao invés de existir uma classificação diferente para cada tipo de problema, as amostras foram classificadas apenas de duas maneiras, sendo uma delas para identificar se o sinal amostrado é de um determinado tipo ou para indicar que o mesmo não é do tipo esperado. Sendo assim os dados de treinamento para todas as RNAs são os mesmos, mas arquivo de classificação é diferente para cada RNA. A Figura 4.13 apresenta um exemplo de um arquivo com amostras de sinais e relaciona este arquivo com os arquivos de classificação de duas RNAs.

Figura 4.13 – Arquivos de amostras e de classificação para RNAs (sinais do tipo IDLE)

A partir da Figura 4.13 é possível notar que no exemplo do arquivo em que a RNA identifica se um sinal Profibus representa curto entre as linhas de dados A e B, todos os tipos de sinais do arquivo de amostras são classificados como “outro tipo de sinal” (valores 0,1) enquanto que apenas as três primeiras amostras são classificadas como “curto A e B” (valores 1,0), afinal, de todos os sinais amostrados no arquivo, apenas os três primeiros realmente representam o problema de curto entre as linhas de dados A e B. Já no segundo caso, onde a RNA é responsável por classificar a rede como sinal *IDLE OK*, os sinais classificados como

“IDLE OK” (valores 1,0) estão na sétima, oitava e nona linhas. As outras amostras que representam outros tipos de formatos de sinal são classificadas como “outro tipo de sinal” (valores 0,1). Vale lembrar que o arquivo das amostras é o mesmo para todas as RNAs.

Para estas RNAs, o treinamento foi feito com 60 bits Profibus (20 bits para cada tipo de sinal), ou seja, o arquivo de amostras foi composto por 60 linhas e 20 colunas, onde cada coluna representa uma amostra de um determinado bit Profibus.

Por fim, tem-se o treinamento do terceiro grupo de sinais. Estas RNAs têm no conjunto de treinamento amostras dos sinais de rede com excesso de terminadores, rede com cabeamento longo, rede com cabeamento rompido ou falta de terminadores e rede com sinal nos padrões corretos. Individualmente, as RNAs irão trabalhar da mesma maneira das três RNAs utilizadas para classificar sinais com níveis de tensão baixos. Sendo assim, a Figura 4.15 apresenta um exemplo de um arquivo com amostras de sinais e relaciona este arquivo com os arquivos de classificação de duas RNAs.

Figura 4.14 – Arquivos de amostras e de classificação para RNAs do terceiro grupo

Para estas 4 RNAs, o treinamento foi feito com 100 bits Profibus, sendo 40 bits amostrados de redes funcionando corretamente e com cabeamentos de 1 e 200 metros, 20 bits de redes com cabeamento longo, 20 bits de redes com cabeamento rompido e 20 bits de redes com excesso de terminadores.

Como já foi comentado anteriormente, mesmo a PMC tendo apresentado bons resultados a partir dos dados utilizados, sabe-se que uma rede Profibus pode apresentar muitas

variações no sinal para um mesmo tipo de problema. Como exemplo pode-se citar o problema de cabeamento longo em que o sinal gerado pode estar com diferentes níveis de atenuação dependendo do tamanho do cabeamento utilizado. Considerando este fato, a taxa de erros ao instalar este sistema em outras redes Profibus pode aumentar devido às variações já citadas.

Levando-se em conta os fatos descritos, um mecanismo de realimentação para a RNA foi criado. Assim, quando um sinal for classificado de maneira equivocada, um especialista poderá inserir esta nova informação na base de conhecimento da RNA e a partir daí, treinar novamente a mesma para as próximas análises.

A fim de uniformizar o funcionamento da ferramenta, o mesmo mecanismo que é utilizado para o treinamento da rede deve ser utilizado para retreina-la. Para isso, o ProfiDoctor fornece a ferramenta apresentada na Figura 4.15.

Figura 4.15 – Janela utilizada para treinamento e teste da RNA

Nesta janela, o usuário pode escolher o local onde os arquivos com as amostras e as classificações estão localizados, o número de neurônios da camada escondida e as redes que o mesmo deseja treinar. Como saída, o usuário pode ver o número de épocas utilizadas no treinamento da rede, a porcentagem de erro e a quantidade de amostras que foram classificadas corretamente ou erroneamente durante o teste.

Durante o treinamento/retreinamento da rede, os pesos sinápticos são definidos e armazenados em um arquivo para o uso do sistema. Assim, o sistema poderá ser executado novamente para os novos sinais oriundos da rede Profibus ou do simulador.

Continuando a análise dos sinais transmitidos na camada física Profibus, tem-se no item 4.2 detalhes sobre o módulo que realiza uma análise com base no nível de tensão de um determinado sinal.

4.2 Análise de níveis de tensão

Este módulo analisa o nível de tensão do sinal transmitido por cada dispositivo ativo na rede Profibus. A Normative Parts of Profibus FMS, DP and PA (1998) recomenda que o nível ideal de funcionamento seja entre 4 e 7 Volts. Este nível de tensão corresponde à diferença entre o nível de tensão positivo do sinal e o nível de tensão negativo. Por exemplo, se um dispositivo está transmitindo seus bits de valor lógico 1 com 3 Volts e os bits de valor lógico 0 com -3 Volts, o nível de tensão deste neste dispositivo é de 6 Volts. A Figura 4.16 apresenta o exemplo descrito graficamente.

Figura 4.16 – Nível de tensão pico a pico

O cálculo do nível de tensão positivo e negativo de um dispositivo é feito através da média aritmética das amostras do sinal de cada um dos bits. Com estas médias calculadas, obtêm-se o nível de tensão através da diferença entre as médias. Desta maneira, a cada telegrama recebido, a média entre cada uma das amostras de um bit positivo e de um bit negativo é calculada e em seguida, a diferença entre estes valores é obtida.

A análise do nível de tensão dos dispositivos da rede permite que especialistas alcancem algumas conclusões. Neste contexto, a fim de inserir o conhecimento destes especialistas em um sistema computacional, um sistema especialista baseado na utilização de uma árvore de decisão foi elaborado. A Figura 4.17 apresenta a árvore de decisão.

Figura 4.17 – Árvore de decisão baseada em níveis de tensão de um bit Profibus

Segundo a Figura 4.17, seis diagnósticos diferentes podem ser obtidos:

- **Rede funcionando normalmente:** este caso acontece caso o nível de tensão de todos os dispositivos da rede sejam menores do que 7 Volts e maiores do que 4 Volts, ou seja, os níveis de tensão encontram-se entre 4 e 7 Volts. A escolha pelo uso dos níveis de tensão 4 e 7 Volts baseia-se nas recomendações encontradas em Normative Parts of Profibus FMS, DP and PA (1998).
- **Problemas com a fonte ou com o *driver* do dispositivo:** este caso acontece em duas situações. Uma delas é quando o nível de tensão está abaixo dos 4 Volts e a outra é quando o nível de tensão está acima dos 7 Volts recomendado. A suspeita de que o problema está diretamente relacionado com o dispositivo e não com a rede baseia-se no fato de que apenas 1 dispositivo está com o nível de tensão fora do intervalo recomendado. Caso mais de 1 dispositivo estiver apresentando este tipo de problema, outros diagnósticos serão obtidos.
- **Curto circuito entre as linhas de dados:** quando as linhas de dados A e B entram em curto o nível de tensão diferencial entre estas linhas observado na rede é próximo zero Volts.
- **Cabeamento longo ou problemas de aterramento:** este caso acontece quando apenas alguns dos dispositivos de uma rede estão transmitindo sinais com o nível de tensão abaixo dos 4 Volts recomendados. Tal problema acontece quando o sinal perde energia durante sua transmissão. A perda de energia pode ser causada pelo mau aterramento da rede ou pelo excesso de tamanho de um determinado cabo. Como já apresentado neste documento, existem alguns limites de comprimento de cabo relacionados à velocidade de transmissão da rede. Por exemplo, para o caso da transmissão com velocidade de 1,5Mbps, velocidade utilizada na obtenção dos resultados desta tese e na maioria das plantas industriais, o limite de comprimento do cabo é de 200 metros.
- **Cabeamento quebrado ou outros problemas de cabeamento:** a diferença deste caso para o caso relacionado a cabeamento longo está no fato de que este cenário apresenta um ou mais dispositivos com nível de tensão zero. O mais comum nesses casos é o problema estar relacionado com cabeamento quebrado. Vale lembrar que caso apenas um dispositivo estiver com o sinal

nulo *off-line*, o sistema diagnosticará o problema como sendo problemas com a fonte ou com o *driver* do dispositivo. É importante notar que os dispositivos que estiverem conectados após a ruptura do cabeamento perdem a comunicação. Já os dispositivos conectados antes da ruptura continuam ligados ao mestre da rede e consequentemente permanecem *on-line*.

É importante notar que a árvore de decisão baseia-se no raciocínio utilizado por um especialista no momento de diagnosticar algum problema no sistema. Em alguns casos a conclusão do sistema poderá não estar correta devido à impossibilidade do sistema conhecer todos os parâmetros necessários para realizar um diagnóstico mais preciso. Para os casos onde o usuário recebe um diagnóstico que gere algum tipo de dúvida, o ProfiDoctor oferece um gráfico de colunas (Figura 4.18) onde podem ser analisados todos os níveis de tensão dos dispositivos da rede.

Figura 4.18 – Interface gráfica de análise de níveis de tensão

A Figura 4.18 apresenta os níveis de tensão para os dispositivos de endereço 1, 4, 5, 6, 7, 8. Para este cenário, o sistema irá detectar algum problema apenas no dispositivo com endereço 5, pois o nível de tensão do mesmo está em 3 Volts. Analisando o cenário e a Figura

4.18, tem-se que o problema será diagnosticado como defeito na fonte de energia ou no *driver* do dispositivo.

4.3 Análise baseada em histórico de mensagens

Outra análise possível em uma rede Profibus é feita analisando-se o histórico de mensagens trafegadas na rede. Este tipo de análise é muito importante no diagnóstico de redes Profibus. Segundo Felser (2006), a fim de testar a qualidade dos telegramas transmitidos em uma rede, informações como a quantidade de telegramas que foram detectados com problemas de paridade ou *checksums* são importantes, mas não suficientes. Analisando apenas isso, o gerenciador da rede pode se confundir, afinal dependendo do ponto onde a obtenção dos telegramas está sendo feita, os problemas poderão acontecer ou não. Sendo assim, além dessas informações, é necessário analisar outros dados estatísticos, como por exemplo, a quantidade de telegramas de sincronismo que estão sendo transmitidos na rede.

Para isso, o software ProfiDoctor armazena em memória uma determinada quantidade de mensagens e após isso, a análise é realizada. A ideia é que sejam armazenadas neste histórico todas as mensagens trafegadas durante um determinado tempo de operação da rede. Antes de apresentar os diagnósticos que são obtidos pelas análises realizadas por este módulo, é importante entender o mecanismo de sincronização de um mestre com um determinado escravo, pois todas as análises descritas são feitas com base em tal procedimento.

Para uma rede Profibus funcionar, é necessário que uma configuração *off-line* seja feita e descarregada no mestre responsável pelo controle da rede. Nesta configuração, está a lista de escravos que o mestre deve controlar. Quando a rede está em funcionamento (*on-line*), o mestre analisa se os escravos de sua lista estão em troca de dados ou não. Caso algum escravo não esteja em troca de dados, o mestre envia uma mensagem de *getDiagnostic* para o escravo. Tal mensagem é uma espécie de pergunta ao escravo para saber se o mesmo está disponível. Caso o escravo não estiver disponível, nenhuma resposta será enviada e nos próximos ciclos de comunicação, o mestre continuará enviando mensagens de *getDiagnostic* até que o escravo em algum momento responda. Se o escravo estiver disponível, o mesmo responde ao mestre com um telegrama de *getDiagnosticResponse* informando que precisa ser parametrizado (bit *parameter request* do telegrama de diagnóstico ativado). No próximo ciclo de comunicação, o mestre envia um comando de *setParameters* com a parametrização

daquele escravo. O escravo responde a este comando com uma mensagem de *acknowledgement* apenas para informar o mestre que a mensagem foi recebida. Após isso, em um novo ciclo, o mestre envia a mensagem de *checkConfiguration*. Esta mensagem possui as informações de configuração do escravo. O escravo então responde novamente apenas com um *acknowledgement*. Feito isso, o mestre envia novamente a mensagem de *getDiagnostic* para confirmar se o processo de sincronização ocorreu de maneira correta. O escravo então responde através da mensagem *getDiagnosticResponse*. Nesta mensagem, podem ser encontrados os erros de *Configuration Fault* ou *Parameter Fault* informando problemas de configuração ou de parametrização. Outro problema que pode ser informado é o *Station not ready* que informa ao mestre que o processo de configuração no escravo ainda não foi finalizado. Neste segundo caso, o mestre espera um novo ciclo e envia novamente a mensagem de *getDiagnostic* até que o escravo não apresente problemas. Neste momento, a troca de dados (*data exchange*) se inicia. O procedimento explanado neste parágrafo é chamado de sincronização (*SYNC*) e pode ser visto de maneira resumida na Figura 4.19.

Figura 4.19 – Processo de sincronização (SYNC) do mestre com um escravo.

Com tal processo entendido, tem-se uma explicação detalhada de cada diagnóstico obtido a partir das mensagens trocadas entre um mestre e um escravo Profibus DP. Antes das explicações detalhadas, a Figura 4.20 apresenta a árvore de decisão com todas as regras necessárias para a análise em questão.

Figura 4.20 – Árvore de decisão para análise baseada em histórico de mensagens Profibus

- Problema de parametrização ou de endereçamento:** normalmente este tipo de problema acontece quando o usuário que configura a rede utiliza um arquivo GSD incorreto ou atribui para um determinado dispositivo um endereço já utilizado por outro dispositivo da rede. É possível detectar tal problema através da análise do campo *IdentNumber*⁶ presente nas mensagens de diagnóstico (*getDiagnosticResponse*) respondidas por um escravo e no telegrama de parametrização (*setParameters*) enviado por um mestre. Caso o valor do *IdentNumber* em ambos os telegramas não seja o mesmo, significa que o sistema está com problemas de parametrização ou de endereçamento. Para implementar esta análise, procura-se no histórico de mensagens por telegramas do tipo *setParameters* e *getDiagnosticResponse*. Estas mensagens

⁶ *IdentNumber* é o número de registro do equipamento na organização Profibus. Todo equipamento criado com o mesmo Profile possui um único *IdentNumber*.

são então analisadas e agrupadas por endereço dos escravos. Este endereço é obtido através do campo *Destination Address* do telegrama *setParameters* e do campo *Source Address* do telegrama *getDiagnosticResponse*. Com as mensagens agrupadas, basta verificar se o valor do *IdentNumber* em cada um dos telegramas agrupados é igual ou diferente. Se algum desses forem diferentes, isso caracteriza um problema de parametrização ou de endereçamento, afinal, o *IdentNumber* utilizado na configuração *off-line* vai ser diferente do dispositivo conectado na rede Profibus.

- **Escravo não encontrado (*missing device*):** este problema acontece quando algum escravo é configurado no mestre, mas não é ativado na rede Profibus real. Como já explicado neste capítulo, o mestre envia telegramas do tipo *getDiagnosticRequest* para os escravos presentes em sua configuração a fim de sincronizar-se com os mesmos e iniciar a troca de dados. Caso o escravo não responde a este tipo de mensagem, significa que o escravo não está ativo na linha (escravo está *off-line*). Para implementar esta análise, busca-se no histórico mensagens do tipo *getDiagnosticRequest* e *getDiagnosticResponse* relacionadas a um determinado escravo. Caso existam no histórico apenas mensagens de *request* sem as respostas do escravo, conclui-se que o dispositivo não está ativo na linha (*Missing Device*). Caso as mensagens de *reponse* sejam encontradas significa que o processo de sincronização está ocorrendo ou já ocorreu sem nenhum problema desta natureza. É importante notar que no caso do escravo parar de se comunicar durante a operação da rede, não será este o diagnóstico dado, pois para o caso da perda da comunicação, mensagens de *retry* normalmente são enviadas antes das mensagens de *getDiagnostic* e o diagnóstico será de queda na conexão por interferência ou outros motivos.
- **Problemas de endereçamento (*wrong address*):** este problema acontece se um usuário configura um escravo no mestre com um determinado endereço e na rede real ativa este escravo com um endereço diferente. Para obter este diagnóstico é necessário que o cenário descrito no parágrafo anterior (escravo não encontrado) aconteça e, além disso, mensagens de resposta às mensagens de *requestFDLStatus* sejam respondidas por um escravo não existente na configuração do mestre. A mensagem de *requestFDLStatus* é enviada pelo

mestre a cada ciclo de comunicação. A ideia é que no decorrer de seus ciclos, o mestre envie este tipo de mensagens para todos os endereços Profibus a fim de detectar todos os dispositivos ativos na linha (*live list*). Quando um escravo responde esta mensagem, significa que o escravo está ativo na linha de comunicação Profibus. Se o escravo não responder, significa que o mesmo não está ativo. Neste contexto, unindo-se o fato de um escravo presente na configuração de um mestre não estar respondendo ao *getDiagnostic* e o fato de que outro escravo que não está configurado no mestre estar respondendo às mensagens de *requestFDLStatus*, suspeita-se que houve algum problema de endereçamento. Por exemplo, um escravo de endereço 10 está configurado no mestre, mas não está ativo na linha. Neste caso, o mestre envia mensagens de *getDiagnostic* e nenhuma resposta é recebida. Ao mesmo tempo, o mestre envia um *requestFDLStatus* a um escravo de endereço 15 que não está presente em sua configuração e este escravo responde ao *requestFDLStatus*. Neste caso, suspeita-se que o operador da rede inseriu na linha um escravo com endereço 15 que deveria estar com endereço 10.

- **Interferência momentânea causada por ruído ou mau contato na conexão:** existem muitas situações durante o funcionamento de uma rede Profibus em que um escravo perde a conexão com a rede e acaba retomando logo em seguida. As causas mais comuns para tal comportamento é algum tipo de ruído esporádico que interfere no sinal da comunicação. Além disso, algum mau contato do conector da rede com um escravo pode ocasionar este tipo de comportamento. Quando isso acontece, o sistema fornecerá o diagnóstico de que o escravo saiu da rede por causa de algum tipo de interferência, mas retornou. Para o sistema detectar isso, é necessária primeiramente uma análise em busca de mensagens de *retry* para algum dispositivo da rede. Quanto tal mensagem é encontrada, procura-se então pela resposta do escravo. Se a resposta acontecer, significa que o escravo teve um problema momentâneo e já está operando normalmente e o diagnóstico descrito neste item será apresentado ao usuário. Outra situação que este diagnóstico será apresentado é se a mensagem de *retry* for detectada e, após isso, um processo de *SYNC* for iniciado e finalizado com sucesso.

- **Escravo perdeu a conexão e não retornou para a rede:** continuando o cenário do item anterior em que um *retry* é detectado, outra situação pode ser observada. Tal situação acontece quando o escravo não retorna para a rede. Neste caso, além do *retry*, mensagens do tipo *getDiagnostic* para o mesmo escravo são detectadas na comunicação sem que a resposta para a mesma seja enviada. Esta mensagem é enviada a um escravo configurado no mestre a fim de iniciar o processo de sincronização. Isso pode acontecer por uma série de motivos, como por exemplo, o fato do usuário remover o dispositivo fisicamente ou o dispositivo ter sofrido algum dano em seu hardware.
- **Escravo com tempo de *watchdog* baixo:** caso um escravo pare de receber telegramas de um mestre por um determinado tempo (tempo de *watchdog*), o mesmo entra em modo seguro, pois conclui que algum problema aconteceu na comunicação da rede. O valor deste tempo é configurado pelo gerenciador da rede e caso este último utilize um tempo muito baixo em sua configuração, o escravo vai sempre entender que a rede está com problemas e vai entrar em modo seguro. Considerando que o mestre continua operando normalmente, uma mensagem será enviada ao escravo no próximo ciclo. Neste momento, o escravo sinaliza ao mestre que está em modo seguro e deve ser parametrizado e configurado novamente. Sendo assim, em uma situação como estas, uma série de tentativas de sincronização acontece e terminam com sucesso, mas logo que o escravo entra em troca de dados, uma nova tentativa de sincronização é iniciada. Ao analisar o histórico de mensagens e observar tal cenário, o diagnóstico de que o tempo de *watchdog* foi configurado com um valor baixo é fornecido ao usuário. Vale destacar que um cenário parecido com este acontece quando ocorrem falhas na configuração ou parametrização de um escravo. A diferença entre estes casos está no fato de que para o caso de erro de configuração ou parametrização, o escravo não entra em troca de dados.

Para a detecção dos problemas descritos até o momento neste capítulo, as análises são feitas em intervalos de tempo relativamente pequenos (valor configurável). Para cada análise realizada, o histórico utilizado é descartado para o início de um novo ciclo de coleta de dados. Além dessas análises, um diagnóstico baseado em um tempo maior de funcionamento da rede

é feito. Para tal, informações estatísticas relacionadas a cada ciclo de coleta de dados são armazenadas. Este tipo de informação capacita o sistema a diagnosticar problemas que acontecem de maneira esporádica na rede como o problema descrito a seguir:

- **Dispositivo entra e sai da linha esporadicamente:** para referenciar este tipo de comportamento de um dispositivo, existe o termo *fall-out*. Por exemplo, se em um período de 6 horas o escravo entrar e sair da linha por 3 vezes em intervalos aleatórios, pode-se concluir que este escravo está recebendo algum tipo de interferência ou outro tipo de ação prejudicial de maneira esporádica. Este tipo de comportamento pode ser causado, por exemplo, por um motor que é ligado apenas em determinadas situações. A implementação desta análise é feita com base nas análises realizadas no parágrafo anterior, ou seja, a cada vez que um escravo sai da linha e após um período retorna, registra-se em um histórico tal fato. A cada 60 minutos (valor configurável), a análise deste histórico é feita e caso haja mais de 3 quedas de comunicação com um determinado escravo, conclui-se que algum dispositivo (motor, inversor, etc.) esteja interferindo esporadicamente neste escravo.

4.4 Análise das mensagens de diagnóstico Profibus

O protocolo Profibus possui telegramas de diagnósticos nativos do próprio protocolo que podem ser utilizados para que um escravo informe ao mestre sobre problemas ou outras informações detectadas. Para que o mestre solicite o diagnóstico do escravo, este último deve enviar uma indicação através de um bit de sinalização de prioridade no campo FC (*Frame Control*) durante a troca de dados. Logo após o mestre receber esta indicação, o mesmo envia ao escravo um comando *getDiagnostics* solicitando este tipo de informação. Estas informações são enviadas através de quatro blocos de dados, como mostra a Figura 4.21.

Diagnóstico Padrão (sempre existe)	Diagnóstico Estendido (Dispositivo)	Diagnóstico Estendido (Módulo)	Diagnóstico Estendido (Canal)
---------------------------------------	---	--------------------------------------	-------------------------------------

6 bytes

Figura 4.21 – Formato dos dados de diagnóstico

O campo de 6 bytes que representa o diagnóstico padrão, é um campo que estará sempre presente em telegramas de diagnóstico. Os outros campos, que representam o diagnóstico estendido, são opcionais. No caso da ferramenta desenvolvida nesta tese, apenas informações de diagnóstico padrão são analisadas, pois as outras informações dependem do arquivo GSD de cada escravo e o escopo do trabalho em questão não aborda a leitura e interpretação deste tipo de arquivo.

Para tal análise, o telegrama recebido é enviado ao módulo de análise do campo de diagnóstico que analisa quais bits estão ativados. Cada um dos bits é verificado e o resultado final é obtido. Este resultado é então classificado como crítico, alarmante ou informação. Caso mais de um bit esteja ativado, a classificação final do diagnóstico prioriza o status de crítico e em seguida o status de alarmante.

Para um entendimento detalhado de cada bit de diagnóstico, tem-se a Figura 4.22, a Figura 4.23 e a Figura 4.24.

Figura 4.22 – Descrição do byte 1 do diagnóstico padrão

Figura 4.23 – Descrição do byte 2 do diagnóstico padrão

Figura 4.24 – Descrição dos bytes 3 a 6 do diagnóstico padrão

Nos marcadores a seguir, tem-se uma explicação para cada bit contido nos três primeiros bytes de diagnóstico:

- **Escravo não existente** (*Station Non Existent*): este caso de diagnóstico nunca será detectado no ProfiDoctor uma vez que o equipamento que não está *on-line* nunca fará um pedido de diagnóstico ao mestre e conseqüentemente nunca enviará uma mensagem de diagnóstico na linha. Como o ProfiDoctor “escuta” a linha de dados Profibus de forma passiva, o mesmo nunca receberá uma mensagem de diagnóstico com este bit ativado. Tal informação é utilizada somente no contexto do mestre da linha.
- **Estação não pronta para troca de dados** (*Station not Ready*) – (alarmante): Este caso acontece na inicialização do escravo, quando o mesmo não terminou o processo de configuração ou de parametrização requisitado pelo mestre da rede ou quando o mesmo está apresentando alguma falha. Para entender mais detalhes destes problemas, existe o diagnóstico estendido.
- **Falha de configuração** (*Configuration Fault*) – (crítico): este bit é ativado pelo escravo se uma configuração enviada pelo mestre não for compatível com a suportada pelo escravo. Este problema acontece também quando um módulo ou um canal de um escravo apresenta problemas durante sua operação na rede.

- **Escravo tem diagnóstico estendido** (*Extended Diagnostics Existent*) – (informação): este bit é ativado pelo escravo caso o equipamento esteja fornecendo um diagnóstico específico dele utilizando para isso os campos destinados ao diagnóstico estendido (bytes 7..240).
- **Escravo não suporta função solicitada** (*Not Supported*) – (crítico): este bit é ativado pelo escravo logo que uma função requerida pelo mestre não é suportada. Este diagnóstico indica que algum problema está acontecendo com o mestre, pois o mesmo está enviando pedidos que não são suportados pelo escravo.
- **Resposta inválida do escravo** (*Invalid Slave Response*): este caso de diagnóstico nunca será detectado no ProfiDoctor uma vez que este bit é ativado apenas no contexto do Mestre e é utilizado para reportar uma resposta inválida de um escravo.
- **Falha de parametrização** (*Parametrization Fault*) – (crítico): este bit é ativado pelo escravo se o último telegrama de parametrização recebido estiver com problemas, como por exemplo, *Ident_Number* errado, parâmetros errados, etc.
- **Escravo controlado por outro mestre** (*Master Lock*) – (crítico): este bit informa que o escravo já foi parametrizado por um mestre. Caso exista mais de um mestre na linha e um segundo mestre pede um diagnóstico para este escravo, o mesmo irá perceber que o escravo em questão já está sob o controle de outro mestre. Este problema é crítico, pois indica que algum mestre presente na rede pode estar configurado de maneira equivocada.
- **Escravo esperando parametrização** (*Parametrization Request*) – (informação): este bit é ativado quando o escravo que o ativou está esperando por uma parametrização e uma configuração. O bit continuará ativo até que a parametrização seja concluída. Este tipo de ação acontece quando o escravo é ligado, mas ainda não está parametrizado por nenhum mestre ou caso o escravo sofra alguma interrupção.
- **Indicação de diagnóstico estático** (*Static Diagnostic*) – (alarmante): com este bit ativado, o escravo indica ao mestre para que o mesmo requisite continuamente o diagnóstico até que este bit seja desativado pelo próprio escravo. Durante o processo de sincronização (*SYNC*) o mestre envia o

comando de configuração e no ciclo seguinte, o escravo pode ainda estar executando internamente a configuração. Nesta situação, o bit *Static Diagnostic* é ativado pelo escravo, afinal, o mesmo não está pronto para a troca de dados (*data exchange*). Tal problema justifica a ativação de um alarme, pois o escravo pode estar apresentando problemas internos para realizar a parametrização.

- **Bit DP (*DP Bit*)** – (ignorar): este bit sempre tem o valor 1. O mesmo será ignorado nas análises de diagnóstico.
- **Monitoração de resposta ligada (*Watchdog ON*)** – (alarmante): este bit será ativado se o *Watchdog* estiver ativado. Caso o *Watchdog* estiver desligado, um alarme é enviado, afinal, um escravo sem a opção de segurança oferece riscos para uma planta industrial.
- **Escravo em modo *Freeze*⁷ (*Slave is on freeze mode*)** – (informação): este bit é ativado assim que um escravo recebe o comando de *Freeze control*.
- **Escravo em modo *Sync*⁸ (*Slave is on sync mode*)** – (informação): este bit é ativado assim que um escravo recebe o comando de *Sync control*.
- **Reservado (*reserved*)** – (ignorar): este bit não é utilizado pelo protocolo, portanto o mesmo será ignorado na análise de diagnósticos.
- **Diagnóstico desativado (*deactivated*)**: este caso nunca será detectado no ProfiDoctor uma vez que este bit é ativado apenas no contexto do mestre.
- **Excesso de diagnóstico estendido (*Overflow*)** – (crítico): O bit ativado indica que os 238 bytes de diagnóstico estendido não foram suficientes para reportar todos os diagnósticos do equipamento. Este problema acontece quando o sistema está sem tempo para fazer comunicação acíclica e os diagnósticos se acumulam ou quando o escravo tem muitos diagnósticos para enviar ao mestre. Em ambos os casos considera-se tal problema como crítico, afinal, um escravo com muitos diagnósticos pode estar com muitos problemas e a falta de tempo para transmitir estas informações ao mestre pode ocultar cenários críticos para uma planta industrial.

