UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

RENATO SANTOS CARRIJO

Gerenciamento de ativos aplicado a instrumentos de campo de protocolos abertos: uma abordagem a partir de dispositivos móveis

São Carlos
2011

RENATO SANTOS CARRIJO

Gerenciamento de ativos aplicado a instrumentos de campo de protocolos abertos: uma abordagem a partir de dispositivos móveis

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

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Dennis Brandão

São Carlos

2011

Trata-se da versão corrigida da dissertação. A versão original encontra-se disponível na EESC/SP que aloja o Programa de Pós-Graduação em Engenharia Elétrica

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

C316g

Carrijo, Renato Santos.
Gerenciamento de ativos aplicado a instrumentos de campo de protocolos abertos: uma abordagem a partir de dispositivos móveis. / Renato Santos Carrijo ; orientador Dennis Brandão. São Carlos, 2011.

Dissertação (Mestrado - 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, 2011.

1. Automação industrial. 2. Gerenciamento de ativos. 3. Dispositivos móveis. 4. PROFIBUS PA. 5. HART. 5. *Foundation Fieldbus*. 6. *Cyber*OPC. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **RENATO SANTOS CARRIJO**

Título da dissertação: "Gerenciamento de ativos aplicado a instrumentos de campo de protocolos abertos: uma abordagem a partir de dispositivos móveis".

Data da defesa: 31/01/2012

Comissão Julgadora:	**Resultado:**
Prof. Dr. **Dennis Brandão (Orientador)** (Escola de Engenharia de São Carlos/EESC)	Aprovado
Prof. Dr. **Nunzio Marco Torrisi** (Universidade Federal do ABC)	Aprovado
Prof. Dr. **Edilson Reis Rodrigues Kato** (Universidade Federal de São Carlos)	Aprovado

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

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo Cesar Lima Segantine**

Agradecimentos

Agradeço, em primeiro lugar, a Deus: autor da vida, fonte inesgotável de amor, sabedoria e ciência;

À minha querida esposa, Rita de Cassia, pelo carinho, compreensão, amor e companheirismo em todas as etapas do mestrado;

Aos meus filhos Gabriel e Mariana, por, juntamente com minha esposa, serem a grande motivação dessa jornada;

Aos meus pais, ao meu sogro e à minha sogra, por todo apoio;

Aos meus irmãos;

Ao professor e amigo Dr. Dennis Brandão, pela oportunidade, incentivo, amizade e motivação que me dedicou ao orientar esse trabalho;

Ao professor Dr. Nunzio Torrisi, pelo apoio no desenvolvimento do projeto;

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

À Smar, pelo incentivo à pós-graduação;

A todos os amigos, que direta ou indiretamente contribuíram para que esse trabalho chegasse até aqui.

Resumo

A tecnologia emergente dos dispositivos móveis como telefones celulares, *handhelds* e *smartphones* trouxe um novo paradigma para o uso desses equipamentos, que, além das convencionais chamadas de áudio, hoje suportam vídeo, internet e redes sem fio. Várias aplicações têm sido desenvolvidas para esse ambiente. Por outro lado, na área da automação industrial, os instrumentos de campo modernos contêm embutidos, internamente, conjuntos de informações que proporcionam grande oportunidade para a gestão de ativos. A necessidade do gerenciamento de ativos ocorre na fase de operação da planta industrial e é realizada por meio de ajustes finos, calibrações, alteração de configurações e manutenções, com o objetivo de se evitar paradas indesejadas. Esse trabalho insere-se no contexto de interface entre esses dois mundos: a computação móvel e o gerenciamento de ativos. Nesse sentido, é desenvolvida uma análise dos padrões abertos *PROFIBUS PA*, *HART* e *Foundation Fieldbus* e o uso das suas características aplicado ao gerenciamento de ativos. É apresentada, ainda, uma proposta e verificação de viabilidade técnica de uma arquitetura aplicada ao gerenciamento de ativos a partir de um dispositivo móvel usando-se um canal de comunicação seguro baseado no padrão *CyberOPC* para acesso à planta industrial.

Palavras-chave: Gerenciamento de Ativos, Dispositivos móveis, PROFIBUS PA, HART, Foundation Fieldbus, Automação industrial, CyberOPC.

Abstract

The emerging technology of mobile devices like cell phones, hand helds and smartphones has brought a new paradigm for the use of such equipment, witch, in addition to conventional audio calls, now supports video, internet and wireless networks. Several applications have been developed for this environment. On the other hand, in the industrial automation area, the modern field devices contain embedded information sets that provide great opportunity for asset management. The need for asset management occurs during the operation of the plant and is performed by means of fine tuning, calibration, changes of configuration and maintenance, in order to avoid undesired downtime. This work fits into the context of interface between these two worlds: mobile computing and asset management. In this sense, is made an analysis of open standards PROFIBUS PA, HART and Foundation Fieldbus and use its features applied to asset management. It also presented a proposal and verification of technical feasibility of an architecture applied to asset management from a mobile device using a secure communication channel based on the standard CyberOPC to access the industrial plant.

Keywords: Asset Management, Mobile devices, PROFIBUS PA, HART, Foundation Fieldbus, Industrial Automation, CyberOPC.

Lista de Figuras

Figura 1- Evolução Tecnológica dos Sistemas de Automação. Fonte: Regh, Swain e Yangula (1999) 7
Figura 2 - Representação da infraestrutura de comunicação industrial Fonte: Brandão (2005) 8
Figura 3 - Topologia física de conexões ponto a ponto e multiponto Fonte: Forouzan (2004), com adaptações 10
Figura 4 - Topologias de redes Fonte: Comer & Droms (2004), com adaptações 11
Figura 5 - Formato do Frame HART Fonte: HART COMMUNICATION FOUNDATION (1996) 13
Figura 6 - Relação entre Blocos, Parâmetros de Blocos e Diretório Fonte: PROFIBUS INTERNACIONAL (2009) 15
Figura 7 - *OctectString* dos 4 bytes do Parâmetro *DIAGNOSIS* do *Physical Block* Fonte: PROFIBUS INTERNACIONAL (2009) 16
Figura 8 - Byte de Status FF Fonte: PANTONI (2006) 20
Figura 9 - Especificações do Padrão OPC. Fonte: OPC FOUNDATION (2010) 24
Figura 10 - Camadas da especificação OPC-UA (Unified Architecture) Fonte: OPC FOUNDATION (2010) 25
Figura 11 -CyberOPC – Arquitetura Geral 26
Figura 12 - Evolução da EDDL ao longo dos protocolos de comunicação 27
Figura 13 - Exemplo de Código EDDL e sua interpretação 27
Figura 14 - Processo de utilização da Tecnologia EDDL 28
Figura 15 - Analogia entre a tecnologia FDT e um Driver de Impressora 30
Figura 16 - Escopo das Interfaces FDT 31
Figura 17 - Interoperabilidade entre fabricantes garantida pela tecnologia FDT 31
Figura 18 - Dimensões de Integração Fonte: Theurich, Lehmann e Wollschlaeger (2010) 33
Figura 19 -Estratégia de manutenção razoável com base na probabilidade de 35
Figura 20 - Três exemplos de acesso aos dados de autodiagnósticos Fonte: Theurich, Lehmann e Wollschlaeger (2010) 36
Figura 21 – Sistemas Operacionais de Dispositivos Móveis Fonte: CANALYS (2009) 39
Figura 22 – Sistemas Operacionais de Dispositivos Móveis Fonte: CANALYS (2010) 39
Figura 23 – Sistemas Operacionais de Dispositivos Móveis até o 3º Quartil de 2011 Fonte: GARTNER (2011) 40
Figura 24–Arquitetura Geral 45
Figura 25 –Modelo do Device Fonte: Theurich, Lehmann e Wollschlaeger (2010), com adaptações 46
Figura 26 –CyberOPC – Arquitetura Geral e Driver de Comunicação Fonte: TORRISI et al.(2007), com adaptações 47
Figura 27 – Arquitetura em detalhes utilizada para os testes 48
Figura 28 – Ciclo da mensagem de comunicação 49
Figura 29 – Mensagem GetStatus (enviada e recebida) 51
Figura 30 –FF_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo FF 53
Figura 31 –HART_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo HART 54
Figura 32 –PROFIBUS_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo PROFIBUS PA 55
Figura 33 – Visual Studio 2008 – Ferramenta utilizada para o desenvolvimento do servidor. 56

Figura 34 – Cliente CyberOPC (Windows Mobile)..... 58
Figura 35 – Client CyberOPC (Symbian) 59
Figura 36 – Client CyberOPC (Android)..... 59
Figura 37 – Servidor CyberOPC com Driver HART Emulado e Cliente Windows Mobile ... 62
Figura 38 – Servidor CyberOPC com Driver FF Emulado e Cliente Windows Mobile 63
Figura 39 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Windows Mobile 64
Figura 40 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Symbian 65
Figura 41 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Android 66
Figura 42 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Windows Desktop 67
Figura 43 – Servidor CyberOPC com Driver PROFIBUS Real e Cliente Windows Mobile .. 68
Figura 44 – FY303, Posicionador de Válvula da Smar (Protocolo PROFIBUS PA) Fonte: Smar (2011)..... 71
Figura 45 – Cliente CyberOPC no Windows Mobile, executando um autosetup do FY303 .. 71

Lista de Tabelas

Tabela 1 - Significado do Byte de Status no Protocolo HART. 13
Tabela 2 - Descrição dos parâmetros do Physical Block..... 16
Tabela 3 - Codificação do Parâmetro DIAGNOSIS do PhysicalBlock..... 17
Tabela 4 - Valores possíveis para o BLOCK_ERR..... 20
Tabela 5 – Resultados estatísticos obtidos do Teste A 62
Tabela 6 – Resultados estatísticos obtidos do Teste B 63
Tabela 7 – Resultados estatísticos obtidos do Teste C 64
Tabela 8 – Resultados estatísticos obtidos do Teste D 65
Tabela 9 – Resultados estatísticos obtidos do Teste E 66
Tabela 10 – Resultados estatísticos obtidos do Teste F..... 67
Tabela 11 – Resultados estatísticos obtidos do Teste G 68
Tabela 12 – Comparação entre o Sistema Operacional, Protocolo e Número de mensagens processadas por minuto 69
Tabela A.13 – Dados de tempo de resposta para as mensagens no Teste A 80
Tabela A.14 – Dados de tempo de resposta para as mensagens no Teste B 83
Tabela A.15 – Dados de tempo de resposta para as mensagens no Teste C 86
Tabela A.16 – Dados de tempo de resposta para as mensagens no Teste D 89
Tabela A.17 – Dados de tempo de resposta para as mensagens no Teste E..... 92
Tabela A.18 – Dados de tempo de resposta para as mensagens no Teste F 95
Tabela A.19 – Dados de tempo de resposta para as mensagens no Teste G 98

Lista de Siglas

AMS	*Asset Management System*
ADT	*Android Development Tools*
AISBL	*Association Industrielle Sans But Lucratif = non-profit*
AJAX	*Asynchronous Javascript And XML*
CDMA	*Code Division Multiple Access*
Windows CE	*Windows Embedded Compact*
CLPs	*Controladores Lógico Programáveis*
COM	*Component Object Model*
CSS	*Cascading Style Sheets*
CyberOPC	*Cybernetic OPC (padrão OPC desenvolvido para uso sob HTTP)*
DCOM	*Distributed Component Object Model*
DCS	*Distributed Control System*
DD	*Device Description*
DDC	*Direct Digital Control*
DDL	*Device Description Language*
DIN	*Deutsches Institut für Normung*
DNS	*Domain Name System*
DP	*Decentrallised Periphery*
ECT	*EDDL Cooperation Team*
EDD	*Electronic Device Description*
EDDL	*Electronic Device Description Language*
EDGE	*Enhanced Data rates for GSM Evolution*
ED-VO	*Evolution-Data Optimized*
ERP	*Enterprise Resource Planning*
FB	*Function Block*
FBAP	*Function Block Application Process*
FCS	*Field Control Systems*

FDI	*Field Device Integration*
FDT/DTM	*Field Device Tool / Device Type Manager*
FF	*Foundation Fieldbus*
FMS	*Message Specification*
FSK	*Frequency Shift Key*
FTP	*File Transfer Protocol*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
HART	*Highway Addressable Remote Transducer*
HSE	*High Speed Ethernet*
HTML5	*Hypertext Markup Language, versão 5*
HTTP	*HyperText Transfer Protocol*
HTTPs	*HyperText Transfer Protocol secure*
I/O	*Input / Output*
ICMP	*Internet Control Message Protocol*
IDE	*Integrated Development Environment*
IDEN	*Integrated Digital Enhanced Network*
IEC	*International Electrotechnical Commission*
IHM	*Interface Homem Máquina*
IPSec	*Internet Protocol Security*
IrDA	*Infrared Data Association*
ISP	*Interoperable System Protocol*
J2ME	*Java 2, Micro Edition*
JIT	*Joint Interest Group*
JSON-RPC	*Java Script Object Notation - Remote Procedure Call*
LTE	*Long Term Evolution*
MES	*Manufacturing Execution Systems*
MIS	*Management Information Systems*
MS/TP	*Master-Slave/Token-Passing*
OLE	*Object Linking and Embedding*
OPC (1)	*OLE for Process Control*
OPC (2)	*Open Connectivity*
OPC-UA	*OPC - Universal Architecture*

OS	*Operational System*
OSI	*Open Systems Interconnection*
PA	*Process Automation*
PAM	*Plant Asset Management*
PB	*Physical Block*
PC	*Personal Computer*
PDAs	*Personal Digital Assistant*
PIMS	*Process Information Management System*
PLC	*Programmable Logic Controllers*
PNO	*Profibus Nutzer Organization*
PPP	*Point-to-Point Protocol*
SAMA	*Scientific Apparatur Markers Association*
SCADA	*Supervisory Control and Data Acquisition*
SDKs	*Standard Development Kits*
SSL	*Secure Sockets Layer*
TB	*Transducer Block*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
UART	*Universal Asynchronous Receiver/Transmiter*
UDP	*User Datagram Protocol*
UMTS	*Universal Mobile Telecommunication System*
VGA	*Video Graphics Array*
WAP	*Wireless Application Protocol*
WiMAX	*Worldwide Interoperability for Microwave Access*
WPF	*Windows Presentation Foundation*
WRT	*Web Runtime*
WTLS	*Wireless Transport Layer Security*
XML	*eXtensible Markup Language*

Sumário

1. APRESENTAÇÃO 1

1.1. MOTIVAÇÃO 2

1.2. OBJETIVOS 3

1.3. MÉTODO UTILIZADO 3

1.4. ESTRUTURA DO TRABALHO 4

2. ESTADO DA ARTE DOS SISTEMAS DE AUTOMAÇÃO 6

2.1. HISTÓRICO DOS SISTEMAS DE AUTOMAÇÃO 6

2.2. REDES DE CAMPO 7

2.2.1 TRANSMISSÃO DE DADOS 9

2.2.2 TOPOLOGIAS DE REDE 10

2.2.3 O MODELO MESTRE-ESCRAVO 11

2.3. HART 12

2.4. PROFIBUS PA 14

2.5. FOUNDATION FIELDBUS 17

2.6. FASES DO CICLO DE AUTOMAÇÃO DE UMA PLANTA VERSUS SISTEMAS FIELDBUS 21

2.7. TECNOLOGIA OPC 22

2.8. O PADRÃO CYBEROPC EM SISTEMAS DE AUTOMAÇÃO 25

2.9. TECNOLOGIA EDDL 26

2.10. TECNOLOGIA FDT/DTM 29

2.11. COMPARAÇÃO: FDT/DTM VERSUS EDDL 32

2.12. TECNOLOGIA FDI 32

2.13. GERENCIAMENTO DE ATIVOS (ASSET MANAGEMENT) 33

3. INOVAÇÃO EMERGENTE DOS DISPOSITIVOS MÓVEIS 38

3.1. APLICAÇÕES PARA DISPOSITIVOS MÓVEIS 38

3.2. SISTEMA OPERACIONAIS PARA DISPOSITIVOS MÓVEIS 41

3.2.1. SYMBIAN OS 41

3.2.2. IOS (IPHONE E IPOD TOUCH) 42

3.2.3. ANDROID 42

3.2.4. WINDOWS MOBILE 43

4. PROPOSTA DE ARQUITETURA 44

4.1. ARQUITETURA DE INTEGRAÇÃO 44

4.2. DETALHAMENTO DA ARQUITETURA E IMPLEMENTAÇÕES UTILIZADAS PARA OS TESTES 47

4.2.1. A TEMPORIZAÇÃO DAS MENSAGENS 49

4.2.2. ESCOPO DA IMPLEMENTAÇÃO DO PROTÓTIPO SERVIDOR 52

4.2.3. ESCOPO DA IMPLEMENTAÇÃO DO PROTÓTIPO CLIENTE 56

4.2.4. ESCOPO DOS TESTES 60

4.3. ANÁLISE E VALIDAÇÃO DOS RESULTADOS 69

4.4. ESTUDO DE CASO APLICADO A UMA OPERAÇÃO TÍPICA DE GERENCIAMENTO DE ATIVOS 70

5. CONSIDERAÇÕES FINAIS E CONCLUSÃO 73

6. REFERÊNCIAS BIBLIOGRÁFICAS 74

APÊNDICE A 80

1. APRESENTAÇÃO

Nos últimos anos, a área de automação tem experimentado um momento de inovação e melhorias. Essas melhorias são o resultado do surgimento de novos e modernos equipamentos que possuem embutidos internamente algum tipo de inteligência.