⁷ Freeze mode: Neste modo, os valores das entradas de um dispositivo são congelados no valor corrente. Nenhuma entrada é modificada até que o mestre da rede envie um comando de *unfreeze*.

⁸ Sync mode: Neste modo, os valores das saídas de um dispositivo são congelados no valor corrente. Nenhuma saída é modificada até que o mestre da rede envie um comando de *unsync*.

Com exceção do bit de *overflow*, o terceiro byte de diagnóstico possui todos os bits sem uso (reservados). O byte número quatro contém o endereço do mestre que parametrizou o escravo. Caso haja outro mestre na linha, este endereço serve para informar qual mestre parametrizou este escravo. Por fim, têm-se os bytes cinco e seis. Estes possuem o valor do *identNumber* de um escravo.

Desta maneira, este módulo é responsável por analisar cada um dos telegramas Profibus que possuam informações de diagnóstico e, caso algum diagnóstico relevante aparecer, tal informação é levada ao usuário de maneira clara, a fim de facilitar a solução do problema.

4.5 Análise dos campos de um telegrama Profibus

Sabe-se o protocolo Profibus possui alguns tipos de telegramas para a comunicação entre os escravos e o mestre. Cada um destes telegramas possui uma série de campos e cada campo possui um conjunto de valores válidos.

Sendo assim, pode-se pensar em uma análise onde cada um destes campos é analisado e caso algum campo possuir valor inválido, um problema é indicado.

É importante notar que este tipo de erro pode acontecer principalmente por dois motivos. O primeiro seria um erro de implementação de um escravo. Tal situação é muito rara, e normalmente quando acontece é originada por dispositivos não certificados, afinal, um dispositivo certificado passa por uma série de testes que elimina esta possibilidade. Outra maneira de acontecer é quando o sinal transmitido é danificado por algum tipo de interferência ou erros de instalação. Este segundo caso é mais comum.

A Tabela 4.2 mostra os possíveis valores para cada um dos campos de um telegrama Profibus. A partir desta tabela, foi implementado um algoritmo que analisa cada um dos campos e em caso de valores inválidos, um diagnóstico de erro é apresentado.

Tabela 4.2 – Campos do telegrama da camada de Enlace Profibus e seus possíveis valores

CAMPO	VALORES POSSÍVEIS
SD (delimitador inicial)	SD1 (10H), SC (E5H), SD2 (68H), SD4 (DCH)
DA (endereço destino) ⁹	0 até 255 (0 até FF) – Todos os valores são permitidos para 1 byte.
SA (endereço fonte) ⁹	0 até 254 (0 até FE)
FC	0 até 127 (são 7 bits, pois o primeiro é sempre 0)
FCS	Não há limites de valores
ED	16H
LE e LER	0 até 249 dependendo do tipo de telegrama (campo aparece apenas no SD2)
DATA_UNIT	Não há limites de valores

Para a implementação deste módulo, uma comparação simples é feita para verificar se o valor inserido em um determinado campo do telegrama é um valor válido ou não. Esta implementação é diferente do padrão utilizado nos outros módulos. Sabe-se que os outros módulos recebem um evento com telegramas Profibus já interpretados e encapsulados em objetos Java. Para o caso da análise realizada neste módulo, esta abordagem não funcionaria. Isto acontece, pois se em um dos campos do telegrama um valor inválido é inserido, a interpretação dos outros campos do mesmo fica comprometida. Por exemplo, se o valor do primeiro byte do telegrama (SD – *start delimiter*) estiver inválido, como o interpretador do telegrama vai saber qual é o próximo campo a ser lido?

Assim sendo, a análise apresentada aqui é feita no momento da interpretação do telegrama (encapsulamento dos bytes em um objeto Java) e caso um erro seja detectado, o problema é apresentado ao usuário.

Para facilitar o entendimento, tem-se um exemplo da análise do primeiro campo do telegrama Profibus. Este campo é chamado de SD (*start delimiter*) e é utilizado para diferenciar o formato dos telegramas Profibus. Sendo assim, quando um telegrama é recebido por este módulo de análise, uma comparação com os quatro valores permitidos (10H, 68H, A2H e DCH) para este campo é feita.

⁹ Sabe-se que os endereços permitidos para este campo podem estar entre 0 e 127. Sabe-se também que apenas 7 bits dos campos DA e SA são utilizados para endereçamento. Sendo assim, ao analisar este campo de maneira integral, os valores podem ir até 255, afinal, nesta análise, todos os bits deste campo são analisados.

4.6 Análise da temporização da rede com sistemas nebulosos (*fuzzy*)

Segundo Lee K.C. (2003a), existe uma série de parâmetros a serem configurados em uma rede Profibus e quando um desses parâmetros recebe um valor incorreto, o desempenho da rede é afetado. Dentre os vários parâmetros a serem configurados, destacam-se alguns parâmetros relacionados à temporização da rede.

Os parâmetros de tempo são importantes em praticamente em todos os tipos de redes de comunicação e para as redes *fieldbus* não é diferente. Husemann e Pereira (2003), afirmam que o processamento das informações em redes adiciona atrasos nos laços de controle e os usuários dessas redes devem considerar a influência destes atrasos no comportamento de sistemas de automação. Os autores afirmam também que diferentes protocolos de comunicação possuem características temporais distintas que dependem de fatores como o comprimento do cabeamento, a taxa de transmissão, o mecanismo de acesso ao meio, entre outros.

Um dos principais parâmetros de tempo e objeto de estudo desta tese é o *Target Rotation Time* - T_{TR} . Este parâmetro refere-se ao intervalo entre o início e o término da comunicação dos mestres com todos os dispositivos escravos de uma rede, ou seja, o intervalo entre o início e o fim de um ciclo de comunicação (NORMATIVE PARTS OF PROFIBUS FMS, DP AND PA, 1998). É importante notar que este parâmetro é configurado pelo usuário no momento da configuração da rede (*off-line*). Quando a rede está em regime de operação (*on-line*), o tempo de um ciclo é chamado *Real Rotation Time* (T_{RR}).

Além do T_{TR} , outro parâmetro de tempo muito importante para uma rede Profibus é o tempo de *watchdog*. Assim como o T_{TR} , este tempo também é configurado pelo usuário. O *watchdog* é utilizado para garantir a operação segura de um sistema no caso de uma falha no mestre ou no cabo de comunicação. Caso o escravo pare de receber comandos do mestre por qualquer motivo, ele aguardará este período (*WatchDog Time* - T_{WD}) antes de ir para o modo de segurança. O T_{WD} não pode ser menor do que o T_{TR} , pois caso isto aconteça, o tempo entre duas requisições ao mesmo escravo excederá o tempo de *watchdog* e este escravo abandonará a troca de dados. Na prática, recomenda-se um valor de T_{WD} entre 5 e 10 vezes o valor do T_{TR} .

Segundo Lee, K.C. (2003a), muitas vezes o usuário ou até mesmo algumas ferramentas de configuração estimam o valor do T_{TR} baseando-se em métodos heurísticos e através da tentativa e erro. Este tipo de atitude pode deixar o tempo de ciclo da rede muito

alto fazendo com que em caso de queda da rede, os dispositivos demorem muito para entrar em modo de segurança deixando uma planta industrial em uma situação de risco.

Nota-se a partir das explicações anteriores que a configuração correta do valor do T_{TR} é fundamental para o bom funcionamento da rede. Entretanto, na prática, a maioria dos mantenedores de redes não executa o cálculo deste parâmetro de maneira precisa. Normalmente, estes apenas aceitam o valor do T_{TR} sugerido pelas ferramentas de configuração que utilizam-se de valores padrão ou executam cálculos a fim de fazer tal configuração de maneira automática evitando erros dos usuários (KAGHAZCHI, 2008).

Estes cálculos são baseados em equações que levam em consideração alguns outros parâmetros de tempo da rede Profibus. Segundo a Normative Parts of Profibus FMS, DP and PA (1998), a equação para o cálculo é:

$$\min T_{TR} = NA * (T_{TC} + highT_{MC}) + k * lowT_{MC} + MT * retT_{MC} \quad (2)$$

Na equação 2, o parâmetro NA representa o número de estações mestre da rede. O T_{MC} é o tempo necessário entre o envio do primeiro bit de um telegrama e o recebimento do último bit da resposta relacionada a este mesmo telegrama. Este último parâmetro envolve o tempo de transmissão de um telegrama além do *station delay time* que representa o tempo necessário para um escravo receber um telegrama, processá-lo e enviar uma mensagem de resposta. Outro parâmetro é o $retT_{MC}$ que é o tempo de ciclo para mensagens de *retry*. Para representar o número de ciclos de mensagens de *retry* por rotação de *token* tem-se o MT. Além disso, existe o T_{TC} , que é o tempo necessário para o envio de um *token* para a próxima estação mestre. Por fim, tem-se o parâmetro K que representa o número estimado de mensagens de baixa prioridade por rotação de *token*. É importante que o cálculo do T_{MC} seja feito de maneira diferente para mensagens de alta prioridade (*high T_{MC}*) e de baixa prioridade (*low T_{MC}*).

Analisando-se esta equação nota-se que alguns parâmetros da mesma são difíceis de serem estimados de forma precisa, como por exemplo, o parâmetro MT, que representa o número de *retries* e pode variar dependendo de fatores como o não cumprimento de requisitos de instalação, entre outros. Outro parâmetro difícil de ser determinado é o número de mensagens de baixa prioridade por rotação de *token*.

Neste contexto, tem-se que na prática, uma série de equações mais simplificadas são utilizadas para este cálculo.

Um exemplo é a equação apresentada por Felser (2011) para definir um valor mínimo para o T_{TR} :

$$\min T_{TR} = n * (T_{TC} + T_{MC}) + k * T_{MC} \quad (3)$$

Na equação 3, o parâmetro n representa o número de estações mestre de uma linha. O parâmetro k é um número estimado de mensagens de baixa prioridade que são transmitidas em um ciclo. T_{TC} é o tempo necessário para o envio de um *token* para a próxima estação mestre. Por fim, tem-se o T_{MC} que possui o mesmo significado do parâmetro apresentado na equação 2, mas neste caso, não existe uma diferenciação para o cálculo do T_{MC} para mensagens de baixa e de alta prioridade. Outra diferença importante entre as equações é que mensagens de *retry* não são consideradas no cálculo. Para considerar as mensagens de *retry*, Felser (2011) sugere um aumento de 10 a 20% no valor calculado para o T_{TR} .

Nota-se que um dos parâmetros destas equações é o T_{MC} . O cálculo deste parâmetro também é aproximado e possui diferentes mecanismos para ser realizado. Segundo a Normative Parts of Profibus FMS, DP and PA (1998) o cálculo do T_{MC} deve ser realizado da seguinte maneira:

$$T_{MC} = T_{S/R} + T_{SDR} + T_{A/R} + T_{ID} + 2.T_{TD} \quad (4)$$

O parâmetro $T_{S/R}$ da equação 4 é o tempo de mensagens de envio e resposta (*send/request*). O T_{SDR} é o tempo necessário para o escravo responder a um pedido. O $T_{A/R}$ é o tempo da mensagem de confirmação (*ACK*). O T_{ID} é o tempo que o barramento fica desocupado (*idle*) e por fim o T_{TD} , que é o tempo de transmissão.

A fim de simplificar estes cálculos, equações mais simples foram criadas para calcular o tempo de ciclo de mensagens. Nestas alguns parâmetros de pouca relevância para o

resultado final foram ignorados ou aproximados. Um exemplo de equação é apresentado por Manchester Metropolitan University Profibus International Competence Centre (2008).

$$T_{mC} \approx (380 + 300 * S + 11 * D) * T_{bit} + 75 \mu s \quad (5)$$

A equação 5 presta-se ao cálculo do tempo de um ciclo de mensagens. Nesta, a variável S representa o número de escravos da rede e D o número total de bytes de dados de entradas e saídas. O valor do parâmetro T_{bit} é dependente da velocidade de transmissão da rede. Por exemplo, para uma rede de 1,5Mbps, o valor do T_{bit} é 0,667us. Vale destacar que o cálculo realizado na equação 5 é bastante aproximado. Os valores numéricos constantes utilizados na equação representam valores aproximados e típicos de parâmetros como o tempo de espera que o mestre deve respeitar entre a recepção do último bit da última resposta e o início do primeiro bit da próxima requisição na rede (T_{ID1}), o tempo que uma estação leva para responder a uma requisição (T_{SDR}), o tempo que cada mensagem leva para transmitir o seu cabeçalho, entre outros.

Vitturi (2004) também propôs uma equação para calcular o tempo de ciclo do Profibus DP e nesta, tanto o tempo de transmissão de dados cíclicos quanto acíclicos são analisados. Além disso, a equação deste autor exige informações como o tempo necessário para a transmissão de um *token* (T_{TK}), o tempo de manutenção da rede onde mensagens de *FDL Status* são enviadas (T_{GAP}), o tempo gasto para troca de dados entre os mestres e escravos da rede (T_P), o tempo necessário para a leitura de diagnóstico dos escravos (T_D), entre outros.

Nota-se que as equações apresentadas exigem do gerenciador da rede Profibus o conhecimento de uma série de parâmetros, dificultando muito a configuração. Além da dificuldade em conhecer de forma precisa todos os parâmetros necessários para o cálculo do T_{TR} , existem configurações onde mais de um mestre controla uma mesma rede. Esta situação tem acontecido cada vez mais a medida que as empresas integram as redes *fieldbus* com o departamento de tecnologia da informação a fim de possibilitar melhores mecanismos de gerenciamento e manutenção dessas redes. Nestes casos é necessário que exista um mestre classe 1 para o controle da rede e um mestre classe 2 para a supervisão e manutenção da mesma (KAGHAZCHI, 2008). Desta maneira, a maioria das ferramentas de configuração não tem conhecimento dos múltiplos mestres e por isso, executam o cálculo automático do T_{TR} de maneira isolada para cada um dos mestres. Sendo assim, o T_{TR} calculado torna-se menor do

que o necessário, afinal, o T_{TR} correto a ser configurado nos dois mestres seria a soma dos tempos de ciclo de todos os mestres da rede. Como nem sempre o engenheiro responsável leva em consideração todas estas condições, muitas vezes o valor do T_{TR} configurado é baixo, fazendo com que a rede fique instável, afinal, os escravos podem interromper a troca de dados em uma situação como esta.

Outra prática utilizada para evitar problemas com o T_{TR} , é configurar os mestres com um valor de T_{TR} muito acima do necessário. Este tipo de prática é muito utilizada a fim de evitar a necessidade de um cálculo mais preciso e trabalhoso deste parâmetro. O problema neste tipo de prática é que um valor muito alto para o T_{TR} influencia no tempo de *watchdog* caso este último esteja sendo calculado com base no T_{TR} , conforme apresentado anteriormente. Assim, um dispositivo não consegue perceber se a rede parou de funcionar em tempo hábil para entrar em modo de segurança, pois ele aguardará um período de *watchdog* elevado.

Uma observação importante é que se um mestre em regime de operação estiver realizando seu ciclo completo em um tempo menor do que o T_{TR} configurado, o mesmo não espera o T_{TR} ser atingido para passar o *token* para outro mestre ou para si mesmo em caso de redes com apenas um mestre.

Não bastando todos os problemas apresentados, em um estudo feito por Kaghazchi et al. (2008), afirma-se que mestres de diferentes fabricantes possuem comportamento diferente em configurações de redes idênticas.

Ainda tratando-se da questão de redes multi-mestres, Lee C.K. (2003a), afirmou que mensagens de alta prioridade são o foco do desempenho da comunicação. Por outro lado, mensagens de baixa prioridade não são sensíveis aos atrasos da rede, afinal, estas podem ser transmitidas quando a rede tem tempo suficiente para tal. Assim, quando o T_{TR} é muito elevado em uma estação, muitas mensagens de baixa prioridade podem ser transmitidas e, neste caso, outros mestres poderão ter dificuldades no momento de transmitir suas mensagens cíclicas. O autor afirma então que o cálculo de um valor de T_{TR} apropriado é crítico na comunicação e este valor deve ser otimizado da melhor maneira possível. Uma afirmação deste mesmo teor é feita por Venturini (2008).

A importância de configurar parâmetros de tempo corretamente também é destacada por Tao (2005). Este autor faz uma análise dos atrasos de uma rede Profibus-DP e suas influências no desempenho de sistemas de controle. Em seu trabalho, o autor conclui que atrasos na rede podem causar fenômenos de *overshooting* e oscilações no controle de um

motor (*pump*). Bao (2009), também apresentou um método para analisar o parâmetro T_{TR} . No caso deste autor, uma rede Profibus com apenas um mestre foi analisada. O autor destacou que para redes bem configuradas, o quesito de que a comunicação aconteça em tempo real é obtido na maioria das vezes.

Tovar et al. (1999) desenvolveu uma análise do pior caso de tempo de resposta de uma mensagem Profibus. O autor afirmou que este tipo de análise está relacionada com o conhecimento do máximo intervalo de tempo entre o recebimento de duas mensagens de *token* por um determinado mestre e que este tipo de estudo é fundamental para garantir os requisitos de tempo de uma rede Profibus.

Ferreira et al. (2004) realizou análises de tempo relacionadas ao uso de redes Profibus com comunicação sem fio trabalhando de maneira concomitante com redes Profibus com transmissão de dados EIA-485. Para a implementação desta rede utilizou-se o *Inter Domain Protocol* (IDP) que oferece compatibilidade com o padrão Profibus e inclui mecanismos para suportar a comunicação entre dispositivos *wireless*.

Uma análise relacionada a atrasos em redes Profibus foi feita também por Lee, S. (2003). Em seu artigo, o autor apresenta um modelo matemático para calcular os atrasos de comunicação em redes Profibus FMS. Para tal, os valores de tempo e as características de tráfego devem ser fornecidos. Os resultados deste modelo foram comparados com experimentos reais a fim de validar o projeto.

Por fim, tem-se a pesquisa realizada por Lee, K.C. (2003a). Este autor propôs um algoritmo de sintonização para o parâmetro T_{TR} para redes Profibus-FMS utilizando para isso algoritmos genéticos. Neste caso, o autor utilizou as condições de tráfego de dados das estações para sintonizar os valores de T_{TR} de cada estação.

A maioria dos trabalhos apresentados fazem propostas relacionadas a metodologias para o cálculo do valor do T_{TR} . Cada um apresenta uma proposta diferente e em muitos casos, o cálculo do mesmo torna-se uma tarefa complexa para os mantenedores das redes Profibus. Além disso, estas diferentes maneiras para obter o valor do T_{TR} enfatizam a imprecisão característica existente no quesito tempo das redes Profibus. Sendo assim, esta quantidade de pesquisas sobre o assunto evidencia a importância do cálculo do mesmo para as redes Profibus, principalmente quando a rede depende da comunicação em tempo real. Segundo Casanova (2006), quando informações de tempo não são essenciais para o controle de uma planta industrial, alguns parâmetros de configuração da rede podem ser negligenciados, mas

quando as necessidades de controle da planta crescem, uma rede mal configurada pode ser uma fonte potencial de instabilidade.

Considerando a importância de uma boa configuração do valor do T_{TR} , todas as imprecisões e dificuldades no momento de definir o mesmo, o desejo de eliminar boa parte da complexidade dos cálculos e a necessidade de garantir um valor que permita um bom desempenho da rede, é proposto nesta tese a utilização de um sistema *fuzzy* para o cálculo do T_{TR} , afinal, a utilização deste modelo de sistema é indicada sempre que um sistema precisa lidar com imprecisões, ambiguidades, abstrações e ambivalência em modelos matemáticos ou conceituais (BURROUGH E MCDONNELL,1998).

Antes de apresentar o sistema *fuzzy*, tem-se que uma alternativa simples para a obtenção do valor do T_{TR} seria através do tempo real gasto para a transmissão de mensagens, ou seja, no tempo que a rede gasta para realizar um ciclo completo quando está em regime de operação (T_{RR}). Sendo assim, o usuário poderia apenas medir este tempo e reconfigurar a rede inserindo este valor como sendo o T_{TR} da mesma. Esta abordagem não é interessante, pois o Profibus transmite mensagens acíclicas que podem mudar o valor do T_{RR} nos diferentes ciclos da rede. Sendo assim, o sistema criado foi modelado para trabalhar em função do tempo de transmissão de mensagens cíclicas (T_C) e do tempo de transmissão de mensagens acíclicas (T_A). A variável de saída é o próprio T_{TR} .

A ideia é que os conjuntos *fuzzy* de cada variável sejam criados a partir do valor do T_{TR} configurado pelo usuário no momento da configuração *off-line* da rede. Com o sistema *fuzzy* criado, o tempo gasto para a transmissão de mensagens cíclicas e acíclicas é obtido pelo ProfiDoctor a partir da análise da rede Profibus em regime de operação. Estes valores são inseridos nas entradas do sistema *fuzzy* que os transforma em variáveis linguísticas como MUITO BAIXO, BAIXO, BOM, ALTO E MUITO ALTO. Estas classificações são feitas sempre em relação ao valor do T_{TR} configurado pelo usuário. A partir desses valores linguísticos, o sistema faz a avaliação das regras e indica, também de maneira linguística, como o T_{TR} deve ser configurado. As variáveis linguísticas utilizadas nos conjuntos *fuzzy* da saída do sistema são as mesmas variáveis utilizadas nos conjuntos de entrada. A partir disso, o modelo de inferência Mamdani é aplicado e por fim, a saída defuzzificada fornece o valor do T_{TR} ideal para o sistema.

Por exemplo, um usuário configura o T_{TR} da rede em 0,9ms. Após isso, com a rede em regime de operação, o T_C máximo é medido a partir de alguns ciclos de mensagens e o valor de 0,4ms é obtido. Para T_A o valor obtido é de 0,55ms. Neste caso, o T_C seria classificado

como BAIXO em relação ao T_{TR} configurado e o T_A seria considerado ALTO. A partir disso, o modelo de Mamdani é aplicado e a saída defuzzificada é apresentada ao usuário. A Figura 4.25 apresenta um diagrama resumindo o processo descrito.

Figura 4.25 – Etapas do sistema para definição do valor do TTR

Com base na Figura 4.25, cada uma das etapas é explicada de maneira detalhada nos próximos parágrafos.

ETAPA 1 – configuração do T_{TR} :

Nesta etapa o usuário é responsável pela configuração do valor do T_{TR} na rede Profibus e no ProfiDoctor. Isso significa que o valor escolhido para o T_{TR} será informado aos mestres da rede através de uma ferramenta de configuração qualquer e também deverá ser informado ao ProfiDoctor através de sua interface gráfica. Este cálculo poderá ser feito de maneira simples e bastante aproximada considerando, por exemplo, apenas o número de dispositivos da rede e um valor médio de tempo gasto por cada dispositivo na comunicação. Outra alternativa é através das ferramentas de configuração Profibus, deixando o sistema *fuzzy* apenas como um mecanismo para aproximar mais o valor configurado do valor de ciclo real da rede. Por fim, caso o usuário prefira, o mesmo pode utilizar as equações sugeridas pelos autores citados neste capítulo, mas neste caso, a vantagem do sistema em permitir que o usuário não tenha dificuldades para a configuração deixa de existir.

ETAPA 2 – Criação do sistema Fuzzy:

O sistema nebuloso será construído automaticamente no momento em que a configuração de uma rede é iniciada. Tal montagem automática acontece, pois o valor do T_{TR} muda para cada configuração. Sendo assim, um valor de T_{TR} que é bom para uma determinada configuração pode ser alto para outra.

A partir do valor de T_{TR} configurado, os três sistemas *fuzzy* serão construídos, sendo dois deles para representar as entradas T_C e T_A e o outro para representar a saída T_{TR} . Cada conjunto *fuzzy* do sistema será construído com base em experiência de especialistas e através da observação do comportamento da comunicação em uma rede Profibus. Considerando que cada variável tem suas próprias características, a criação de cada um será feita de maneira independente. São utilizadas as funções de pertinência trapezoidal e triangulares. A Figura 4.26 apresenta estas duas funções. É importante atentar-se para os parâmetros a , m , n e b da função trapezoidal e para os parâmetros a , b e c da função triangular, afinal, a modelagem de cada conjunto *fuzzy* criado neste módulo do projeto está baseada nestes parâmetros.

Figura 4.26 – Funções trapezoidal e triangular

Para a obtenção dos valores desses parâmetros, uma unidade de medida foi criada com base na equação 6:

$$U = \left(\frac{T_{TR}}{20} \right) \quad (6)$$

A ideia da equação 6 é que a partir do valor do T_{TR} , a unidade de medida U seja calculada. Esta unidade é basicamente um vinte avos do valor do T_{TR} . Por exemplo, se o T_{TR} foi configurado com o valor de 0,9 microssegundos, o valor da unidade de medida U será de 0,045. Para o entendimento da utilização desta unidade e da modelagem dos conjuntos *fuzzy*, segue:

- **Criação do sistema para a variável de saída T_{TR} .** Para uma visão geral de todos os conjuntos criados, tem-se na Figura 4.27 um exemplo deste sistema considerando 0,9ms o valor do T_{TR} configurado pelo usuário. Para este exemplo, a unidade de medida U tem o valor de 0,045:

Figura 4.27 – Sistema fuzzy de saída com o TTR configurado em 0,9ms

- Conjunto BOM: será considerado bom o valor que esteja em um intervalo de dois U , acima e abaixo do valor do T_{TR} configurado pelo usuário. Por exemplo, para o T_{TR} configurado com o valor de 0,9, pode-se considerar bom, valores entre 0,81 e 0,99. Devido a este intervalo, a função de pertinência escolhida para este caso foi a função trapezoidal. Para os parâmetros a e b são atribuídos, respectivamente, valores cinco U abaixo e acima do valor do T_{TR} .
- Conjunto BAIXO: para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor do T_{TR} subtraído de sete vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, três vezes o valor de U no valor de c .
- Conjunto MUITO BAIXO: este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito baixo. Para isso, atribui-se para a variável n o valor do T_{TR} subtraído de onze vezes o valor de U . Para o cálculo da variável b , acrescenta-se duas vezes o valor de U em n . As variáveis m e a deverão receber o valor zero, pois o T_{TR} nunca será negativo.
- Conjunto ALTO: para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor do T_{TR} , acrescido de sete vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, três vezes o valor de U no valor de c .
- Conjunto MUITO ALTO: este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito alto. Para isso, atribui-se para a variável m o valor do T_{TR} acrescentado de onze vezes o valor de U . Para o cálculo da variável a , subtrai-se duas vezes o valor de U em m . O valor de n e b é de duas vezes o valor atribuído para o T_{TR} .

- **Criação do sistema para a variável de entrada T_C .** Para uma visão geral de todos os conjuntos criados, tem-se na Figura 4.28 um exemplo deste sistema considerando 0,9ms o valor do T_{TR} configurado pelo usuário. Para este exemplo, a unidade de medida U tem o valor de 0,045:

Figura 4.28 – Sistema fuzzy da entrada TC com o TTR de 0,9ms

- **Conjunto BOM:** neste caso, será considerado bom um valor que esteja próximo a 70% do valor do T_{TR} configurado. Este valor foi estipulado, pois este sistema prioriza a transmissão de mensagens cíclicas, afinal, este tipo de mensagem tem fundamental importância no controle de uma planta industrial. Considerando que não existe apenas um valor bom para este tipo de configuração, a função de pertinência escolhida para este caso foi a função trapezoidal. Sendo assim, será considerado bom, o valor que esteja em um intervalo de dois U , acima e abaixo do valor dos 70% do T_{TR} configurado pelo usuário. Para os parâmetros a e b , que definem os limites da função trapezoidal, subtrai-se e soma-se, respectivamente, duas vezes o valor de U no valor de m e n .
- **Conjunto BAIXO:** para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor de 70% do T_{TR} subtraído de cinco vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, duas vezes o valor de U no valor de c .
- **Conjunto MUITO BAIXO:** este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito baixo. Para isso, atribui-se para a variável n o valor

de 70% do T_{TR} subtraído de nove vezes o valor de U . Para o cálculo da variável b , acrescenta-se três vezes o valor de U em n . As variáveis m e n deverão receber o valor zero, pois o T_C nunca será negativo.

- Conjunto ALTO: para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor de 70% do T_{TR} , acrescido de cinco vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, duas vezes o valor de U no valor de c .
- Conjunto MUITO ALTO: este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito alto. Para isso, atribui-se para a variável m o valor de 70% do T_{TR} acrescentado de oito vezes o valor de U . Para o cálculo da variável a , subtrai-se duas vezes o valor de U em m . O valor de n e b é de duas vezes o valor atribuído para o T_{TR} .

- **Criação do sistema para a variável de entrada T_A .** Para uma visão geral de todos os conjuntos criados, tem-se na Figura 4.29 um exemplo deste sistema considerando 0,9ms o valor do T_{TR} configurado pelo usuário. Para este exemplo, a unidade de medida U tem o valor de 0,045:

Figura 4.29 – Sistema fuzzy da entrada T_A com o valor do T_{TR} de 0,9ms

- Conjunto BOM: neste caso, será considerado bom um valor que esteja próximo a 40% do valor do T_{TR} configurado. Este valor foi estipulado,

pois este sistema prioriza a transmissão de mensagens cíclicas, pois a maioria destas tem fundamental importância no controle de uma planta industrial. Sendo assim, um sistema com este tipo de prioridade não pode ocupar o seu tempo de ciclo com muitas mensagens acíclicas. Considerando que não existe apenas um valor bom para este tipo de configuração, a função de pertinência escolhida para este caso foi a função trapezoidal. Sendo assim, será considerado bom, o valor que esteja em um intervalo de U , acima e abaixo do valor dos 40% do T_{TR} configurado pelo usuário. Para os parâmetros a e b , que definem os limites da função trapezoidal, subtrai-se e soma-se, respectivamente, duas vezes o valor de U no valor de m e n .

- Conjunto BAIXO: para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor de 40% do T_{TR} , subtraído de três vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, uma vez o valor de U no valor de c .
- Conjunto MUITO BAIXO: este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito baixo. Para isso, atribui-se para a variável n o valor de 40% do T_{TR} subtraído de cinco vezes o valor de U . Para o cálculo da variável b , acrescenta-se duas vezes o valor de U em n . As variáveis m e n deverão receber o valor zero, pois o T_C nunca será negativo.
- Conjunto ALTO: para este conjunto, uma função de pertinência triangular é utilizada. Neste, o valor de c é configurado com o valor de 40% do T_{TR} , acrescido de quatro vezes o valor de U . Para as variáveis a e b , que definem os limites da função triangular, subtrai-se e soma-se, respectivamente, duas vezes o valor de U no valor de c .
- Conjunto MUITO ALTO: este utiliza a função trapezoidal, afinal, depois de um determinado valor, qualquer valor configurado deve ser considerado muito alto. Para isso, atribui-se para a variável m o valor de 40% do T_{TR} acrescentado de sete vezes o valor de U . Para o cálculo

da variável a , subtrai-se duas vezes o valor de U em m . O valor de n e b é de duas vezes o valor atribuído para o T_{TR} .

ETAPA 3 – obtenção de T_C e T_A a partir da rede Profibus

Neste momento, o ProfiDoctor realiza a coleta das mensagens transmitidas na rede Profibus em regime de operação e contabiliza o tempo utilizado para o envio e recebimento das mensagens cíclicas e acíclicas. Para este projeto, as mensagens de *data exchange*, *requestFDLStatus* e *pass token* serão consideradas cíclicas, pois nos mestres analisados, este tipo de mensagem acontece em praticamente todos os ciclos de transmissão. Os outros tipos de mensagens, como por exemplo, *retries*, *read inputs*, *read outputs*, *getDiagnostic*, *checkConfiguration*, *setParameters*, etc. serão consideradas acíclicas.

Para tal, são levados em consideração os tempos obtidos durante 300 ciclos de comunicação, sendo que este valor é configurável. Este mecanismo é importante principalmente para configurações onde existam mestres classe 2, pois o intervalo de flutuação do tempo de mensagens que um mestre deste tipo envia no barramento é muito grande, afinal o tamanho e a quantidade de mensagens transmitidas depende do tamanho dos dados requisitados por este tipo de mestre (KAGHAZCHI, 2008). Esta mesma afirmação é encontrado no artigo de Venturini (2008), que destaca que mensagens acíclicas transmitidas na rede variam em cada ciclo de *token*.

Após a obtenção dos tempos de T_C e T_A estes valores são inseridos nas entradas do sistema.

ETAPA 4 – fuzzificação das entradas:

Neste momento, as variáveis T_C e T_A são fuzzificadas, ou seja, os valores *CRISP* obtidos da rede são classificados em MUITO BAIXO, BAIXO, BOM, ALTO OU MUITO ALTO de acordo com os conjuntos *fuzzy* criados na etapa 2.

Assim como citado anteriormente neste documento, com um T_{TR} configurado com o valor de 0,9ms, um T_C de 0,4ms e um T_A de 0,55ms teria-se que T_C é BAIXO e o T_A é ALTO.

ETAPA 5 – aplicação das regras de inferência:

As regras utilizadas encontram-se na forma:

SE (T_C É A) E (T_A É B) ENTÃO (T_{TR} DEVE SER C).

Seguindo o exemplo da etapa 4, uma possível regra seria:

SE (T_C É BAIXO) E (T_A É ALTO) ENTÃO (T_{TR} DEVE SER BOM).

Considerando os valores linguísticos que cada variável pode assumir e o conhecimento de especialistas, a base de conhecimento apresentada na Tabela 4.3 foi criada:

Tabela 4.3 – Regras de inferência fuzzy

T_C	T_A	T_{TR}
MUITO BAIXO	MUITO BAIXO	MUITO BAIXO
MUITO BAIXO	BAIXO	MUITO BAIXO
MUITO BAIXO	BOM	BAIXO
MUITO BAIXO	ALTO	BAIXO
MUITO BAIXO	MUITO ALTO	BOM
BAIXO	MUITO BAIXO	BAIXO
BAIXO	BAIXO	BAIXO
BAIXO	BOM	BAIXO
BAIXO	ALTO	BOM
BAIXO	MUITO ALTO	BOM
BOM	MUITO BAIXO	BOM
BOM	BAIXO	BOM
BOM	BOM	BOM
BOM	ALTO	BOM
BOM	MUITO ALTO	ALTO
ALTO	MUITO BAIXO	ALTO
ALTO	BAIXO	ALTO
ALTO	BOM	ALTO
ALTO	ALTO	MUITO ALTO
ALTO	MUITO ALTO	MUITO ALTO
MUITO ALTO	MUITO BAIXO	MUITO ALTO
MUITO ALTO	BAIXO	MUITO ALTO
MUITO ALTO	BOM	MUITO ALTO
MUITO ALTO	ALTO	MUITO ALTO
MUITO ALTO	MUITO ALTO	MUITO ALTO

Desta forma, este sistema irá definir o valor que o T_{TR} **deve ser** configurado.

ETAPA 6 – definição das regras ativas e agregação:

A partir dos valores linguísticos das variáveis T_A e T_C definidos, dos conjuntos *fuzzy* criados e da base de conhecimento, analisa-se quais são as regras ativas.

Com as regras ativas definidas, aplica-se o operador *MIN*, afinal, as regras foram construídas a partir do conectivo E. O resultado desta operação é então agregado através do operador *MAX*. O resultado deve então ser defuzzificado.

ETAPA 7 – defuzzificação:

Para a defuzzificação do valor, utilizou-se o método do centroide. Após isso, o valor *CRISP* resultante é apresentado na interface gráfica do usuário em forma de sugestão.

4.7 Análise baseada em informações dos sistemas inteligentes

Este é o módulo que irá receber os resultados de todas as análises inteligentes e elaborar uma única saída para o usuário. A execução deste é feita após todas as análises sugeridas nos módulos descritos nos itens anteriores, afinal, a análise em questão depende de informações oriundas destes itens.

Para realizar esta combinação um sistema responsável por unir diagnósticos de diferentes módulos e, a partir disso, fornecer um diagnóstico mais completo foi criado. Os marcadores a seguir apresentam cada uma das análises realizadas:

- a) Os módulos que analisam os sinais (RNA e níveis de tensão) detectam deformações de bits Profibus e indicam uma possível causa. Muitas vezes, apesar do sinal não estar em perfeitas condições, a rede continua funcionando normalmente. Para este caso, o usuário recebe apenas uma notificação alertando-o de que a rede não está suficientemente robusta e apontando a causa do problema. Caso o problema seja detectado e os telegramas recebidos estiverem corrompidos, o usuário recebe uma mensagem de erro apontando a causa do erro. Para saber se a rede está funcionando corretamente ou não, o módulo de análise dos campos de um telegrama Profibus é consultado. Para exemplificar este cenário, pode-se citar o problema de falta de terminadores no barramento. Muitas vezes, apesar do barramento estar sem terminadores a rede continua operando normalmente.

- b) Na análise de sinais obtida a partir do nível médio de tensão, um dos diagnósticos é de que uma rede está com problemas no driver. Este diagnóstico baseia-se no fato de que todos os escravos estão transmitindo algum sinal, mas apenas um escravo está *off-line*. O problema é que se o cabeamento se romper no final da rede e deixar apenas um escravo desconectado ou um cabeamento longo estiver sendo utilizado para a comunicação com apenas um escravo, este módulo vai diagnosticar o problema de maneira equivocada apontando um erro no próprio escravo. Considerando que o módulo de RNA é capaz de analisar a forma de onda e detectar problemas de cabeamento rompido ou cabeamento longo, pode-se cruzar as duas informações e apresentar ao usuário o diagnóstico correto. O mesmo raciocínio pode ser utilizado caso mais de um escravo estiver apresentando problemas em sua fonte. Se este tipo de problema acontecer, o módulo de análise de níveis de tensão vai diagnosticar erroneamente o problema como cabeamento longo ou rompido. Utilizando a informação das RNAs é possível perceber que o problema no sinal não foi causado por estes dois últimos motivos e desta maneira, modifica-se o diagnóstico para indicar problemas no dispositivo.

Além das situações descritas nos tópicos (a) e (b), tem-se que todas as vezes que um diagnóstico de falha fornecido pelos outros módulos do sistema é recebido por este módulo, uma comparação para analisar se diferentes módulos obtiveram o mesmo diagnóstico é realizada. Caso mais de um módulo fornecer o mesmo diagnóstico de falha, este diagnóstico é informado ao usuário com uma notificação extra de que tal informação possui um grau de certeza maior do que as outras. Caso os diagnósticos de falhas sejam diferentes entre os módulos, uma notificação comum é fornecida. A fim de apresentar cada tipo de problema oriundo dos outros módulos do sistema e identificar qual módulo pode fornecer tal diagnóstico, tem-se a Tabela 4.4. Além disso, esta tabela apresenta o momento do ciclo de vida da rede que tal diagnóstico é fornecido, ou seja, se o problema detectado acontece durante o momento em que a rede está em regime de operação ou quando o sistema está sendo instalado ou modificado, afinal, o diagnóstico de falhas é uma função importante em todos os estágios do ciclo de vida de qualquer produto (FENTON, 2002).

Tabela 4.4 – Erros possíveis do sistema relacionados aos módulos

DIAGNÓSTICO	MÓDULO	MOMENTO DE DETECÇÃO
Sinal com rede funcionando normalmente com cabeamento de 1m/200m (sinal de <i>IDLE</i> e de dados)	RNA e Nível de tensão.	Instalação/Manutenção e Operação: sinal funcionando corretamente vai ser detectado e nenhuma ação é feita, afinal, o mesmo está correto.
Falta de terminador, cabeamento quebrado ou conector desconectado	RNA e Nível de tensão.	Instalação/Manutenção e Operação: terminador pode ser desativado, cabeamento pode quebrar, o conector pode ser retirado do dispositivo, etc.
Rede utilizando cabeamento longo (1000m)	RNA e Nível de tensão.	Instalação/Manutenção: quando um cabeamento longo é utilizado o sinal apresenta-se deformado logo que a rede é inicializada.
Excesso de terminadores de linha	RNA	Instalação/Manutenção e Operação: terminador pode ser ativado a qualquer momento.
Curto circuito entre as linhas de dados A e B	RNA e Nível de tensão	Instalação/Manutenção: neste caso a rede não inicia a comunicação
Terminador sem energia	RNA	Instalação/Manutenção e Operação: rede pode ser inicializada com operador sem energia ou pode perder a fonte de energia com a rede em regime de operação.
Interferência eletromagnética	RNA – Pré-análise.	Instalação/Manutenção e Operação: instalação feita incorretamente ou algum motor ou outro tipo de dispositivo instalado próximo à rede Profibus. Além disso, dispositivo pode ligar e desligar esporadicamente.
Problemas com a fonte ou com o driver do dispositivo	Nível de tensão.	Instalação/Manutenção e Operação: o dispositivo pode estar com problemas logo no momento de inicializar o funcionamento da rede ou passa a apresentar defeito causado por algum fenômeno prejudicial.
Problema de parametrização ou de endereçamento	Histórico de mensagens e Campos do telegrama de diagnóstico	Instalação/Manutenção: vai acontecer no momento que um dispositivo novo é inserido na rede ou caso um dispositivo for trocado durante o funcionamento.
Escravo não encontrado	Histórico de mensagens	Instalação/Manutenção: caso a rede seja inicializada sem que um determinado escravo não esteja se comunicando.

DIAGNÓSTICO	MÓDULO	MOMENTO DE DETECÇÃO
Interferência momentânea causada por ruído ou mau contato na conexão	Histórico de mensagens	Operação: se alguma interferência acontece a comunicação que estava funcionando normalmente é prejudicada
Escravo perdeu a conexão e não retornou para a rede	Histórico de mensagens	Operação: durante o funcionamento da rede um escravo pode ser removido, desligado, apresentar problemas, etc.
Dispositivo entra e sai da linha esporadicamente	Histórico de mensagens	Operação: causada por algum dispositivo que provoca EMI.
<i>Watchdog baixo</i>	Histórico de mensagens	Instalação/Manutenção: no início da operação da rede caso o <i>watchdog</i> esteja configurado de forma equivocada.
Estação não pronta para troca de dados	Campos do telegrama de diagnóstico	Instalação/Manutenção: no momento da instalação ou durante uma manutenção o escravo pode levar tempo para executar os procedimentos de configuração ou parametrização
Falha de configuração	Campos do telegrama de diagnóstico	Instalação/Manutenção: a configuração feita pode estar errada.
Escravo não suporta função solicitada	Campos do telegrama de diagnóstico	Operação: o escravo recebe um pedido de um mestre e não suporta a função requisitada ou não consegue executar o pedido por causa de falhas internas.
Escravo esperando parametrização	Campos do telegrama de diagnóstico	Instalação/Manutenção e Operação: sempre que um escravo é iniciado ele apresenta este diagnóstico para o mestre. Uma série de problemas faz com que um escravo entre e saia da linha de dados. Sempre que isso acontece, o mesmo precisa ser parametrizado novamente.
Indicativo de diagnóstico estático	Campos do telegrama de diagnóstico	Operação: escravo tenta informar algo durante a operação da rede.
Problema em algum dos campos do telegrama Profibus	Análise dos campos do telegrama Profibus	Instalação/Manutenção ou Operação: o problema pode acontecer já na instalação do sistema por problemas de EMI ou mesmo erro de implementação no escravo ou durante a operação em algum telegrama de dados ou em alguma função específica.

A partir da Tabela 4.4, é possível obter uma visão ampla de todos os problemas que cada módulo poderá diagnosticar. Com a plataforma para diagnóstico inteligente apresentada em detalhes, têm-se no capítulo 5 uma descrição dos materiais utilizados na obtenção dos resultados desta tese.

5 Descrição da plataforma experimental

A fim de obter os resultados necessários para comprovar a tese proposta neste documento, uma série de ferramentas, incluindo softwares, dispositivos Profibus, entre outros foram utilizadas. Sendo assim, este capítulo tem a função de apresentar estas ferramentas.

A Tabela 5.1 apresenta as informações dos dispositivos físicos utilizados.

Tabela 5.1 – Ferramentas utilizadas

Dispositivo	Tipo	Endereço	Fabricante
ET 200S HIGH FEATURE	Escravo	24	Siemens
DP100V	Escravo	34	Altus
DP-B-FDN-M	Escravo	59	Sense
DP-KD-2EA-D	Escravo	6	Sense
XM-210DP	Escravo	8	DLG
DF73 - HSE/Profibus-DP Control	Mestre Classe 1	2	Smar
Proficaptain/Proficore Ultra - V 1.5	Mestre Classe 1 e Classe 2	1	Procentec
Hardware de aquisição	Ferramenta Passiva	—	—
3000 Series Oscilloscope - DSO3152A-150MHz - 1GSa/s	Ferramenta Passiva	—	Agilent
Computador Sony Vaio VPCY115FX com Win 7	Ferramenta Passiva	—	Sony
Computador IBM T60 com Win XP SP3	Ferramenta Passiva	—	IBM

Mais detalhes sobre os dispositivos escravos e mestres apresentados na Tabela 5.1 são apresentados no APÊNDICE B.

Além dos dispositivos físicos, alguns softwares foram utilizados nesta tese. Detalhes destes são apresentados na Tabela 5.2.

Tabela 5.2 – Softwares utilizados

Software	Fabricante	Versão
3000 Series Connect Software	Agilent	1.1.27
Matlab	MathWorks	7.7.0.471 (R2008b)
AnyBus NetTool	HMS Industrial Networks	1.09.0004
Sycon	Hielscher	2.918 (56)
Simatic Manager Step7	Siemens	5.5
Profitrace	Procentec	2.5
ProfiDoctor	—	1.0
Simulador	Torres (2011)	1.0

A maior parte dos materiais apresentados pode ser visualizada na Figura 5.1.

Figura 5.1 – Fotografia da bancada de testes utilizada na coleta dos resultados

A partir da Figura 5.1 tem-se:

- 1) Computador IBM T60 com Win XP SP3;
- 2) 3000 Series Oscilloscope - DSO3152A-150MHz - 1GSa/s;

- 3) DF73 - HSE/Profibus-DP Control
- 4) Kits com os escravos apresentados na Tabela 5.1;
- 5) Proficaptain/Proficore Ultra - V 1.5;
- 6) Computador Sony Vaio VPCY115FX com Win 7;
- 7) Cabeamento Profibus de 1000 metros;

Dentre todas as ferramentas apresentadas, três fazem parte do projeto proposto por esta tese. Uma delas é o próprio software de análise denominado ProfiDoctor, já apresetado no capítulo 4. As outras duas ferramentas são o hardware responsável pela coleta das informações a partir de redes Profibus e o simulador. Sendo assim, uma apresentação mais detalhada dos mesmos é feita nos itens 5.1 e 5.2.

5.1 Hardware para aquisição do sinal e dos telegramas Profibus

Para a criação do hardware a ser utilizado neste sistema uma série de componentes foram utilizados e uma especificação prévia do sistema foi desenvolvida. A Figura 5.2 apresenta o esquema geral especificado.

Figura 5.2 – Arquitetura interna do Hardware

Como apresentado na Figura 5.2 o sinal Profibus chega através da linha RS485 e é, ao mesmo tempo, enviado para dois blocos. O primeiro deles, denominado MAU, é responsável

por interpretar os sinais da linha RS485 e transformá-los em valores lógicos (bits). A saída deste bloco será os telegramas Profibus.

A montagem apenas do telegrama Profibus seria suficiente para algumas análises do protocolo, mas, além dessas camadas, sabe-se que o sistema possui um diagnóstico relacionado à camada física Profibus. Para isso, o hardware precisa enviar ao ProfiDoctor os sinais da camada física amostrados. Tal função é desempenhada pelo ADC (conversor analógico-digital). Assim, este componente irá receber o sinal RS485 e irá amostrar cada bit em vinte amostras. Neste caso, a saída deste componente será um conjunto de vinte amostras para cada bit de dados trafegado na rede. O ADC escolhido é fabricado pela *Texas Instruments* e é denominado ADS6128. Este tem uma frequência de amostragem de até 210 MSPS (*Mega Samples Per Second*), o que é suficiente para a amostragem dos dados com vinte amostras por segundo até a velocidade de 6 Mbits/s. Segundo a Normative Parts of Profibus FMS, DP and PA (1998), o protocolo pode operar em até 12 Mbits/s. Nesta velocidade, tem-se que a quantidade máxima de amostras seria de dez amostras por bit.

Este resultado baseia-se no teorema de *Nyquist-Shannon* que resume o cálculo como $FD = N * FS$. Onde FD é a frequência de amostragem, N é o número de amostras e FS é a frequência do sinal.

Por fim, as saídas do ADC e do MAU são utilizadas como entradas em um último bloco denominado FPGA + Nios II. Seu fabricante é o Altera, e o seu modelo é o Arria II GX. A escolha deste modelo baseou-se no fato de que a Altera é um fabricante renomado no mercado, possui um suporte técnico ativo e este componente cobriu todas as necessidades para o funcionamento deste hardware. Sabe-se que para as análises feitas a partir do ProfiDoctor, o mesmo precisa receber a informação do telegrama Profibus e juntamente com estes telegramas, as amostras dos respectivos sinais da camada física.

Tal informação, é enviada através do telegrama ProfiDoctor, apresentado na Figura 4.2. Para a montagem deste telegrama com os dados oriundos do MAU e do ADC, é necessária a sincronização dessas duas informações. Quem faz isso é o próprio FPGA, que após a sincronização, armazena estes dados em uma memória RAM.

Estes dados então são lidos por um processador *softcore* denominado *NIOS II*. Este processador faz parte da estrutura do próprio FPGA. A função do mesmo é ler a memória que armazena os dados descritos no parágrafo anterior, montar o telegrama ProfiDoctor e enviá-lo ao software através de uma conexão TCP/IP. Esta comunicação também é controlada pelo *NIOS II*.

Vale lembrar que o desenvolvimento deste hardware está sendo realizado em um projeto de pesquisa concomitante a este e não será utilizado para a coleta dos resultados desta tese. É aqui documentado por motivo de documentação da proposta.

5.2 Simulador

A importância da simulação, principalmente de sistemas complexos, já está consolidada. Segundo (HOSSEINPOUR, 2009), sistemas simulados podem auxiliar na seleção de uma melhor alternativa de projeto, assim como na solução de problemas em projetos já implantados.

O objetivo do desenvolvimento de um simulador de redes Profibus DP é auxiliar no desenvolvimento e validação da ferramenta de análise e diagnóstico ProfiDoctor. Como já apresentado, além de ser muito importante na execução de testes e na obtenção dos resultados desta tese, o simulador pode ser útil em treinamentos, afinal, o grande crescimento deste tipo de protocolo e a consequente necessidade de profissionais capazes de lidar com este tipo de rede impulsionam uma grande demanda de ambientes para capacitação.

Para atingir tal objetivo, alguns requisitos são necessários:

- Transferir os telegramas resultantes para a ferramenta ProfiDoctor através de uma comunicação TCP/IP;
- Permitir construir uma configuração de rede, simular a mesma e modificar o estado dos equipamentos durante a execução da simulação;
- Permitir simular erros relacionados a cada camada do protocolo;
- Manter uma interface amigável semelhante à interface do ProfiDoctor;
- Permitir dividir o simulador em módulos de simulação de mestre e escravos para validar ambos separadamente;

A partir da revisão bibliográfica relacionada ao tema de simulação de redes Profibus desenvolvido por Torres (2011), constatou-se que não existe nenhuma ferramenta com objetivos semelhantes aos descritos anteriormente. Apesar disso, existem simuladores para este tipo de protocolo com objetivos e funcionalidades diferentes. Sendo assim pode-se destacar o Simulador de Mestre Profibus DP/DP-V1 B+W que é dotado de um software e um conversor serial RS232/RS485 ou USB/RS485 com a finalidade de fazer interface com um computador pessoal. A pilha de protocolos é executada no computador e a porta serial (ou

USB) é utilizada para fazer interface com a rede Profibus DP real. A velocidade da rede é limitada a 19.200 kbps pela ferramenta. Isso torna possível a implementação e execução da camada de enlace do mestre Profibus DP através do Windows sem prejudicar a operação da rede (HUANG, 2005). Outra ferramenta que merece destaque é o SIMIT fornecido pela Siemens. Este equipamento tem a função de emular os escravos Profibus DP. Seus dados de aplicação podem ser modificados através de uma estação de operação dedicada a este fim. Com a mesma função do SIMIT, mas com menos destaque comercial, tem-se um sistema desenvolvido pela empresa Sprintprosi. Este trata-se de uma placa que utiliza circuitos integrados dedicados a controladores Profibus permitindo simular até 16 escravos. Por fim, em Antonopoulos (2004) há o desenvolvimento de uma aplicação em Visual Basic onde o usuário pode configurar os parâmetros de rede em conjunto com parâmetros experimentais (número de estações, número de mestres, etc.) com o objetivo de determinar o atraso médio de resposta dos escravos e o *throughput*, permitindo criar um modelo de simulação para o Profibus DP.

Quanto ao desenvolvimento do simulador, é importante destacar que o mesmo foi desenvolvido seguindo as mesmas definições do ProfiDoctor, como por exemplo, a linguagem de programação Java que permite uma independência entre os sistemas operacionais mais populares. Como ambos estariam sendo desenvolvidos na mesma linguagem, além do aproveitamento de código, evitam-se possíveis problemas de interoperabilidade como formatos numéricos, datas, etc. Além disso, o uso de uma mesma linguagem facilita o desenvolvimento de classes e telegramas comuns a ambos os aplicativos, acelerando o desenvolvimento e evitando duplicação de trabalho. Para auxílio do trabalho em grupo, a ferramenta SVN (sistema de controle de revisões) foi utilizada facilitando assim o controle de versões desses componentes comuns. Um servidor de uso gratuito foi escolhido para permitir o compartilhamento e controle de versão do código (APÊNDICE A).

O simulador pode ser utilizado para simular uma rede com qualquer número de equipamentos dentro da especificação. A maneira como o simulador foi estruturado evita que o uso do PC como plataforma para simulação cause problemas de atraso ou imprecisões devido a características do sistema operacional que não incluem a operação das aplicações em tempo real. Nenhum módulo ou biblioteca de tempo real foi utilizado. A operação do simulador ocorre como um processo qualquer e o *timestamp* dos telegramas são calculados de acordo com as regras de tempo exigidas pelo protocolo, permitindo o uso do simulador sem

sobrecarregar o computador e também possibilitando a paralização e o reinício da simulação a qualquer momento sem comprometer os resultados.

Considerando que o objetivo é fornecer ao ProfiDoctor as mesmas informações que o hardware de aquisição oferece, o simulador cria amostras de sinais obedecendo o mesmo padrão do hardware, ou seja, para cada bit, 20 amostras são criadas. A criação dessas amostras foi feita com base em sinais reais obtidos em laboratório. Para o caso dos sinais com EMI e com excesso de terminadores utilizou-se como base os sinais obtidos em PROCENEC (2012).

Além dos formatos de sinais da camada física, outros problemas podem ser simulados inserindo-se ou removendo-se equipamentos da rede durante a simulação e alterando-se outros atributos. A Figura 5.3 apresentam os painéis que permitem as operações descritas.

Figura 5.3 – Painel para modificar atuar sobre os escravos simulados

Outra possibilidade está relacionada com o telegrama de *getDiagnostic response*. Em um dos painéis do simulador, é possível modificar o diagnóstico padrão de resposta de um escravo durante a simulação da rede.

Todas as ações do simulador apresentadas até aqui tem relação com o funcionamento da rede enquanto a mesma está em regime de operação (*on-line*), não possuindo correlação com a configuração *off-line* do mestre. A configuração do mestre fica armazenada no arquivo *master.properties*, no formato *ASCII*. Esta configuração é carregada quando o simulador é inicializado. Este arquivo tem uma sintaxe baseada em chave e atributo e permite atribuir uma configuração simples para cada um dos escravos e definir se o endereço esta presente na simulação ou não. A Figura 5.4 apresenta um exemplo deste arquivo.

```
master.properties_.txt - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

BaudRate=93750
SlotTime=4095
MinTSDR=22
MaxTSDR=1000
QuietTime=0
SetupTime=150

TTR=200000
GAP=2
Retries=2
HSA=126
WDG=5000

Address0.isActive=no
Address0.IdentNumber=0000
Address0.Cfg=00

Address1.isActive=no
Address1.IdentNumber=0000
Address1.Cfg=00

Address2.isActive=no
Address2.IdentNumber=0000
Address2.Cfg=00

Address3.isActive=no
Address3.IdentNumber=0000
Address3.Cfg=00

Address4.isActive=yes
Address4.IdentNumber=0000
Address4.Cfg=1020

REP = 5

Address5.isActive=yes
Address5.IdentNumber=1020
Address5.Cfg=1020

Address6.isActive=no
Address6.IdentNumber=0000
Address6.Cfg=00
```

Figura 5.4 – Arquivo master.properties
FONTE: TORRES (2011)

Para cada endereço existem 3 atributos:

- **isActive**: Determina se o endereço está na configuração do mestre ou não.
- **IdentNumber**: Determina o modelo do equipamento a ser utilizado. Existe um único *IdentNumber* para cada modelo de equipamento.
- **Cfg**: Determina qual a configuração a ser enviada no comando *Check Configuration*, enviado pelo mestre.

Para tornar o endereço parte da configuração, deve ser atribuído o valor de *yes*, para o atributo *isActive* do endereço selecionado. A configuração deve coincidir com a configuração do escravo para que este entre em troca de dados.

O parâmetro REP indica a presença de um repetidor na sequência após a estação anterior, isso acarreta em um *delay* na comunicação definido pelo valor do REP em T_{bit} .

Este arquivo obedece a sintaxe tipo *java properties* (JAVA PROPERTIES, 2011), o que facilita a leitura dos atributos do arquivo texto sem a necessidade da implementação de um *parser* adicional.

Mais detalhes sobre a implementação e sobre os resultados obtidos para o simulador em destaque podem ser encontrados em Torres (2011).

Com a descrição da plataforma utilizada nos experimentos realizados nesta tese, têm-se no capítulo 6 os resultados obtidos.

6 Resultados

Os resultados desta tese foram obtidos em laboratório. Alguns cenários de análise não foram reproduzidos em laboratório e nestes casos, dados sintéticos foram utilizados.

A velocidade de transmissão da rede foi configurada em 1,5Mbps, por ser esta a velocidade mais utilizada em redes Profibus em operação.

A topologia básica da rede utilizada praticamente em todos os cenários criados para a coleta das informações pode ser visualizada na Figura 6.1.

Figura 6.1 – Topologia da rede Profibus real utilizada maioria dos testes

6.1 Resultados relacionados às análises dos sinais Profibus

Este item foi estruturado e organizado conforme a evolução do trabalho, norteado de acordo com os experimentos realizados em ordem cronológica.