Nos anos 80, os instrumentos de campo utilizavam tecnologias proprietárias, tornando difícil o processo de integração e interoperabilidade quando era necessário reunir equipamentos de diferentes fabricantes. A padronização começou a surgir a partir dos anos 90, quando se deu o início do desenvolvimento de padrões abertos com a formação de grupos de usuários que contribuíram para a criação de protocolos abertos como *HART*, *PROFIBUS* e *Foundation Fieldbus*.

A definição de rede industrial está relacionada aos protocolos de comunicação utilizados para supervisionar e controlar um determinado processo, bem como a troca de informações entre sensores, atuadores, *CLPs*, *Linking Devices*, entre outros (MAHALIK, 2003).

Os protocolos *HART*, *PROFIBUS PA* e *Foundation Fieldbus* são padrões abertos de redes de campo, que possuem aplicações em processos, manufatura e automação e estão consolidados no padrão IEC 61158 - norma esta criada no ano 2000 pelo Comitê Internacional de Eletrotécnica (IEC, 2000).

Os padrões *PROFIBUS PA* e *Foundation Fieldbus* possuem a característica da comunicação puramente digital, enquanto o padrão *HART* possui uma comunicação híbrida: analógica e digital (BERGE, 2002).

Essas redes industriais podem ser interligadas, por exemplo, a sistemas de gerenciamento através de padrões industriais como o *OPC* (*OLE for Process Control*) (HOLLENDER, 2010).

A configuração desses instrumentos é feita a partir de uma estação de engenharia - um computador contendo um software de configuração. Devido à característica dos equipamentos desses protocolos publicarem a informação sobre seus diagnósticos, eles são chamados de instrumentos inteligentes (THOMPSON, 2005).

Os sistemas de diagnóstico e manutenção, conhecidos como gerenciamento de ativos, fazem uso das informações vindas dos instrumentos inteligentes e são capazes de monitorar suas condições e seus autodiagnósticos (BERGE, 2002).

Por outro lado, em outra área do conhecimento, especificamente no domínio da computação móvel, é possível perceber um crescimento do uso de equipamentos portáteis que, ao longo da última década, agregaram várias funcionalidades e também receberam a conotação de equipamentos inteligentes: os *smartphones*.

Os dispositivos móveis chamados *smartphones* são o resultado de um processo de evolução e convergência de dispositivos que começou nas décadas de 80 e 90, a partir das agendas eletrônicas, servindo como uma maneira de armazenar números de telefones ou fazer anotações rápidas, além de suporte para criação de alarmes e compromissos.

Nota-se uma evolução em direção à mobilidade que se inicia pelos computadores pessoais do tipo desktop, avança com o uso dos *notebooks* e *netbooks*, passa pelos computadores ultraportáteis do tipo *Pocket PC*, *PDAs* e *Tablets*, chegando a dispositivos menores do tipo *smartphones*.

Essa variedade de ambientes gera um grande desafio, principalmente, porque as aplicações destinadas aos computadores pessoais, nos últimos dez anos, não puderam prever esse uso simultâneo em plataformas múltiplas (LARYSZ; NEMEC; FASUGA, 2011).

Os desafios estão relacionados a diferentes capacidades de processamento, tamanhos de telas de visualização e interação *IHM* (interface homem-máquina). Uma aplicação para um dispositivo móvel precisa, portanto, adaptar-se a utilizar recursos de maneira limitada em termos de: a) processamento; b) interfaces de entrada; c) *display* com o usuário e d) consumo de energia.

O presente trabalho insere-se no contexto de interface entre os dois mundos: a automação industrial e a computação móvel.

Hoje, os *smartphones* agregam praticamente todas as funções existentes em um computador pessoal e, a cada dia, surgem novos aplicativos executados a partir desses dispositivos aplicados a diversas áreas do conhecimento.

Todavia, especificamente na área de automação industrial, o gerenciamento de ativos pode usufruir dos vários benefícios proporcionados por uma computação móvel.

1.1. Motivação

Os modernos instrumentos de campo contêm embutidos conjuntos de informações que proporcionam grande oportunidade para a gestão de ativos (MULLER et al., 2003).

O uso de redes de campo de protocolos abertos, como *HART*, *PROFIBUS* e *Foundation Fieldbus*, permite a maximização dos ativos da planta em potencial, uma vez que os instrumentos de campo inteligentes são capazes de enviar informações sobre o autodiagnóstico, status, tendências e alarmes. Esse tipo de informação pode ser coletado e classificado por um sistema de gestão de ativos que pode prever algumas interrupções indesejáveis. Dessa forma, o custo com manutenções não programadas pode ser reduzido, aumentando o ciclo de vida do ativo na planta de automação (AKESSON, 1999).

As operações de gerenciamento de ativos também contemplam atividades de calibrações, ajustes e manutenções, que na maioria das vezes, necessitam ser feitas no chão de fábrica. O acesso às redes de automação por meio de dispositivos móveis/*smartphones* torna-se, portanto, um elemento facilitador, pois permite operações em campo sem a necessidade de conexões físicas.

Apesar das inovações emergentes proporcionadas pelos dispositivos portáteis, os estudos relacionados à sua utilização e aplicação na automação industrial ainda são bastante escassos. Nesse sentido, este trabalho propõe uma contribuição para o uso da computação móvel aplicada à área de gerenciamento de ativos, que, na última década, tomou grande destaque e importância no processo produtivo industrial.

1.2. Objetivos

Esse trabalho tem como objetivos:

1) Análise tecnológica e científica (crítica) dos benefícios dos padrões abertos de redes de campo: *HART*, *PROFIBUS PA* e *Foundation Fieldbus*; e uso do seu potencial de informações de diagnósticos aplicado ao gerenciamento de ativos.

2) Proposta e verificação da viabilidade técnica (protótipo) de uma arquitetura capaz de usufruir dos benefícios da mobilidade por meio da integração da computação móvel ao gerenciamento de ativos de uma planta industrial.

1.3. Método utilizado

O método utilizado tem como base um caráter experimental e exploratório, fazendo-se uma análise das características importantes de cada um destes três protocolos abertos (*HART*, *Profibus PA* e *Foundation Fieldbus*) aplicado ao gerenciamento de ativos.

Para assegurar os objetivos propostos, foram desenvolvidas as seguintes atividades:

i) Análise das tecnologias existentes e mais usadas nos sistemas de automação industrial;
ii) Análise de alguns sistemas operacionais e plataformas de desenvolvimento para dispositivos móveis;
iii) Análise das características necessárias à operação de gerenciamento de ativos em um ambiente industrial;
iv) Proposta de arquitetura para integração da computação móvel a um sistema de gerenciamento de ativos;
v) Escolha de 03 plataformas de desenvolvimento para dispositivos móveis para implementação de um protótipo para verificação e validação da arquitetura;
vi) Aplicação do protótipo à solução de um estudo de caso típico da operação de gerenciamento de ativos em um ambiente industrial.

1.4. Estrutura do Trabalho

No primeiro capítulo é apresentado o contexto das áreas de conhecimento nas quais esse trabalho propõe uma interface de integração: a computação móvel e o gerenciamento de ativos de uma planta industrial.

O capítulo 2 aborda: a) o estado da arte dos sistemas de automação e o seu histórico; b) os diferentes padrões abertos de redes de campo e as tecnologias envolvidas para a integração de um sistema de automação industrial; c) uma análise crítica das características de diagnósticos presentes nos protocolos *HART*, *PROFIBUS PA*, *Foundation Fieldbus* e que são importantes para o gerenciamento de ativos.

No capítulo 3 é feita uma análise da inovação tecnológica emergente dos dispositivos móveis.

O capítulo 4, por sua vez, apresenta a proposta de uma arquitetura para integração, levanta características do desenvolvimento de software para plataformas móveis, sua aplicação no gerenciamento de ativos e apresenta os resultados obtidos por meio de testes comparativos feitos com equipamentos simulados e um equipamento real. Esse capítulo ilustra, ainda, um estudo de caso do uso da arquitetura para resolver um problema típico da área de gerenciamento de ativos.

O capítulo 5 faz as considerações finais, apresentando as conclusões e propostas de trabalhos futuros.

Finalmente, o capítulo 6 apresenta uma lista das referências bibliográficas.

2. ESTADO DA ARTE DOS SISTEMAS DE AUTOMAÇÃO

2.1. Histórico dos Sistemas de Automação

Nos anos 60, o início dos sistemas de controle foi marcado pela denominada "era do controle pneumático", onde o controlador ficava localizado no campo e sua operação era local. Os instrumentos pneumáticos, por sua vez, permitiam a transmissão de sinais relacionados às variáveis de processo à distância, possibilitando o uso de controladores em um único local, dando origem às salas de controle de processo. Os transmissores pneumáticos usavam um sinal pneumático variável, linear, modulado de 3 a 15 psi para uma faixa de medida de 0 a 100% de uma variável. Essa faixa de transmissão foi adotada pela *SAMA* (*Scientific Apparatur Markers Association*), Associação de Fabricantes de Instrumentos e também por grande parte dos fabricantes de transmissores e controladores os Estados Unidos (COFFIN, 2010).

Considerando-se os chamados "sistemas de 3 a 15 psi", pode-se observar que o valor mínimo do sinal pneumático não é zero e sim 3psi. Dessa forma, é possível a calibração correta de um instrumento e também a detecção de vazamento de ar na linha de transmissão. Introduz-se, portanto, o conceito do "zero vivo", que tem por base detectar algum tipo de problema caso a grandeza esteja abaixo do valor mínimo.

Na década de 70, o controle passou a ser feito por meio de um sinal elétrico analógico. O sinal era levado para os transmissores no campo e o modelo é chamado de controle direto centralizado (*DDC – Direct Digital Control*). Nessa arquitetura, utilizava-se um sinal de corrente de 4 a 20 mA, mantendo-se ainda o conceito do "zero vivo", uma vez que o valor mínimo adotado é 4 mA. Esse conceito permite, por exemplo, a detecção de um rompimento dos fios caso o valor da variável esteja abaixo do valor mínimo. Ainda nesse cenário, o dispositivo controlador ficava em uma sala de controle.

Ainda ao longo dos anos 70, o *DDC* passou pelos controladores lógicos programáveis (*PLC – Programmable Logic Controllers)*. Contudo, esse modelo possui problemas, pois qualquer falha no controlador seria capaz de causar uma parada em todo o processo. O problema da possibilidade da falha levou à criação de uma nova arquitetura chamada de sistema de controle distribuído (*DCS – Distributed Control Systems*), em 1976. O DCS pode ser considerado como sendo apenas menos centralizado, comparando-o ao DDC, pois uma falha simples pode gerar falhas em cadeia (BERGE, 2002).

Em 1984, os sistemas *DCS*, por sua vez, deram origem ao sistema de controle *fieldbus* (*FCS – Field Control Systems*). Nesse modelo, os instrumentos de campo não são apenas periféricos, mas podem atuar como possíveis controladores (BERGE, 2002).

A Figura 1 ilustra a migração do controle iniciada pelos Sistemas do tipo *DDC*, passando pelos Sistemas *DCS* e, finalmente, chegando aos Sistemas *FCS*.

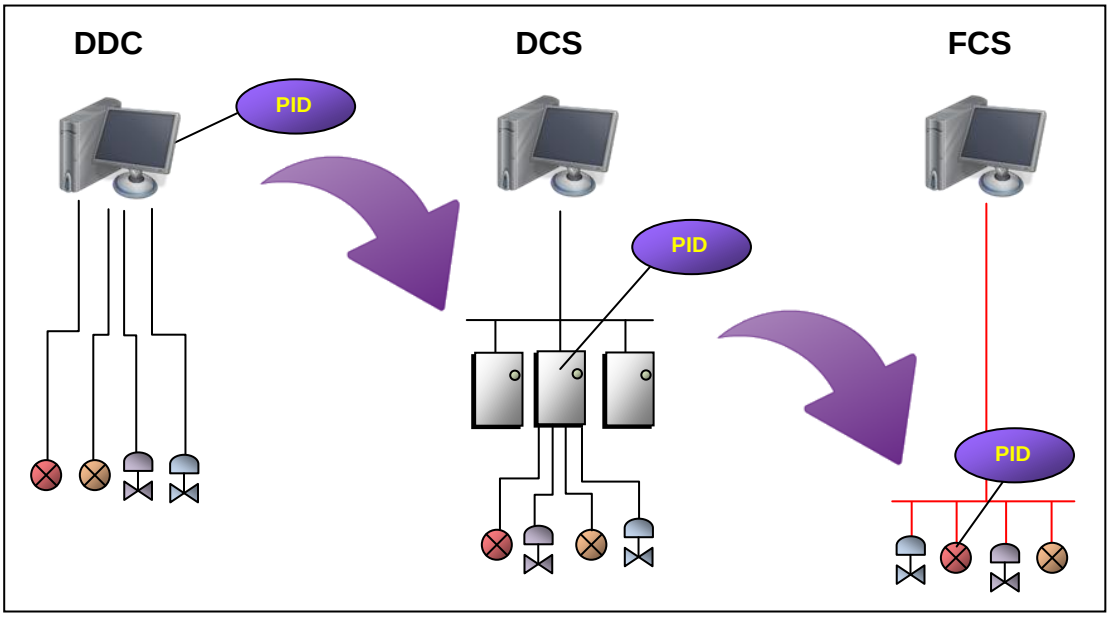

Figura 1- Evolução Tecnológica dos Sistemas de Automação.
Fonte: Regh, Swain e Yangula (1999)

Na década de 90, devido à necessidade de um protocolo totalmente digital, surge o protocolo *fieldbus*. Desta maneira, o *Fieldbus* pode transmitir dados serialmente através do meio de comunicação ou barramento, permitindo uma comunicação bidirecional (INTERNATIONAL ELECTROTECHNICAL COMMISION, 2000).

Atualmente, os barramentos de campo – *fieldbuses* – têm se tornado pontos importantes no ambiente industrial graças às vantagens como distribuição do controle, facilidade de expansão, interoperabilidade e intercambiabilidade, segurança, determinismo e possibilidade de acesso remoto (BRANDÃO, 2005).

2.2. Redes de Campo

A evolução tecnológica proporcionou grandes benefícios como eficiência, flexibilidade e agilidade por meio da interligação de vários sistemas e equipamentos em rede.

Essa abordagem foi estendida para a área de automação industrial através das redes de campo conhecidas como *fieldbuses* (TIPSUWAN & CHOW, 2003).

Conceitualmente, um sistema *fieldbus* pode ser definido da seguinte forma:

> "[...] um sistema distribuído composto por dispositivos de campo e equipamentos de controle e de monitoramento integrados em um ambiente físico de uma planta ou uma fábrica. Os dispositivos do *fieldbus* trabalham em conjunto para realizar entradas/saídas e controle em operações e processos automáticos" (FIELDBUS FOUNDATION, 1999a, p.1)

Mahalik (2003) comenta que "*fieldbus* é um sistema digital, bidirecional de comunicação *multidrop*, que se baseia na estrutura de camadas do modelo OSI (*Open System Interconnection*) e pode ser considerado a coluna vertebral de um sistema distribuído em tempo real".

Thomesse (1998), por sua vez, afirma que "*fieldbus* é uma rede para conexão de dispositivos de campo, tais como: sensores, atuadores, controladores de campo (*PLC – Programable Logical Controller*), reguladores, controladores de percursos, etc."

De uma maneira mais geral, pode-se, então, entender redes de campo como redes locais de comunicação, bidirecionais, concebidas e utilizadas para interligar instrumentos de medida, equipamentos de controle e sistemas de automação.

A infra-estrutura de comunicação digital de uma indústria que possui processos produtivos automáticos é demonstrada na Figura 2. As redes de campo estão na base da pirâmide mostrada.

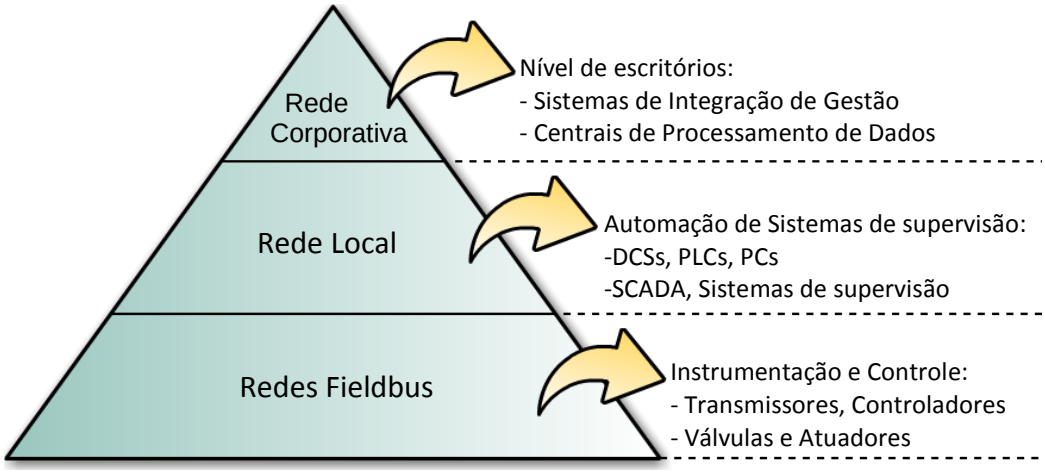

Figura 2 - Representação da infraestrutura de comunicação industrial
Fonte: Brandão (2005)

Os instrumentos de campo conectam-se entre si de acordo com o protocolo de comunicação utilizado no processo de controle. Pode-se deduzir, portanto, os *"fieldbuses"* são utilizados para a comunicação, supervisão e controle de um determinado processo, como a troca de informações entre sensores, atuadores, *CLPs*, *LinkingDevices*, entre outros.

Os protocolos *HART*, *PROFIBUS PA* e *Foundation Fieldbus* são padrões abertos de redes de campo, que possuem aplicações em processos, manufatura e automação e estão consolidados no padrão IEC 61158.