A captura dos sinais de forma amostrada foi realizada através do software DSO3000 da Agilent, que captura as amostras de um sinal oriundo do osciloscópio e salva estas

amostras em um documento Excel. Dessa maneira, para cada tipo de erro, alguns arquivos Excel foram criados. Após isso, um único arquivo de texto contendo todas as amostras de todos os tipos de sinais foi criado e utilizado nos módulos de análises de sinal. Vale lembrar que para os sinais com EMI e com excesso de terminadores, valores sintéticos obtidos a partir de PROCENEC (2012) foram utilizados. Além dos treinamentos e testes com dados coletados a partir de redes Profibus reais, algumas análises, detalhadas no decorrer deste capítulo, foram realizadas a partir de sinais gerados de maneira sintética com o objetivo de obter-se uma quantidade maior de amostras e consequentemente uma maior variedade de sinais no treinamento das RNAs. A criação destes sinais sintéticos foi feita com base em sinais reais que foram modificados de maneira que sua forma de onda mantivesse um padrão que permitisse a identificação por parte do sistema e ao mesmo tempo sofresse pequenas alterações de uma amostragem para outra. Para um melhor entendimento, têm-se na Tabela 6.1 os gráficos de dez formas de ondas geradas pelo simulador juntamente com a forma de onda obtida em laboratório ou em PROCENEC (2012). Desta maneira, é possível observar a semelhança entre cada forma de onda. Os gráficos possuem no eixo y os valores dos níveis de tensão e no eixo x as amostras coletadas em um determinado período de tempo. É importante observar que as escalas de valores dos níveis de tensão (eixo Y) são diferentes de um gráfico para outro.

Tabela 6.1 – Ondas sintéticas (cor azul) e reais (cor vermelha) para cada tipo de sinal

Tipo de sinal	Gráficos
Sinal com rede configurada corretamente com cabeamento de 1m e 200m
Sinal de IDLE com rede configurada corretamente com cabeamento de 1m e 200m;

Tipo de sinal	Gráficos
Curto circuito em linhas de dados A e B
Falta de terminador, cabeamento quebrado ou conector desconectado
Terminador sem alimentação de energia
Excesso de terminadores de linha;
Rede utilizando cabeamento longo de 1000m

Tipo de sinal	Gráficos
EMI

Observando a Tabela 6.1 é possível perceber que o tipo de sinal com exemplares sintéticos mais diferentes entre si é aquele que representa o problema de EMI. Tal fato ocorre devido à aleatoriedade deste tipo de sinal. Um sinal sintético que também variou bastante é o sinal que representa cabeamento longo. Tal diferença foi implementada a fim de simular cabeamentos longos com diferentes tamanhos, afinal, a atenuação varia de acordo com o comprimento do cabeamento.

Com os detalhes de como os dados foram obtidos, tem-se que a primeira análise realizada considerou a utilização de apenas uma RNA responsável por realizar a classificação de cada tipo de problema. Para esta RNA, todas as amostras foram obtidas a partir de redes Profibus reais, com excessão das amostras com EMI e com excesso de terminadores. Considerando que 8 formas de onda diferentes foram selecionadas para serem classificadas e que para cada tipo de forma de onda 20 bits Profibus foram amostrados, tem-se que a RNA foi treinada com 160 bits Profibus. Vale lembrar que cada bit Profibus foi amostrado em 20 pontos. Desta maneira, o arquivo de treinamento foi composto por 160 linhas e por 20 colunas. A saída foi composta por 8 neurônios e, assim como em todas as RNAs utilizadas nesta tese, utilizou-se 20 neurônios na camada de entrada. Os testes foram realizados através do toolbox de Redes Neurais do Matlab (Neural Network Toolbox), inicializado através do comando *nprtool*.

Para este caso, uma alta taxa de erros foi encontrada sendo que os melhores resultados são apresentadas na Tabela 6.2. Vale destacar aqui que diferentes números de neurônios foram utilizados na camada escondida das RNAs.

Tabela 6.2 – Resultados utilizando uma RNA para a classificação de todos os sinais

Neurônios camada escondida	Épocas	Porcentagem de Erro	MSE ¹⁰
10	42	33,33%	6,49235 e-2
15	73	29,62%	6,49586 e-2
20	90	14,81%	3,22406 e-2
30	90	33,33%	7,41699 e-2

Considerando os resultados ruins, uma nova abordagem foi criar uma RNA independente para a classificação de cada sinal. Sendo assim, cada RNA criada é responsável por indicar se um determinado sinal representa ou não um determinado tipo de problema. Esta solução apresenta-se com a mesma abordagem descrita no item 4.1.2, com a diferença de que neste caso, são utilizadas oito RNAs e que amostras de sinais com EMI são inseridos nos dados de treinamento. Nesta nova organização, as RNAs também foram treinadas com 160 bits Profibus amostrados a partir de uma rede Profibus real (exceto sinais com EMI e excesso de terminadores), com 20 neurônios de entrada e, diferentemente da RNA única criada anteriormente, a saída de cada RNA foi composta por 2 neurônios. Os testes foram realizados novamente através do toolbox de Redes Neurais do Matlab. Para este caso, os resultados dos treinamentos e testes de cada RNA são apresentados na Tabela 6.3.

Tabela 6.3 – Resultados utilizando oito RNAs para todos os sinais

RNA	Camada escondida	Épocas	Porcentagem de Erro	MSE
OK 1m e 200m	10 neurônios	66	3,70%	2,06286e-7
	15 neurônios	61	0%	1,57565e-2
	20 neurônios	78	0%	1,78691e-7
	30 neurônios	74	0%	1,08191e-7
IDLE OK 1m e 200m	10 neurônios	105	3,70%	3,70750e-2
	15 neurônios	90	3,70%	3,70304e-2
	20 neurônios	29	3,70%	4,20770e-2
	30 neurônios	35	3,70%	4,88340e-2
Curto circuito A e B	10 neurônios	85	3,70%	3,70380e-2
	15 neurônios	74	0%	1,66607e-7
	20 neurônios	90	3,70%	3,70371e-2
	30 neurônios	85	3,70%	3,70271e-2
Falta de terminadores ou cabo rompido	10 neurônios	38	0%	1,72520e-2
	15 neurônios	39	0%	4,78240e-4
	20 neurônios	45	0%	1,63655e-3
	30 neurônios	49	0%	1,43564e-5

¹⁰ O MSE (*Mean Square Error*) é o erro em relação aos valores observados e os valores obtidos.

RNA	Camada escondida	Épocas	Porcentagem de Erro	MSE
Terminador sem energia	10 neurônios	109	0%	6,52202e-3
	15 neurônios	55	0%	9,84731e-3
	20 neurônios	136	0%	2,10843e-7
	30 neurônios	139	0%	1,11098e-5
Excesso de terminadores	10 neurônios	47	0%	2,59778e-6
	15 neurônios	27	0%	3,37027e-3
	20 neurônios	54	0%	3,93364e-7
	30 neurônios	65	0%	1,52069e-7
Cabeamento longo	10 neurônios	64	3,70%	3,28953e-2
	15 neurônios	38	7,40%	6,33052e-2
	20 neurônios	38	3,70%	1,99339e-2
	30 neurônios	23	3,70%	2,82789e-2
EMI	10 neurônios	48	7,40%	6,59644e-2
	15 neurônios	31	3,70%	4,51407e-2
	20 neurônios	64	3,70%	2,40888e-2
	30 neurônios	21	3,70%	3,81490e-2

A partir da comparação entre a Tabela 6.1 e a Tabela 6.2, é possível notar que os resultados melhoraram consideravelmente, mas mesmo assim algumas RNAs ainda apresentam erros.

Desta maneira, os padrões dos sinais utilizados nos treinamentos foram analisados e notou-se que os sinais com EMI possuíam uma variância muito alta que poderia estar prejudicando as RNAs. Decidiu-se então eliminar das amostras estes tipos de sinais e desta maneira, diminuiu-se o número de RNAs de oito para sete. Assim sendo, estas RNAs foram treinadas com 140 bits Profibus amostrados, a saída de cada uma foi composta por 2 neurônios e utilizou-se 20 neurônios de entrada. Novamente os testes foram realizados no Matlab, sendo que desta vez, obteve-se uma taxa de acertos muito boa. Os resultados de cada RNA encontram-se na Tabela 6.4.

Tabela 6.4 – Sete RNAs para a classificar todos os sinais (exceção EMI)

RNA	Camada escondida	Épocas	Porcentagem de Erro	MSE
OK 1m e 200m	10 neurônios	65	0%	1,09956e-7
	15 neurônios	70	0%	3,13185e-7
	20 neurônios	85	0%	2,56773e-7
	30 neurônios	81	0%	2,40090e-7
IDLE OK 1m e 200m	10 neurônios	65	0%	3,08814e-7
	15 neurônios	86	0%	3,13469e-4
	20 neurônios	70	0%	8,13959e-8
	30 neurônios	81	0%	2,82674e-7

Curto circuito A e B	10 neurônios	57	0%	5,19004e-8
	15 neurônios	73	0%	8,66367e-8
	20 neurônios	45	0%	1,44709e-7
	30 neurônios	75	0%	8,91363e-8
Falta de terminadores ou cabo rompido	10 neurônios	43	0%	2,24910e-7
	15 neurônios	64	0%	7,58683e-7
	20 neurônios	45	0%	5,28910e-7
	30 neurônios	54	0%	2,41698e-7
Terminador sem energia	10 neurônios	41	0%	1,14719e-7
	15 neurônios	50	0%	3,90998e-7
	20 neurônios	40	0%	6,05059e-7
	30 neurônios	39	0%	1,12312e-7
Excesso de terminadores	10 neurônios	60	0%	9,49151e-8
	15 neurônios	84	0%	1,32190e-7
	20 neurônios	58	0%	2,67235e-7
	30 neurônios	70	0%	1,35770e-7
Cabeamento longo	10 neurônios	71	0%	4,81645e-8
	15 neurônios	45	0%	6,84545e-7
	20 neurônios	42	0%	2,84642e-7
	30 neurônios	80	0%	5,75154e-8

Com o bom resultado, decidiu-se pela eliminação dos sinais com EMI das amostras de treinamento. Para tal tarefa, optou-se pela análise da variância de cada sinal. A partir disso, um novo mecanismo, já apresentado no item 4.1.1 foi criado. Considerando que este mecanismo foi criado a partir da observação de sinais reais coletados em laboratório, outras formas de ondas devem ser obtidas para realizar os testes deste módulo, afinal, com as mesmas amostras, o resultado certamente seria de 100% de acerto. Sendo assim, as formas de onda sintéticas apresentadas na Tabela 6.1 localizada no início deste capítulo são utilizados para os testes.

Foram utilizados 10 bits Profibus sintéticos para cada tipo de forma de onda. Para cada caso, a variância foi calculada e os valores de cada uma estão representados no gráfico da Figura 6.2.

Figura 6.2 – Variância das amostras simuladas

Todos estes dados sintéticos foram inseridos no módulo de pré-análise dos sinais e classificados. A partir da Figura 6.3 é possível observar que para todos os casos a variância é menor do que 0,4, com exceção do cabeamento longo e EMI. Vale enfatizar que a variância do cabeamento longo varia acima e abaixo de 0,4.

Figura 6.3 – Quantidade de amostras acima e abaixo do valor limite da variância

Considerando que os sinais com cabeamento longo oscilam no intervalo escolhido para dividir os sinais com EMI dos outros tipos de sinais, ter-se-á, como resultado de tal divisão, um conjunto de amostras com formas de onda com EMI misturadas com formas de ondas com cabeamento longo. Desta maneira, um mecanismo para diferenciar estas duas

formas de onda também foi criado. Para tal solução, utilizou-se novamente uma RNA do tipo PMC. O conjunto de amostras utilizadas nos treinamentos e testes possuíam amostras de bits Profibus obtidas em laboratório que representavam sinal com cabeamento longo de 1000m com variância maior do que 0,4 e amostras sintéticas que representavam sinais com EMI. Todas estas amostras compuseram um arquivo de entradas de treinamento de 40 linhas, sendo que cada linha representava um bit Profibus amostrado em 20 pontos. A saída da rede foi composta por 2 neurônios. A Tabela 6.5 apresenta os resultados obtidos.

Tabela 6.5 – Resultados da RNA para classificar EMI e cabeamento longo

Camada escondida	Épocas	Porcentagem de Erro	MSE
10 neurônios	31	0%	3,04466 e-2
15 neurônios	34	0%	2,62248 e-4
20 neurônios	26	0%	1,49823 e-3
30 neurônios	18	0%	8,79584 e-3

A partir deste mecanismo em que sete RNAs classificam sinais com variância menor ou igual a 0,4 e outra RNA classifica sinais com variância maior do que 0,4, os resultados obtidos foram ótimos. Mesmo assim, novas análises sobre os sinais a serem coletados continuaram a ser realizadas e um fato importante foi observado. Tal fato tem relação com sinais que são coletados da rede Profibus no momento em que nenhuma informação está sendo transmitida. Estes sinais representam redes com terminadores sem energia, sinais *IDLE* da rede funcionando corretamente e redes com curto circuito entre as linhas de dados A e B. Sendo assim, a fim de aumentar ainda mais a robustez das RNAs, o grupo com as sete redes foi dividido novamente e os resultados das novas análises estão na Tabela 6.6 e na Tabela 6.7.

Para as RNAs cujo resultado está apresentado na Tabela 6.6 foram utilizados 60 bits Profibus amostrados em 20 pontos, 20 neurônios na camada de entrada e 2 neurônios na camada de saída. Para as outras RNAs (Tabela 6.7), a mesma arquitetura foi utilizada, mas os dados de treinamento foram compostos por 80 bits Profibus. Novamente as análises foram executadas através do *toolbox* de RNAs do Matlab.

Tabela 6.6 – Resultados utilizando três RNAs para a sinais IDLE

RNA	Camada escondida	Épocas	Porcentagem de Erro	MSE
IDLE OK 1m e 200m	10 neurônios	21	0%	2,95621e-8
	15 neurônios	22	0%	6,14079e-8
	20 neurônios	22	0%	1,07269e-8
	30 neurônios	26	0%	2,87757e-8
Curto circuito A e B	10 neurônios	16	0%	3,56569e-8
	15 neurônios	24	0%	8,45584e-8
	20 neurônios	19	0%	1,72429e-8
	30 neurônios	23	0%	1,56548e-8
Terminador sem energia	10 neurônios	22	0%	5,48267e-8
	15 neurônios	21	0%	7,77654e-8
	20 neurônios	27	0%	5,42727e-8
	30 neurônios	16	0%	6,05325e-8

Tabela 6.7 – Resultados utilizando quatro RNAs

RNA	Camada escondida	Épocas	Porcentagem de Erro	MSE
OK 1m e 200m	10 neurônios	72	0%	7,48172e-8
	15 neurônios	80	0%	9,99231e-8
	20 neurônios	90	0%	5,73231e-8
	30 neurônios	65	0%	5,91123e-8
Falta de terminadores ou cabo rompido	10 neurônios	43	0%	9,19921e-8
	15 neurônios	41	0%	8,12322e-8
	20 neurônios	79	0%	9,44212e-9
	30 neurônios	50	0%	3,81521e-8
Excesso de terminadores	10 neurônios	62	0%	3,91723e-8
	15 neurônios	62	0%	2,34422e-8
	20 neurônios	70	0%	8,44212e-9
	30 neurônios	80	0%	1,91521e-8
Cabeamento longo	10 neurônios	28	0%	7,40477e-8
	15 neurônios	38	0%	5,15448e-8
	20 neurônios	35	0%	4,14767e-8
	30 neurônios	33	0%	4,37397e-8

Comparando-se os resultados apresentados na Tabela 6.4 com a Tabela 6.6 e a Tabela 6.7, percebe-se uma melhora, afinal, o MSE das duas últimas foi relativamente menor. Outro fato a ser notado foi que a melhor configuração para as RNAs criadas foi a que utilizou 20 neurônios na camada escondida da rede.

Todas as análises apresentadas neste capítulo foram realizadas através do Matlab e considerando os ótimos resultados, esta última configuração do sistema foi implementada no

software ProfiDoctor e, a partir deste último, praticamente os mesmos resultados foram obtidos.

Foi destacado que formas de ondas sintéticas foram utilizadas na validação e análise do sistema desenvolvido. A fim de analisar a legitimidade das amostras sintéticas de sinais criadas pelo simulador, outras análises foram realizadas. Nestas, todas as amostras utilizadas no treinamento das RNAs foram criadas de maneira sintética, ou seja, as RNAs foram treinadas com valores sintéticos. No momento de realizar os testes, os valores utilizados foram os valores reais obtidos em laboratório, com exceção das amostras de sinais para excesso de terminadores e dos sinais com EMI. Todos estes testes foram realizados na plataforma ProfiDoctor utilizando 20 neurônios na camada escondida e a mesma configuração utilizada nos testes cujo resultados foram apresentados na Tabela 6.6 e na Tabela 6.7. Para testar cada tipo de forma de onda, 10 amostras para cada tipo de sinal foram inseridas no sistema. Os resultados deste último procedimento estão na Tabela 6.8.

Tabela 6.8 – Treinamento com dados sintéticos e testes com dados reais

RNA	Acertos	Erros
IDLE OK 1m e 200m	10	0
Curto circuito A e B	10	0
Terminador sem energia	10	0
OK 1m e 200m	10	0
Falta de terminadores ou cabo rompido	10	0
Cabeamento Longo	10	0

Analisando-se estes dados, conclui-se que a criação dos dados sintéticos seguem o padrão necessário e que os mesmos são válidos para este tipo de análise.

Desta maneira, completou-se a análise dos sinais através das RNAs. No ProfiDoctor, este sistema será executado em intervalos definidos pelo usuário e como saída, fornecerá um diagnóstico. A Figura 6.4 apresenta um exemplo de diagnóstico fornecido.

Figura 6.4 – Exemplo de diagnóstico fornecido pelo ProfiDoctor

Uma conclusão importante obtida a partir das amostras de sinais coletadas e da experiência em analisar as mesmas de forma visual é que uma rede Profibus pode apresentar muitas variações no sinal para um mesmo tipo de problema. Por exemplo, para o problema de cabeamento longo, o sinal gerado pode estar com diferentes níveis de atenuação. Isso dependerá de quão longo é um determinado cabo. Outra característica que pode causar modificações no formato do sinal é a velocidade que a rede Profibus está operando, portanto, caso uma rede Profibus em outra velocidade for analisada por este software, será necessária a coleta novas amostras para treinar novamente as RNAs. Estas amostras poderão ser obtidas a partir da própria rede Profibus que será analisada ou pode ser gerada sinteticamente pelo próprio software.

Outra conclusão a ser destacada é que este sistema é capaz de detectar um problema caso o mesmo esteja acontecendo de maneira isolada na rede Profibus. Caso a rede Profibus estiver com o seu sinal deformado por causa de mais de um problema ao mesmo tempo, por exemplo, quando a rede possui cabeamento longo e também não possui terminadores ativos, a detecção deste problema não acontecerá de maneira correta, afinal, a RNA não está treinada para obter resultados para estes casos.

6.2 Resultados relacionados à análise de níveis de tensão

A análise deste módulo não está baseada em resultados quantitativos como foi feito no módulo das RNAs. Para este caso, os resultados baseiam-se em conclusões relacionadas a cada cenário além da confirmação de que a árvore de decisão implementada fornece uma resposta correta ao usuário do sistema. A rede Profibus utilizada é a mesma apresentada no início deste capítulo.

Neste módulo, cinco cenários foram analisados, sendo que na maioria o Proficore (hardware para obtenção dos sinais da rede) foi conectado no escravo ET200S:

- a) **Rede operando de maneira correta:** este é o caso em que nenhum problema é detectado, pois todos os dispositivos estão transmitindo dados com o nível de tensão correto, sendo 6,80 Volts para o escravo DP100V, 5,33 Volts para o escravo ET 200S e 5,64 para o escravo DP-B-FDN-M. A Figura 6.5 apresenta os níveis de tensão obtidos.

Figura 6.5 – Níveis de tensão para rede operando de maneira correta

- b) **Curto-circuito entre as linhas de dados:** este caso é percebido quando todos os dispositivos da rede estão com nível de tensão muito próximo ou igual a zero. Foi possível a reprodução e coleta de amostras em laboratório e para isso, os cabos de dados A e B foram colocados em contato um com o outro no final do barramento Profibus. Quando este tipo de situação acontece, o software não consegue calcular o nível de tensão do dispositivo de maneira padrão, pois o mesmo não detecta médias de tensão com sinais opostos, afinal a média de tensão é zero ou muito próxima deste valor. Para solucionar tal situação, o nível de tensão é calculado apenas com a média aritmética de um único bit amostrado da linha. A média obtida

neste teste foi de 0,078. Outro ponto observado é que não é possível detectar nem mesmo as mensagens de requisição do mestre e por isso, a saída do sistema não relaciona o problema com um dispositivo específico. A partir das amostras coletadas, o ProfiDoctor informa em sua interface de diagnósticos que a rede está sem comunicação e que este tipo de cenário é típico de curto circuito entre as linhas de dados A e B.

Figura 6.6 – Níveis de tensão com rede com curto circuito entre A e B

- c) **Cabeamento longo ou problemas no aterramento:** este caso é percebido quando mais de um dispositivo da rede está com nível de tensão abaixo do recomendado pela norma (4v) e acima de zero. A coleta de amostras foi realizada em laboratório e para isso, utilizou-se um cabo Profibus de 1000 metros para interligar o mestre da rede aos escravos. Para analisar este caso, dois cenários foram criados, sendo que no primeiro apenas um escravo foi ligado à rede através do cabeamento longo e no segundo, dois escravos foram conectados desta forma. Na Figura 6.7 é possível observar que para o primeiro cenário, a média obtida é 6,80 Volts para o escravo DP100V que está a apenas 1 metro do ponto de medição do sinal, 5,06 para o escravo ET 200S que está a 2 metros da medição e 1,75 Volts para o escravo DP-B-FDN-M que está a 1000 metros do ponto de medição. Na mesma figura, têm-se também as médias para o segundo cenário. Neste, o escravo DP100V continua com os mesmos valores, o escravo ET 200S fica com o nível de tensão em 1,98 Volts e o escravo DP-B-FDN-M em 1,73 Volts.

Figura 6.7 – Níveis de tensão para escravos com cabeamento longo

A partir destas configurações, uma falha deste módulo foi detectada. A questão aqui é que as regras da árvore de decisão implementada estão configuradas de maneira que quando apenas um escravo está com nível de tensão baixo, o diagnóstico é de que o problema está no escravo e não na instalação física da rede. Sendo assim, em vez do sistema diagnosticar este caso como sendo problemas de cabeamento longo, o mesmo, de maneira equivocada, classifica-o como problemas no dispositivo. Para solucionar este problema é necessário que estas informações sejam confrontadas com as informações do módulo de análise de sinais através de RNAs. A confrontação das saídas destes módulos é realizada no módulo de análise geral apresentado no item 4.7. Um detalhe importante observado nestas análises é que nestes casos o endereço do escravo é obtido através da análise do campo *destination address* do telegrama enviado pelo mestre, pois a resposta do escravo não é interpretada corretamente.

d) Problemas com a fonte ou com o driver do dispositivo: a reprodução deste cenário depende de sinais transmitidos por algum dispositivo com nível de tensão acima de 7 Volts ou abaixo de 4 Volts. Para detectar um nível de tensão acima de 7 Volts, a medição do sinal é feita a partir do escravo DP100V. Vale lembrar aqui que nos outros cenários testados, a medição é feita a partir do escravo ET200S. Neste ponto, o nível médio de tensão do mesmo ficou em torno de 8 Volts, o escravo ET200S apresentou nível de tensão de 5,02 Volts, o DP-B-FDN-M apresentou nível de 5,73 Volts. A Figura 6.8 apresenta os níveis de tensão utilizados nesta análise.

Figura 6.8 – Níveis de tensão acima do recomendado pela norma Profibus

Quando este cenário foi testado, observou-se que apesar de fora da norma, a interpretação dos sinais com níveis de tensão acima de 7 Volts por parte dos dispositivos da rede ocorreu normalmente. Sendo assim, conclui-se que este tipo de problema deve ser apresentado ao usuário apenas como um alerta de que a rede não está no padrão recomendado.

Outro cenário que ocasiona este tipo de diagnóstico é o caso em que apenas um dispositivo da rede encontra-se com o nível de tensão abaixo do recomendado. Para realizar este teste, a mesma configuração utilizada no item (a), com 3 escravos, foi preparada novamente, sendo que desta vez, o escravo de endereço 59 teve a sua fonte desligada. A Figura 6.9 apresenta os níveis de tensão para este cenário.

Figura 6.9 – Níveis de tensão abaixo do recomendado pela norma Profibus

- e) **Cabeamento quebrado:** o mesmo tipo de problema descrito no cenário de cabeamento longo é encontrado nesta situação, ou seja, caso apenas um

escravo estiver conectado após a ruptura do cabeamento, o diagnóstico realizado é de que o escravo está com problemas em vez de informar o problema de cabeamento quebrado. Sendo assim, na primeira análise, o nível de tensão para o escravo DP100V foi de 6,46, para o escravo ET 200S foi de 5,06 e para o escravo DP-B-FDN-M, localizado após a ruptura do cabeamento foi de zero Volts. Já no segundo caso, em que dois escravos foram instalados após a ruptura do cabeamento, tem-se o escravo DP100V com 6,53 Volts e os outros dois escravos com nível de tensão nulo. A Figura 6.10 apresenta os níveis de tensão para ambos os cenários testados.

Figura 6.10 – Níveis de tensão para cabeamento quebrado

Através destas análises e dos gráficos apresentados neste item, é possível observar a validade das regras implementadas pela árvore de decisão criada, com exceção das regras de cabeamento longo e quebrado que dependem de informações de outros módulos para que sejam validadas com um grau de certeza maior. Vale destacar que as duas regras questionadas neste módulo funcionam bem para a maioria dos casos, pois normalmente quando mais de um dispositivo apresenta os problemas descritos aqui, o diagnóstico será o de cabeamento longo ou quebrado, afinal, mais de um dispositivo sem funcionar por problemas internos em uma única rede não é algo comum.

6.3 Resultados relacionados à análise baseada em histórico de mensagens

Basicamente, este módulo analisa os telegramas armazenados pelo sistema e a partir destes dados fornece ao usuário alguns diagnósticos.

Os telegramas utilizados para a obtenção dos resultados para este módulo são obtidos a partir de dados reais coletados através da ferramenta Profitrace e exportados em forma de arquivo. O ProfiDoctor então realiza o *parser* deste arquivo e, a partir das informações obtidas nele, telegramas ProfiDoctor são criados e armazenados em memória. Por fim, estes dados são processados pelo sistema e um diagnóstico é fornecido.

Além das análises com dados reais, foram realizados testes com dados provenientes do simulador. Desta maneira foi possível comprovar a validade da utilização do simulador para este tipo de análise. Segue então a análise detalhada de cada cenário:

a) Problema de parametrização ou de endereçamento:

Para reproduzir este problema a configuração criada para a rede Profibus descrita no início deste capítulo foi modificada. Ao invés de configurar o escravo DP-B-FDN-M com o GSD correto de *identNumber* 08B5, utilizou-se o GSD de um escravo WAGO 750-333, com *identNumber* B754.

Para obter estes valores, primeiramente analisou-se a mensagem *setParameters* enviada do mestre ao escravo. Esta mensagem é do tipo SD2 e a interpretação da mesma a partir dos dados obtidos é apresentada na Figura 6.11.

Figura 6.11 – Mensagem de *setParameters* enviada ao escravo DP-B-FDN-M

Analisando a Figura 6.11 é possível notar que o valor do *identNumber* enviado para parametrizar o escravo DP-B-FDN-M é diferente de 08B5.

Quando o escravo recebe este telegrama ele tenta realizar as operações necessárias para sua parametrização detectando assim este problema. Para enviar esta

informação ao mestre, o escravo espera pelo pedido de *getDiagnostic* que será enviado no final do processo de sincronização. Desta maneira, analisando a sequência dessas mensagens, obteve-se a resposta do escravo. A interpretação desta resposta encontra-se na Figura 6.12.

Figura 6.12 – *getDiagnosticResponse* com o *identNumber* de outro escravo

Para realizar os mesmos testes a partir do simulador, utilizou-se um mecanismo que permite que o valor do *IdentNumber* seja modificado manualmente, afinal, as ferramentas implementadas nesta tese não oferecem suporte para a obtenção de dados de arquivos GSD. Fazendo tal modificação, o valor do *IdentNumber* ficou diferente nos telegramas de *slaveDiagnostic* e de *setParameters* e os mesmos resultados obtidos a partir de dados reais foram conquistados através da simulação.

O resultado destas análises foi um diagnóstico informando que o escravo está apresentando problemas no GSD utilizado na configuração ou que o endereço do escravo configurado está incorreto.

Realizando este tipo de análise, o software acelera a percepção do problema por parte do operador, pois sem este tipo de interpretação, o usuário contaria apenas com a notificação de erro de parametrização do escravo enviada através de um bit do campo de diagnósticos da mensagem *getDiagnosticResponse*.

b) Erro de endereçamento

Para reproduzir este problema a mesma configuração criada para a rede Profibus descrita no início deste capítulo foi mantida. Neste caso, uma mudança de endereço foi feita fisicamente no escravo DP-B-FDN-M, ou seja, o escravo DP-B-FDN-M recebeu o endereço 69, mas em sua configuração *off-line* o mesmo escravo foi configurado com o endereço 59. Desta maneira, o mestre tenta entrar em troca de dados com o endereço 59 configurado, mas o escravo DP-B-FDN-M está conectado na rede com o endereço 69.