Os instrumentos *HART* são capazes de suportar o legado de redes de automação analógicas (4 a 20 mA). Por sua vez, os instrumentos *PROFIBUS PA* e *Foundation Fieldbus* atendem a características mais rígidas de operação, podendo ser usados em ambientes cujos requisitos contemplam a segurança intrínseca.

2.2.1 Transmissão de Dados

Por definição, uma rede é constituída por um conjunto de módulos processadores, que por sua vez possuem capacidade de trocar informações e compartilhar recursos, interligados por um sistema de comunicação composto por enlaces físicos (meios de transmissão) e de um conjunto de regras com o objetivo de organizar a comunicação, denominados protocolos. (TANEMBAUM, 1997).

Ainda, de acordo com Tanembaum (1997), podem-se enumerar alguns conceitos importantes quando se deseja comparar diferentes sistemas:

a) ***Retardo de Acesso***: é o intervalo de tempo decorrido desde o momento em que uma mensagem a ser transmitida é gerada por um nó da rede até o momento em que ela consegue acessar o meio para transmissão, sem que haja colisão no meio.

b) ***Retardo de Transmissão***: é o intervalo de tempo decorrido desde o início da transmissão de uma mensagem por um nó de origem até o momento em que ela chega ao nó de destino.

c) ***Retardo de Transferência***: é a soma entre o Retardo de Transmissão e o Retardo de Acesso.

d) ***Desempenho***: é a capacidade efetiva de transmissão da rede. Tem como base medidas como: retardo de transferência, velocidade, fluxo, etc. O desempenho é influenciado por fatores como, por exemplo: escolha adequada da arquitetura e estrutura de conexão, protocolo de comunicação e meio de transmissão.

e) ***Compatibilidade ou Interoperabilidade***: é a capacidade que um sistema possui de se interligar a dispositivos de diferentes fabricantes.

2.2.2 Topologias de Rede

Uma rede é constituída por dois ou mais dispositivos juntos por meio de um *link*. (FOROUZAN, 2004). A topologia refere-se à forma como os enlaces físicos e os nós da comunicação estão organizados. Forouzan (2004) afirma, ainda, que é possível classificar a comunicação quanto à ligação física em ponto a ponto e multiponto:

a) Ponto a Ponto: um enlace que liga dois dispositivos únicos. Toda a capacidade do *link* é reservada para a comunicação entre esses dois dispositivos.

b) Multiponto: Uma conexão multiponto (*multipoint* ou *multidrop*) é aquela em que mais de dois dispositivos compartilham um único *link*.

A Figura 3 ilustra as opções de conexão física ponto a ponto e multiponto. É possível verificar que o meio físico *link* pode ser realizado tanto por meio de fios (*wired*) quanto sem fios (*wireless*).

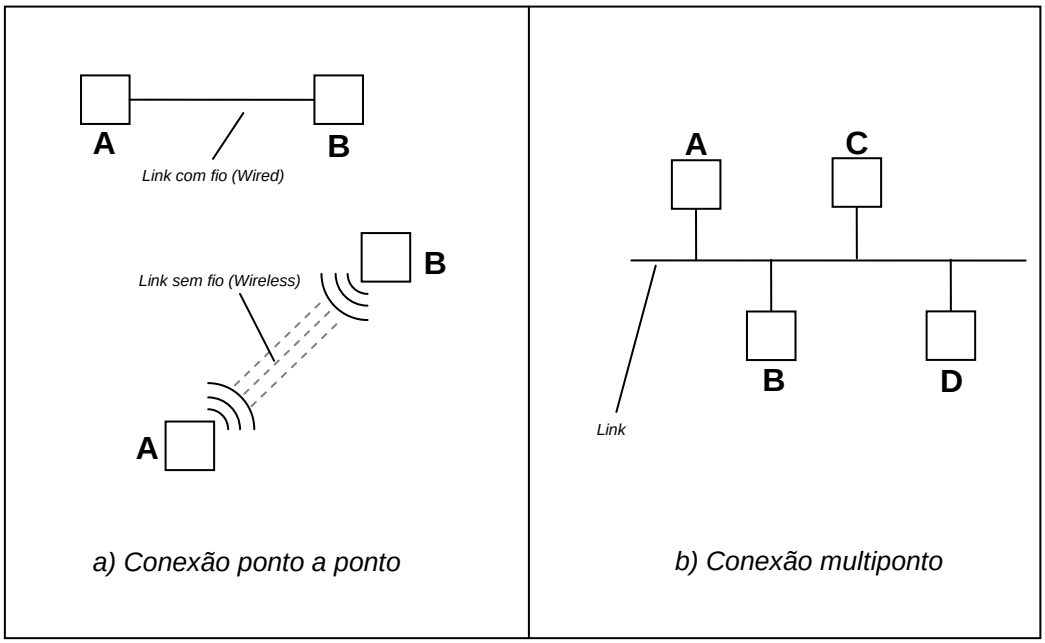

Figura 3 - Topologia física de conexões ponto a ponto e multiponto
Fonte: Forouzan (2004), com adaptações

Segundo Comer & Droms (2004), as topologias de redes mais utilizadas são multiponto do tipo:

a) Estrela;

b) Barramento;

c) Anel;

d) Malha.

A Figura 4, a seguir, ilustra as topologias de rede citadas:

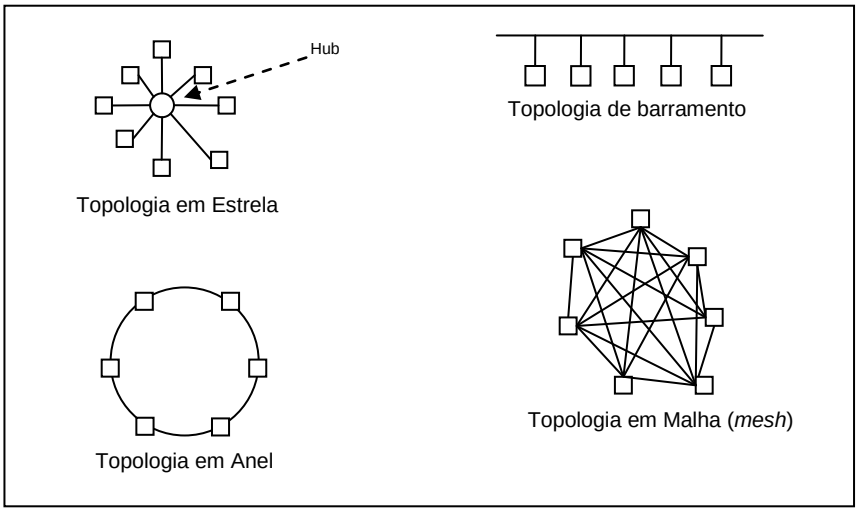

Figura 4 - Topologias de redes
Fonte: Comer & Droms (2004), com adaptações

2.2.3 O Modelo Mestre-escravo

Nas redes de controle há um padrão de arquitetura de comunicação chamado modelo *master-slave*. Esse padrão, por sua vez, é baseado em dois tipos distintos de elementos em uma rede: o nó *master*(mestre) e o nó *slave* (escravo).

Segundo esse modelo, o dispositivo ou processo, definido como “mestre”, é usado para controle de um ou mais dispositivos ou processos, definidos como “escravos”. Uma vez estabelecida a relação mestre-escravo, a direção de controle é sempre do mestre para o escravo (ZHANG, 2008).

De acordo com Zhang (2008), o elemento mestre é responsável por dividir as tarefas entre os escravos disponíveis, e mantê-los em funcionamento sem necessariamente usar de muitos recursos de comunicação. Se o elemento escravo estiver disponível e houver tarefas a serem realizadas, o mestre irá designá-las aos escravos, caso contrário, os escravos serão colocados no modo de espera aguardando a chegada de novas tarefas.

Com base no modelo *master-slave*, o protocolo *Master-Slave/Token-Passing* (MS/TP) foi desenvolvido para automação predial e sistemas de controle usando um único microprocessador com a *Universal Asynchronous Receiver/Transmiter* (UART). As redes

MS/TP podem ser configuradas como uma rede *master/slave*, uma rede *peer-to-peer token passing*, ou uma rede mista (combinação de mais de um tipo). O *token* é transmitido com a finalidade de regular o acesso ao meio, circulando de um nó mestre a outros de acordo com o endereço lógico. (LIU; REN, 2007).

Como exemplos de protocolos que utilizam o modelo *master-slave*, pode-se citar HART e PROFIBUS PA, os quais serão detalhados adiante.

2.3. HART

O protocolo *HART* surgiu com a Fisher Rosemount em 1980. *HART* é o acrônimo de "*Highway Addressable Remote Transducer*". Em 1990 um grupo de usuários fundou o chamado *HART Users Group* que foi o começo do *HART* como um padrão aberto. Em junho de 1993, o *HART Users Group* tornou-se a *HART Communication Foundation* (HART COMMUNICATION FOUNDATION, 2010).

Uma grande vantagem desse protocolo é a possibilidade do uso de instrumentos inteligentes através dos cabos 4-20 mA tradicionais. A velocidade baixa de comunicação (1200 bps) permite que os cabos normalmente usados na instrumentação analógica possam ser mantidos. Os dispositivos que executam essa comunicação híbrida são denominados *smart*.

O sinal *HART* possui modulação *FSK* (*Frequency Shift Key*) e é sobreposto ao sinal analógico de 4-20 mA. A transmissão do sinal digital é obtida pela seguinte convenção: o dígito 1 é representado por um sinal de 1mA pico a pico na frequência de 1200 Hz e o dígito 0 é representado pela fequência de 2400Hz.

A comunicação é bidirecional, *half-duplex* e realizada nos sentidos:

a) *Master to Slave*;
b) *Slave to Master*.

O protocolo *HART* introduz o conceito de comandos, onde a resposta para os mesmos retorna juntamente com uma informação genérica de diagnóstico. Os comandos são enviados e recebidos de acordo com o *Frame* mostrado na Figura 5.

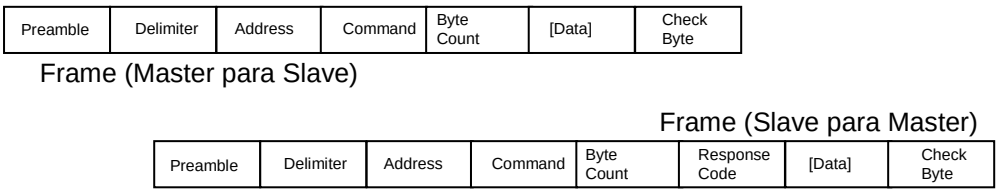

Figura 5 - Formato do Frame HART
Fonte: HART COMMUNICATION FOUNDATION (1996)

Uma grande vantagem introduzida por esse padrão é a possibilidade de verificação da integridade da mensagem, por meio do campo *Check Byte*. Esse campo contém um valor composto de um cálculo matemático feito a partir de todos os dados da mensagem.

As mensagens enviadas no sentido *Slave* para *Master* possuem um campo específico para informações de diagnósticos: o *Response Code*. Esse, por sua vez, possui informações a respeito do status de execução de um comando solicitado, bem como o envio das informações básicas e estendidas de diagnósticos. O *Byte* de *Status* do *Response Code* contém as informações descritas na Tabela 1.

Tabela 1 - Significado do Byte de Status no Protocolo HART.

Bit	Status	Descrição
#7	Field Device Malfunction	Indica um erro que impede o funcionamento adequado do instrumento de campo. Quando este bit está sinalizado, os hosts devem considerar uma condição de alarme e o instrumento não deve ser usado para aplicações de controle de processo.
#6	Configuration Changed	Este bit indica que algum item de configuração no dispositivo de campo foi modificado. Essa indicação é mantida mesmo com falhas de energia e só pode ser redefinida pelo host por meio do comando 38. Esse status é usado juntamente com um contador para permitir ao host monitorar a configuração do instrumento de campo.
#5	Cold Start	Este status é definido quando acontece um reset do instrumento de campo. O host pode usar esse status para detectar uma perda de energia no instrumento de campo.
#4	More Status Available	Este bit indica que, além do status padrão, o instrumento de campo possui mais informações de diagnósticos.
#3	Analog Output Fixed	Este bit indica que a corrente está em um valor fixo e não irá responder a alterações do processo. Isso indica que o host não deverá usar o sinal de corrente para uso no controle do processo.
#2	Analog Output Satured	Os instrumentos HART calculam de maneira contínua o valor digital das suas variáveis. Esses cálculos compreendem um valor válido entre os limites inferior e superior. Na maioria dos casos, o sinal de corrente representa somente uma porção da faixa do valor. Quando o valor digital exceder a faixa de valor, o sinal de corrente irá saturar. Esse status é definido, portanto, quando esta condição estiver presente.
#1	Non-Primary Variable Out of Limits	Este status é definido quando qualquer outra variável, diferente do sinal de corrente, atinge o seu limite inferior ou superior. Os limites podem ser elétricos (por exemplo, um sensor elétrico em um transmissor) ou mecânicos (por exemplo, um atuador no seu fim de curso – indicado por um sensor mecânico de fim de curso).
#0	Primary Variable Out of Limits	Este status é definido quando o sinal de corrente atinge o seu limite superior ou inferior.

Fonte: HART COMMUNICATION FOUNDATION (1997)

Essas informações de status podem, portanto, ser enviadas a um sistema de gerenciamento de ativos, que fará uso desses diagnósticos de forma a otimizar o uso do instrumento no processo industrial.

2.4. Profibus PA

O protocolo *PROFIBUS* surgiu em 1987 na Alemanha, a partir da colaboração de 15 empresas e 5 institutos de pesquisa financiados pelo governo alemão, com a contribuição de indústrias como Siemens, Bosh e Klocker-Moeller. Em 1991, os resultados desses trabalhos deram origem à publicação da norma DIN 19254 (BRANDÃO, 2005).

PROFIBUS é um padrão aberto *fieldbus* que tem aplicação em processos, manufatura e automação e é consolidado pela Norma IEC 61158. A arquitetura do *PROFIBUS* possui três variantes: *Profibus DP* (*Decentralized Periphery*), *Profibus FMS* (*Field Message Specification*) e *Profibus PA* (*Process Automation*), essa última é usada para automação de processos (PROFIBUS INTERNACIONAL, 2010).

A comunicação é bidirecional, *half-duplex* e realizada nos sentidos: *Master to Slave e Slave to Master.*

Um instrumento de campo *Profibus PA* é capaz de realizar uma ou mais aplicações críticas, como, por exemplo, fazer a aquisição de dados por meio de um sensor enquanto, simultaneamente, executa um algoritmo de controle de processo.

Cada aplicação é composta de um conjunto de elementos modelado de acordo com o conceito de blocos funcionais (*FB – Function Blocks)* (PROFIBUS INTERNACIONAL, 2009).

Um bloco funcional ou *FB* (*Function Block*) pode ser entendido como uma função de automação básica executada por uma aplicação, de tal maneira a abstrair os detalhes específicos dos níveis de hardware e de rede de comunicação. Os blocos funcionais processam as informações que chegam à sua entrada, e geram saídas que poderão ser utilizadas por outros blocos.

Existem dois tipos diferentes de equipamentos que executam uma aplicação do tipo *Profibus PA Function Block*:

a) Equipamentos compactos para uso em áreas de controle de processo, como, por exemplo, um transmissor ou atuador.

b) Equipamentos modulares compostos de entradas e saídas binárias. Esses são frequentemente usados para ligar e desligar válvulas.

Em ambos os tipos, um equipamento é representado pelo *Physical Block* bem como as funções de gerenciamento e parâmetros. As variáveis e parâmetros de um equipamento ou módulo são estruturados em blocos. Esses blocos apresentam algumas vistas dos instrumentos aplicados a atividades de comissionamento, operação e diagnósticos.

No *Profibus PA*, três tipos de blocos são criados a partir de um perfil de parâmetros: a) *Function Block*; b) *Transducer Block*; e c) *Physical Block*. Os blocos do equipamento estão disponíveis em uma área de memória do instrumento, chamada *Directory* (ou Diretório) que contém referências para todos os outros blocos. A Figura 6 ilustra esse modelo.

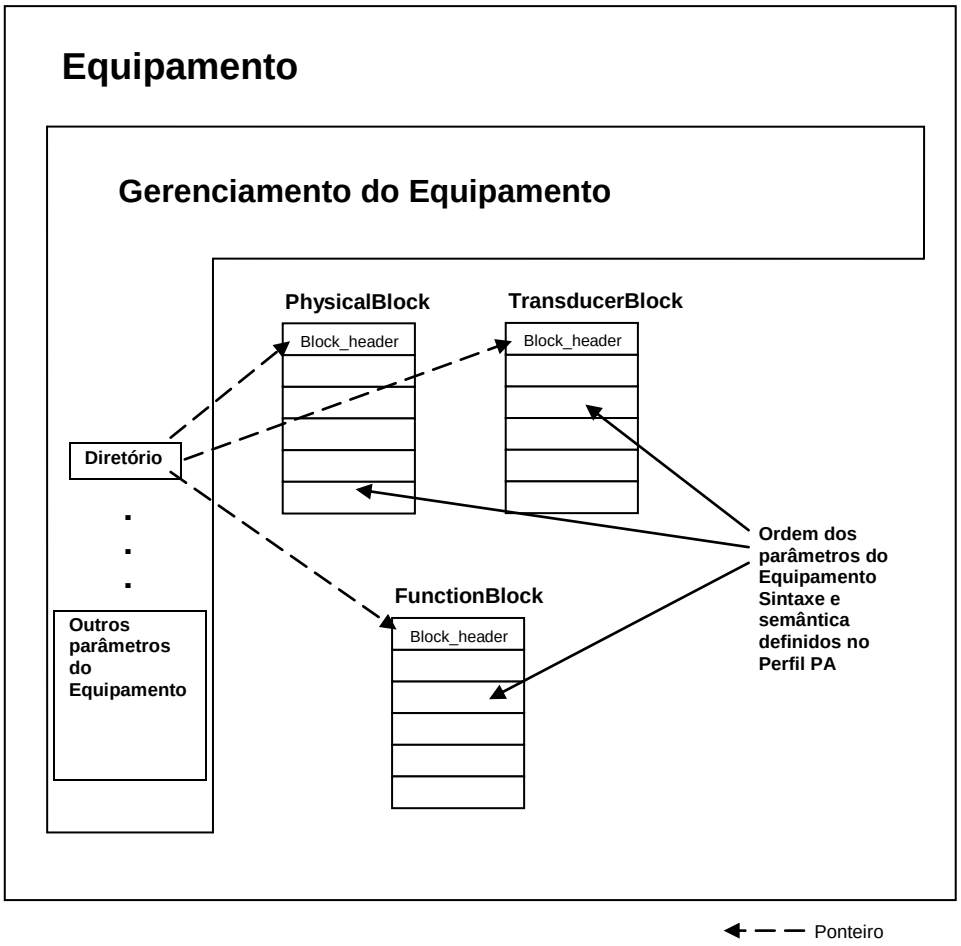

Figura 6 - Relação entre Blocos, Parâmetros de Blocos e Diretório
Fonte: PROFIBUS INTERNACIONAL (2009)

Os *Function Blocks* (*FB*) descrevem as funções dos equipamentos quando estão presentes em um sistema de automação. Os *Transducer Blocks* (*TB*), por sua vez, disponibilizam o acesso aos parâmetros necessários para configuração e ajuste dos equipamentos.

Durante a operação existe, normalmente, um *Transducer Block* e um *Function Block* conectados. Essa conexão funciona como referência, ou seja, o *FB* possui um ponteiro para o

TB e seus parâmetros. Já o *Physical Block* (*PB*) descreve os parâmetros e funções necessárias à operação do próprio hardware do instrumento. Portanto, cada instrumento *Profibus PA* contém um *Physical Block* (*PB*).

Os parâmetros do *PhysicalBlock* (*PB*), interessantes para aplicação em gerenciamento de ativos, são apresentados em detalhes na Tabela 2.

Tabela 2 - Descrição dos parâmetros do Physical Block

Índice Relativo	Status	Descrição
. . .		
13	DIAGNOSIS	Este parâmetro possui informações detalhadas sobre o instrumento, codificada de acordo com o padrão bitwise, ou seja, é possível sinalizar mais que um diagnóstico ao mesmo tempo. Se o bit MSB do byte 4 estiver sinalizado, significa que mais informações estão disponíveis no próximo parâmetro (DIAGNOSIS_EXTENSION)
14	DIAGNOSIS_EXTENSION	Este parâmetro possui informações adicionais, específicas do fabricante do instrumento. É seguido o padrão bitwise. Portanto, é possível sinalizar mais que um diagnóstico ao mesmo tempo.
15	DIAGNOSIS_MASK	Definição de suporte ao diagnóstico padrão: 0 – não suportado. 1 – suportado.
16	DIAGNOSIS_MASK_EXTENSION	Definição de suporte ao diagnóstico estendido: 0 – não suportado. 1 – suportado.
. . .		

Fonte: PROFIBUS INTERNACIONAL (2009), com adaptações

O parâmetro *DIAGNOSIS* do *Physical Block* (*PB*) é composto de 4 bytes, conforme a Figura 7, a seguir:

Octeto 1				Octeto2				Octeto3				Octeto4			
Bit 7	. . .	. . .	Bit 0	Bit 7	. . .	. . .	Bit 0	Bit 7	. . .	. . .	Bit 0	Bit 7	. . .	. . .	Bit 0

Figura 7 - *OctectString* dos 4 bytes do Parâmetro *DIAGNOSIS* do *Physical Block*
Fonte: PROFIBUS INTERNACIONAL (2009)

A convenção utilizada para os bits desse parâmetro é a seguinte:

0 – não sinalizado;

1 – sinalizado.

A Tabela 3 ilustra os detalhes da codificação do parâmetro *DIAGNOSIS* do *Physical Block.*

Tabela 3 - Codificação do Parâmetro DIAGNOSIS do PhysicalBlock

Octeto	Bit	Mnemônico	Descrição	Indicação
1	0	DIA_HW_ELECTR	Falha no hardware eletrônico	R
	1	DIA_HW_MECH	Falha no hardware mecânico	R
	2	DIA_TEMP_MOTOR	Temperatura do motor muito alta	R
	3	DIA_TEMP_ELECTR	Temperatura eletrônica muito alta	R
	4	DIA_MEM_CHKSUM	Erro de memória	R
	5	DIA_MEASUREMENT	Falha na medição	R
	6	DIA_NOT_INIT	Device não inicializado (não auto-calibração)	R
	7	DIA_INIT_ERR	Auto-calibração falhou	R
2	0	DIA_ZERO_ERR	Erro do ponto zero (posição limite)	R
	1	DIA_SUPPLY	Fornecimento de energia falhou (elétricos/pneumáticos)	R
	2	DIA_CONF_INVAL	Configuração não válida	R
	3	DIA_WARMSTART	Deve estar ativo após uma partida ou após um FACTORY_RESET = 2506 ter sido executado	A
	4	DIA_COLDSTART	Deve estar ativo após um FACTORY_RESET = 1 ter sido executado	A
	5	DIA_MAINTENANCE	Manutenção requerida	R
	6	DIA_CHARACT	Caracterização inválida	R
	7	IDENT_NUMBER_VIOLATION	Definido para 1 (um) se o Ident_Number executando a transferência cíclica de dados e o valor doparametro IDENT_NUMBER_SELECTOR do PhysicalBlock não correspondem. Se o IDENT_NUMBER_SELECTOR = 127 (modo de adaptação), então o bit IDENT_NUMBER_VIOLATION de diagnóstico não é definido	R
3	0..7	Reservado	Reservado	
4	0..6	Reservado	Reservado	
4	7	EXTENSION_AVAILABLE	Estão disponíveis mais informações de diagnósticos	

Indicação:
R – Indicação permanece ativa enquanto a razão para a mensagem existe
A - A indicação ter que estar presente no mínimo por 10s e deve ser desativada por não mais que 10s após a ação estar finalizada.
Fonte: PROFIBUS INTERNACIONAL (2009)

Assim como nas redes *HART*, essas informações de diagnósticos das Redes *Profibus PA* podem, portanto, ser utilizadas por um sistema de gerenciamento de ativos.

2.5. Foundation Fieldbus

Em 1994, a organização *Fieldbus Foundation*, composta pela união de duas organizações *fieldbus:* a *ISP* (*Interoperable System Protocol*) e uma divisão da *WorldFIP*

francesa, mais um grupo de 85 empresas, fizeram a proposta do protocolo *Foundation Fieldbus* (ZHANG, 2008)

O protocolo *Foundation Fieldbus* é um padrão aberto de comunicação digital, em série e bidirecional que permite a conexão entre equipamentos "*fieldbus*" como sensores, atuadores e controladores. O *Foundation Fieldbus* permite o controle distribuído em campo.

A tecnologia é controlada pela *Fieldbus Foundation*, organização não lucrativa, hoje composta de mais de 150 dos principais fornecedores e usuários de controle e instrumentação do mundo. O protocolo mantém muitas características operacionais do padrão analógico (4 a 20 mA), como dispositivos alimentados por um único par de fios e ainda recursos de segurança intrínseca, além de vários benefícios adicionais.

Os benefícios podem ser enumerados como: a) interoperabilidade; b) dados de processo mais completos; c) vista expandida do processo; d) melhor segurança da planta; e) redução de custos de cabeamento e f) redução dos custos efetivos de manutenção por meio do uso da manutenção preditiva (BERGE, 2002).

Por meio da interoperabilidade, um dispositivo *FF* (*Foundation Fieldbus*) pode ser substituído por um dispositivo similar com maior funcionalidade de outro fornecedor na mesma rede *FF*, mantendo as características originais. Os dispositivos podem transmitir e receber a informação de multivariáveis, comunicando-se diretamente um com o outro sobre o barramento *FF*, permitindo que novos dispositivos sejam adicionados ao barramento sem interromper o controle. A interoperabilidade entre equipamentos é provida por meio da tecnologia de descrição de dispositivos chamada *ElectronicDeviceDescription* (*EDD*).

O protocolo define ainda, de acordo com a norma IEC 61784, três *profiles* ou tipos: *H1*, *H2* e *HSE* (*High Speed Ethernet*) (FELSER; SAUTER, 2002).

O *FF H1* é um protocolo digital, que trabalha a 31,25 Kbps, onde: a) as informações são transmitidas em forma de mensagens; b) serial (transmissão bit a bit); c) bidirecional *half-duplex* e d) *multidrop* (comunicação entre os vários equipamentos conectados à rede) (ZHANG, 2008).

Segundo Zhang (2008), o *FF HSE* baseia-se no protocolo Ethernet de camada física e trabalha a uma taxa de 100 Mbits/s entre dispositivos não alimentados pelo barramento. Nas redes *HSE*, são possíveis quatro tipos de dispositivos:

a) *LinkingDevice*: estabelece a comunicação entre a rede *HSE* e um ou mais canais *H1*;
b) *Ethernet Field Device*: dispositivo com a camada de aplicação de blocos funcionais ligado diretamente à rede HSE;

c) *I/O Gateway*: permite o acesso a equipamentos de outras tecnologias, diferentes da *FF*;
d) *Host*: é o dispositivo pelo qual o operador acessa a rede, por meio de aplicativos de configuração e controle.

O protocolo *FF H1* pode ser conectado a um backbone HSE para uso em grandes sistemas. O *HSE* suporta toda a gama de capacidades *fieldbus*, incluindo os *Standard Functions Blocks*, DDs (*Device Description*) bem como os *FFBs* (*Flexible Function Blocks*) para o controle avançado em aplicações do tipo discretas, híbridas e batelada (FIELDBUS FOUNDATION, 1999a).

Outra característica do *Fieldbus Foundation* é sua capacidade intrínseca de distribuir o controle da aplicação pela rede industrial, compartilhando com os instrumentos a "inteligência" do controle, ao contrário de outros protocolos que concentram este controle em *CLPs*, por exemplo. A distribuição permite uma maior flexibilidade no design da estratégia de controle (BERGE, 2002).

As funções de comunicação permitem o acesso aos dados de autodiagnóstico, providos pelos instrumentos inteligentes, e sua análise em softwares como *Management Information Systems* (MIS) e *Asset Management Systems* (AMS).

O *Fieldbus Foundation* foi concebido para ser interoperável entre diversos fabricantes, e as funções dos instrumentos foram padronizadas. Esses instrumentos, por sua vez, possuem os chamados blocos funcionais (*Function Blocks – FB*). A interligação entre os blocos funcionais define a estratégia de controle e programação do processo a ser controlado.

Os blocos funcionais podem ser entendidos como elementos de software capazes de transformar, por meio da modelagem de algoritmos paramétricos, parâmetros entrada em parâmetros de saída. (FIELDBUS FOUNDATION, 1999a)

Os parâmetros e variáveis que controlam um bloco funcional são caracterizados por:

a) nome ou mnemônico;
b) tipo de dado (variável simples: booleano; estruturas; etc);
c) tipos de uso (entrada, saída ou internos);
d) modo de armazenamento (estático ou dinâmico);

Juntamente com os parâmetros de entrada ou saída, há um *Byte* de *Status*, composto pela seguintes partes: qualidade, *sub-status* e limite.

A Figura 8, a seguir, ilustra a composição do Byte de Status FF.

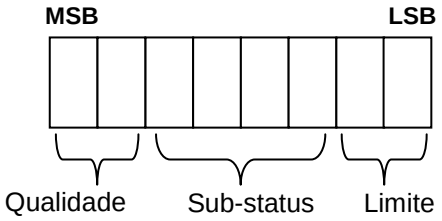

Figura 8 - Byte de Status FF
Fonte: PANTONI (2006)

A qualidade do *Byte* de *Status* pode ser dos tipos:

a) *Good (Cascade)* – O dado é confiável e pode ser usado no controle do tipo cascata.
b) *Good (Non-cascade)* – O dado é confiável, porém o bloco não pode ser usado no controle do tipo cascata.
c) *Uncertain* – O dado possui qualidade duvidosa. É possível haver erros provenientes da aquisição ou no processo de cálculos.
d) *Bad* – O dado não possui um valor válido, portanto, não deve ser utilizado.

Por sua vez, o sub-status, complementar à qualidade, é usado em alarmes no início ou finalização de malhas de controle do tipo cascata. Finalmente, o limite define se o valor é ou não limitado, ou seja, se foi restringido por extrapolar valores pré-determinados.

Entre os parâmetros dos blocos funcionais *FF*, é possível observar o Bloco de Erro ou BLOCK_ERR, que é interessante para uma aplicação em gerenciamento de ativos. Esse parâmetro reflete o status de erro dos componentes de hardware ou software associados com o impacto direto na operação correta do bloco. A tabela 4, a seguir, ilustra os valores possíveis:

Tabela 4 - Valores possíveis para o BLOCK_ERR

Bit	Status
0	Other (LSB)
1	Block Configuration Error
2	Link Configuration Error
3	Simulate Active
4	Local Override
5	Device Fault State Set
6	Device Needs Maintenance Soon

Continua na próxima página...

Continuação da página anterior:

Bit	Status
7	Sensor Failure detected by this block / process variable has a status of BAD, Sensor Failure
8	Output Failure detected by this block / back calculation input has a status of BAD, Device Failure
9	Memory Failure
10	Lost Static Data
11	Lost NV Data
12	Readback Check Failed
13	Device Needs Maintenance Now
14	Power-up
15	Out-of-Service (MSB)

Fonte: FOUNDATION FIELDBUS (1999c)

A característica única da arquitetura *Foundation Fieldbus*, que assegura a interoperabilidade entre equipamentos é o uso de uma camada de usuário baseada em blocos e *DDs* (*Device Description*). A camada de usuário (*User Layer*) define a *FBAP* (*Function Block Application Process*) por meio do uso de *Resource Blocks*, *Function Blocks*, *Transducer Blocks*, Sistemas de Gerenciamento, Gerenciamento de Rede e Tecnologia *DD*.

O protocolo *Foundation Fieldbus* suporta, ainda, o conceito de Alarmes e Eventos. Segundo norma FF, alarmes e eventos são comumente conhecidos como alertas e representam alterações no estado das aplicações do tipo *Function Block* (FOUNDATION FIELDBUS, 1999c).

Os objetos de alertas são usados para comunicar um evento aos outros equipamentos. Esses alertas, por sua vez, podem ser enviados a sistemas de gerenciamento de ativos para efeito de reconhecimento de alarmes, satisfazendo os requisitos de monitoração de falhas no processo.

2.6. Fases do Ciclo de Automação de uma Planta versus Sistemas Fieldbus

Independentemente do protocolo especificado para o projeto, o ciclo de automação de uma planta tem início na chamada fase de engenharia, anterior à operação propriamente dita. Nessa fase, são feitas as configurações de parâmetros dos instrumentos, desenvolvidas as estratégias das malhas de controle, intertravamento e criação de lógicas (THEURICH; LEHMANN; WOLLSCHLAEGER, 2010).

A fase posterior é a operação da planta, que tem seu início no chamado *startup*. A partir desse momento, o processo produtivo entrará em funcionamento, realizando o controle conforme projetado na fase de engenharia.

Segundo PANTONI (2006), existem três categorias de softwares de IHM inseridos na automação industrial baseados em *fieldbus*:

- Os configuradores de dispositivos: usados na fase de engenharia para programação da estratégia de controle e *startup* da planta;

- Os sistemas supervisores: usados na operação da planta, são responsáveis por monitorar as variáveis do processo e informar a condição de operação por meio de telas de sinóticos do próprio processo;

- Os sistemas de gerenciamento de ativos ou *asset management*: usados para gerenciar a planta com a finalidade de se evitar as paradas não programadas do processo. Esses tipos de sistema são capazes de monitorar a condição dos equipamentos, possibilitando operações de ajustes finos, calibrações, manutenções nos instrumentos e serão abordados em mais detalhes adiante.

Hoje, as tecnologias mais utilizadas para integração dos sistemas *Fieldbus* são baseadas em padrões abertos, como, por exemplo: *OPC* (*OLE for Process Control), EDDL* (*Enhanced Device Description Language*) e *FDT/DTM* (*Field Device Tool / Device Type Manager*). Esses padrões, portanto, serão detalhados a seguir.

2.7. Tecnologia OPC

Em 1995, algumas empresas se reuniram com a finalidade de estabelecer um padrão baseado na tecnologia *OLE/DCOM* da Microsoft, para acesso a dados por meio do sistema operacional Windows. Este grupo sem fins lucrativos é composto por várias empresas e gerenciado pela *OPC Foundation*. O padrão *OPC* estabelece regras para que sejam desenvolvidos sistemas com interfaces padrões para comunicação dos dispositivos de campo (*CLPs*, atuadores, sensores) com sistemas de automação (*SCADA, MES, Asset Management*).

A tecnologia *OLE* (*Object Linking and Embedding*) foi desenvolvida pela Microsoft em meados de 1990 para integração entre diferentes aplicações dentro da plataforma Windows.

O *DCOM* (*Distributed Componente Object Model*) surgiu com o Windows NT é um conjunto de definições que permite a implementação de aplicações distribuídas no modelo de arquitetura cliente-servidor.