Para detectar este tipo de problema através da análise dos telegramas transmitidos na rede, primeiramente analisou-se a mensagem *getDiagnosticRequest* enviada do mestre ao escravo que supostamente estaria no endereço 59. Notou-se então que sempre que esta mensagem é enviada, nenhuma resposta é recebida pelo mestre. Através da sequência apresentada na Figura 6.13 é possível visualizar o mestre enviando o pedido de diagnóstico por duas vezes sem nenhuma resposta.

Endereço	Serviço	Req/Res	Dados
1->1	Token Passing	Request	DC0101
1->24	Data Exchange	Request	6805056818015D00007616
0->0	Ack	Request	E5
1->34	Data Exchange	Request	6804046822017D00A016
34->1	Data Exchange	Response	6805056801220801002C16
1->59	Get Diagnostics Request	Request	68050568BB816D3C3E2316
1->1	Token Passing	Request	DC0101
1->1	Token Passing	Request	DC0101
1->24	Data Exchange	Request	6805056818017D00009616
0->0	Ack	Request	E5
1->34	Data Exchange	Request	6804046822015D008016
34->1	Data Exchange	Response	6805056801220801002C16
1->59	Get Diagnostics Request	Request	68050568BB816D3C3E2316
1->1	Token Passing	Request	DC0101

Figura 6.13 – Mensagens de requisição de diagnóstico sem respostas

Quando esta situação é detectada pelo sistema, uma nova busca por novas mensagens é realizada. Nesta busca, procura-se por mensagens de *requestFDLStatusResponse*. Este procedimento é o mesmo utilizado durante a criação da *LiveList*.

Ao termino desta busca, foi possível então observar que dentre todos os escravos *on-line*, existia um escravo de endereço 69 respondendo aos *requestFDLStatus*. Este escravo não estava configurado no mestre. Já o escravo 59, configurado no mestre, não está ativo na rede e por isso não responde às requisições do mestre. A Figura 6.14 mostra estas mensagens e suas respostas relativas aos escravos ativos na rede. É importante notar que as mensagens apresentadas não acontecem nesta sequência. Neste exemplo, as mensagens que não são de interesse para esta análise são filtradas e descartadas.

Endereço	Mensagem	Req/Res	Dados
1->24	Request FDL-Status	Request	101801496216
24->1	ACK positive (OK)	Response	100118001916
1->34	Request FDL-Status	Request	102201496C16
34->1	ACK positive (OK)	Response	100122002316
1->59	Request FDL-Status	Request	103B01498516
1->69	Request FDL-Status	Request	104501498F16
69->1	ACK positive (OK)	Response	100145004616

Figura 6.14 – RequestFDL para endereço 59 sem e com resposta para endereço 69

Para realizar os mesmos testes a partir do simulador, configurou-se o arquivo *master.properties* para que os escravos com endereço 24, 34 e 59 estivessem na configuração do mestre simulado. No simulador, por padrão, estes escravos presentes na configuração do mestre são inicializados com o status *on-line*, mas para a reprodução deste caso, antes de inicializar o simulador, modificou-se o status do endereço 59 para *off-line* e o endereço 69 para *on-line*. A partir disso, o simulador foi inicializado, as mensagens foram coletadas e analisadas e o diagnóstico esperado foi obtido.

c) Escravo não encontrado

Esta análise é muito parecida com a apresentada no item anterior. A diferença aqui fica por conta do escravo 69 que não será ativado fisicamente na rede e por isso, não responde a mensagens de *requestFDLStatus*.

Para realizar os mesmos testes a partir do simulador, configurou-se o arquivo *master.properties* para que os escravos com endereço 24, 34 e 59 estivessem na configuração do mestre simulado. Antes de iniciar a criação das mensagens, o status do escravo 59 foi modificado para *off-line*. A partir disso, as mensagens foram analisadas e o diagnóstico esperado foi obtido.

d) Interferência momentânea

Este diagnóstico é fornecido em duas situações. Em ambas o mestre envia mensagens de *retry* para o escravo que está apresentando problemas. O que diferencia uma situação da outra é que em uma, o escravo responde aos comandos de *retry* retomando imediatamente a comunicação com a rede e na outra o escravo não responde aos *retries* e considerando tal fato, o mestre inicia o processo de sincronização com uma mensagem de *getDiagnostic*. Para os dois cenários o diagnóstico é de que a comunicação foi perdida e retomada de imediato, pois a

diferença de tempo entre um caso e outro na escala temporal perceptível para o ser humano é muito pequena (casa dos microssegundos).

Para a reprodução de ambos os casos utilizou-se a mesma configuração apresentada no início deste capítulo. Para emular em laboratório o problema momentâneo de falha da comunicação em que apenas telegramas de *retry* aparecem na transmissão, um curto entre as linhas de dados A e B foi causado por um intervalo de tempo mínimo. No exemplo apresentado na Figura 6.15, este valor está próximo de 280 microssegundos.

Tempo	Endereço	Serviço	Dados
353771	1->1	Token Passing	DC0101
353818	1->24	Data Exchange	6805056818017D00009616
353908	0->0	Ack	E5
353938	1->34	Data Exchange	6804046822015D008016
354215	1->34	Data Exchange	6804046822015D008016
354299	34->1	Data Exchange	6805056801220801002C16
354440	1->59	Data Exchange	103B015D9916
354492	59->1	Data Exchange	680B0B68013B08000000000000004416
354642	1->80	Request FDL ...	105001499A16
354886	1->1	Token Passing	DC0101

Figura 6.15 – Mensagem de *retry* enviada pelo mestre e respondida pelo escravo 34

O segundo cenário foi reproduzido da mesma maneira, mas sem tantas preocupações com a questão do tempo que as linhas de dados A e B ficavam em contato uma com a outra, pois a detecção deste segundo problema é feita com base em telegramas de *retry* seguidos *getDiagnostic* que acontecem durante um período de coleta de dados relativamente grande (1000 ciclos). A Figura 6.16 mostra um caso que ilustra os escravo 59 recebendo *retries* durante a troca de dados e em seguida um comando de *getDiagnostic*, reiniciando assim o processo de sincronização com o mestre.

Endereço	Serviço	Dados
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818017D00009616
0->0	ACK	E5
1->34	Data Exchange	6804046822017D00A016
34->1	Data Exchange	6805056801220801002C16
1->59	Data Exchange	1038015D9916
1->59	Data Exchange	1038015D9916
1->59	Data Exchange	1038015D9916
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818015D00007616
0->0	ACK	E5
1->34	Data Exchange	6804046822015D008016
34->1	Data Exchange	6805056801220801002C16
1->59	Get Diagnostics Request	68050568BB816D3C3E2316
59->1	Slave Diagnostics Response	6827276881BB083E3C000C000108B51C0000000201...
1->1	Token Passing	DC0101

Figura 6.16 – Mensagens de retry e de sincronização

A reprodução deste caso no simulador não foi possível, pois a versão atual do simulador não implementa este tipo de cenário.

e) Escravo perdeu a conexão e não retornou para a rede

Para reproduzir este cenário o escravo DP-B-FDN-M teve sua ligação com a fonte de energia desligada com a rede em regime de operação. A sequência de mensagens obtida foi a mesma apresentada na Figura 6.16, com a exceção de que o mestre passou a enviar a mensagem de *getDiagnosticRequest* em todos os ciclos seguintes ao desligamento da fonte e em nenhum deles obteve algum tipo de resposta do escravo. A Figura 6.17 apresenta a sequência de mensagens com as tentativas sem sucesso de sincronização com o escravo DP-B-FDN-M.

Endereço	Serviço	Dados
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818017D00009616
0->0	ACK	E5
1->34	Data Exchange	6804046822017D00A016
34->1	Data Exchange	6805056801220801002C16
1->59	Data Exchange	1038015D9916
1->59	Data Exchange	1038015D9916
1->59	Data Exchange	1038015D9916
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818015D00007616
0->0	ACK	E5
1->34	Data Exchange	6804046822015D008016
34->1	Data Exchange	6805056801220801002C16
1->59	Get Diagnostics Request	68050568BB816D3C3E2316
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818017D00009616
0->0	ACK	E5
1->34	Data Exchange	6804046822017D00A016
34->1	Data Exchange	6805056801220801002C16
1->59	Get Diagnostics Request	68050568BB816D3C3E2316
1->1	Token Passing	DC0101
1->24	Data Exchange	6805056818015D00007616

Figura 6.17 – Escravo 59 sendo retirado da rede

A reprodução deste caso no simulador não foi possível, pois a versão atual do simulador não implementa este tipo de cenário.

f) WatchDog Baixo

A reprodução deste problema deu-se através de uma configuração com 5 escravos, sendo estes: DLG XM-210DP, SENSE DP-KD-2EA-D, SIEMENS ET 200S, SENSE DP-B-FDN-M e ALTUS DP100V. O valor do tempo de *watchdog* configurado foi de 50 ms. A partir disso, o problema foi diagnosticado através dos telegramas recebidos e respondidos pelo escravo DLG XM-210DP com o endereço 8. A Figura 6.18 apresenta a sequência de telegramas detectados para a elaboração do diagnóstico. É importante notar que estes telegramas estão filtrados, ou seja, apenas os telegramas referentes ao escravo de endereço 8 estão sendo apresentados.

Tempo	Endereço	Serviço	Dados	
1548609	1->8	Data Exchange	6804046808017D008616	ESCRAVO 8 EM TROCA DE DADOS
1548691	8->1	ACK (RS)	100108030C16	
1590186	1->8	Get Diagnostics Request	6805056888815D3C3EE016	
1590276	8->1	Slave Diagnostics Respo...	680B0B688188083E3C020500FF...	ESCRAVO 8 EM PROCESSO DE SYNC
1636938	1->8	Set Parameter Request	68EFEF6888817D3D3E8801050...	
1638743	0->0	ACK	E5	
1683852	1->8	Check Configuration Re...	6855556888815D3E3E42010001...	
1684528	0->0	ACK	E5	
1730480	1->8	Get Diagnostics Request	6805056888817D3C3E0016	ESCRAVO 8 EM TROCA DE DADOS
1730570	8->1	Slave Diagnostics Respo...	680B0B688188083E3C000C000...	
1777766	1->8	Data Exchange	6804046808015D006616	
1777848	8->1	Data Exchange	68292968010808000000000000...	ESCRAVO 8 EM TROCA DE DADOS
1829462	1->8	Data Exchange	6804046808017D008616	
1829544	8->1	ACK (RS)	100108030C16	
1870951	1->8	Get Diagnostics Request	6805056888815D3C3EE016	ESCRAVO 8 EM PROCESSO DE SYNC
1871041	8->1	Slave Diagnostics Respo...	680B0B688188083E3C020500FF...	
1917947	1->8	Set Parameter Request	68EFEF6888817D3D3E8801050...	
1919752	0->0	ACK	E5	
1964388	1->8	Check Configuration Re...	6855556888815D3E3E42010001...	
1965064	0->0	ACK	E5	ESCRAVO 8 EM TROCA DE DADOS
2011441	1->8	Get Diagnostics Request	6805056888817D3C3E0016	
2011530	8->1	Slave Diagnostics Respo...	680B0B688188083E3C000C000...	
2058164	1->8	Data Exchange	6804046808015D006616	ESCRAVO 8 EM TROCA DE DADOS
2058246	8->1	Data Exchange	68292968010808FEFFDFFFCF...	

Figura 6.18 – Telegramas do escravo DLG XM-210DP com watchdog baixo

Observando a Figura 6.18 é possível observar que o escravo DLG XM-210DP estava em troca de dados com o mestre e em determinado momento, este escravo recebeu um pedido de diagnóstico, sinalizando que o mestre está iniciando um processo de sincronização com este escravo. Este processo segue até o seu final e o escravo volta ao estado de troca de dados. Após alguns telegramas, o escravo entra novamente em processo de sincronização e em troca de dados. Estes fatos se repetem durante todo o bloco de telegramas coletado, caracterizando assim uma suspeita de que o tempo configurado para o *watchdog* está muito baixo.

A reprodução deste caso no simulador não foi possível, pois a versão atual do simulador não implementa este tipo de cenário.

g) Dispositivo entra e sai da linha esporadicamente (*Fall-outs*)

A obtenção deste diagnóstico é feita de maneira diferente das outras, pois neste caso, a análise é feita com base em dados de um período de horas ou até mesmo dias (valores configuráveis).

Para a reprodução deste caso foram utilizadas amostras coletadas esporadicamente durante um período de aproximadamente uma hora. A mesma configuração da rede Profibus apresentada no início deste capítulo foi utilizada aqui. Os telegramas foram coletados a partir da ferramenta Profitrace e exportados para arquivos de texto. Durante a coleta de dados,

alguns curtos-circuitos entre as linhas A e B foram causados a fim de emular uma interferência momentânea em um ou mais dispositivos da rede.

Com os telegramas coletados e exportados para arquivos, carregaram-se estes arquivos para o ProfiDoctor e executaram-se as análises necessárias. Por fim, o diagnóstico de que alguns dispositivos perderam a comunicação com a rede em diferentes momentos foi obtido.

Considerando que este cenário depende de muitos telegramas Profibus para a sua reprodução, apresentar o conjunto de mensagens coletadas neste documento torna-se inviável. Apesar disso, sabe-se que os curtos-circuitos causados de maneira esporádica causam a interrupção da comunicação e em seguida a retomada da mesma. Sendo assim, a Figura 6.16, já apresentada anteriormente, dá uma ideia do que cada coleta de telegramas obteve.

A partir destes cenários criados em todas as análises realizadas neste capítulo, tem-se a Tabela 6.9 que resume as principais informações analisadas na obtenção dos resultados.

Tabela 6.9 – Resumo dos cenários utilizados nas análises deste módulo

Diagnóstico	Mensagens analisadas	Origem dos dados utilizados
GSD ou endereço incorreto	<i>setParameters</i> <i>getDiagnosticResponse</i>	Dados reais e Simulador
Erro de endereçamento	<i>getDiagnosticResponse</i> <i>requestFDLResponse</i>	Dados reais e Simulador
Escravo não encontrado	<i>getDiagnosticRequest</i> <i>getDiagnosticResponse</i>	Dados reais e Simulador
Interferência momentânea	<i>retries</i> <i>getDiagnosticRequest</i> <i>getDiagnosticResponse</i>	Dados reais
Escravo perdeu a conexão e não retornou	<i>retries</i> <i>getDiagnosticRequest</i> <i>getDiagnosticResponse</i>	Dados reais
WatchDog baixo	<i>retries</i> <i>getDiagnosticRequest</i> <i>getDiagnosticResponse</i>	Dados reais
Fall-Outs	<i>retries</i> <i>getDiagnosticRequest</i> <i>getDiagnosticResponse</i>	Dados reais

6.4 Resultados relacionados à análise baseada em mensagens de diagnóstico

O diagnóstico deste módulo baseia-se em analisar alguns bits de diagnósticos nativos do protocolo Profibus. Considerando que este tipo de análise envolve apenas a implementação de verificações baseadas em especificações existentes na norma Profibus, os resultados apresentados aqui focam em testes e análises da implementação do software de interpretação de diagnósticos e no funcionamento da integração entre o ProfiDoctor e o simulador.

Sabe-se que o diagnóstico possui uma série de bits e que cada um deles representa um tipo de problema ou informação. Neste caso, um teste completo envolveria a combinação de todos estes bits em cada mensagem de diagnóstico, algo inviável e desnecessário para a ocasião. Sendo assim, algumas combinações de bits ativados foram criadas e a partir dessas combinações os resultados foram avaliados.

Antes de apresentar a tabela com os resultados de cada combinação, tem-se a Figura 6.19 que apresenta a interface gráfica do simulador onde cada bit de diagnóstico é ativado ou desativado.

Figura 6.19 – Janela de configuração de mensagens de diagnóstico do Simulador

Através desta interface, o simulador permite a criação das combinações apresentadas na Tabela 6.10. Esta tabela apresenta quais os bits estão ativados, os objetivos do teste realizado e o resultado obtido a partir do ProfiDoctor.

Tabela 6.10 – Combinação de testes para as mensagens de diagnóstico

BITS ANALISADOS	OBJETIVO	RESULTADO
TODOS OS BITS ATIVADOS	Verificar se a explicação de cada um dos bits presentes em uma mensagem de diagnóstico e a classificação do tipo de erro é apresentada corretamente.	A classificação do tipo de erro foi “crítico” e a mensagem de erro foi um texto com a explicação de cada um dos bits de diagnóstico. A grande quantidade de informação pode prejudicar a interpretação do operador da rede, mas este fato não foi levado em consideração, pois este tipo de situação não acontece em uma comunicação real.
PARAMETRIZATION FAULT, MASTER LOCK	Analisar dois tipos de erro do nível crítico acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “crítico” e a mensagem de erro foi um texto com a explicação de que algum problema na parametrização do escravo aconteceu e que a estação está sendo controlada por outro escravo.
WATCHDOG ON (desativado), STATIC DIAG.	Analisar dois tipos de erro do nível alarmante acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “alarmante” com a informação de que sem <i>watchdog</i> a planta controlada pode estar correndo riscos e que o escravo está demorando a entrar em troca de dados.
FREEZE MODE, EXTENDED DIAGNOSTICS	Analisar dois tipos de erro do nível informação acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “informação”, pois normalmente um escravo entra em modo <i>freeze</i> através de um comando enviado pelo próprio operador e neste caso, o operador provavelmente está consciente da ação que está executando. Nenhuma informação relacionada à existência de diagnóstico estendido foi apresentada,

BITS ANALISADOS	OBJETIVO	RESULTADO
		afinal, esta informação é utilizada apenas para controle do protocolo.
STATION NOT READY, CONFIGURATION FAULT	Analisar um tipo de erro crítico e um alarmante acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “crítico”, pois o escravo foi configurado de maneira equivocada. Além disso, na explicação do problema uma mensagem indicando que algum problema não está permitindo que a estação não seja inicializada é apresentada.
CONFIGURATION FAULT, FREEZE MODE	Analisar um tipo de erro crítico e uma informação acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “crítico”, o escravo foi configurado de maneira equivocada. Além disso, na explicação do problema uma mensagem informando que a rede está com seus valores congelados é apresentada.
WATCHDOG ON, FREEZE MODE	Analisar um tipo de erro alarmante e uma informação acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “alarmante” e uma mensagem alertando o operador sobre os riscos de não utilizar o <i>watchdog</i> é apresentada. Além disso, o operador é informando que a rede está com seus valores congelados.
CONFIGURATION FAULT, WATCHDOG ON, FREEZE MODE	Analisar um tipo de erro crítico , um alarmante e uma informação acontecendo na mesma mensagem de diagnóstico.	A classificação do tipo de erro foi “crítico”, pois uma falha na configuração ocorreu. Além disso, mensagens alertando sobre o risco que a rede corre ao deixar o <i>watchdog</i> desabilitado e de que todos os valores da rede estão congelados é apresentada ao operador.

Após a análise realizada e apresentada na Tabela 6.10, tem-se que o resultado esperado para este módulo foi alcançado.

Um fato importante a ser destacado aqui é que apesar deste módulo não fornecer resultados quantitativos, o mesmo é muito importante para algumas análises realizadas no módulo de análise geral descrito no item 4.7.

6.5 Resultados relacionados à análise dos campos dos telegramas

Profibus

Este módulo baseia-se apenas em verificar se determinados campos de um telegrama Profibus possuem os valores especificados pela norma Profibus. Para a análise deste módulo, é necessário que mensagens com campos inválidos sejam criadas. Para a reprodução deste caso, o mesmo procedimento adotado no item 6.3 – (d) foi utilizado, ou seja, um curto entre as linhas de dados A e B foi causado a fim de interromper a transmissão correta do sinal. A Figura 6.20 ilustra o momento em que a falha acontece.

Endereço	Serviço	Dados
1->59	Data Exchange	103B017DB916
59->1	Data Exchange	680B0B68013B0800000000000000004416
1->65	Token Passing	DC4101
-64->1	Token Passing	DC01C0
1->65	Token Passing	DC4101
1->24	Data Exchange	6805056818016D000008616
0->0	Ack	E5
1->34	Data Exchange	6804046822016D0009016
1->59	Data Exchange	103B016DA916
0->0	No Function	6A2B2B2DC4
0->0	No Function	761824088428849446A4FF

Figura 6.20 – Telegramas com valores inválidos

É possível notar a partir da Figura 6.20 que os dois últimos telegramas da análise não foram interpretados corretamente.

Quando este tipo de erro acontece, os dados dos campos dos telegramas Profibus podem assumir quaisquer valores, pois estes dependem de como a interferência agiu no sinal transmitido. Sendo assim, é difícil criar cenários específicos para a análise individual de cada campo do telegrama Profibus. Desta maneira, para a realização de testes individuais em cada campo do telegrama, utilizaram-se valores sintéticos e para cada caso, apenas o valor a ser

analisado foi modificado. A Tabela 6.11 apresenta os valores utilizados a fim de originar os erros.

Tabela 6.11 – Testes e resultados para a análise dos campos de um telegrama Profibus

CAMPO	VALORES POSSÍVEIS	VALOR UTILIZADO
SD	SD1 (10H), SC (E5H), SD2 (68H), SD4 (DCH)	11H
LE/LEr	0 até 249	250
SA	0 até 254 (0 até FEH)	255
DA	Todos são permitidos	-
FC	0 até 127 (são 7 bits, pois o primeiro é sempre 0)	150
ED	16H	17H

A conclusão obtida a partir destas análises é que apenas detectar este tipo de falha não ajuda muito o usuário na detecção da causa do problema. Sendo assim, estas informações são mais úteis no módulo de análise final (item 4.7) em que mais informações são relacionadas e novas e melhores conclusões são obtidas.

Além desta conclusão foi possível observar que quando este tipo de problema acontece por causa de uma deformação do sinal, o protocolo automaticamente resolve o problema através de um *retry* realizado no mesmo ciclo de mensagens ou com a sincronização em um próximo ciclo. Se este tipo de erro permanecer por outros ciclos, o módulo de análise baseada em histórico de mensagens vai fornecer alguma saída mais informativa ao usuário final.

6.6 Resultados da análise da temporização da rede com sistemas *fuzzy*

Basicamente, este módulo sugere um novo valor a ser configurado para o T_{TR} a partir da combinação do valor do T_{TR} originalmente configurado e dos valores do tempo real utilizados para a transmissão dos telegramas cíclicos (T_C) e acíclicos (T_A).

Para o sistema obter o valor original do T_{TR} , uma interface gráfica requisita o mesmo no início da utilização do sistema. O valor a ser inserido nesta ocasião pode ser obtido a partir de alguma equação apresentada no capítulo 4.6 ou então, utiliza-se o mesmo valor sugerido pela ferramenta de configuração utilizada pelo operador da rede Profibus. Sabe-se que existem várias ferramentas de configuração no mercado e que cada uma utiliza um método diferente para a obtenção dos valores para esta variável. Devido a este fato, para cada configuração diferente que será analisada neste módulo, os valores de T_{TR} sugeridos em diferentes ferramentas de configuração serão analisados. Além disso, o valor do T_{TR} obtido a

partir da equação 5 também é analisado. A opção pela utilização apenas da equação 5 deve-se ao fato de que este capítulo tem como objetivo a comparação dos valores obtidos pelos operadores das redes Profibus durante sua aplicação prática e, normalmente, nenhum operador calcula minuciosamente estes valores de tempo como sugere a norma Profibus. Além disso, esta equação é sugerida pelos treinamentos fornecidos pela associação Profibus.

De maneira geral, tem-se que o valor do T_{TR} sugerido pelas ferramentas ou obtido a partir da equação 5 varia de acordo com o número de escravos, o número de entradas e saídas e a velocidade da rede. A fim de ilustrar a variação do T_{TR} em função dos parâmetros citados, a Tabela 6.12 apresenta gráficos dos valores de T_{TR} sugeridos pelas principais ferramentas do mercado em função do número de escravos e do número de entradas e saídas. Esta mesma tabela apresenta também um gráfico com os valores de T_{TR} obtidos a partir da equação 5. A velocidade de transmissão utilizada na obtenção dos valores de T_{TR} é de 1,5Mbps, afinal, todas as análises realizadas nesta tese são executadas nesta velocidade.

Tabela 6.12 – T_{TR} de acordo com o número de escravos e de entradas/saídas

<p style="text-align: center;">TTR a partir do AnyBus</p>	<p>A variação do valor do T_{TR} ocorre de maneira similar à equação 5. A diferença dos dois casos é que o valor sugerido pelo AnyBus é praticamente 3 vezes maior do que o valor sugerido pela configuração.</p>
<p style="text-align: center;">TTR a partir do Hielscher Sycon</p>	<p>Diferentemente dos casos do Anybus e da equação 5, o valor do T_{TR} sugerido aqui varia apenas com o número de entradas e saídas do sistema. A variação do número de escravos não influencia em nada no parâmetro analisado. Além disso, os valores sugeridos são muito menores do que nos casos anteriores.</p>

<p style="text-align: center;">Step 7</p>	<p>Assim como no AnyBus e na equação 5, a variação do valor do T_{TR} aqui possui uma influência muito grande do número de escravos da rede. O número de entradas e saídas influencia o T_{TR} de maneira praticamente insignificante.</p>
<p style="text-align: center;">Profitrace</p>	<p>O Profitrace não leva em consideração a quantidade de entradas/saídas ou o número de escravos de uma configuração no momento de sugerir um valor de T_{TR}. Nesta ferramenta, o valor do T_{TR} sugerido é fixo (13,3ms)</p>

Além do valor do T_{TR} , o sistema *fuzzy* proposto depende dos valores de tempo utilizados na transmissão dos telegramas Profibus quando a rede está em regime de operação (T_A e T_C). Para a obtenção destes valores, as redes a serem analisadas foram montadas e, a partir do Profitrace, os telegramas transmitidos na comunicação foram capturados e armazenados em arquivos. Estes arquivos foram então exportados em forma de texto e interpretados pelo ProfiDoctor. A partir disso, os tempos gastos para a transmissão de mensagens cíclicas e acíclicas foram obtidos. Com todas estas informações registradas, o sistema *fuzzy* sugere um novo valor para o T_{TR} .

A fim de analisar o comportamento do sistema nas arquiteturas de redes mais comumente utilizadas, alguns cenários foram criados e são apresentados na Tabela 6.13.

Tabela 6.13 – Cenários para análise do sistema *fuzzy*

Cenário	Mestres	Escravos	Telegramas Acíclicos	Objetivo
A	1	3	Não	Analisar o comportamento do sistema para uma configuração básica mono-mestre diante de valores de T_{TR} sugerido pelas principais ferramentas de configuração.
B	2	5	Não	Analisar o comportamento do sistema para uma rede multi-mestre, considerando duas situações: o valor do T_{TR} igual a soma dos tempos de ciclo de cada mestre; o valor do T_{TR} igual ao tempo de ciclo de cada mestre obtido de maneira isolada.
C	1	5	Sim	Analisar o comportamento do sistema para uma configuração mono-mestre com tráfego de telegramas acíclicos diante valores de T_{TR} sugerido pelas principais ferramentas de configuração.
D	1	5	Não	Analisar o comportamento do sistema para uma configuração mono-mestre em que o valor do T_{TR} configurado está abaixo do valor do T_{RR} .

Além dos cenários apresentados na Tabela 6.13, uma análise que combina diferentes entradas foi criada sinteticamente com o objetivo de analisar todas as regras criadas para o sistema *fuzzy*. Esta última análise é apresentada no item E.

A. Configuração mono-mestre sem telegramas acíclicos

A primeira análise deu-se sobre a configuração apresentada no início deste capítulo.

O primeiro método para a obtenção do T_{TR} utilizado foi a partir da equação 5. Analisando esta configuração em detalhes tem-se que o escravo DP100V possui um módulo com duas saídas e nenhuma entrada. O escravo DP-B-FDN-M possui um módulo com oito

entradas e nenhuma saída. Por fim, o escravo ET 200S possui três módulos, sendo que, em conjunto, compõem duas entradas e uma saída. Sendo assim, tem-se que o número de escravos é 3, o número total de entradas e saídas é 13 e o T_{BIT} é de 0,6667 microssegundos, afinal, a velocidade configurada para a rede Profibus foi de 1,5Mbps.

Aplicando estes valores na equação 5 tem-se:

$$T_{MC} \approx (380 + 300 * 3 + 11 * 13) * 0.6667 \mu s + 75 \mu s \approx 1.0237 ms$$

Após isso, a configuração utilizada foi criada em algumas ferramentas de configuração Profibus e, a partir destas, o valor do T_{TR} sugerido foi obtido. A partir destes valores tem-se a Tabela 6.14.

Tabela 6.14 – T_{TR} com um mestre e 3 escravos

Mecanismo para o cálculo	T_{TR} (ms)
Equação	1,0237
AnyBus NetTool	10,1000
Hielscher Sycon	2,2920
Profitrace	13,3000
Siemens Step 7	23,3000

Cada um destes valores foram então inseridos no sistema e um novo valor para o T_{TR} foi sugerido. Além destes valores, obteve-se o valor de 1,1150 ms para a variável T_C e zero para T_A . As mensagens transmitidas e seus respectivos tempos encontram-se na Figura 6.21.

Tempo	Endereço	Serviço	Dados
353678	1->1	Token Passing	DC0101
353724	1->1	Token Passing	DC0101
353771	1->1	Token Passing	DC0101
353818	1->24	Data Exchange	6805056818017D00009616
353908	0->0	ACK	E5
353938	1->34	Data Exchange	6804046822015D008016
354215	1->34	Data Exchange	6804046822015D008016
354299	34->1	Data Exchange	6805056801220801002C16
354440	1->59	Data Exchange	103B015D9916
354492	59->1	Data Exchange	680B0B68013B0800000000000000004416
354642	1->80	Request FDL ...	105001499A16
354886	1->1	Token Passing	DC0101
354934	1->1	Token Passing	DC0101
354980	1->1	Token Passing	DC0101

Figura 6.21 – Mensagens de um ciclo de configuração com um mestre e 3 escravos

A partir de todos estes dados apresentados, os resultados para cada valor de T_{TR} utilizados no sistema são apresentados a seguir:

• **Equação:**

Utilizando-se a equação, tem-se o valor do T_{TR} de 1,0237 ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.22.

Figura 6.22 – Sistema para T_{TR} obtido a partir da equação 5 ($T_{TR} = 1,0237$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

**SE (T_C É *MUITO_ALTO*) E (T_A É *MUITO_BAIXO*)
 ENTÃO (T_{TR} DEVE SER *MUITO_ALTO*).**

Considerando que o conectivo **E** foi utilizado nas regras, o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.23.

Figura 6.23 – Resultado do operador mínimo nas regras ativas de T_C e T_A

Como apenas uma regra foi ativada, a agregação dos resultados utilizando o operador máximo obteve o mesmo resultado apresentado na Figura 6.23. Aplicando-se o centroide nesta região gráfica obteve-se então um valor de T_{TR} de 1,7873 ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} um pouco maior do que o valor obtido através da equação. Utilizando-se a equação, um valor de T_{TR} inferior ao T_{RR} foi obtido. Considerando que para o cálculo do T_{TR} recomenda-se utilizar um valor um pouco maior do que o T_{RR} , o valor obtido pelo sistema *fuzzy* pode ser considerado bom. Desta maneira, conclui-se que o sistema está bem sintonizado e pode substituir a utilização das equações e, desta maneira, diminuir a quantidade de trabalho dos desenvolvedores das redes Profibus.

- **AnyBus NetTool:**

Utilizando-se esta ferramenta, tem-se o valor do T_{TR} de 10,1000 ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.24.

Figura 6.24 – Sistema para T_{TR} obtido a partir do AnyBus NetTool ($T_{TR} = 10,10\text{ms}$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

SE (T_C É *MUITO_BAIXO*) **E** (T_A É *MUITO_BAIXO*)
ENTÃO (T_{TR} **DEVE SER** *MUITO_BAIXO*).

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.25.

Figura 6.25 – Resultado do operador mínimo (AnyBus NetTool)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de T_{TR} de 2,5144ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} praticamente duas vezes maior do que o valor do T_{RR} . O resultado acima do valor ideal deve-se ao fato de que o cálculo é realizado de maneira proporcional ao valor do T_{TR} configurado pelo usuário. Apesar disso, para este caso, o valor obtido não foi ruim se comparado aos valores sugeridos pelas ferramentas de configuração disponíveis no mercado.

Um fato importante para este cenário é que o sistema *fuzzy* criado pode ser realimentado pelo usuário com o novo valor de T_{TR} obtido a partir da primeira tentativa. Para analisar como o sistema se comporta, inseriu-se novamente o valor para o T_{TR} , sendo que desta vez, o mesmo foi de 2,5144ms. Após isso, o valor do T_{TR} foi calculado novamente e como resultado obteve-se um T_{TR} sugerido de 1,6239ms. Este segundo valor está de acordo com o esperado pelo sistema.

- **Profitrace:**

Os resultados obtidos para esta ferramenta foram muito parecidos com os resultados obtidos na AnyBus NetTool. O valor do T_{TR} sugerido pelo Profitrace foi de 13,3000 ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.26.

Figura 6.26 – Sistema para T_{TR} obtido a partir do Profitrace ($T_{TR} = 13,30\text{ms}$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

SE (T_C É *MUITO_BAIXO*) **E** (T_A É *MUITO_BAIXO*)
ENTÃO (T_{TR} **DEVE SER** *MUITO_BAIXO*).

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.27.

Figura 6.27 – Resultado do operador mínimo (Profitrace)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de T_{TR} de 3,3111ms. Este resultado reforça a conclusão de que quanto maior o valor de T_{TR} configurado pelo usuário, maior será o valor de T_{TR} sugerido pelo sistema *fuzzy*. Da mesma maneira feita nas análises dos resultados do T_{TR} sugerido pela ferramenta AnyBus, o valor do T_{TR} obtido (3,3111ms) foi inserido novamente no sistema e um resultado melhor foi atingido.

- **Hielscher Sycon:**

Utilizando-se a ferramenta da Hielscher, tem-se o valor do T_{TR} sugerido de 2,2920ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.28.

Figura 6.28 – Sistema para T_{TR} obtido a partir do Sycon (Hielscher) ($T_{TR} = 2,2920$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

**SE (T_C É BAIXO) E (T_A É MUITO_BAIXO)
ENTÃO (T_{TR} DEVE SER BAIXO).**

Considerando que o conectivo **E** foi utilizado nas regras, o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.29.

Figura 6.29 – Resultado do operador mínimo (Hielscher)

Como apenas uma regra foi ativada, a agregação dos resultados utilizando o operador máximo obteve o mesmo resultado apresentado na Figura 6.29. Aplicando-se o centroide nesta região gráfica obteve-se então um valor de T_{TR} de 1,4840 ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} muito próximo ao valor do T_{RR} obtido a partir da rede Profibus. Este valor tem uma margem de erro pequena, porém suficiente para permitir que em algumas situações de erro em que mensagens de *retry* sejam transmitidas sem prejudicar o funcionamento da rede.

- **Siemens Step 7:**

Utilizando-se a ferramenta Step 7, tem-se o valor do T_{TR} sugerido de 23,3ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.30.

Figura 6.30 – Sistema para T_{TR} obtido a partir do Step 7 ($T_{TR} = 23,3$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

**SE (T_C É *MUITO_BAIXO*) E (T_A É *MUITO_BAIXO*)
 ENTÃO (T_{TR} **DEVE SER** *MUITO_BAIXO*).**

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.31.

Figura 6.31 – Resultado do operador mínimo (Step 7)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de T_{TR} de 5,8007ms. As mesmas conclusões obtidas com os valores sugeridos pelo Profitrace e pelo AnyBus NetTool cabem aqui. Sendo assim, o valor do T_{TR} obtido (5,8007ms) foi inserido novamente no sistema o T_{TR} obtido foi de 1,4440ms.

A fim de apresentar todos os resultados obtidos de maneira compacta, tem-se a Tabela 6.15.

Tabela 6.15 – Resultados a partir de uma configuração com 1 mestre e 3 escravos

Mecanismo	Regra ativada	T_{TR} (ms) Inserido	T_{TR} (ms) Obtido	Análise
Equação	SE (T_C É MUITO_ALTO) E (T_A É MUITO_BAILO) ENTÃO (T_{TR} DEVE SER MUITO_ALTO).	1.0237	1.7873	Considerando o T_{RR} de 1.115, o resultado foi ótimo.
AnyBus NetTool	SE (T_C É MUITO_BAILO) E (T_A É MUITO_BAILO) ENTÃO (T_{TR} DEVE SER MUITO_BAILO)	10.1000	2.5144	Valor bom se comparado ao sugerido pela ferramenta, mas ainda alto se comparado ao T_{RR} . Após nova tentativa obteve-se 1.6239, que pode ser considerado ideal.
Hielscher Sycon	SE (T_C É BAILO) E (T_A É MUITO_BAILO) ENTÃO (T_{TR} DEVE SER BAILO)	2.2920	1.4840	Considerando o T_{RR} de 1.115, o resultado foi ótimo.
Profitrace	SE (T_C É MUITO_BAILO) E (T_A É MUITO_BAILO) ENTÃO (T_{TR} DEVE SER MUITO_BAILO)	13.3000	3.3111	Valor bom se comparado ao sugerido pela ferramenta, mas ainda alto se comparado ao T_{RR} . Após nova tentativa obteve-se 1.5210, que pode ser considerado ideal.
Step 7	SE (T_C É MUITO_BAILO) E (T_A É MUITO_BAILO) ENTÃO (T_{TR} DEVE SER MUITO_BAILO)	23.3	5.8007	Valor bom se comparado ao sugerido pela ferramenta, mas ainda alto se comparado ao T_{RR} . Após nova tentativa obteve-se 1.4440, que pode ser considerado ideal.

A partir desses valores pode-se concluir que para sistemas que utilizam basicamente apenas dados cíclicos este mecanismo para o cálculo do T_{TR} tem um ótimo resultado. Vale lembrar que para situações em que o operador configura a rede com valores muito maiores do que o necessário, como é o exemplo dos valores sugeridos pelas ferramentas AnyBus NetTool, Profitrace e Step7, o sistema *fuzzy* fornece como saída um valor relativamente maior do que o necessário para o bom funcionamento da rede. Apesar disso, este tipo de comportamento não interfere nos bons resultados, afinal, uma rede que originalmente seria configurada com um valor de T_{TR} de 10 a 20 vezes maior do que o valor ideal não está sendo configurada para controlar uma planta com valores de tempo críticos para seu funcionamento. Além disso, caso o operador desejar, o mesmo pode reconfigurar o valor do T_{TR} até que um valor ideal seja obtido.

Figura 6.32 – Mensagens de um ciclo de configuração com um mestre e 3 escravos

A partir de todos estes dados apresentados, os resultados para cada valor de T_{TR} utilizados no sistema são apresentados.

Analisando a primeira configuração em que o mestre Proficaptain foi utilizado tem-se que o escravo XM-210DP possui 38 entradas e uma saída, contabilizando 39 entradas/saídas. Já o escravo DP-KD-2EA-D possui 4 entradas. Por fim, o escravo da Siemens possui três módulos, sendo que, em conjunto, compõem duas entradas e uma saída.

Sendo assim, tem-se que o número de escravos desta configuração é 3 e o número total de entradas/saídas é 49. O T_{BIT} para as duas configurações utilizadas aqui é de 0,6667 microssegundos, afinal, a velocidade configurada para a rede Profibus foi de 1,5Mbps.

Aplicando estes valores na equação 5 tem-se:

$$T_{MC} \approx (380 + 300 * 3 + 11 * 49) * 0,6667 \mu s + 75 \mu s \approx 1,2877 ms$$

Com o valor para a primeira configuração obtido, tem-se que a segunda configuração foi composta por um escravo DP100V que possui um módulo com duas saídas e nenhuma entrada e um escravo DP-B-FDN-M que possui um módulo com oito entradas e nenhuma saída. Neste cenário, esta configuração contabiliza 2 escravos e 10 entradas/saídas.

Aplicando estes valores na equação 5 tem-se:

$$T_{MC} \approx (380 + 300 * 2 + 11 * 10) * 0,6667 \mu s + 75 \mu s \approx 0,8017 ms$$

A partir destes valores, o usuário que configura o valor do T_{TR} pode, de maneira equivocada, configurar a ferramenta Profitrace com o T_{TR} de 1,2877ms e o *Sycon* da Hielscher com 0,8017ms. Além deste cenário, tem-se a situação em que o usuário configura o valor do T_{TR} a partir da soma dos valores de T_{TR} obtidos em cada uma das equações, ou seja, o valor do T_{TR} a ser configurado nas ferramentas Profitrace e *Sycon* corresponde à soma de 1,2877ms com 0,8017ms, resultando em 2,0894ms. Vale destacar que este tipo de ação é considerada correta durante a configuração deste parâmetro de tempo.

Considerando estas situações, os resultados a seguir foram obtidos.

○ **Valores de T_{TR} independentes:**

Para este caso, os valores de T_{TR} a serem inseridos no sistema são de 1,2877ms e 0,8017ms. Vale lembrar que os valores de tempo obtidos da rede durante a comunicação foram de 1,570 ms para a variável T_C e zero para T_A

Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.33.

Figura 6.33 – Sistemas para T_{TR} obtidos a partir da equação 5 ($T_{TR} = 1,2877$ e $0,8017$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, para ambas as configurações, a regra ativada foi:

**SE (T_C É MUITO_ALTO) E (T_A É MUITO_BAIXO)
ENTÃO (T_{TR} DEVE SER MUITO_ALTO).**

A partir disso, o operador mínimo foi aplicado e como resultado obteve-se as regiões apresentadas na Figura 6.34.

Figura 6.34 – Resultado do operador

Como apenas uma regra foi ativada em ambos os casos, a agregação dos resultados utilizando o operador máximo obteve o resultado apresentado na Figura 6.34. Aplicando-se o centroide nestas regiões obtiveram-se então um valor de T_{TR} de 2,2515ms para a primeira configuração e de 1,4018 ms para a segunda.

Pode-se concluir então que para o primeiro caso, o sistema *fuzzy* foi capaz de fornecer um novo valor ao T_{TR} compatível com a necessidade do sistema. Para o segundo caso, um valor abaixo do esperado foi fornecido, afinal o T_{RR} do sistema é de 1,570ms. Para este segundo caso, o valor de 1,4018 ms foi novamente inserido no sistema. Nesta segunda tentativa, um valor satisfatório de 2,4373ms foi obtido.

○ Valores de T_{TR} somados:

Somando-se os dois T_{TRs} obtidos (1,2877ms e 0,8017ms) tem-se que o T_{TR} a ser configurado no sistema é de 2,0894ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.35.

Figura 6.35 – Sistema para T_{TR} obtido a partir soma ($T_{TR} = 2,0894$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

SE (T_C É BOM) E (T_A É MUITO_BAIXO)
ENTÃO (T_{TR} DEVE SER BOM).

A partir disso, o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.36.

Figura 6.36 – Resultado do operador (soma dos T_{TR})

Como apenas uma regra foi ativada, a agregação dos resultados utilizando o operador máximo obteve o mesmo resultado apresentado na Figura 6.36. Aplicando-se o centroide nesta região gráfica obteve-se então um valor de T_{TR} de 2,0841 ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} maior do que o valor do T_{RR} e menor do que o valor sugerido pela equação. Sendo assim, pode-se considerar este valor compatível com as necessidades da rede configurada.

C. Configuração mono-mestre com telegramas acíclicos

O objetivo desta análise foi observar como o sistema se comporta para obter o valor do T_{TR} quando telegramas acíclicos são transmitidas em conjunto com telegramas cíclicos. Para reproduzir este cenário em laboratório utilizou-se o mestre Proficaptain em conjunto com 5 escravos. A fim de causar o envio de mensagens acíclicas no barramento, uma operação oferecida pelo software Profitrace foi executada. Esta operação chama-se *Network scan* e quando é executada, o Proficaptain envia no barramento mensagens acíclicas do tipo *getDiagnostic*.

Os escravos utilizados foram o XM-210DP, o escravo DP-KD-2EA-D, o escravo ET 200S HighFeature, o escravo DP-B-FDN-M e o escravo da DP100V.

O primeiro método para a obtenção do T_{TR} utilizado foi a partir da equação 5. Analisando esta configuração em detalhes tem-se que o escravo DP100V possui um módulo com duas saídas e nenhuma entrada. O escravo DP-B-FDN-M da Sense possui um módulo com oito entradas e nenhuma saída. O escravo ET 200S possui três módulos, sendo que, em conjunto, possuem duas entradas e uma saída. O escravo XM-210DP possui 38 entradas e uma saída, contabilizando 39 entradas/saídas. Por fim, o escravo DP-KD-2EA-D possui 4 entradas. Sendo assim, tem-se que o número de escravos é 5, o número total de entradas e

saídas é 56 e o T_{BIT} é de 0,6667 microssegundos. Aplicando estes valores na equação 5 tem-se:

$$T_{MC} \approx (380 + 300 * 5 + 11 * 56) * 0,6667 \mu s + 75 \mu s \approx 1,7390 ms$$

Após isso, a configuração utilizada foi criada em algumas ferramentas de configuração Profibus e, a partir destas, o valor do T_{TR} sugerido foi obtido. A Tabela 6.14 apresenta estes valores.

Tabela 6.16 – T_{TR} com um mestre e 5 escravos

Mecanismo para o cálculo	T_{TR}
Equação	1,7830 ms
AnyBus NetTool	11,5000 ms
Hielscher Sycon	2,8827 ms
Profitrace	13,3000 ms
Siemens Step 7	29,6000 ms

Cada um destes valores foram então inseridos no sistema e um novo valor para o T_{TR} foi sugerido. Além destes valores, obteve-se o valor de 1,5640ms para a variável T_C e 0,0150 para T_A . As mensagens transmitidas e seus respectivos tempos encontram-se na Figura 6.37.

Tempo	Endereço	Serviço	Dados
7994416	1->1	Token Passing	DC0101
7994462	1->1	Token Passing	DC0101
7994509	1->6	Data Exchange	1006015D6416
7994562	6->1	Data Exchange	68070768010608001000102F16
7994682	1->8	Data Exchange	6804046808015D006616
7994764	8->1	Data Exchange	68292968010808FDFFFEFFDFFFCFFDFFDFFDFFDFFDFFDFFDFFDFF...
7995134	1->24	Data Exchange	6805056818017D00009616
7995224	0->0	No Function	E5
7995254	1->34	Data Exchange	6804046822017D00A016
7995338	34->1	Data Exchange	6805056801220801002C16
7995444	1->59	Data Exchange	103B017DB916
7995496	59->1	Data Exchange	680B0B68013B0800000000000000004416
7995646	1->104	Get Diagnostics Request	68050568E8816D3C3E5016
7995930	1->100	Request FDL Status	10640149AE16
7996176	1->1	Token Passing	DC0101

Figura 6.37 – Mensagens de um ciclo de configuração com um mestre e 3 escravos

A partir de todos estes dados apresentados, os resultados para cada valor de T_{TR} utilizados no sistema são apresentados a seguir:

- **Equação:**

Utilizando-se a equação, tem-se o valor do T_{TR} de 1,7390ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.38.

Figura 6.38 – Sistema para T_{TR} obtido a partir da equação 5 ($T_{TR} = 1,7390$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, as seguintes regras foram ativadas:

SE (T_C É BOM) E (T_A É MUITO_BAIXO)

ENTÃO (T_{TR} DEVE SER BOM).

SE (T_C É ALTO) E (T_A É MUITO_BAIXO)

ENTÃO (T_{TR} DEVE SER ALTO).

Considerando que o conectivo **E** foi utilizado nas regras, o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.39.

Figura 6.39 – Resultado do operador mínimo ($T_{TR} = 1,7390$)

Considerando que duas regras foram ativadas, uma agregação dessas regras utilizando o operador máximo foi executada e obteve-se a região apresentada no gráfico da Figura 6.40.

Figura 6.40 – Região obtida a partir da agregação ($T_{TR} = 1,7390$)

Aplicando-se o centroide nesta região obteve-se então um valor de T_{TR} de 2,327 ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} muito próximo ao valor do T_{RR} obtido a partir da rede Profibus. Este valor tem uma margem de erro suficiente para permitir que em algumas situações de erro telegramas de *retry* ou outros telegramas assíncronos sejam transmitidas sem prejudicar o funcionamento da rede.

- **AnyBus NetTool:**

Utilizando-se esta ferramenta, tem-se o valor do T_{TR} de 11,5000 ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.41.

Figura 6.41 – Sistema para T_{TR} obtido a partir do AnyBus NetTool ($T_{TR} = 11,5500\text{ms}$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

**SE (T_C É *MUITO_BAIXO*) E (T_A É *MUITO_BAIXO*)
 ENTÃO (T_{TR} **DEVE SER** *MUITO_BAIXO*).**

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.42.

Figura 6.42 – Resultado do operador mínimo (AnyBus NetTool)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de T_{TR} de 2,8754ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} praticamente 50% maior em relação ao valor do T_{RR} . Apesar disso, para este caso, o valor obtido foi muito bom se comparado aos valores sugeridos pelas ferramentas de configuração disponíveis no mercado e da uma margem razoável para permitir a transmissão de outros telegramas assíncronos.

- **Profitrace:**

O valor do T_{TR} sugerido pelo Profitrace foi exatamente o mesmo sugerido para a configuração em que 3 escravos foram utilizados. Sendo assim os mesmos sistemas *fuzzy* da Figura 6.26 foram utilizados.

Assim como no exemplo para 3 escravos, para este caso, os valores de T_C e T_A também foram considerados MUITO_BAIXO, sendo que o grau de pertinência destes valores neste conjunto foi de 1. Sendo assim, o valor do T_{TR} sugerido nesta situação também foi de 3,3111ms.

- **Hielscher Sycon:**

Utilizando-se a ferramenta da Hielscher, tem-se o valor do T_{TR} sugerido de 2,8827ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.43.

Figura 6.43 – Sistema para T_{TR} obtido a partir do Sycon ($T_{TR} = 2,8827$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, as seguintes regras foram ativadas:

SE (T_C É BAIXO) E (T_A É MUITO_BAIXO)

ENTÃO (T_{TR} DEVE SER BAIXO).

SE (T_C É BOM) E (T_A É MUITO_BAIXO)

ENTÃO (T_{TR} DEVE SER BOM).

O operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.44.

Figura 6.44 – Resultado do operador mínimo ($T_{TR} = 2,8827$)

Neste caso duas regras foram ativadas. Sendo assim, uma agregação dessas regras utilizando o operador máximo foi aplicada e obteve-se a região apresentada no gráfico da Figura 6.45.

Figura 6.45 – Região obtida a partir da agregação utilizando o operador máximo.

Aplicando-se o centroide nesta região obteve-se então um valor de T_{TR} de 2,7783 ms.

- **Siemens Step 7:**

Utilizando-se esta ferramenta, tem-se o valor do T_{TR} de 29,6000 ms. Inserindo-se estes valores no sistema ProfiDoctor, obtêm-se os sistemas *fuzzy* apresentados na Figura 6.46.

Figura 6.46 – Sistema para T_{TR} obtido a partir do Step 7 ($T_{TR} = 23,3\text{ms}$)

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

**SE (T_C É *MUITO_BAIXO*) E (T_A É *MUITO_BAIXO*)
 ENTÃO (T_{TR} **DEVE SER** *MUITO_BAIXO*).**

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.47.

Figura 6.47 – Resultado do operador mínimo (Step 7)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de T_{TR} de 6,6223ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} bem mais próximo ao ideal de comparado ao T_{TR} sugerido pelo Step 7, mas mesmo assim este valor pode ser considerado alto em relação ao valor do T_{RR} . Neste caso inseriu-se novamente o valor para o T_{TR} obtido (6,6223ms). Após isso, o valor do T_{TR} foi calculado novamente e como resultado obteve-se um T_{TR} sugerido de 2,2214ms. Este segundo valor está de acordo com o esperado pelo sistema.

A fim de apresentar todos os resultados obtidos de maneira compacta, tem-se a Tabela 6.17.

Tabela 6.17 – Resultados (1 mestre, 5 escravos e telegramas assíncronos)

Mecanismo	Regras ativas	T_{TR} (ms) Inserido	T_{TR} (ms) Obtido	Análise
Equação	SE (T_C É BOM) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER BOM). SE (T_C É ALTO) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER ALTO).	1.7390	2.3270	Considerando o T_{RR} de 1.714, o resultado foi ótimo.
AnyBus NetTool	SE (T_C É MUITO_BAIXO) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER MUITO_BAIXO)	11.5500	2.8754	Considerando o T_{RR} de 1.714, o resultado foi ótimo.
Hielscher Sycon	SE (T_C É BAIXO) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER BAIXO). SE (T_C É BOM) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER BOM)	2.8827	2.7783	Considerando o T_{RR} de 1.714, o resultado foi ótimo.
Profitrace	SE (T_C É MUITO_BAIXO) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER MUITO_BAIXO)	13.3000	3.3111	Considerando o T_{RR} de 1.714, o resultado pode ser considerado bom.
Step 7	SE (T_C É MUITO_BAIXO) E (T_A É MUITO_BAIXO) ENTÃO (T_{TR} DEVE SER MUITO_BAIXO)	29.6	6.6223	Valor bom se comparado ao sugerido pela ferramenta, mas ainda alto se comparado ao T_{RR} . Após nova tentativa obteve-se 2.2214, que pode ser considerado ideal.

valor final do T_{RR} obtido. Assim, o valor desses parâmetros será obtido corretamente mesmo em situações como esta.

A partir disso, foi obtido o valor de 1,432ms para T_C e de zero para T_A . Com o valor do T_{TR} configurado, os sistemas *fuzzy* foram construídos. A Figura 6.49 apresenta os mesmos.

Figura 6.49 – Sistemas *fuzzy* para T_{TR} baixo

Com os conjuntos *fuzzy* criados, inseriu-se no sistema o valor de T_C e T_A obtidos a partir da rede Profibus. Assim sendo, a seguinte regra foi ativada:

SE (T_C É *MUITO_ALTO*) E (T_A É *MUITO_BAIXO*)
ENTÃO (T_{TR} DEVE SER *MUITO_ALTO*).

A partir disso o operador mínimo foi aplicado e como resultado obteve-se a região apresentada na Figura 6.50.

Figura 6.50 – Resultado do operador mínimo (AnyBus NetTool)

Aplicando-se o centroide nesta região gráfica obteve-se então um valor defuzzificado de 0,8742ms.

A partir deste valor é possível observar que o sistema criado obteve um valor de T_{TR} ainda abaixo do necessário para seu bom funcionamento. Tal fato acontece devido aos sistemas *fuzzy* criados serem proporcionais ao valor do T_{TR} configurado pelo usuário. Desta maneira, quando um usuário configurar o T_{TR} com um valor muito abaixo do ideal, recomenda-se que o valor resultante gerado seja inserido novamente no sistema. Após realizar esta prática, obteve-se o valor de 1,5285 que pode ser considerado correto em uma comunicação sem problemas de interferência e sem a transmissão de telegramas acíclicos.

E. Testes para análise de todas as regras implementadas

Neste item são apresentados os resultados dos testes realizados com o objetivo de validar o funcionamento de todas as regras criadas para o sistema *fuzzy* desenvolvido nesta tese. Para tal teste, um valor de T_{TR} de 2ms foi utilizado. É importante lembrar que os sistemas *fuzzy* são construídos de maneira proporcional ao valor de T_{TR} inserido pelo usuário e, desta maneira, qualquer valor de T_{TR} é considerado válido para a execução deste teste.

A partir dessa informação, os sistemas *fuzzy* são criados. A Figura 6.51 apresenta os mesmos.

Figura 6.51 – Sistemas *Fuzzy* para o T_{TR} configurado em 2ms.

A partir destes sistemas, diferentes valores de T_A e T_C foram utilizados. Estes valores foram selecionados de forma que os mesmos ativassem cada uma das regras criadas para este sistema. A Tabela 6.18 apresenta cada valor inserido, o valor linguístico das variáveis T_A e T_C , o valor linguístico sugerido para a variável T_{TR} e, por fim, o valor em milissegundos sugerido para o T_{TR} .

Tabela 6.18 – Análise detalhada de cada cenário possível do sistema *fuzzy*.

T_C	T_A	T_{TR} deve ser	T_{TR} sugerido
350ms (MB)	150ms (MB)	MB	490ms
350ms (MB)	500ms (BA)	MB	490ms
350ms (MB)	800ms (BO)	BA	1290ms
350ms (MB)	1200ms (AL)	BA	1290ms
350ms (MB)	1600ms (MA)	BO	1990ms
900ms (BA)	150ms (MB)	BA	1290ms
900ms (BA)	500ms (BA)	BA	1290ms
900ms (BA)	800ms (BO)	BA	1290ms

T_C	T_A	T_{TR} deve ser	T_{TR} sugerido
900ms (BA)	1200ms (AL)	BO	1990ms
900ms (BA)	1600ms (MA)	BO	1990ms
1500ms (BO)	150ms (MB)	BO	1990ms
1500ms (BO)	500ms (BA)	BO	1990ms
1500ms (BO)	800ms (BO)	BO	1990ms
1500ms (BO)	1200ms (AL)	BO	1990ms
1500ms (BO)	1600ms (MA)	AL	2690ms
1900ms (AL)	150ms (MB)	AL	2690ms
1900ms (AL)	500ms (BA)	AL	2690ms
1900ms (AL)	800ms (BO)	AL	2690ms
1900ms (AL)	1200ms (AL)	MA	3490ms
1900ms (AL)	1600ms (MA)	MA	3490ms
2400ms (MA)	150ms (MB)	MA	3490ms
2400ms (MA)	500ms (BA)	MA	3490ms
2400ms (MA)	800ms (BO)	MA	3490ms
2400ms (MA)	1200ms (AL)	MA	3490ms
2400ms (MA)	1600ms (MA)	MA	3490ms

Assim como no item C, alguns resultados apresentados na Tabela 6.18 não são satisfatórios. Nestes cenários, o valor do T_{TR} sugerido pelo sistema *fuzzy* deve ser inserido novamente no sistema. Em todas as ocasiões, nesta segunda tentativa, um valor ideal para este parâmetro foi obtido.

A partir destas análises, é possível concluir que o sistema dá uma prioridade muito pequena para a transmissão de mensagens acíclicas, afinal, a mudança do valor de T_A na maioria das vezes não afeta o valor de T_{TR} calculado pelo sistema. Esta solução prioriza sempre o envio de mensagens cíclicas.