A primeira especificação produzida foi publicada em agosto de 1996, denominada *OPC Specification Version 1.0*. Em setembro de 1997 foi liberada a primeira atualização da especificação *OPC* que passou a ser chamada de *OPC Data Access Specification Version 1.0A* (OPC FOUNDATION, 2010). Hoje, existem as seguintes especificações publicadas ou em processo de aprovação:

• *OPC Overview* – Descrição geral dos campos de aplicação das especificações *OPC*;

• *OPC Common Definitions and Interfaces* – Definição das funcionalidades básicas para as demais especificações;

•*OPC Data Access Specification* – Definição da interface para leitura e escrita de dados de tempo real;

•*OPC Alarms and Events Specification* – Definição da interface para monitoração de eventos;

•*OPC Historical Data Access Specification* – Definição da interface para acesso a dados históricos;

•*OPC Batch Specification* – Definição da interface para acesso aos dados de processos por batelada (batch). Esta especificação é uma extensão da OPC Data Access Specification;

•*OPC Security Specification* – Definição da interface para utilização de políticas de segurança;

•*OPC and XML*– Integração entre OPC e XML para aplicações via Internet (web);

•*OPC-UA* (*Unified Architecture*) – Definição de um padrão de integração independente de plataformas.

A Figura 9 ilustra o diagrama das especificações do padrão *OPC* e a Figura 10 mostra a divisão em camadas da última versão, liberada em 2009, que é a *OPC-UA* (*Unified Architecture*), proposta para ser independente de plataforma.

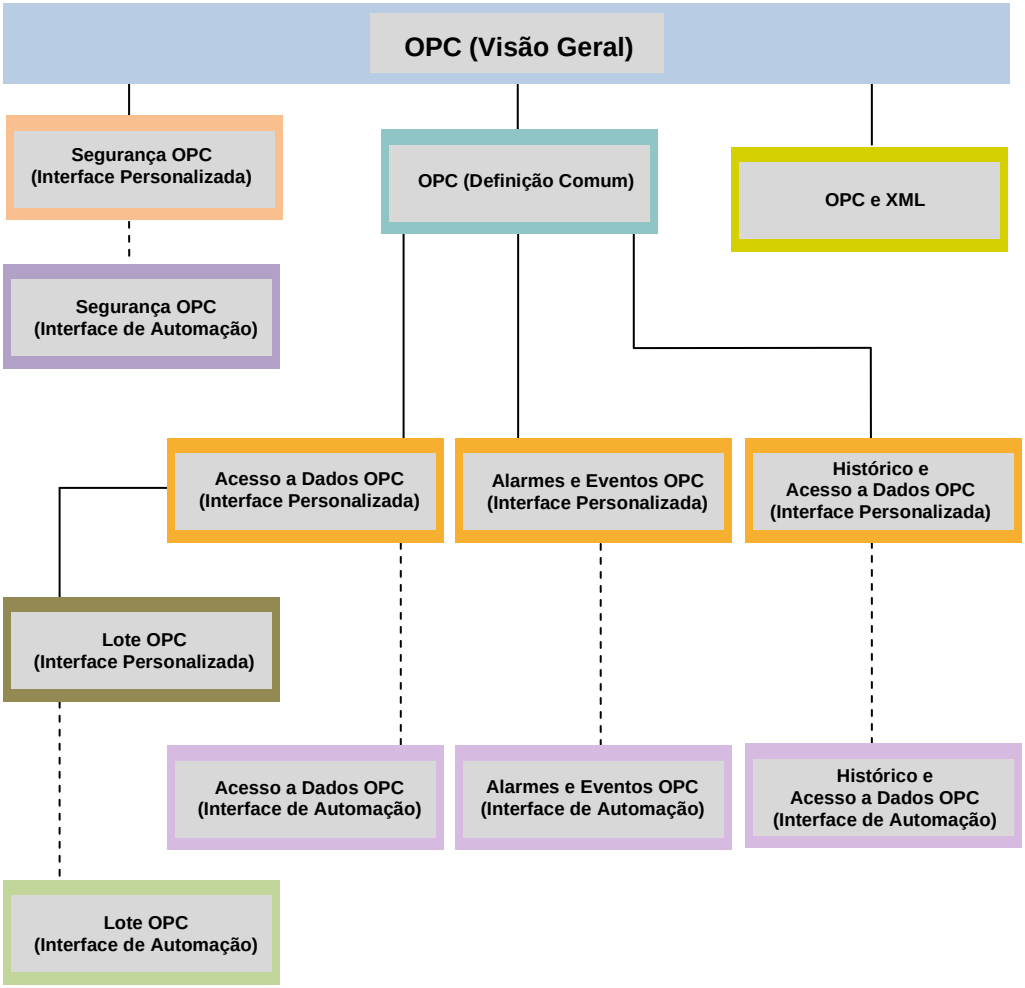

Figura 9 - Especificações do Padrão OPC.
Fonte: OPC FOUNDATION (2010)

O padrão OPC é nativo da plataforma Windows e é usado por diversas aplicações de automação como Sistemas *SCADA*, *PIMS* (*Process Information Management System*), Sistemas *MES* (*Manufacturing Execution Systems*), *AMS* (*Asset Management Systems*) e sistemas supervisores. Com a criação da especificação OPC-UA, o padrão se tornou independente de plataforma.

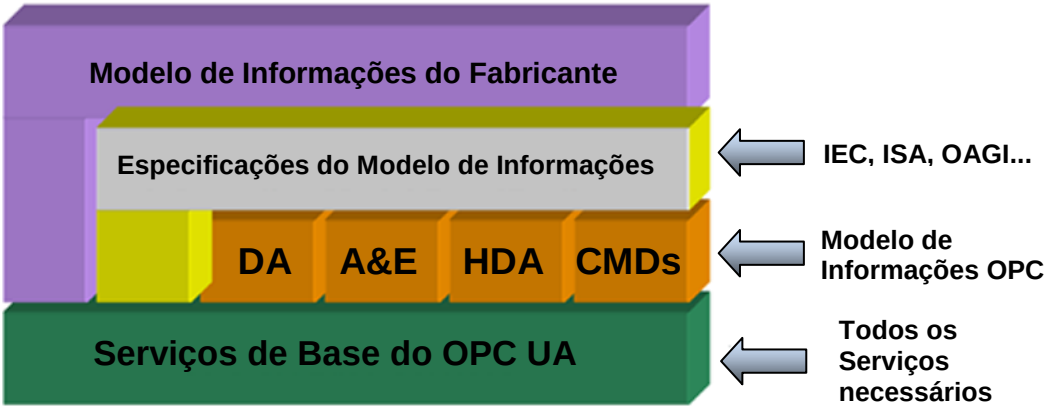

Figura 10 - Camadas da especificação OPC-UA (Unified Architecture)
Fonte: OPC FOUNDATION (2010)

No início de sua criação, *OPC* era entendido como OLE for Process Control. Hoje, *OPC* significa *Open Connectivity via Open Standards*, ou seja, conectividade aberta através de padrões abertos (OPC FOUNDATION, 2010).

2.8. O Padrão CyberOPC em Sistemas de Automação

O padrão aberto *CyberOPC* é um projeto de pesquisa acadêmico iniciado em 2006, que inclui metodologias para redes de computadores e segurança digital para o desenvolvimento de novos sistemas de comunicação para processos industriais com os seguintes objetivos: minimizar o atraso entre as transmissões de dados para controle, garantindo a segurança do canal de comunicação utilizados, garantir a integridade e confidencialidade dos dados transmitidos (TORRISI et al., 2007).

A arquitetura é baseada em um servidor *Web*, com suporte *SSL* (*Secure Socket Layer*) e podem ser publicados na internet para acesso criptografado via *HTTPS* (*Hyper Text Transfer Protocol Secure*). Os serviços são prestados de acordo com a especificação *JSON-RPC* (*Java Script Object Notation - Remote Procedure Call*). A Figura 11 ilustra o diagrama da arquitetura geral.

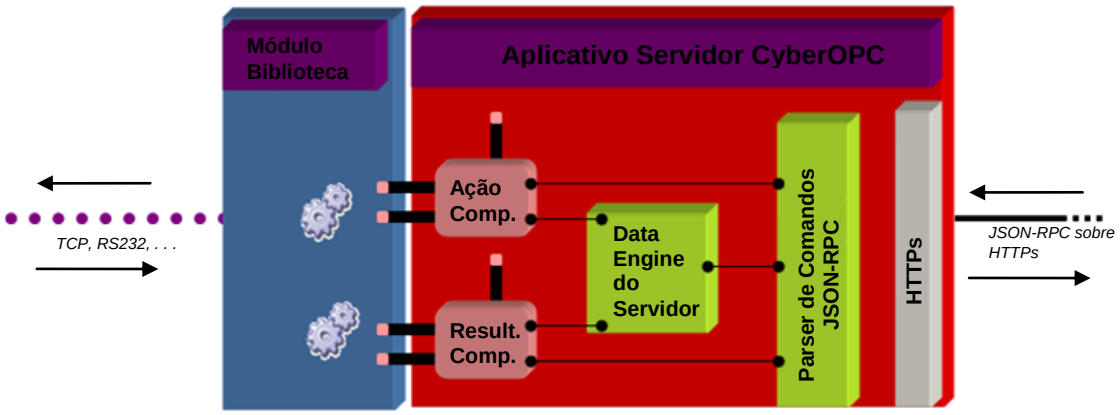

Figura 11 -CyberOPC – Arquitetura Geral
Fonte: TORRISI et al. (2007)

O *CyberOPC* permite o uso de streaming HTTP, que é um fluxo contínuo de dados entre o servidor eo cliente, sem a necessidade de atualizações de pedido, conhecido como polling.

Essas características dão ao *CyberOPC* a vantagem de ser totalmente assíncrono. Não é preciso esperar o próximo ciclo de polling para receber as informações mais atuais. Esse uso é muito adequado em sistemas de automação, pois mecanismos assíncronos são mais leves quando comparados aos síncronos.

O cliente *CyberOPC* é basicamente um cliente AJAX (*Asynchronous Javascript and XML*) capaz de processar mensagens *JSON-RPC*, com comandos encapsulados para ler, escrever e assinar para começar a transmitir a partir do servidor.

2.9. Tecnologia EDDL

A *EDDL* ou *Electronic Device Description Language* é uma linguagem baseada em texto para descrição de características de dispositivos inteligentes. Entre estas características estão variáveis, dados de diagnósticos, detalhes de calibração, configuração e métodos.

A *EDDL* é independente de plataforma e protocolo de comunicação, surgiu a partir da *DDL* (*DeviceDescriptionLanguage*) e tornou-se muito importante na área de instrumentação e controle.

Em março de 2004, ela foi aprovada como uma linguagem padrão IEC 61804-2, reunindo características dos protocolos FF, HART e PROFIBUS. A evolução da EDDL é mostrada na Figura 12, a seguir:

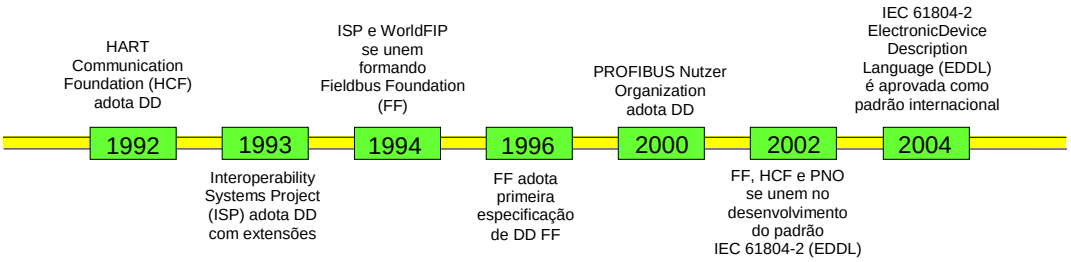

Figura 12 - Evolução da EDDL ao longo dos protocolos de comunicação
Fonte: MITSCHKE (2004)

Entre as vantagens da EDDL, podemos enumerar: a) portabilidade; b) consistência; c) uniformidade; d) controle de revisão; e) testes e f) suporte a um padrão internacional (IEC 61804-2). Algumas desvantagens são: a) a linguagem de desenvolvimento não é amigável; b) o desenvolvedor de dispositivos necessita de um compilador proprietário; c) o desenvolvedor de configuradores necessita de um interpretador proprietário.

Os arquivos DD do protocolo FF, por exemplo, possuem uma estrutura completa e hierárquica de informações, dentre elas: *blocks*, *variables*, *menus*, *edit displays*, *methods*, *relations*, *arrays*, *records*, *variablelists*, *programs*, *domains*, *response codes*, *item arrays* e *collections*.

Um pequeno exemplo de definição de variável pode ilustrar a implementação da EDDL, conforme a Figura 12:

Código em EDDL	Interpretação do Sistema

VARIABLE temperature
{
LABEL “Temperature”;
TYPE FLOAT
{
DISPLAY FORMAT “####.#”;
MIN_VALUE “-273,1”;
MAX_VALUE “300,1”;
}
CONSTANT_UNIC “ºC”;
}
	Temperature | 100 ºC

Figura 13 - Exemplo de Código EDDL e sua interpretação
Fonte: PANTONI (2006)

Os arquivos de *DD* (*Device Description*) são necessários, mas sozinhos não são suficientes para que se tenha a comunicação com os equipamentos de campo. Algumas outras camadas de software são necessárias entre os arquivos de *DD* e a comunicação com o equipamento.

O processo completo consiste no desenvolvimento da *EDD*, por meio da linguagem *EDDL*, na compilação do código fonte (através do compilador ou *tokenizer*), na geração de um arquivo binário e interpretação deste binário por outra ferramenta (interpretador de DD). Tanto o compilador (*tokenizer*) quanto o interpretador são ferramentas proprietárias. Essas ferramentas são distribuídas pelas organizações de cada um dos protocolos. A Figura 13, a seguir, ilustra o processo de utilização da EDDL:

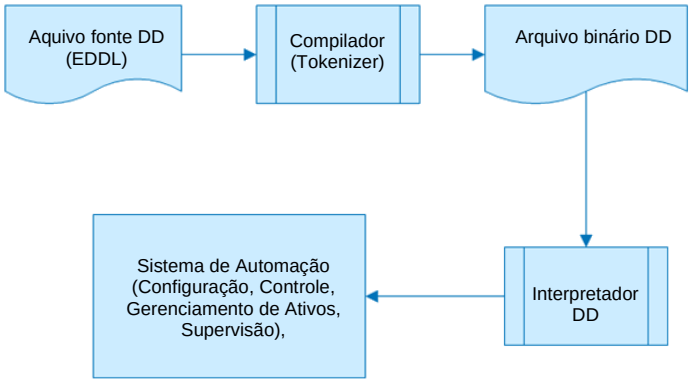

Figura 14 - Processo de utilização da Tecnologia EDDL

A existência de um compilador/interpretador único (assegurado pela organização correspondente - *FF*, *HART* ou *PROFIBUS*) garante a interoperabilidade entre as ferramentas e instrumentos que se utilizam deste modelo.

Enhanced EDDL são melhorias que foram adicionadas à *EDDL* original, permitindo o suporte a interfaces de janelas gráficas para os equipamentos. Além da lista de parâmetros, os programas escritos em *Enhanced DD Language* suportam a criação de *tabs* para isolar as partes da interface. É possível ainda a criação de gráficos bidimensionais para apresentar dados, a execução de funções matemáticas e o armazenamento de arquivos e figuras.

Pode-se ver a *Enhanced EDDL* como uma *"C-like" language*, ou seja, como a *EDDL* original, compilada e distribuída em um formato único por todas as organizações *fieldbus*.

O código está em arquivo e pode ser interpretado em tempo de execução pelo software interpretador disponibilizado pela organização *FF*, *HART* ou *PROFIBUS*, sendo cada organização a responsável pela distribuição dos interpretadores.

2.10. Tecnologia FDT/DTM

Em 1998, alguns fabricantes de equipamentos da área de automação industrial começaram o início do desenvolvimento de um padrão aberto, o *FDT/DTM* (*Field Device Tool / Device Type Manager*). Alguns anos mais tarde, foi criado o *FDT Joint Interest Group* (*JIT*), para promover o uso do *FDT* em aplicações industriais. Posteriormente, o *FDT-JIG* foi convertido em uma associação chamada *FDT Group AISBL*(*Association Industrielle Sans But Lucratif*). Em maio de 2009, o padrão *FDT/DTM* foi aprovado como um padrão internacional - IEC 62453.

O objetivo da tecnologia é prover uma interface padronizada e não proprietária para integração de equipamentos de campo em sistemas de engenharia, automação e gerenciamento de ativos, permitindo a interoperabilidade entre a *IHM* (Interface Homem-Máquina), os instrumentos de campo e outros equipamentos (FDT GROUP, 2007).

É possível fazer uma analogia da tecnologia *FDT/DTM* a um *driver* de impressora para um sistema operacional. O fabricante de produtos de informática desenvolve uma impressora. Têm-se o equivalente ao fabricante de equipamentos que desenvolve um instrumento de campo. Juntamente com a impressora, o fabricante fornece os *drivers* para que a mesma possa ser instalada no sistema operacional. Esses *drivers* são componentes de software binários e são especialistas no uso de todos os recursos da impressora. Da mesma maneira, são disponibilizados os *drivers* ou *DTMs* dos instrumentos de campo, que são componentes de software binários, especialistas no funcionamento e uso de todos os recursos do instrumento. Uma vez instalado o driver da impressora, qualquer software - seja ele um editor de textos, um software de planilhas ou qualquer outro programa – é capaz de imprimir, sem ter que conhecer os detalhes de como o processo funciona. Isso é tarefa do *driver* que já está instalado do sistema operacional. De maneira equivalente, qualquer ferramenta *FDT* pode fazer o uso dos *DTMs* instalados na máquina e ter acesso a todos os recursos do instrumento de campo, sem ter que conhecer os detalhes de funcionamento do mesmo.

A Figura 14, a seguir, ilustra a analogia entre a tecnologia FDT e um driver de impressora.

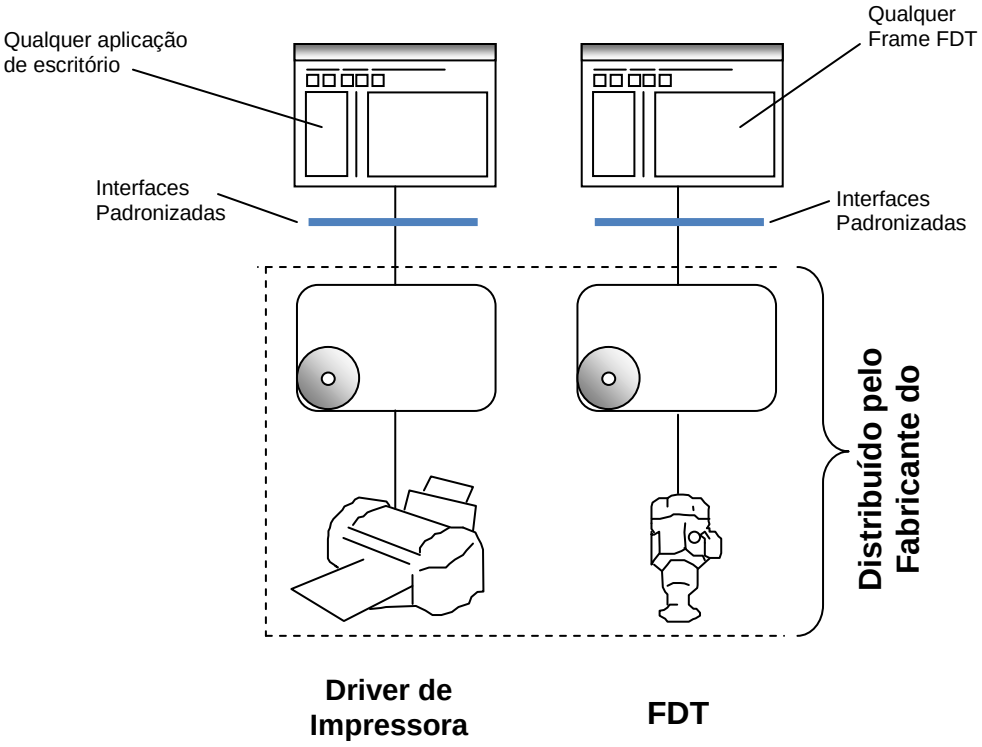

Figura 15 - Analogia entre a tecnologia FDT e um Driver de Impressora
Fonte: FDT Group (2007)

Basicamente, existem 3 tipos de DTMs:

- DTMs de Comunicação: são responsáveis por representar o elemento da comunicação física, seja um *PLC*, um *Linking Device*, um Controlador, um Multiplexador ou outro elemento de comunicação;
- DTMs de Device: são responsáveis por representar o instrumento de campo propriamente dito;
- DTMs de Gateway: são responsáveis por representar elementos que fazem a interligação de protocolos diferentes.

Os DTMs trocam dados entre si e com o Frame *FDT* através de interfaces padronizadas e arquivos no padrão XML (*eXtensible Markup Language*).

A Figura 16 ilustra o escopo das interfaces FDT e a Figura 17, por sua vez, ilustra a interoperabilidade decorrente da padronização feita pela tecnologia.

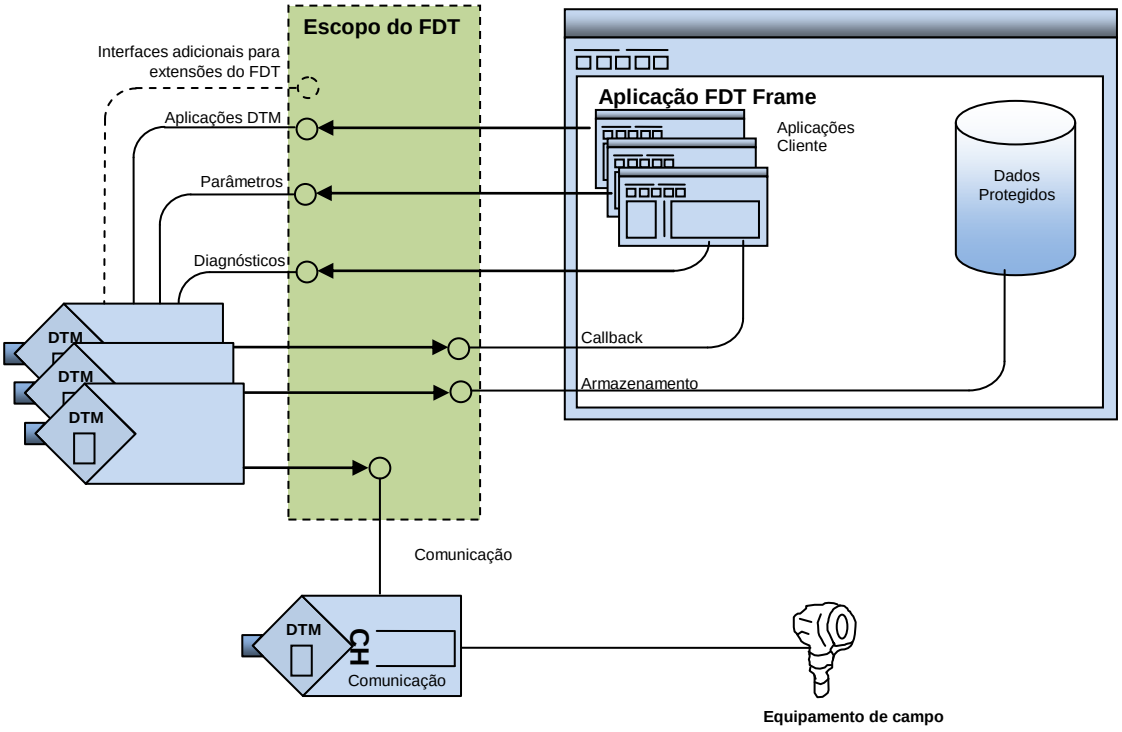

Figura 16 - Escopo das Interfaces FDT
Fonte: FDT Group (2007)

Essa padronização garante o funcionamento de um DTM de um fabricante A, com um fabricante B e um fabricante C, dentro de um Frame FDT do fabricante X, Y ou Z.

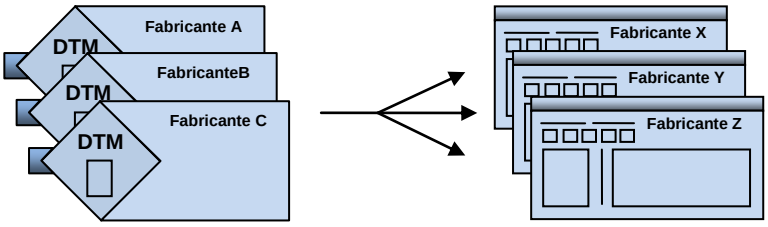

Figura 17 - Interoperabilidade entre fabricantes garantida pela tecnologia FDT
Fonte: FDT Group (2007)

O FDT/DTM foi concebido para ser executado em plataforma Microsoft, por meio do uso da tecnologia COM (*Component Object Model*).

A tecnologia FDT/DTM oferece também um suporte padronizado de interfaces gráficas através de um guia de estilo. Isso garante ao usuário final uma sensação de uniformidade e conforto visual ao navegar por telas de operação construídas por diferentes fabricantes.

2.11. Comparação: FDT/DTM versus EDDL

A *EDDL* fornece, de maneira padronizada, a descrição eletrônica do equipamento, onde estão listadas todas as funcionalidades, variáveis, métodos, enfim, todos os recursos disponíveis em um determinado instrumento. Por sua vez, o *DTM* pode ser considerado como um *driver* binário, especialista no funcionamento de um determinado instrumento, capaz de explorar todas as informações fornecidas pela *EDDL*, armazenando-as de forma binária, tornando possível a integração de um determinado instrumento a qualquer sistema que suporte a tecnologia *FDT*. Hoje, ambos são padrões *IEC*: tanto o *FDT/DTM* quanto a *EDDL*. No entanto, a padronização de interfaces acontece em camadas diferentes. Uma comparação entre ambas as tecnologias que as considere concorrentes, não estaria contemplando o que há de melhor em cada um desses padrões.

2.12. Tecnologia FDI

Em 2003, três organizações: *Fieldbus Foundation*, *HART Communication Foundation* e *Profibus Nutzer Organization* (*PNO*) se juntaram para um trabalho de cooperação de desenvolvimento de uma especificação comum para visualização gráfica e persistência de dados disponível a partir da *EDDL*. Surge, então, o *ECT* (*EDDL Cooperation Team*). Em 2004, o *OPC* se juntou ao time de cooperação (LI; LIU, 2011).

O *FDT/DTM* possui um mecanismo de extensão e suporte de melhorias. Através de *Schemas XML* é possível a adição de novos protocolos. Além disso, uma nova especificação - a *FDT 2.0* - já está sendo desenvolvida desde 2008. Essa nova especificação irá incorporar os novos recursos disponíveis em novas versões do Windows por meio do uso de .NET *Framework* e *WPF* (*Windows Presentation Foundation*).

Em abril de 2007, na feira de Hannover, Alemanha, o *EDDL Cooperation Team* (*ECT*) e o *FDT Group* anunciaram que iriam combinar esforços para trabalhar no sentido de uma solução unificada para integração de equipamentos e manutenção da compatibilidade com ambas tecnologias (*EDDL* e *FDT/DTM*).

Nos próximos anos estará disponível um padrão resultante do esforço desses dois times, capaz de explorar o que há de melhor no potencial de cada uma dessas tecnologias. Este padrão é o *FDI* (*Field Device Integration*) que irá suportar o legado do *FDT/DTM* e da *EDDL* (NAGASHIMA et al., 2008).

2.13. Gerenciamento de Ativos (Asset Management)

As modernas indústrias têm encontrado grandes desafios relacionados ao mercado, resultante da necessidade de redução de custos e aumento da flexibilidade para acompanhar os concorrentes (THEURICH; LEHMANN; WOLLSCHLAEGER, 2010).

Para enfrentar esses desafios, estudos mostram que é necessária a integração das diversas áreas que compõe o universo industrial. Nessa perspectiva, a integração vertical, apresentada na Figura 18, tem ganhado grande importância. Todavia, a solução para os desafios exige tanto a integração vertical quanto a horizontal e a contribuição é necessária desde os níveis mais altos, a começar pelo *ERP* (*Enterprise Resource Planning*) até o chão de fábrica, onde se encontram os instrumentos de campo (THEURICH; LEHMANN; WOLLSCHLAEGER, 2010).

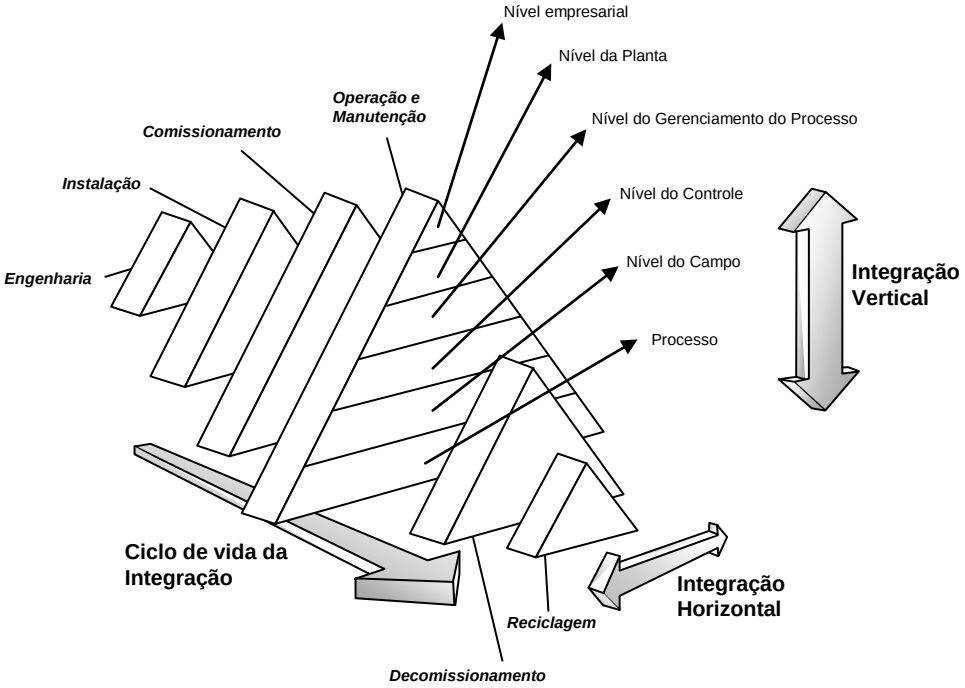

Figura 18 - Dimensões de Integração
Fonte: Theurich, Lehmann e Wollschlaeger (2010)

É nesse contexto que está inserido o uso do gerenciamento de ativos. Tão importante quanto iniciar um processo industrial é mantê-lo em funcionamento uma vez iniciado.

A NAMUR NE 91 (NAMUR, 2001), define *Asset Management*, como:

> "[...] as atividades e medições efetuadas para conservar e estender o valor da planta. Isso inclui o gerenciamento da planta, gerenciamento do processo, otimização do processo, conservação dos valores e melhoria na manutenção".

Segundo Mohseni (2003), "*Enterprise Asset Management* é uma disciplina para otimização e aplicação de estratégias relacionadas ao ciclo de vida do ativo e as decisões de planejamento de trabalho".

Pode-se, então, entender o gerenciamento de ativos ou *asset management* como o conjunto de operações(calibração, manutenções e ajustes) responsável pelo cuidado do ativo, neste caso, o equipamento de campo. Esse gerenciamento tem seu início a partir do *startup* (partida) da planta e dura todo o tempo de vida do equipamento.

O uso de "*fieldbuses*" padrões (*HART, PROFIBUS PA* e *Foundation Fieldbus*) com equipamentos "inteligentes" habilita a maximização do potencial dos ativos de uma planta, onde os equipamentos de campo inteligentes são capazes de enviar informação sobre autodiagnósticos, status, tendências e alarmes. Esse tipo de informação pode ser coletada e classificada por um sistema de gerenciamento de ativos que pode predizer algumas situações indesejáveis. Dessa forma, o custo de manutenção corretiva pode ser reduzido, aumentando o ciclo de vida do ativo na planta de automação (BERGE, 2002).

Segundo Hollender (2010), as manutenções no ambiente industrial podem ser classificadas nos tipos: a) Manutenção Corretiva; b) Manutenção Preventiva; c) Manutenção Preditiva e d) Manutenção Proativa.

Por definição, a manutenção corretiva é aquela executada a partir de um defeito propriamente dito. O defeito pode ocasionar uma parada indesejada no processo produtivo e os custos relacionados a este tipo de manutenção, em geral, são altos.

Por outro lado, a manutenção preventiva é aquela executada com intervalos pré-programados, de forma a tentar prevenir o acontecimento de defeitos indesejados. Quando comparada à manutenção corretiva, ela implica em custos menores. Todavia, essa abordagem não é a ideal, uma vez que pode gerar custos desnecessários pela intervenção em equipamentos que não necessariamente precisariam de alguma manutenção.

A manutenção preditiva, por sua vez, procura aprimorar o conceito da manutenção preventiva. Baseada em condições de uso e nos dados de autodiagnósticos fornecidos pelos equipamentos de campo inteligentes, é possível verificar tendências e antecipar o acontecimento de defeitos, de tal forma a se prever determinadas condições antes mesmo que

elas possam acontecer. Sendo assim, torna-se possível chegar ao limite de utilização de um instrumento. Os custos da manutenção preditiva são, portanto, menores quando comparados à manutenção preventiva.

Finalmente, a manutenção proativa é aquela que possui como base analisar os resultados das causas raízes que originaram distúrbios no processo.

A Figura 19 ilustra uma representação da manutenção com base na probabilidade de falha e no impacto sobre a operação por ativo.

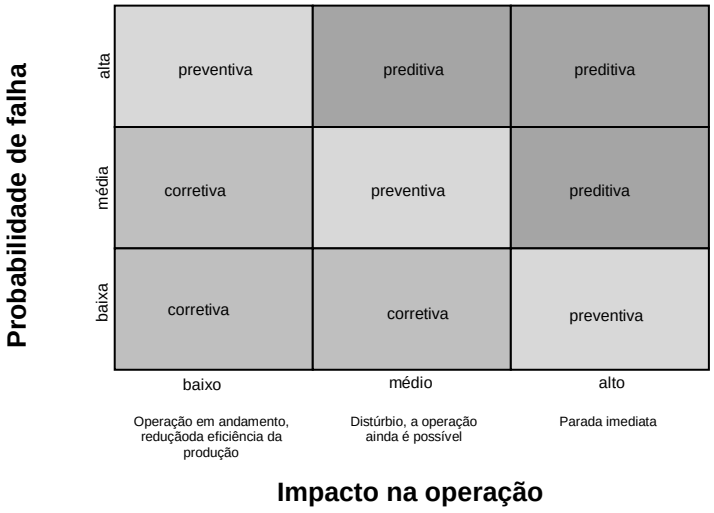

Figura 19 -Estratégia de manutenção razoável com base na probabilidade de falha e impacto sobre a operação por ativo
Fonte: Hollender (2010)

É tarefa, portanto, de um sistema de gerenciamento de ativos, incorporar no seu mecanismo de funcionamento, o tratamento desses tipos de manutenções.