6.7 Resultados da análise baseada em resultados dos sistemas especialistas

Este módulo recebe os resultados das análises realizadas pelos outros módulos descritos neste capítulo e processa essas informações a fim de obter uma saída mais compacta e de obter diagnósticos mais precisos de determinados problemas. A obtenção dos resultados aqui baseou-se na coleta de dados a partir da ferramenta Profitrace e do osciloscópio. A partir disso, um arquivo de entrada para o ProfiDoctor foi criado. Este arquivo possuía os telegramas Profibus em formato hexadecimal, o tempo em microssegundos que o telegrama foi enviado e as amostras de um bit Profibus deste telegrama.

O primeiro cenário a ser analisado é quando os módulos de RNA ou de níveis de tensão detectam deformações de bits Profibus e ao mesmo tempo o módulo que analisa os campos do telegrama Profibus indica a presença de telegramas corrompidos. Para analisar tal caso utilizou-se a configuração de rede apresentada no início deste capítulo (mestre Proficaptain + 3 escravos de endereço 24, 34 e 59). Além disso, utilizou-se um cabeamento de 1000m para conectar o escravo 59. Desta maneira, todas as vezes que um telegrama foi enviado para este escravo, os sinais ficaram com a deformação típica de um cabeamento longo e consequentemente a RNA detectou e classificou o problema. Ao mesmo tempo, o módulo de análise dos campos dos telegramas Profibus detectou valores indevidos e apontou erro. Desta maneira, o módulo descrito neste item uniu as duas saídas e apresentou um diagnóstico de **erro** informando que a causa do problema foi a utilização de cabeamento longo. Vale destacar que no caso da RNA detectar um problema de cabeamento longo, mas o módulo de análise dos campos do telegrama Profibus não diagnosticar nenhum problema, apenas um aviso informando ao usuário que a rede pode apresentar problemas devido à utilização de cabeamento longo é apresentado. A Figura 6.52 apresenta a amostra do sinal utilizada pela RNA, os telegramas analisados e o diagnóstico obtido a partir destes dados.

Figura 6.52 – Análise com base em sinais e nos campos do telegrama Profibus

Outro cenário a ser analisado aqui é o caso em que o módulo de análise de níveis de tensão fornece um diagnóstico de que um determinado escravo está com problemas e ao mesmo tempo, o módulo de RNA informa que existe um problema de cabeamento longo na rede. Para analisar tal caso utilizou-se a mesma rede utilizada no exemplo anterior. Desta

maneira, todas as vezes que um telegrama foi enviado para o escravo de endereço 59, os sinais ficaram com a deformação típica de um cabeamento longo e consequentemente a RNA detectou e classificou o problema. Ao mesmo tempo, o módulo que realiza a análise com base no nível de tensão indicou que o escravo 59 estava com o nível médio de tensão em 1,05 Volts e que consequentemente estava com problemas de driver ou de fonte, afinal, este era o único escravo da rede com nível de tensão abaixo de 4 Volts. Nota-se então que este último diagnóstico está equivocado. Caso o módulo de RNA não detectasse nenhum problema do sinal e este segundo módulo fornecesse o diagnóstico descrito, o diagnóstico final do sistema seria de que o escravo está com problemas. Considerando que além deste diagnóstico o módulo de RNA detectou um problema de cabeamento longo, o diagnóstico final foi apenas o de cabeamento longo. A Figura 6.53 apresenta a amostra do sinal utilizada pela RNA e pelo módulo de análise de níveis de tensão.

Figura 6.53 – Módulo de RNA com módulo de análise de níveis de tensão

Além dessas análises específicas, vários testes envolvendo a detecção de erros apresentados na Tabela 4.4 foram realizados. Em todos os cenários testados obteve-se a saída esperada para o sistema.

6.8 Conclusões obtidas a partir dos resultados

Com o objetivo de fornecer uma visão geral dos resultados e das análises realizadas nesta tese, tem-se a Tabela 6.19 com os resultados da análise dos sinais realizada através de RNAs, a Tabela 6.20 com os resultados da análise do sinal através do nível de tensão, a Tabela 6.21 com os resultados da análise realizada a partir do histórico de mensagens Profibus e por fim, a Tabela 6.22, com os resultados da análise do T_{TR} a partir de sistemas *fuzzy*.

Tabela 6.19 – Resultados da análise através de RNA

Análise do sinal através de RNA		
Cenário	Procedimento	Conclusões
Análise da variância	Analisou-se a variância de todos os tipos de sinais gerados de maneira sintética e real	Em todos os casos, mesmo com a modificação das formas de onda dos sinais analisados, este método mostrou-se eficaz e separou corretamente os sinais com EMI e com cabeamento longo dos outros sinais analisados.
Uma RNA para analisar todos os sinais	Utilizou-se apenas uma RNA e todas as amostras de sinais foram consideradas no treinamento da mesma.	Resultados ruins, sendo que o melhor caso foi em uma RNA com 20 neurônios na camada escondida (14,81% de erro).
Uma RNA para análise de cada sinal	Utilizou-se uma RNA para classificar isoladamente cada sinal. Vale lembrar que cada RNA possui amostras de todos os tipos de sinais.	Algumas RNAs apresentaram taxas de erro de 3,40% ou 7,40%. Outras apresentaram 100% de acerto.
Sete RNAs para análise de cada sinal sem amostras com EMI e uma RNA para analisar sinais com EMI e cabo longo	Utilizou-se uma RNA para classificar isoladamente cada sinal. Neste caso, os treinamentos e testes foram feitos sem considerar as amostras com EMI.	Todas RNAs apresentaram taxas de erro de 0%.
RNAs com sinais divididos em 3 grupos	Agrupou-se os sinais em 3 grupos diferentes. O primeiro grupo com 4 tipos de sinais e 4 RNAs para classificação, o segundo com 3 sinais e 3 RNAs e o terceiro grupo com uma RNA para diferenciar sinais com EMI e cabeamento longo	Ótimos resultados. Todas RNAs apresentaram 100% de acerto. Neste caso, os resultados foram ainda melhores do que os apresentados no cenário da linha anterior, pois os MSEs medidos foram menores.

A partir da Tabela 6.19, confirma-se a tese de que redes neurais artificiais são ferramentas extremamente eficazes para a análise de padrões, sendo que no caso desta tese, os

padrões analisados foram os formatos de onda dos sinais transmitidos pelo protocolo Profibus DP. Logicamente, antes de apresentar as amostras dos sinais para as RNAs, um pré-processamento foi necessário e a análise da variância dos sinais apresentou-se como uma ótima opção para a realização desta tarefa.

Tabela 6.20 – Resultados da análise através do nível de tensão

Análise do sinal através do nível de tensão		
Cenário	Procedimento	Conclusões
Rede operando de maneira correta	Utilizaram-se três dispositivos, sendo que todos estavam com o nível de tensão nos padrões corretos (4V e 7V)	O sistema não apresenta diagnóstico, afinal, está tudo funcionando corretamente.
Curto-circuito entre as linhas de dados	Os cabos de dados A e B foram colocados em curto circuito	Nível de tensão fica em zero e o sistema sugere que o problema pode ser um curto circuito nas linhas de dados.
Cabeamento longo ou problemas no aterramento	Dois testes utilizando cabeamento com 1000m: 1 escravo ligado por este cabeamento; 2 escravos ligados por este cabeamento.	Falha para o caso de apenas um escravo estar com nível de tensão baixo. O diagnóstico é de que o problema está no escravo e não na instalação física da rede. Análise final corrige isso.
Problemas com a fonte ou com o driver do dispositivo	Dois cenários analisados: 1 escravo com tensão de 8 Volts; 1 escravo teve sua fonte desligada.	Ótimos resultados. Sistema apresenta diagnóstico de problemas com a fonte. Lembrando que mais de um dispositivo com nível de tensão abaixo de 4V leva o sistema a indicar cabeamento longo.
Cabeamento quebrado	Roupeu-se o cabeamento da rede e dois testes foram feitos: 1 escravo ligado após a ruptura; 2 escravos ligados após a ruptura	As mesmas conclusões obtidas para o caso do cabeamento longo foram obtidas aqui.

Analisando os resultados apresentados na Tabela 6.20, nota-se que alguns diagnósticos fornecidos por este módulo aparecem em forma de sugestões, pois os sintomas analisados não garantem que um determinado problema está ou não acontecendo. Além disso, o sistema classifica de maneira equivocada alguns cenários, mas este tipo de falha não é um problema de grande relevância, afinal, unindo-se as informações destas análises com as análises do módulo de RNA, um diagnóstico mais preciso pode ser alcançado.

Tabela 6.21 – Resultados da análise através dos históricos de mensagens

Análise baseada nos históricos de mensagens		
Cenário	Procedimento	Conclusões
Problema de parametrização ou de endereçamento	Utilizou-se um GSD incorreto no momento da configuração	Sistema indica o problema e evita que o usuário receba apenas a notificação de erro de parametrização.
Erro de endereçamento	Um escravo teve seu endereço modificado fisicamente, deixando-o com valor diferente do configurado <i>off-line</i> .	O escravo com endereço modificado não respondeu aos <i>getDiag.</i> e um outro escravo respondeu aos <i>requestFDL</i> . O sistema sugere que o problema tem relação com endereçamento.
Escravo não encontrado	Mesmo cenário do erro de endereçamento. A diferença é que apenas os escravos configurados no mestre ficam ativos na linha.	O sistema indica a falta de um escravo, colaborando para que o usuário perceba rapidamente que algum escravo configurado no mestre não está ativo na rede.
Interferência momentânea	Um curto entre as linhas de dados A e B foi causado por um intervalo de tempo mínimo	Comunicação foi perdida e retomada de imediato. Este problema é de difícil detecção por parte do usuário e por isso, o grau de importância deste diagnóstico é muito alto.
Escravo perdeu a conexão e não retornou para a rede	Fonte de energia de um escravo foi desligada com a rede em regime de operação	Diagnóstico apenas informativo, pois nenhum detalhe da causa do motivo do problema é fornecida.
WatchDog Baixo	Configuração com 5 escravos e <i>watchdog</i> configurado de 50ms	Um escravo entra e sai de troca de dados. Somente um usuário experiente, analisando todos os frames notaria isso.
Dispositivo entra e sai da linha esporadicamente (Fall-outs)	Análise de amostras coletadas esporadicamente durante aproximadamente uma hora. Neste período, alguns curtos-circuitos foram causados.	Este problema é de difícil detecção por parte do usuário e por isso, o grau de importância deste diagnóstico é muito alto.

Na Tabela 6.21 são apresentados diagnósticos de muita importância e utilidade, pois a detecção de alguns dos problemas apresentados sem a utilização deste tipo de ferramenta torna-se praticamente impossível, como por exemplo, os casos em que um dispositivo entra e sai da linha de forma esporádica. Esta dificuldade acontece por causa da enorme quantidade de dados transmitidos em curtos períodos de tempo. Além destes problemas de difícil detecção, o sistema indica também alguns problemas mais simples, como o caso do “escravo não encontrado”. Este tipo de diagnóstico, apesar de ser simples, também colabora muito na detecção da origem de um problema no momento da instalação de uma rede Profibus.

Tabela 6.22 – Resultados da análise do T_{TR} através de sistema *Fuzzy*

Análise do T_{TR} através de sistemas <i>FUZZY</i>		
Cenário	Procedimento	Conclusões
1 Mestre Proficaptain + 3 escravos	Análises comparativas com as principais ferramentas do mercado e com a equação 5.	Ótimos resultados para a maioria dos casos, sendo que alguns casos um valor de T_{TR} acima do necessário foi obtido e uma realimentação do sistema foi necessária para o resultado ideal ser alcançado.
1 Mestre Proficaptain + 3 escravos + 1 Mestre DF73 + 2 escravos	Análise dos valores de T_{TR} calculados a partir da equação 5, sendo que em um caso o cálculo foi feito considerando a soma do tempo de ciclo dos dois mestres e no outro caso os tempos de ciclo foram analisados isoladamente	Ferramenta mostrou-se eficaz, indicando um valor ideal para todos os cenários, sendo que para o cenário onde o mestre com menos escravos foi analisado de maneira isolada, um valor de T_{TR} menor do que o T_{RR} foi sugerido, fazendo que o resultado ideal fosse obtido a partir da realimentação do sistema <i>fuzzy</i> .
Mestre Proficaptain + 5 escravos + telegramas acíclicos	Análises comparativas com as principais ferramentas do mercado e com a equação 5.	Ótimos resultados para a maioria dos casos, sendo que alguns casos um valor de T_{TR} acima do necessário foi obtido e uma realimentação do sistema foi necessária para o resultado ideal ser alcançado.
1 Mestre Proficaptain + 5 escravos + T_{TR} muito baixo	Análise através de um valor de T_{TR} muito abaixo do valor do T_{RR}	Pelo fato do valor ser muito baixo, o T_{TR} sugerido continuou abaixo do necessário. Um valor ideal foi obtido após a realimentação do sistema com o valor do T_{TR} sugerido na primeira iteração.
Testes para análise de todas as regras implementadas	Análise para validar o funcionamento de cada regra do sistema	Para praticamente todos os casos o valor sugerido foi ideal. Notou-se através destes testes que o sistema <i>fuzzy</i> implementado prioriza a troca de telegramas cíclicos.

Uma conclusão geral permitida a partir dos resultados apresentados na Tabela 6.22 é a de que este tipo de sistema mostrou-se muito útil durante a elaboração da configuração de uma rede Profibus, afinal, o mesmo elimina a necessidade do usuário de calcular o valor do T_{TR} . Além disso, notou-se que a maioria das ferramentas de configuração sugere um valor de T_{TR} muito acima do necessário, negligenciando assim as necessidades temporais de uma planta industrial. Um ponto negativo que pode ser destacado é que a rede precisa entrar em operação para que o valor do T_{TR} seja calculado pelo sistema *fuzzy*. Devido a isso, em caso de um eventual ajuste do valor do mesmo, a nova configuração precisa ser descarregada novamente para a rede Profibus. Sendo assim, este tipo de sistema é indicado para ser utilizado nos primeiros momentos do funcionamento de uma rede, evitando assim a interrupção do funcionamento de uma planta industrial.

Sendo assim, após analisar cada módulo desenvolvido, conclui-se que a utilização dos sistemas inteligentes propostos atingiram os resultados esperados. Outras conclusões relacionadas a esta tese são apresentadas no capítulo 7.

7 Conclusões

O Profibus é um padrão aberto de rede de campo independente de fornecedores. Estas redes apresentam uma alta sensibilidade a falhas relacionadas à sua instalação física. Inobservância a requisitos técnicos podem afetar o desempenho da comunicação e alterar características elétricas do sistema de transmissão, degradando assim o sinal a ser transmitido. Além disso, erros durante o processo de configuração prejudicam o funcionamento dessas redes, causando uma série de prejuízos.

Neste contexto esta tese apresentou a proposta de uma ferramenta que, através de tecnologias como Redes Neurais Artificiais, sistemas nebulosos e sistemas especialistas baseados em regras, procura automaticamente fazer um diagnóstico relacionado a problemas de instalação e configuração de uma rede Profibus.

A utilização de redes neurais artificiais para a classificação dos sinais mostrou-se bastante versátil e robusta, proporcionando excelentes resultados. Através das RNAs, foi possível o diagnóstico dos problemas de instalação que causam distúrbios na forma de onda dos sinais transmitidos na camada física do protocolo Profibus DP, garantindo-se assim, a viabilidade do projeto proposto nesta tese. Outra questão observada foi a capacidade de generalização destas redes, afinal, uma série de padrões foram criados sinteticamente e em todos os casos as RNAs classificaram corretamente os problemas apresentados. Logicamente, sabe-se que quando a análise é feita em outras redes Profibus um domínio maior de formatos de sinais e problemas podem ser encontrados, pois, o domínio desses dados em laboratório é limitado. Assim sendo, mesmo com a capacidade de generalização das RNAs, o retreinamento das RNAs utilizadas no sistema com os formatos de sinais obtidos em outras redes pode ser necessário. Devido a este fato, um especialista em redes Profibus dificilmente poderá ser dispensado, pois uma operação de retreinamento de uma RNA necessita da entrada de novas informações (como a classificação de um novo formato de sinal). Sendo assim, este sistema pode ser melhor aproveitado quando utilizado para o apoio a decisões dos operadores das redes e até mesmo dos especialistas. Foi possível também constatar a contribuição que um sistema especialista oferece, afinal, analisar a grande quantidade de telegramas transmitidos durante a comunicação e encontrar possíveis erros é uma tarefa que exige muita experiência

de um usuário. Por fim, a utilização dos modelos de inferência *fuzzy* com regras desenvolvidas a partir de conhecimentos de especialistas e da observação do comportamento das redes Profibus se mostrou bastante apropriado para a definição de um valor para o T_{TR} mais próximo do T_{RR} .

Considerando que os resultados de parte desta tese foram analisados a partir de dados gerados sinteticamente, conclui-se também que a utilização dessas informações permitem simular diferentes configurações de rede, possibilitando assim a realização de testes que demandariam muito tempo e esforço se fossem realizados a partir de uma rede real. A flexibilidade proporcionada pelo simulador permite avaliar o desempenho das redes segundo a variação dos equipamentos na configuração do mestre e dos parâmetros de rede. Além disso, este tipo de simulação permite que treinamentos de instalação e manutenção deste tipo de rede sejam fornecidos sem a necessidade de equipamentos reais.

Outra conclusão importante está relacionada à análise dos sinais da camada física Profibus. Considerando que os dados são transmitidos no padrão EIA-485, o uso deste tipo de sistema pode ser considerado para todas as redes que se comunicam desta forma. Para tal generalização, apenas alguns detalhes relacionados à comunicação do hardware de coleta dos dados com o ProfiDoctor devem ser ajustados.

Por fim, pode-se destacar que este tipo de sistema fornece como benefício a redução do tempo de instalação e de manutenção nas plantas industriais, afinal, o mesmo detecta os sintomas de uma falha e indica as possíveis causas da mesma.

Assim sendo, os objetivos desta tese foram atingidos, deixando como legado, uma ferramenta que poderá contribuir para a diminuição da dependência dos sistemas de automação industrial em relação aos seus operadores através da utilização de sistemas computacionais inteligentes como as RNAs, os sistemas especialistas e os sistemas nebulosos.

Além de atingir os objetivos almejados, o trabalho criou uma série de possibilidades de melhorias em alguns módulos já desenvolvidos do sistema além da criação de novas características que possibilitarão aos usuários novas possibilidades. Desta maneira, pode-se destacar:

- **Treinamento da Rede Neural:** como foi destacado no decorrer deste documento, o treinamento da rede neural utilizada na classificação dos sinais foi realizado a partir de amostras coletadas em laboratório com a velocidade de 1,5Mbps. Sabe-se que conforme a velocidade da rede aumenta, o formato do

sinal sofre deformações. Neste contexto, um trabalho futuro seria a coleta de dados em outras velocidades e, em seguida, o treinamento das RNAs.

- **Novos métodos para análise do sinal:** nesta tese, apenas uma maneira de analisar e classificar um sinal Profibus foi utilizada. Existem muitos outros mecanismos para realizar este tipo de análises. No Laboratório de Automação Industrial da Escola de Engenharia de São Carlos, já está em fase de estudos e implementação a realização do pré-processamento do sinal com uma técnica denominada Clipper. Neste mesmo laboratório, também está em fase de estudos uma análise do sinal através do valor quadrático médio das amostras dos sinais.
- **Uso de GSD:** Como explanado no item 4.4, um telegrama Profibus pode conter informações relacionadas a diagnóstico nativas do próprio protocolo. Alguns bytes de informações são obrigatórios nesses telegramas, mas além desses bytes, informações de diagnóstico estendidas podem estar presentes. A interpretação dessas informações dependem dos arquivos GSD de um determinado escravo. Este trabalho não contemplará tal interpretação. Assim sendo, tem-se a sugestão de um novo trabalho para a implementação de tal solução.
- **Hardware:** como foi descrito neste trabalho, sabe-se que detalhes de como o hardware será desenvolvido já foram especificados, mas a implementação do mesmo está sendo realizada em um trabalho de mestrado orientado pelo mesmo professor orientador do projeto descrito neste documento. Sendo assim, como trabalho futuro tem-se a finalização do hardware para posterior integração do mesmo com o ProfiDoctor.

Referências Bibliográficas

ALTROCK, C. V.; GEBHARDT, J. (1996). Recent successful fuzzy logic applications in industrial automation. In: FIFTH IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS, New Orleans, LA. **Proceedings...** New York: IEEE, 1996. p. 1845-1851. v.3.

AMAYA, J.E.; ALVARES, J.A. (2010). SIMPREBAL: An Expert System for Real-Time Fault Diagnosis of Hydrogenerators Machinery. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA), Bilbao, SPAIN. **Proceedings...** New York: IEEE, 2010. p. 1-8.

AMBRÓSIO, P. E. **Redes neurais artificiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmonares**. 2002. Dissertação (Mestrado em Física Aplicada à Medicina e Biologia) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2002. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/59/59135/tde-26102002-155559/>>. Acesso em: 2012-05-08.

AMINIAN, M., AMINIAN, F. (2000). Neural Network Based Analog-Circuit Fault Diagnosis Using Wavelet Transform as Preprocessor. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v. 47, n. 2, p.151-156, 2000.

ANDRADE, A. A. **Desenvolvimento de sistema especialista com operacionalidade de aprendizado para operar em tempo real com sistemas industriais automatizados**. 2007. Tese (Doutorado em Sistemas de Potência) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/3/3143/tde-28032008-180226/>>. Acesso em: 2012-07-10.

ANGELI, C. (2008), Online expert systems for fault diagnosis in technical processes. **Expert Systems**, v.25, n.2, p.115–132. doi: 10.1111/j.1468-0394.2008.00442.x.

ANGRISANI, L., DAPONTE, P., D'APUZZO, M. (2001). Wavelet Network-Based Detection and Classification of Transients. **IEEE Transactions on Instrumentation and Measurement**, v. 50, n. 5, p. 1425-1435, 2001.

ANTONPOULOS, C. P., V. KAPSALIS, S. A. KOUBIAS. (2004). An Efficient Simulator for the PROFIBUS-DP MAC-Layer Protocol. In: IFAC CONFERENCE ON MANUFACTURING, MODELLING, MANAGEMENT AND CONTROL (MiM'04), 2004, Athens, GREECE, **Proceedings of the IFAC Conference**. 2004. p.21-22.

AWADALLAH, M.A., MORCOS, M.M. (2003). Application of AI tools in fault diagnosis of electrical machines and drives-an overview. **IEEE Transactions on Energy Conversion**. v.18, n.2, pg.245-251, 2003.

BABUŠKA, R., **Fuzzy Systems, Modeling and Identification**, Technical report, Delft University of Technology, 2001. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.9152&rep=rep1&type=pdf>>. Acesso em: 20/05/2012.

BAO, W., ZHANG, H., LI, H., HUANG, W., PENG, D. (2009). Analysis and Research on the Real-Time Performance of Profibus Fieldbus. In: WRI World Congress on Software Engineering 2009, WCSE 2009, Xiamen, CHINA, 2009. **Proceedings of WRI World Congress**, p.136-140.

BÁRDOSSY, A; DUCKSTEIN, L. **Fuzzy rule-Based Modeling with Applications to Geophysical, Biological and Engineering Systems**. CRC Press Taylor & Francis Group. USA, 1995.

BECRAFT, W. R. (1993). Diagnostic Applications of Artificial Neural Networks. In: International Joint Conference on Neural Network. Nagoya, JAPAN, 1993. **Proceedings...** New York: IEEE, 1993. p.2807-2810. v.3.

BOLAÑOS, J.R.M. (2007). **Controlador supervisorio inteligente para sistemas híbridos eólico-diesel-bateria de pequeno porte**. 2007. Dissertação (Mestrado em Sistemas de Potência) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/3/3143/tde-02042008-095246/>>. Acesso em: 2012-05-25.

BUENO, Elaine Inacio. **Utilização de redes neurais artificiais na monitoração e detecção de falhas em sensores do Reator IEA-R1**. 2006. Dissertação (Mestrado em Tecnologia Nuclear - Reatores) - Instituto de Pesquisas Energéticas e Nucleares, University of São Paulo, São Paulo, 2006. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/85/85133/tde-22052007-153127/>>. Acesso em: 2012-05-22.

BURROUGH, P.A. e MCDONNELL, R.A (1998). **Principles of Geographical Information Systems**. Oxford, Oxford University Press, 1998.

CASANOVA, V. (2006). Networked Control Systems over Profibus-DP: Simulation Mode. In: IEEE International Conference on Control Application. Munich, GERMANY, 2006. **Proceedings...** New York: IEEE, 1993. p.1337-1342.

CASSIOLATO, C. (2010). Minimizando Ruídos em Instalações PROFIBUS. **Revista Mecatrônica Atual, Automação Industrial de Processos e Manufatura**. Ed. Europa. n.46. p.40-42, 2010.

CAVANI, F. A. (2007). **Análise de cenas de pomares de laranjeiras através de segmentação de imagens e reconhecimento de padrões**. 2007. Dissertação (Mestrado em Manufatura) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18145/tde-17012011-123819/>>. Acesso em: 2012-05-08.

CHEN, T. (2003). A fuzzy back propagation network for output time prediction in a wafer fab, **Applied Soft Computing**, Elsevier, Soft Computing in Manufacturing Enterprise Systems, v.2, n.3, p.211-222, 2003.

CHIANG, J. e HSU, J. (2002). Fuzzy classification trees for data analysis. **Journal Fuzzy Sets and Systems**. v.130 n.1, p.87-99, Agosto 2002.

D'EMERY, R. A. (2007). **Aplicação de Análises Estatística e Neural para Reconhecimento de Sinais de Odores**. 2007. Dissertação (Mestrado) – Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco, 2007. Disponível em: < <http://www.pgbiom.ufrpe.br/dissertacoes/2007/d2007-06.pdf>>. Acesso em: 10/07/2012.

DIEDRICH,C.; BANGEMANN, T. (2007). **Profibus PA: Instrumentation Technology for the Process Industry**. Oldenbourg Industrieverlag GmbH.

DINIZ, P. S. R.; SILVA, E. A. B.; NETTO, S. L. (2004). **Processamento Digital de Sinais – Projeto e Análise de Sistemas**. Rio de Janeiro, Bookman, 2004.

EL AYEB, B. (1994). Toward systematic construction of diagnostic systems for large industrial plants: methods, languages, and tools. **IEEE Transactions on Knowledge and Data Engineering**. v6, n.5, p. 698-712.

FELSER, M. (2006). Quality of profibus installations. In: IEEE WORKSHOP ON FACTORY COMMUNICATION SYSTEMS, Torino, ITALY, 2006. **Proceedings...** New York: IEEE, 2006. p. 113-118.

FELSER, M. (2011). **Profibus Manual**. Oldenbourg Industrieverlag GmbH. epubli GmbH, Berlin. ISBN 978-3-8442-1435-2.

FENTON, B., MCGINNITY, M., MAGUIRE, L. (2002). Fault diagnosis of electronic systems using artificial intelligence. **IEEE Instrumentation & Measurement Magazine** 2002. n.3, v.5, p. 16-20.

FERGANY, A. A., YOUSEF, M. T. and EL_ALAILY, A. A.(2001). Fault Diagnosis in Power Systems - Substation Level - Through Hybrid Artificial Neural Networks and Expert System. In: TRANSMISSION AND DISTRIBUTION CONFERENCE AND EXPOSITION, Atlanta, GA, 2001. **Proceedings...** York: IEEE, 2001. v.1. p.207-211.

FERREIRA, L.; TOVAR,E. (2004).Timing analysis of a multiple logical ring wired/wireless PROFIBUS network. In: INTERNATIONAL WORKSHOP FACTORY COMMUNICATION SYSTEMS, 2004. **Proceedings...** New York: IEEE, 2004. p.81-90.

FLORES, C. D. Fundamentos dos Sistemas Especialistas. In: BARONE, D. A. C. (Ed.). **Sociedades Artificiais: a nova fronteira da inteligência nas máquinas**. Porto Alegre: Bookman, 2003. p.127-154.

FONG, C.K.; LAI, E.M.-K. (1994). AILAN: a local area network diagnostic expert system. In: INTERNATIONAL CONFERENCE ON EXPERT SYSTEMS FOR DEVELOPMENT, 1994. **Proceedings...** New York: IEEE, p. 238-243.

FORNARO,O.; MAGRO,M.C.; PINCETI,P. (2004). Diagnostics for measure transmitters. In: 21st IEEE INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE (IMTC 04), 2004. **Proceedings...** New York: IEEE, 2004. v.2, p.1450-1453.

GÜLER, I., ÜBEYLI, E. D. (2005). An Expert System for Detection of Electrocardiographic Changes in Patients with Partial Epilepsy using Wavelet-based Neural Networks. **Expert**

Systems. Wiley Online Library, 2005, v.22, n.2, p.67-71. DOI: 10.1111/j.1468-0394.2005.00295.x. Disponível em: <<http://onlinelibrary.wiley.com/doi/10.1111/j.1468-0394.2005.00295.x/pdf>> . Acesso em: 10/07/2012.

HARIKUMAR, R.; NARAYANAN, B.S. (2003). Fuzzy Techniques for Classification of Epilepsy risk level from EEG Signals. In: IEEE CONFERENCE ON CONVERGENT TECHNOLOGIES FOR ASIA-PACIFIC REGION (TENCON). 2003. **Proceedings...** New York: IEEE, 2003. v.1, p.209-213.