Um sistema de gerenciamento de ativos deve, então, ser capaz de coletar as informações de diagnósticos presentes nos instrumentos de campo e disponibilizá-las de tal forma a se otimizar o uso do equipamento de campo por meio da redução do custo com manutenções e paradas não programadas. Para isso, as tecnologias mais comumente usadas são baseadas nos padrões *EDDL*, *FDT/DTM* e *OPC*(MULLER et al., 2003). Essas tecnologias permitem uma padronização de interfaces de software e suportam completamente o seu uso em protocolos abertos como *HART*, *PROFIBUS PA* e *Foundation Fieldbus*.

O mecanismo utilizado para coletar os autodiagnósticos dos instrumentos de campo pode ser ilustrado a partir da Figura 20:

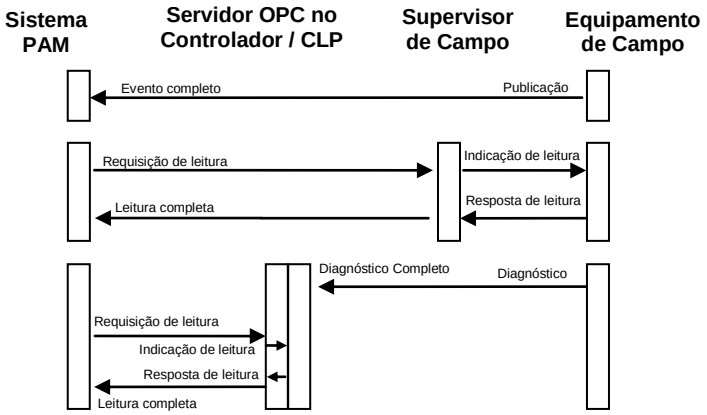

Figura 20 - Três exemplos de acesso aos dados de autodiagnósticos
Fonte: Theurich, Lehmann e Wollschlaeger (2010)

Theurich, Lehmann e Wollschlaeger (2010) apresentam três maneiras distintas para acesso às informações de diagnósticos:

a) No primeiro exemplo, existe o conceito de *publish-subscriber*. Ou seja, o sistema *PAM* (*Plant Asset Management*) assina o evento interessante a ser monitorado. Assim que ocorrer, esse evento é enviado ao assinante.
b) No segundo exemplo, o sistema *PAM* lê os dados diretamente do dispositivo por meio do adaptador de barramento.
c) Na última opção, o sistema *PAM* lê os dados a partir de um servidor *OPC*.

Dessa forma, é possível concluir que o uso desses padrões, como, por exemplo, o *OPC*, pode facilitar o mecanismo de integração de um instrumento de campo em um ambiente de *asset management* (BANGEMMANN; HAHNICHE; NEUMANN, 1998).

Por outro lado, tomando como referência as fases do ciclo de automação de uma planta, citadas anteriormente, é possível notar que os sistemas responsáveis pela programação da estratégia de controle e *startup* da planta não são os sistemas de gerenciamento de ativos, mas sim os sistemas configuradores. Uma vez iniciado, o processo industrial possui mecanismos de intertravamento e lógicas de controle capazes de garantir a estabilidade da operação e ir para um estado seguro caso alguma situação inesperada aconteça.

Uma função do gerenciamento de ativos é utilizar as informações dos autodiagnósticos dos instrumentos de maneira a se evitar paradas indesejadas causadas por situações inesperadas. Todavia, os níveis de precisão em termos de tempo de resposta de um sistema de *asset management* não são tão críticos quanto o controle do processo.

Uma informação sobre um diagnóstico crítico deverá chegar, necessariamente, ao *asset management*, entretanto, o responsável por levar a planta a um estado de segurança, caso algum problema aconteça, é o processo de controle. Por sua vez, esse processo de controle funciona independentemente do *asset management*.

Pode-se deduzir, portanto, que uma informação de diagnóstico de um instrumento de campo tipicamente definida pela NAMUR NE 91, como, por exemplo: "*MaintenanceRequired Bit*" (NAMUR, 2001), que indica o status de um transmissor com necessidade de uma manutenção, terá a mesma importância caso chegue ao *PAM* 1 milissegundo após ter acontecido ou 5 minutos após ter acontecido. Essa informação poderá gerar a necessidade de uma calibração – operação típica de manutenção, que deverá ser executada por um operador de manutenção. Uma vez que uma calibração exige a intervenção de um fator humano, é possível concluir que, nesse caso, a informação não necessita chegar, necessariamente, no instante em que ocorre, mas pode sofrer a tolerância de alguns minutos.

Essa característica de tolerância ao tempo de resposta é muito importante, uma vez que torna flexível o mecanismo de busca de diagnósticos, permitindo o uso de algoritmos de escalonamento para varredura de instrumentos de campo de protocolos distintos, como, por exemplo: *HART, Profibus PA* e *Foundation Fieldbus*. Esses protocolos não necessariamente compartilham o mesmo tempo de resposta, todavia, ainda assim, podem ter seus instrumentos de campo consolidados e integrados em uma única ferramenta de *asset management*.

Segundo Bixler (2008), os desafios para gestão de ativos não se limitam à simples captação de diagnósticos, mas incluem ainda: a) uma gestão do monitoramento de diagnósticos; b) monitoração sob demanda versus monitoração contínua; c) anunciação de alarmes e falhas; d) operações de armazenamento de eventos; e) armazenamento de tendência de dados; e f) uso de estatísticas do processo de controle.

Conclui-se, portanto, que o gerenciamento de ativos é composto não somente da coleta de diagnósticos a partir de instrumentos de campo, mas do uso desses diagnósticos associados a um planejamento estratégico e tático, capaz de se traduzir em um planejamento de gestão e programas de investimento e manutenção, incluindo suporte para gestão de risco e análise de decisão, de forma a criar um valor para os ativos.

3. INOVAÇÃO EMERGENTE DOS DISPOSITIVOS MÓVEIS

3.1. Aplicações para Dispositivos Móveis

Nos últimos anos é possível experimentar o surgimento de novos dispositivos móveis com acesso à internet. Nota-se uma evolução em direção à mobilidade que se inicia pelos computadores pessoais do tipo *desktop*, avança com o uso dos notebooks e netbooks, passa pelos computadores ultraportáteis do tipo *Pocket PCs*, *PDAs* e *Tablets*, chegando a dispositivos menores do tipo *smartphones*. Um *smartphone* é um telefone celular com recursos adicionais provenientes da possibilidade de rodar aplicativos em seu sistema operacional.

Essa variedade de ambientes gera um grande desafio, principalmente, porque as aplicações destinadas aos computadores pessoais, nos últimos 10 anos, não puderam prever esse uso simultâneo em plataformas múltiplas. Os desafios estão relacionados a diferentes capacidades de processamento, tamanhos de telas de visualização e interação IHM (Interface Homem-máquina). Uma aplicação para um dispositivo móvel precisa adaptar-se a utilizar recursos de maneira limitada em termos de: a) processamento; b) interfaces de entrada; c) display com o usuário; d) consumo de energia.

Embora a maioria dos celulares atuais seja capaz de executar aplicativos escritos em J2ME (*midlets*), esses aparelhos não podem ser considerados como *"true" smartphone*, pois a funcionalidade de aplicativos J2ME é limitada. O acesso a recursos de hardware e software não são feitas diretamente, mas através da Java Virtual Machine (ZAYKOVSKIY & SCHMITT, 2008).

Hoje, os principais sistemas operacionais para dispositivos móveis são: a) Symbian; b) BlackBerry OS; c) Windows Mobile; d) iPhone OS (iOS); e) Palm; f) WebOS; g) Samsung Bada; h) Maemo e i) Android.

De acordo com estatística mundial Canalys (2009) e Gartner (2011), os sistemas operacionais de dispositivos móveis são distribuídos de acordo com as Figuras 21, 22 e 23 (2009, 2010 e 2011, respectivamente):

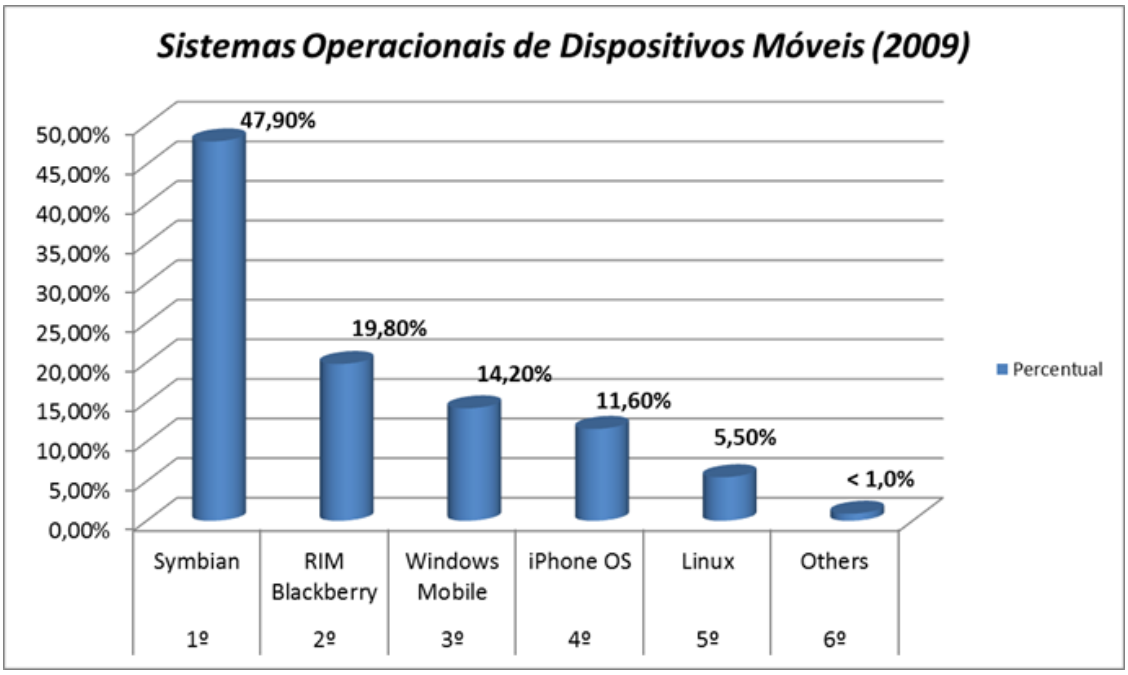

Figura 21 – Sistemas Operacionais de Dispositivos Móveis
Fonte: CANALYS (2009)

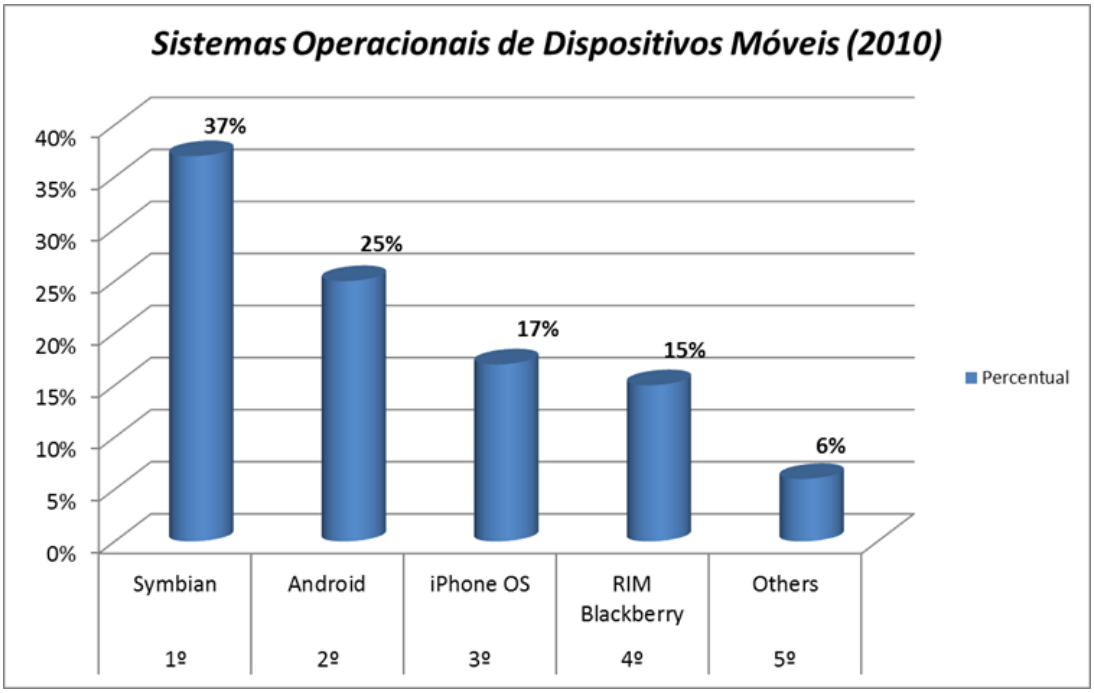

Figura 22 – Sistemas Operacionais de Dispositivos Móveis
Fonte: CANALYS (2010)

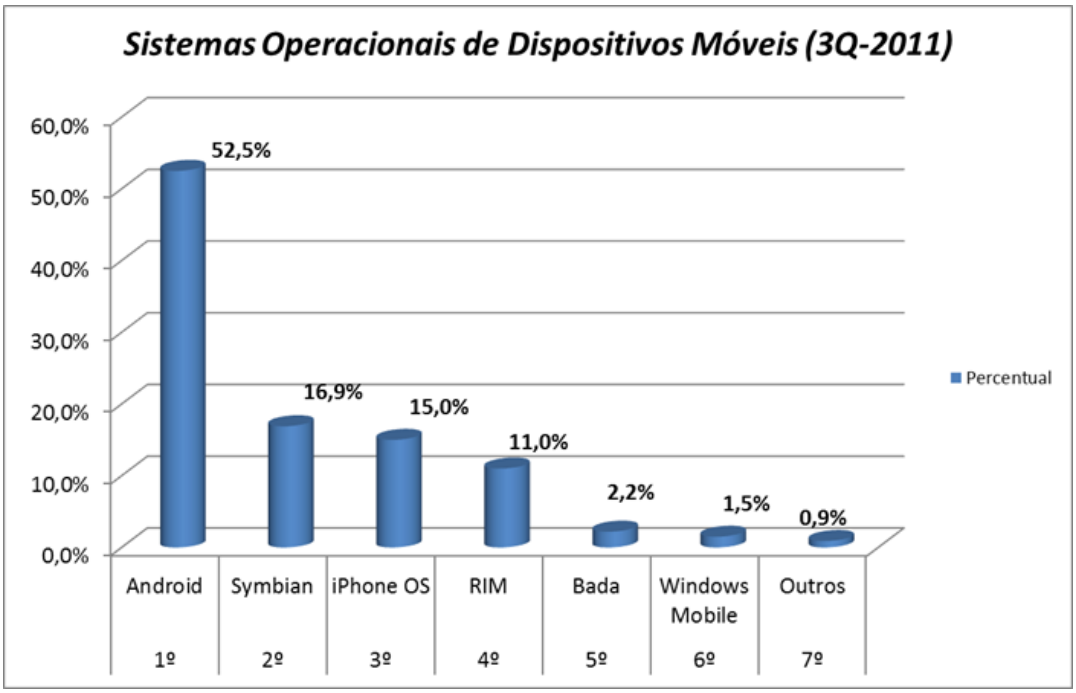

Figura 23 – Sistemas Operacionais de Dispositivos Móveis até o 3º Quartil de 2011
Fonte: GARTNER (2011)

Com base nessas estatísticas, quase metade dos dispositivos móveis possuía o Symbian OS ao final de 2009. As últimas versões do Symbian permitem a criação de aplicações, tais como *widgets* (componentes de uma interface gráfica que inclui janelas, botões, menus, ícones, barras de rolagem, etc.)

Em 2010, nota-se um crescimento do Android, seguido pelo iOS. Ainda neste mesmo ano o sistema operacional Symbian torna-se "*open source*" (SYMBIAN FOUNDATION, 2010).

Segundo análises de Canalys (2010) e Gartner (2011), entre 2010 e 2011, houve um crescimento da utilização do Android, um pequeno crescimento do Windows Mobile e uma diminuição do Symbian.

Apesar das diferenças entre os números dos sistemas operacionais, os dispositivos smartphones apresentam uma tendência de crescimento e uso cada vez mais frequente. Isso implica no surgimento de novas aplicações ou adaptações do atual uso do computador pessoal para uma plataforma móvel.

3.2. Sistema operacionais para dispositivos móveis

3.2.1. Symbian OS

O projeto Symbian começou em 1998, por meio de um consórcio formado pelas empresas Nokia, Siemens, Samsung, Ericsson, Sony Ericsson e Panasonic. Hoje, pertence à Nokia, que adquiriu quase todas as suas ações em 2008. Empresas não-pertencentes ao consórcio podem licenciar o Sistema Operacional para utilização em seus produtos. Em 2010, o Symbian tornou-se um projeto *"open-source"*(SYMBIAN FOUNDATION, 2010).

O Symbian é um sistema operacional de 32 bits, multitarefa, robusto, desenvolvido especificamente para um ambiente móvel e com restrições de hardware, como telefones celulares. É uma plataforma aberta, dirigida para o desenvolvimento de aplicações com compromissos com padrões abertos.

O desenvolvimento de aplicativos pode ser feito em C++, Java ME, Flash Lite, HTML5, Perl, Phyton, Ruby, QT e WRT, facilitado pelo uso de *SDKs* (*Standard Development Kits*) (NOKIA, 2010).

O padrão para o desenvolvimento de widgets Symbian é o *WRT* (*Web Runtime*) (NOKIA, 2010), que estende a funcionalidade do Web Browser do Symbian.