HORI, S.; OMORI, T. (1991). VGM: an expert system supporting operators to repair facilities. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL ELECTRONICS, CONTROL AND INSTRUMENTATION (IECON), 1991. **Proceedings...** New York: IEEE, 1991. v.1, p.1525-1530.

HOSSEINPOUR, F.; HAJIHOSSEINI, H. Importance of Simulation in Manufacturing. **World Academy of Science, Engineering and Technology**. 2009, v.51, p.285-288. ISSN:2070-3724.

HUANG, Y. (2005). Time characteristics of PROFIBUS on Windows XP, Individual Project, Control Laboratory, **University of Twente Publications**, the Netherlands, 2005. Disponível em: <<https://www.ce.utwente.nl/aigaion/publications/show/910>>. Acesso em: 10/07/2012.

HUANG, Y. (2009). Research on Internet-based Open Remote Fault Diagnosis Expert System. In: FIRST ASIAN HIMALAYAS INTERNATIONAL CONFERENCE, Kathmandu, NEPAL, 2009. **Proceedings...** New York: IEEE, 2009, p.1-4.

HUANG, X.; QI, H.; LUI, X. (2006). Implementation of Fault Detection and Diagnosis System for Control Systems in Thermal Power Plants. In: SIXTH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (WCICA), Dalian, CHINA, 2006. **Proceedings...** New York: IEEE, v.2, p. 5777-5781.

HUSEMANN, R.. **Sistema de validação temporal para redes de barramento de campo**. 2003. Dissertação (Mestrado em Engenharia Elétrica) - Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/6114/000525124.pdf?sequence=1>. Acesso em: 18/07/2012.

HUSEMANN, R.; PEREIRA, C.E. (2003). BR-Tool: a real-time bus monitoring and validation system for fieldbus-based industrial automation applications. In: IEEE CONFERENCE EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA). Lisbon, PORTUGAL, 2003. **Proceedings...** New York: IEEE, v.1, p. 145-152.

JAIN, M. B., JAIN, A. and SRINIVAS M. B. (2008). A Web based Expert System Shell for Fault Diagnosis and Control of Power System Equipment. In: INTERNATIONAL CONFERENCE ON CONDITION MONITORING AND DIAGNOSIS, Beijing, CHINA, 2008. **Proceedings...** New York: IEEE, p.1310-1313.

JAVA PROPERTIES. Java Documentation. Disponível em: <<http://download.oracle.com/javase/1.4.2/docs/api/java/util/Properties.html>>. Acesso em: 10/07/2012.

KAGHAZCHI, H.; HONGXIN LI ; ULRICH, M. (2008). Influence of Token Rotation Time in Multi Master PROFIBUS Networks. In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS (WFCS), Dresden, GERMANY, 2008. **Proceedings...** New York: IEEE, p.189-197.

KAMEL, T. S.; HASSAN, M. A. M.; EL-MORSHEDY, A. (2009). Using a Combined Artificial Intelligent Approach In Distance Relay for Transmission Line Protection in EPS. In: FIFTH INTERNATIONAL SOFT COMPUTING, COMPUTING WITH WORDS AND PERCEPTIONS IN SYSTEM ANALYSIS, DECISION AND CONTROL (ICSCCW), Famagusta, 2009. **Proceedings...** New York: IEEE, p.1-6.

KASZKUREWICZ E.; BHAYA A.; EBECKEN N.F.F (1997). A Fault Detection and Diagnosis Module for Oil Production Plants in Offshore Platforms. **International Journal of Expert Systems with Applications**, Elsevier, v.12, n.2, p.189-194, 1997.

KHANH, T. D. (2009). An Open Implementation of Profibus DP. In: ELEVENTH REAL-TIME LINUX WORKSHOP, Dresden, GERMANY, 2009. **Proceedings of Real Time Linux Foundation**. p.65-72. Disponível em: <http://lwn.net/images/conf/rtlws11/papers/proceedings_s_2009.pdf>. Acesso em: 10/07/2012.

KAVIDIR, I.; GUYER, D. E. (2003). Apple grading using fuzzy logic, **Turkish Journal of Agriculture**, Tübitak, 2003, v.27, n.6, p.375-382.

LAI, E.Y. (1989). A knowledge-based approach to intelligent network trouble analysis. In: GLOBAL TELECOMMUNICATIONS CONFERENCE AND EXHIBITION (GLOBECOM), Dallas, TX, 1989. **Proceedings...** New York: IEEE, v.1. p.512-516.

LEE, K.C.; KIM, H.H.; LEE, S.; LEE, M.H. (2003a). Timer Selection Algorithm for Real-time Requirements of Profibus Protocol using GA. In: IEEE/ASME INTERNATIONAL CONFERENCE ON ADVANCED INTELLIGENT MECHATRONICS (AIM), Kobe, JAPAN, 2003. **Proceedings...** New York: IEEE, v.2. p. 741-746.

LEE, K.C.; LEE, S.; LEE, M.Y. (2003b). Remote Fuzzy Logic Control of Networked Control System via Profibus-DP. **IEEE Transactions on Industrial Electronics**, v.50, n.4, p.784-792. 2003.

LEE, S; LEE, K.C.; KIM H.H. (2003). Development of Performance Model for Profibus Token Passing Protocol. In: IEEE CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN), Seville, SPAIN, 2003. **Proceedings...** New York: IEEE, p.130-135.

LIRA, M.M.S. (2004). **Wavelets na Compactação e Processamento de Sinais de Distúrbio em Sistemas de Potência para Classificação via Redes Neurais Artificiais**. Recife. 132 p. Tese (Doutorado) – Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Programa de Pós-graduação em Engenharia Elétrica, Pernambuco, 2004.

MAKI, Y.; LOPARO, K. A. (1997). A Neural Network Approach to Fault Detection and Diagnosis in Industrial Processes. **IEEE Transactions on Control Systems Technology**, v.5, n.6. p. 529-541. 1997.

MANCHESTER METROPOLITAN UNIVERSITY PROFIBUS INTERNATIONAL COMPETENCE CENTRE (2008). Pratical Steps for a successful PROFIBUS Project. Manchester Metropolitan University. Disponível em: <http://www.profibus.com/uploads/media/pxddamkey%5B9101%5D_Successful_Profibus_Project_XuiJi.pdf>. Acesso em 20/05/2012.

MATAIX, C.; MARTIN, P.; RODRIGUEZ, F.J.; MANZANO, M.J.; POZO, J.; DONATO, P.G. (2003). Design and implementation of a safety communication network in railways. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA), Lisbon, PORTUGAL, 2003. **Proceedings...** New York: IEEE, v.2, p.109-112.

MENDES, R. D. (1997). Inteligência Artificial: Sistemas Especialistas no Gerenciamento da Informação. **Ciência da Informação**, 1997, v.26, n.1, ISSN:0100-1965. Disponível em: <<http://dx.doi.org/10.1590/S0100-19651997000100006>>. Acesso em: 10/07/2012.

MITCHELL, R. W. (2004) **PROFIBUS – A Pocket Guide**. Research Triangle Park: ISA – The Instrumentation, Systems and Automation Society, 2004.

MOORE, R.L. (1992). The G2 development and deployment environment. In: FOURTH INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, Arlington, VA, 1992. **Proceedings...** p. 458-459.

MOSSIN, E. A.; PANTONI, R. P.; BRANDÃO, D. (2008). Sistema Fuzzy para avaliação de desempenho de alunos na área de automação industrial. In: XVII CONGRESSO BRASILEIRO DE AUTOMÁTICA, 2008, Juiz de Fora, BRASIL. **Anais do XVII Congresso Brasileiro de Autômática**, 2008. 1 CD-ROM.

NAIDU, S. R.; ZAFIRIOU, E.; McAVOY, T. J. (1990). Use of Neural Networks for Sensor Failure Detection in a Control System. In: AMERICAN CONTROL CONFERENCE. Pittsburgh, PENNSYLVANIA, 1990. **Proceedings...** New York: IEEE, v.10, n.3, p.49-55.

NARENDRA, K. S., PARTHASARATHY K. (1990). Identification and Control of Dynamical Systems Using Neural Networks. **IEEE Transactions on Neural Networks**. v.1, n.1, 1990.

NEDELJKOVIC I. (2004). Image classification based on fuzzy logic. In: THE INTERNATIONAL SOCIETY FOR PHOTOGRAMMETRY AND REMOTE SENSING (ISPRS), Istanbul, TURKEY, 2004. **Proceedings of XXth ISPRS Congress**, v.34, p. 83-89.

NORMATIVE PARTS OF PROFIBUS FMS, DP AND PA. (1998): according to the European Standard **EN50170**. v.2. Edition 1.0.

ORHAN, U.; HEKIM, M.; ÖZER, M. (2010). Discretization Approach to EEG Signal Classification Using Multilayer Perceptron Neural Network Model. In: 15th NATIONAL BIOMEDICAL ENGINEERING MEETING (BIYOMUT), Antalya, TURKEY, 2010. **Proceedings...** New York: IEEE, p.1-4.

PROCEN TEC (2012). **Profitrace 2.6.2 User Manual**. Document Version 3.5.0. May, 2012. Disponível em: <<http://www.procentec.com/downloads/profitrace/ProfiTrace2-Manual-EN.pdf>>. Acesso em 18/05/2012.

PROFIBUS SCOPE (2010). **Profibus Scope User Manual**. Document Version 1.2. Feb, 2010. Disponível em: <http://www.t-h.de/fileadmin/inhalt/IC/downloads/profibus_scope/software/User_Manual_PROFIBUS_Scope.pdf>. Acesso em 18/05/2011.

RAMOS, A. C. (2006). **Sistemas Especialistas no Auxílio ao Diagnóstico para Clínicas de Pediatria**. Trabalho de Conclusão do Curso em Sistemas de Informação, Rio Grande do Sul, 2006.

REZENDE, S.O. (2003). **Sistemas inteligentes: fundamentos e aplicações**. 1th ed. Editora Manole, 2003

ROSEMOUNT INC. HUISENGA, G.D.; LEWIS, R.A.; LATTIMER, D.R. **Field device for digital process control loop diagnostics**. US20090083001. September 15, 2009.

ROSEMOUNT INC. HUISENGA, G.D.; LONGSDORF, R.J.; LATTIMER, D.R. **Two-wire process control loop diagnostics**. US7321846. January 22, 2008

ROSS, T. J. (2005). **Fuzzy logic with engineering applications**. 2th ed. USA, Willey, 2005.

SHAMSUDDIN, N., MUSTAFA, M. N., HUSIN, S., TAIB, M. N. (2005). Classification of Heart Sounds Using a Multilayer Feed-forward Neural Network. In: CONFERENCE ON SENSORS AND THE INTERNATIONAL CONFERENCE ON NEW TECHNIQUES IN PHARMACEUTICAL AND BIOMEDICAL RESEARCH, Kuala Lumpur, MALAYSIA, 2005. **Proceedings...** New York: IEEE, p.87-90.

SHERWALI, H.; CROSSLEY, P. (1997). An expert system to evaluate the operating response of the distance relays during a fault. In: SIXTH INTERNATIONAL CONFERENCE ON DEVELOPMENTS IN POWER SYSTEM PROTECTION, Nottingham, UK, 1997. **Proceedings...** New York: IEEE, p.361-364.

SILVA, W.L. (2008). **Métodos para Diagnóstico de Falhas e Avaliação de Desempenho em Redes PROFIBUS DP**, 2008. Monografia (Especialização em Automação Industrial) – Escola de Engenharia, UFMG, Belo Horizonte.

SILVA, W.L.; MOTA, C.H. (2010). Métodos para Diagnóstico em Redes PROFIBUS DP. **Profibus Brasil – Artigos Técnicos**. Disponível em: <http://www.profibus.org.br/detalhe_noticia.php?id=24>. Acesso em: 16/05/2011.

SIMEÓN, E.J.A. (2008). **Aplicação de técnicas de inteligência artificial no desenvolvimento de um sistema de manutenção baseada em condição**. Dissertação (mestrado) - Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Mecânica, 2008. Disponível em: <<http://repositorio.bce.unb.br/handle/10482/1089>>. Acesso em: 11/07/2012.

SOFTING AG. ENDL, H.; PALME, D. **Diagnostic procedure and device for a field bus system**. US US7966525. June 21, 2011

SOFTING (2012). **PROFIBUS Tester 4 (BC-600-PB) User Manual**. Disponível em: <<http://industrial.softing.com/en/products/functionality/network-diagnostics/mobile-diagnostics-commissioning-and-troubleshooting/profibus/profibus-tester-physical-installation-quality-and-protocol-analysis.html>>. Acesso em 18/05/2012.

SOUZA, R.C. (2012). **Diagnóstico da camada física de redes Profibus DP baseado em redes neurais artificiais**. Dissertação (Mestrado em Sistemas Dinâmicos) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18153/tde-14052012-101221/>>. Acesso em: 2012-07-11.

SPATTI, D.H. (2007). **Sistema inteligente para controle de relés reguladores de tensão**. Dissertação (Mestrado em Engenharia Elétrica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18133/tde-29032007-103518/>>. Acesso em: 2012-05-25.

STEMMER, M. R. (2001). **Sistemas Distribuídos e Redes de Computadores para Controle e Automação Industrial**. Santa Catarina: Universidade Federal de Santa Catarina.

SUSSMAN B. C., FITZPATRICK B. W., PILATO, C. M. (2002). **Version Control with Subversion**. E-book. Disponível em: <<http://svnbook.red-bean.com/en/1.4/svn-book.html>>. Acesso em: 03/08/2011.

TAN, J.C.; CROSSLEY, P.A.; MCLAREN, P.G. (2001). Fuzzy expert system for on-line fault diagnosis on a transmission network. In: IEEE POWER ENGINEERING SOCIETY WINTER MEETING, Columbus, OH, 2001. **Proceedings...** New York: IEEE, p.775-780, v.2.

TAO, J.; WANG, Z. (2005). Analysis of PROFIBUS.DP Network Delay and Its Influence on the Performance of Control Systems. **Wuhan University Journal of Natural Sciences (WUJNS)**. p.877-882, v.10, n.5, 2005.

TATAI, V. K. (2003). **Técnicas de Sistemas Inteligentes aplicadas ao desenvolvimento de Jogos de Computador**. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2003. Disponível em: <www.dca.fee.unicamp.br/~gudwin/ftp/publications/TeseTatai.pdf>. Acesso em: 11/07/2012.

PROFIBUS GUIDELINE (1998). **Installation Guideline for Profibus-DP/FMS**. Version 1.0 September 1998.

TORRES, R. V. (2011). **Simulador de redes Profibus DP dedicado à ferramenta de diagnóstico**. Dissertação (Qualificação de Mestrado). Escola de Engenharia de São Carlos, USP.

TOVAR, E.; VASQUES, F. (1999). Cycle time properties of the PROFIBUS timed-token protocol. **Computer Communications**, Elsevier Science, v.22, n.13, p.1206-1216, 1999.

UPTON, G.; COOK, I. (1998). **Introducing Statistics**. 2nd Edition. Oxford University Press, Oxford, p.56-65. 1998.

USIDA, W.F. (2007). **Controle fuzzy para melhoria do perfil de tensão em sistemas de distribuição de energia elétrica**. Dissertação (Mestrado em Engenharia Elétrica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18133/tde-13042007-121149/>>. Acesso em: 2012-04-10.

VENTURINI, V.; BRANDÃO, D.; PINOTTI, M.J. (2008). Análise Experimental da Influência do Tempo Alvo de Rotação do Token No Tempo de Varredura de Escravos em Redes Profibus. In: VIII CONFERÊNCIA INTERNACIONAL DE APLICAÇÕES INDUSTRIAIS, Poços de Caldas, BRASIL, 2008. **Anais...** Disponível em: <<http://www.labplan.ufsc.br/congressos/Induscon%202008/pdfs/42393.pdf>>. Acesso em: 24/05/2012

VITTURI, S. (2004). On the effects of the acyclic traffic on PROFIBUS DP networks, **Computer Standards & Interfaces**, v.26, n.2, p.131-144, 2004.

WANG, Y. et al. (2009). Fault diagnosis expert system based on integration of fault-tree and neural network. In: COMPUTATIONAL INTELLIGENCE AND SOFTWARE ENGINEERING, Wuhan, CHINA, 2009. **Proceedings...** New York: IEEE, p.1-4.

WU, S., CHOW, W. S. (2004). Induction Machine Fault Detection Using SOM-Based RBF Neural Networks. **IEEE Transactions on Industrial Electronics**, v. 51, n.1, p.183-194, 2004.

WU, Z. Y. B.; ZARGARI, N. (2000). Online Mechanical Fault Diagnosis of Induction Motor by Wavelet Artificial Neural Network Using Stator Current. In: 26th ANNUAL INDUSTRIAL ELECTRONICS SOCIETY (IECON), Nagoya, JAPAN, 2000. **Proceedings...** New York: IEEE, p.1183-1188, v.2.

ZADEH, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision process. **IEEE Transactions on Systems, man, and cybernetics**, v.3, n.1, p.28-44, 1973.

ZALIS, K. (2004). Application of Expert Systems in Diagnostics of High Voltage Insulating Systems. In: IEEE INTERNATIONAL CONFERENCE ON SOLID DIELECTRICS (ICSD), 2004. **Proceedings...** New York: IEEE, p.691-694, v.2, 2004.

ZHENG, X.; WANG, Z.; QIAN, F. (2006). An Expert System for Real-time Fault Diagnosis and Its Application in PTA Process. In: IEEE SIXTH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (WCICA), 2006. **Proceedings...** New York: IEEE, p.5623-5627, v.2, 2006.

ZIMMERMANN, H.J. (1997). **Fuzzy sets, decision making, and expert systems**. International Series in Management Science Operations Research. Boston, Springer, 1th ed. ISBN:0-898-38149-5.

APÊNDICE A

Detalhes do software ProfiDoctor

Neste apêndice, algumas funcionalidades importantes do software desenvolvido para a análise dos resultados deste trabalho são apresentadas. Para o desenvolvimento deste sistema, foram utilizadas algumas bibliotecas Java de terceiros:

- SwingX : esta biblioteca permite a criação de interfaces gráficas com recursos mais avançados. O projeto desta é um subprojeto do SwingLabs (<http://www.swinglabs.org/>) que é um laboratório de desenvolvimento de código aberto a fim de fazer com que a criação de interfaces gráficas Java fique mais simples, com melhor desempenho e um design mais aprimorado.
- JFreeChart: considerando que o sistema possui um osciloscópio e um gráfico que apresenta o nível médio de tensão dos dispositivos, uma biblioteca com recursos para a construção desses gráficos também foi incorporada ao software. Esta última é denominada JFreeChart-1.0.13. O JFreeChart é uma biblioteca 100% livre para a criação gráficos de qualidade profissional em aplicações desenvolvidas em Java.
- JAMA: para facilitar a implementação do algoritmo das RNAs utilizadas neste projeto, utilizou-se a biblioteca JAMA, que possui uma série de objetos que facilitam o trabalho com matrizes entre outros objetos voltados para álgebra linear. A versão utilizada neste pacote foi a 1.0.2. Este software é um produto criado cooperativamente pela The MathWorks e pelo Instituto Nacional de Padrões e Tecnologia (NIST) e é liberado para o domínio público.

Considerando a complexidade e a quantidade de classes e bibliotecas desenvolvidas no projeto do ProfiDoctor, um sistema para controle de versões denominado *SubVersion* (SVN) também foi utilizado. Além de facilitar o gerenciamento de versões este tipo de sistema contribui para o trabalho em grupo. Esta característica foi muito utilizada neste projeto, afinal, o ProfiDoctor foi realizado em conjunto com o simulador desenvolvido por Torres (2011), pois os dois softwares possuem classes que são utilizadas de maneira comum.

Segundo Sussman et al. (2002), *Subversion* é um sistema de controle de versão livre/open-source que tem o objetivo de gerenciar arquivos e diretórios através do controle de suas mudanças. Dessa maneira, é possível a recuperação de versões antigas de seus dados além de permitir a análise do histórico de como seus dados foram modificados.

Para que um sistema deste tipo funcione, é necessário um repositório de dados onde todo o código fonte do software será armazenado e disponibilizado aos membros do grupo de desenvolvimento. Neste projeto, um repositório remoto com acesso disponibilizado para a Internet foi criado. Tal repositório, o *ProjectLocker* (<http://www.projectlocker.com/>) possui controle de usuários e é grátis.

Para facilitar o acesso aos dados desse repositório, o software cliente *SmartSVN* (<http://www.syntevo.com/smartsvn/index.html>) foi utilizado. Através deste software, é possível efetuar todas as operações permitidas pelo SVN, destacando-se o envio de novas versões de arquivos para o repositório (*commit*) e a atualizações das novas versões de arquivos que encontram-se no repositório (*update*).

Dessa maneira, três projetos foram inseridos no controle de versões. O primeiro projeto refere-se ao código fonte do ProfiDoctor. O segundo, refere-se à parte comum entre o ProfiDoctor e o Simulador. O terceiro armazena o código do Simulador desenvolvido por Torres (2011).

A tabela abaixo possui o caminho para acesso a cada um dos projetos:

Nome do Projeto	Caminho (SVN-Location)
PROFIDOCTORAPP	https://free1.projectlocker.com/emossin/PROFIDOCTOR/svn
SIMULATOR	https://pl3.projectlocker.com/renatoveigatorres/profisim/svn
ProfiDoctorCommon	https://pl3.projectlocker.com/profidoctorcommon/svn

Com as bibliotecas e as ferramentas de suporte para o desenvolvimento apresentadas, tem-se então uma breve descrição de cada uma das funcionalidades fornecidas pelo ProfiDoctor.

Live List

A *Live List* é um componente onde todos os dispositivos de uma rede Profibus e seus respectivos status são mostrados. No sistema em questão, a *Live List* é apresentada ao usuário através de duas interfaces gráficas distintas. A primeira é mostrada na figura abaixo.

Nota-se através da figura que os dispositivos 4, 7, 25, 55, 95, 96 e 115 estão *on-line*. O dispositivo de endereço 1 é o mestre da rede.

A outra possibilidade de visualizar a *Live List* apresenta os dispositivos de uma maneira mais compacta através de uma *TreeView*.

É garantido que o funcionamento das duas *Live List* seja exatamente o mesmo, pois ambas baseiam-se em uma estrutura de controle que está implementada em uma classe que fica ativa durante todo o funcionamento do software. Esta classe recebe os telegramas Profibus e os analisa a fim de determinar quais os dispositivos estão em funcionamento.

Sabe-se que o Profibus é um sistema mestre/escravo e que os escravos se comunicam apenas quando o mestre envia mensagens permitindo isso. Sabe-se também que uma rede Profibus possui escravos endereçados com endereços que variam entre 0 e 125. O endereço 126 é um valor padrão utilizado para que o escravo entre na rede.

Desta maneira, para determinar quais dispositivos estão em funcionamento ou não, é necessário analisar cada telegrama oriundo de uma rede ou do simulador Profibus e baseando-se em quem está se comunicando ou não, monta-se a *Live List*. Em períodos pré-determinados (1000 ciclos por padrão), esta análise é feita e quando um dispositivo para de se comunicar ou inicia uma nova comunicação, o status do mesmo é atualizado na classe que controla a *Live List*. Com a mesma periodicidade que tal análise é feita, as interfaces gráficas que apresentam a *Live List* ao usuário também são atualizadas.

A figura a seguir mostra o mecanismo que é implementado para o funcionamento da *Live List*.

Nota-se que o gerenciador de telegramas envia um evento de recebimento de telegrama ProfiDoctor para a classe chamada *LiveListController*. Esta classe recebe este telegrama e armazena o endereço do dispositivo (através do campo *Source Address*) na lista de dispositivos do bloco de análise corrente. Este procedimento é feito para todos os telegramas recebidos até o final de bloco de análise (padrão 1000 ciclos). Neste momento, uma análise é feita. Nesta análise, todos os endereços armazenados na lista de dispositivos do bloco atual são considerados dispositivos ativos (ícone verde). Após isso, uma comparação com a lista de dispositivos armazenadas no bloco de análise anterior é feita e para cada

dispositivo que estava presente no bloco anterior e ausente no bloco atual, um status de “ausência” é configurado (ícone vermelho). Por fim, os dispositivos que não estão em nenhuma das duas listas é configurado com o status de desligado (ícone cinza).

Ao final dessa análise, um evento de atualização é enviado para as duas interfaces gráficas que apresentam a *Live List*.

Interface de análise geral

Assim como nos outros módulos, esta interface percorre a lista de mensagens armazenada em memória interpretando cada mensagem e mostrando detalhes da mesma na interface gráfica detalhada a seguir.

Na prática, esta será a janela mais utilizada, pois ela apresenta ao usuário:

1. Detalhes de um determinado telegrama Profibus selecionado na tabela (4);
2. Live List compacta;
3. Gráfico dos sinais amostrados de um determinado dispositivo selecionado na Live List (2);
4. Tabela com os telegramas e os diagnósticos resumidos para cada telegrama;
5. Painel de controle de filtros para facilitar a visualização dos telegramas.

Osciloscópio

Outra funcionalidade necessária para este tipo de software é a visualização dos sinais da camada física. Para isso, desenvolveu-se a interface gráfica apresentada a seguir. Para a

implementação deste módulo, as mensagens armazenadas no histórico são obtidas e as amostras dos sinais contidas nelas são inseridas no gráfico.

Como pode ser observada, esta interface conta com um osciloscópio que plota os gráficos dos sinais amostrados da camada física Profibus. Para isso, o usuário deve selecionar, através do filtro, quais os dispositivos possuirão seus sinais apresentados.

Média da tensão do sinal dos dispositivos

Outra funcionalidade importante é importante a visualização da média de tensão dos sinais de cada dispositivo. Assim como no osciloscópio, as mensagens armazenadas no histórico são percorridas e as suas amostras são obtidas. A partir disso, calcula-se a média das amostras dos sinais positivos e a média das amostras dos sinais negativos. Por fim, a diferença entre estas médias são calculadas e inseridas no gráfico de colunas apresentado na figura a seguir.

APÊNDICE B

Dispositivos Profibus Utilizados

Mestre DF73

Este mestre faz parte da plataforma DFI302. Possui capacidade de *gateway* PROFIBUS DP- High Speed Ethernet. Fornece um canal PROFIBUS Mestre DP classe 1 e 2. Permite comunicação DP e PA, sendo que a utilizada nesta tese foi apenas a DP.

A versão de firmware utilizada na coleta dos resultados foi a 2.0.0.53. A figura a seguir apresenta este dispositivo.

FONTE: SMAR WEB SITE

Mestre (Proficaptain)

Este mestre é implementado em software e age através do Proficore Ultra. Este hardware traduz os dados Profibus para USB e vice-versa. A versão deste hardware é a 1.5. Uma imagem ilustrando o Proficore Ultra é apresentada a seguir.

FONTE: PROCENTEC WEB SITE

Escravo – ET 200S

Características obtidas a partir do arquivo GSD:

- Vendor Name: SIEMENS
- Model Name: ET 200S High Feature
- Ident Nr.: 0x80E0
- Hardware release: A1.0
- Software release: Z1.0

Em sua configuração foram utilizados três módulos:

- 6ES7 138-4CA01-0AA0 PM24V S, com uma entrada;
- 6ES7 131-4BB01-0AB0 2DI DC24V, com uma entrada;
- 6ES7 132-4BB01-0AA0 2DO DC24V, com uma saída;

Este escravo é apresentado na figura a seguir.

FONTE: SIEMENS WEB SITE

Escravo – DP100V

Características obtidas a partir do arquivo GSD:

- Vendor Name: ALTUS Sistemas de Informatica
- Model Name: DP100V
- Ident Nr.: 0x04D4
- Hardware release: 1.0
- Software release: 1.3

Em sua configuração foi utilizado o módulo 122.4BH00 DO16xDC24V. Este módulo é composto por duas saídas.

Este escravo é apresentado na figura a seguir.

FONTE: ALTUS WEB SITE

Escravo – DP-B-FDN-M

Características obtidas a partir do arquivo GSD:

- Vendor Name: SENSE ELETRONICA LTDA
- Model Name: DP-B-FDN
- Ident Nr.: 0x08B5
- Hardware release: 1.0
- Software release: 1.0

Em sua configuração foi utilizado apenas um módulo com nome de INPUT - 8byte. Este módulo possui 8 entradas.

Este escravo é apresentado na figura a seguir.

FONTE: SENSE WEB SITE

Escravo – DP-KD-2EA-D

Características obtidas a partir do arquivo GSD:

- Vendor Name: SENSE ELETRONICA LTDA
- Model Name: DP-KD-2EA-D
- Ident Nr.: 0x0799
- Hardware release: A1.0
- Software release: Z1.0

Durante a configuração deste escravo é necessária a inserção do módulo denominado DP-KD-2EA-D que possui 4 entradas. Este escravo é apresentado na figura a seguir.

FONTE: SENSE WEB SITE

Escravo – XM-210DP

Características obtidas a partir do arquivo GSD:

- Vendor Name: DLG Automação Industrial LTDA
- Model Name: XM-210DP
- Ident Nr.: 0x0C05
- Hardware release: 1.00
- Software release: 1.00

Para sua correta configuração é necessária a inserção manual e em ordem de todos os módulos oferecidos por este dispositivo, sendo estes: Channel 1,2,3 ... 16; Digital Inputs; Ambient temp.; Alarms 1 and 2; Relay 1 and 2. No total, tem-se que este escravo foi configurado com 38 entradas e uma saída.

A figura a seguir apresenta o dispositivo descrito.

FONTE: DLG WEB SITE