O *WRT* permite um desenvolvimento simples, devido à facilidade de acesso a recursos nativos, como páginas *HTML*, *Java Script*, *CSS Stylesheet*, figuras e *AJAX*.

As principais vantagens do Symbian são: a) é um sistema aberto; b) possui recursos para gerenciamento e utilização de pouca memória e bateria; c) permite instalação de softwares de terceiros; d) baseado em padrões de comunicação e dados e e) é estável e seguro. (FORSTNER et al, 2005).

As principais ferramentas de desenvolvimento são SDKs para desenvolvimento em *C++Symbian* (FORSTNER et al., 2005) e QT e o desenvolvimento pode ser feito a partir de estações usando *Windows*, *Linux*, *MacOS* e *Unix*. Com a ferramenta *Aptana Studio*, *IDE open source*, é possível o desenvolvimento em *WRT* (*WidgetRuntime*), que é uma plataforma de desenvolvimento web rápido em HTML5 e AJAX.

3.2.2. iOS (iPhone e iPod Touch)

Em 2007, a *Apple* lançou uma versão otimizada do sistema operacional *MacOS* chamada *iPhoneOS*. A partir de junho de 2010, o *iPhoneOS* passa a ser conhecido como iOS e é o sistema operacional da *Apple* para plataformas móveis. Originalmente desenvolvido para o *iPhone*, o *iOS* tem seu suporte estendido para ser usado também no *iPod Touch*, *iPad* e *Apple TV*.

O *iOS* é derivado do *MacOS X*, que faz parte da *Darwin Foundation*, e portanto é um sistema operacional do tipo *Unix*. No *iOS* existe uma abstração de 4 camadas: a) *Core OS layer*; b) *Core Services layer*; c) *Media layer* e d) *Cocoa Touch layer* (APPLE, 2010).

A interface do *iOS* é baseada no conceito de manipulação direta, usando o suporte a multi-toque. A interação com o sistema operacional inclui gestos como: tocar a tela, deslizar o dedo e o movimento de "pinça" utilizado para se ampliar ou reduzir a imagem.

O *iOS* é um sistema fechado e devido a restrições de licença não pode ser instalado em hardware de terceiros.

O desenvolvimento de aplicativos para o *iOS* é feito através da suíte chamada *XCode*. É necessário um computador com o *MacOS X* para executar as ferramentas do *XCode*. Portanto, o desenvolvimento é proprietário e somente pode ser feito em computadores *Apple*. O código é desenvolvido em linguagem *Objective-C* (linguagem de programação reflexiva orientada a objeto, que adiciona transmissão de mensagens no estilo *Smaltalk* para o *C*).

3.2.3. Android

Em 2005, a *Google* adquiriu a Android Inc., pequena empresa em Palo Alto, Califórnia, USA. A partir de então, foi desenvolvida uma plataforma de telefone móvel baseado em *Linux*, com o objetivo de ser uma plataforma aberta, flexível e de fácil migração para os fabricantes de dispositivos móveis.

O *Android* é um sistema operacional móvel, baseado no *Kernel* 2.6 do *Linux*. Inicialmente, foi desenvolvido pela *Google* e posteriormente pela *Open Handset Alliance*, entretanto a Google é responsável pela gerência do produto e engenharia de processos.

Entre as principais características pode-se enumerar que a plataforma é adaptada tanto para dispositivos *VGA* maiores, gráficos *2D*, bibliotecas *3D* baseadas em *OpenGL ES* e os layouts tradicionais de smartphones.

Em termos de conectividade, é possível a utilização do *GSM/EDGE, IDEN, CDMA, ED-VO, UMTS, LTE, Bluetooth, 3G, Wi-Fi* e *WiMAX*.

Um ponto importante é que as aplicações Java são compiladas em *bytecodes Dalvik* e executadas usando a máquina virtual *Dalvik* (máquina virtual especializada desenvolvida para o uso em dispositivos móveis), o que permite que programas sejam distribuídos em formato binário (*bytecode*) e possam ser executados em qualquer dispositivo Android, independente do hardware. Portanto, o *Android* não executa *bytecode JVM*.

As ferramentas de desenvolvimento são de plataforma aberta. Uma característica interessante é que as aplicações podem fazer uso do *Google Cloud* (GRONLI; HANSEN e GHINEA, 2010). É possível utilizar o *IDE Eclipse*, juntamente com *plugins* do tipo *Android Development Tools* (*ADT*). Isso implica em possibilidade de desenvolvimento a partir de estações *Windows*, *Linux* ou *Unix*.

3.2.4. Windows Mobile

Em 1996, a *Microsoft* iniciou o desenvolvimento de uma versão do *Windows* direcionada a equipamentos portáteis. Surgia, então o *Windows CE* 1.0. A versões posteriores surgiram em 1997 (*Windows CE* 2.0), 1998 (*Windows CE* 2.11), 2000 (*Windows CE* 3.0), 2002 (*Windows CE* 4- *.NET*), 2004 (*WindowsCE* 5) e 2006 (*WindowsCE* 6) (MICROSOFT, 2010).

Em 2003, a *Microsoft* decidiu lançar as variações desse sistema operacional, dando origem ao *Windows Mobile 2003* que evoluiu até o *Windows Mobile 6*, e hoje é o *Windows Phone 7*.

Praticamente todas as características do sistema operacional *Windows* estão presentes, como suporte a protocolos de rede, *Bluetooth*, *3G*, *Wi-Fi*, entre outros.

As ferramentas de desenvolvimento são proprietárias, fornecidas pela *Microsoft*. O principal ambiente de desenvolvimento é o *Microsoft Visual Studio* e as estações de desenvolvimento devem ser, necessariamente, computadores executando o *Windows*.

Entretanto, é possível desenvolver código na plataforma *.NET*, uma vez que o *Windows Mobile* suporta o *.NET Compact Framework*. Esse é um fator relevante, uma vez que a plataforma *.NET* possui recursos de orientação a objeto, é de fácil aprendizagem e possui vasta documentação (SCHULT & POLZE, 2002).

4. PROPOSTA DE ARQUITETURA

4.1. Arquitetura de Integração

Um software de gerenciamento de ativos tem a necessidade de acesso a informações que estão localizadas nos instrumentos de campo. Para tanto, algumas ferramentas realizam esse acesso por meio de padrões industriais como *EDDL*, *FDT/DTM* e *OPC*, que possuem definições de interfaces de software concebidas com o objetivo de integração entre diferentes componentes de software e hardware de diversos fabricantes (ENDRESS+HAUSER, 2008).

O padrão OPC tem sido usado como interface de integração entre os diversos componentes de software de diversos fabricantes. Entretanto, algumas versões como, por exemplo, *OPC Data Access* e *OPC Alarms and Events* são dependentes da plataforma Windows, uma vez que são baseadas na tecnologia *COM (Component Object Model)*.

Algumas opções para uso independente de plataforma surgem com o *OPC XML DA* (integração entre *OPC* e *XML* para aplicações via internet) ou *OPC-UA* (*Unified Architecture*), sendo essa última capaz de ser executada até mesmo em uma plataforma embarcada, como, por exemplo, um *PLC* (*Programmable Logic Controller*).

Contudo, essas duas últimas opções citadas, apesar da independência de plataforma, apresentam uma complexidade no desenvolvimento de um software para dispositivos móveis. A versão *OPC XML DA* exige um interpretador *XML* no dispositivo móvel, enquanto a versão *OPC-UA* exige um cliente capaz de interpretar *XML* ou trabalhar no modo binário de transferência de informações.

Um desafio de arquitetura surge, então, quando se deseja aplicar o conceito de gerenciamento de ativos e permitir o seu uso a partir de um dispositivo móvel: a escolha de um padrão comum, capaz de ser utilizado a partir de um *smartphone* e também se integrar a um sistema de automação industrial, por exemplo.

Nesse sentido, o projeto acadêmico *CyberOPC* trouxe uma grande contribuição para a solução da arquitetura, uma vez que propõe a integração de aplicações industriais por meio de tecnologias de acesso web (TORRISI et al., 2007). Essas tecnologias web, por sua vez, estão disponíveis para o uso a partir de dispositivos móveis como os smartphones.

Portanto, a escolha da arquitetura prevê o *CyberOPC* como sendo o elemento de ligação entre os mundos da computação móvel e automação industrial.

Esse trabalho apresenta como proposta a implementação de um servidor *CyberOPC* para ser executado em plataforma móvel, capaz de ser interligado a outros servidores OPC

DA (*Data Access*), que possam prover acesso aos equipamentos dos protocolos HART, Foundation Fieldbus e PROFIBUS.

No modelo proposto, o dispositivo móvel executa uma aplicação do tipo *CyberOPC Client* e se conecta a um servidor *CyberOPC*, que pode estar localizado tanto numa intranet quanto na internet.

O canal de comunicação escolhido contempla uma infra-estrutura de comunicação sem fio, que pode ser feita de duas maneiras: a) através de um *Acess Point Wireless*; ou b) por meio de um canal de comunicação *3G* de uma operadora de telefonia. Para efeitos de criptografia, o *CyberOPC* provê um mecanismo de conexão baseado em *HTTPS*, tornando assim o mecanismo de acesso seguro (TORRISI, 2007).

A Figura 24, a seguir, ilustra essa arquitetura:

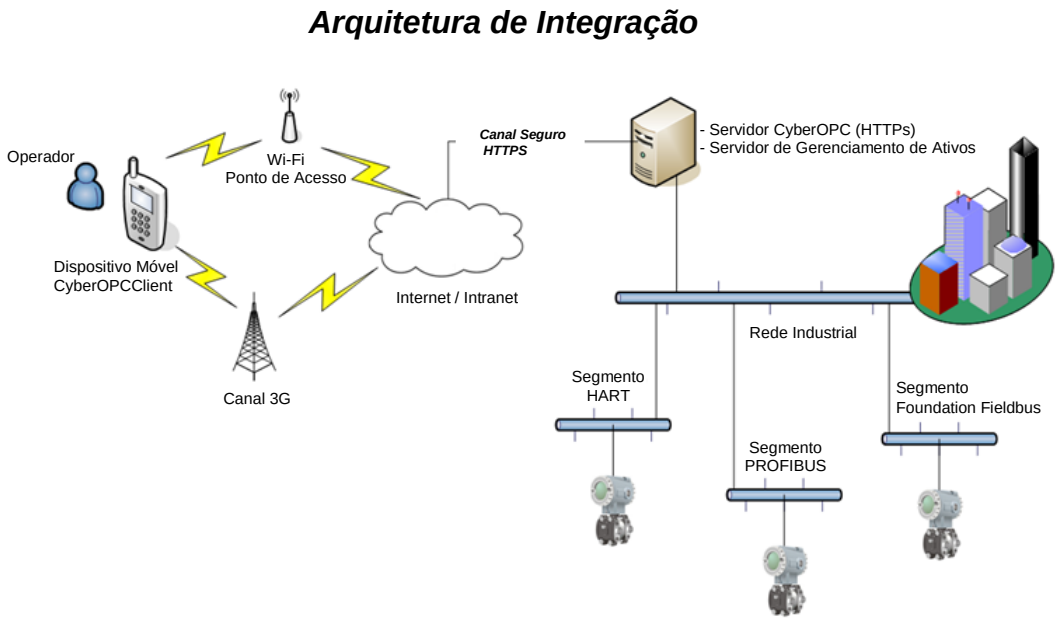

Figura 24–Arquitetura Geral

Provavelmente, uma vez que o uso da comunicação para o *asset management* pode ser feito com certa tolerância em termos de tempo de resposta, conforme citado anteriormente no capítulo referente ao gerenciamento de ativos, a latência proveniente da comunicação através da rede *wireless* ou da rede *3G* não deveria impor impactos significativos para os testes realizados. Todavia, foram efetuados testes de velocidade e desempenho para determinação dos limites de utilização.

Os equipamentos de campo escolhidos são de três tipos de protocolos distintos: *HART*, *Profibus PA* e *Foundation Fieldbus*.

Para efeito de normalização, foi aplicado um modelo genérico de instrumento de campo, baseado na proposta de Theurich, Lehmann e Wollschlaeger (2010), no qual são representadas suas características como: a) entradas; b) saídas; c) diagnósticos; d) parâmetros somente de leitura; e) parâmetros de leitura e escrita. Esse modelo é apresentado na Figura 25:

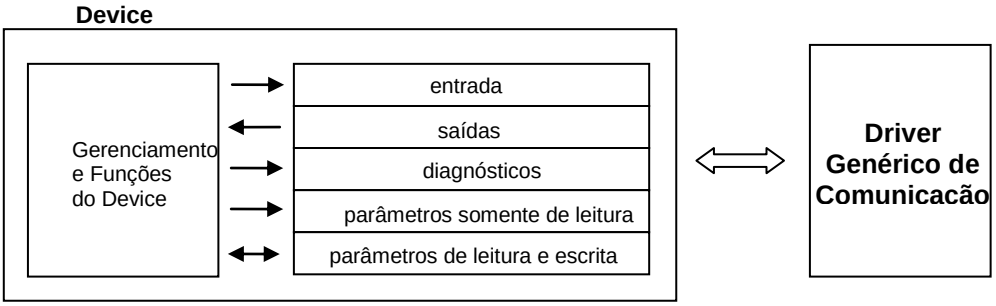

Figura 25 –Modelo do Device
Fonte: Theurich, Lehmann e Wollschlaeger (2010), com adaptações

O driver de comunicação genérico apresentado na Figura 25 foi substituído pelo driver de comunicação específico *HART*, *PROFIBUS PA* e *Foundation Fieldbus*, para acesso a cada instrumento de campo de cada uma dessas redes, respectivamente.

A Figura 26 ilustra esse mecanismo:

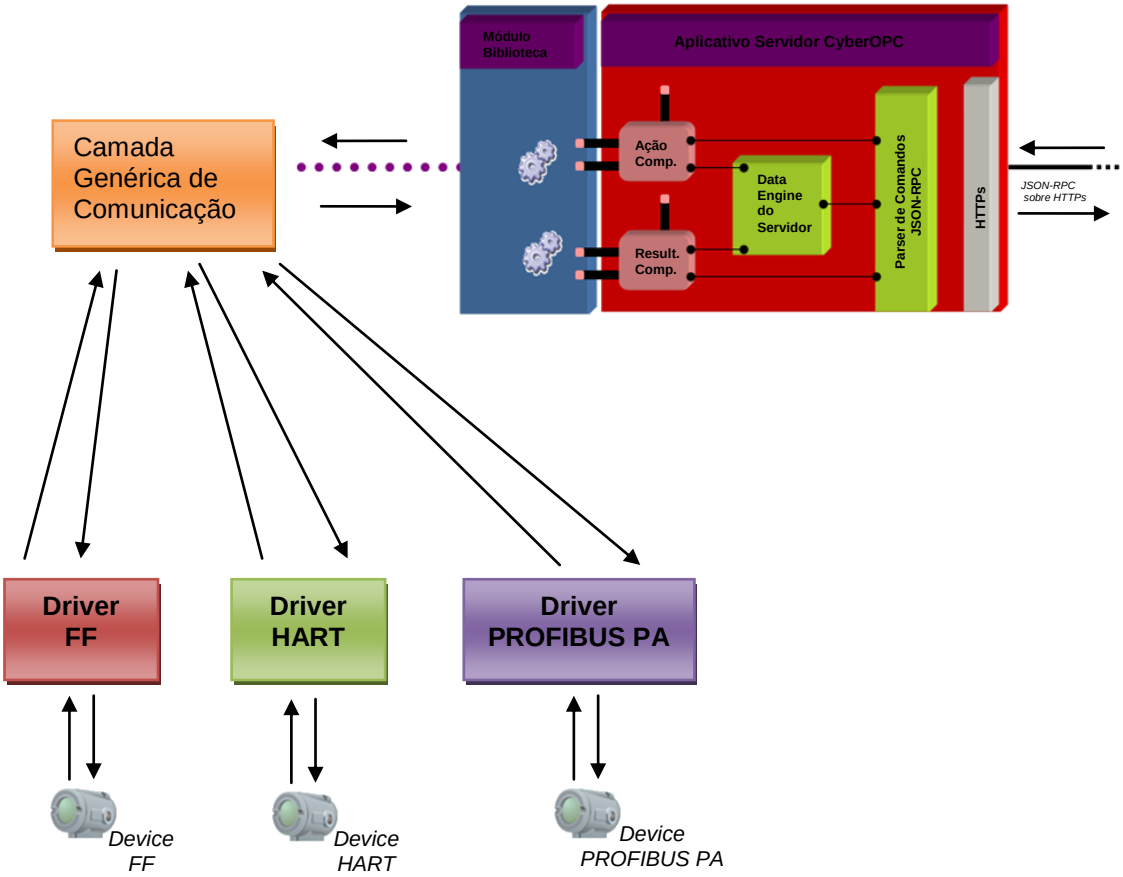

Figura 26 –CyberOPC – Arquitetura Geral e Driver de Comunicação
Fonte: TORRISI et al.(2007), com adaptações

Por sua vez, cada *driver* de comunicação é responsável pelo envio e recebimento das mensagens específicas de cada um dos protocolos envolvidos.

A validação dessa arquitetura será descrita adiante, por meio da implementação de um protótipo e ensaios com instrumentos de campo.

4.2. Detalhamento da arquitetura e implementações utilizadas para os Testes

Para a execução dos testes, foi implementado um servidor CyberOPC com suporte aos protocolos: a) HART (em modo emulado por meio de um servidor OPC DA); b) FF (em modo emulado por meio de um servidor OPC DA) e c) PROFIBUS (em modo emulado por meio de um servidor OPC DA e também em modo real, usando-se um servidor OPC conectado a um Mestre Classe 2 do Protocolo PROFIBUS: o DF73, da Smar).

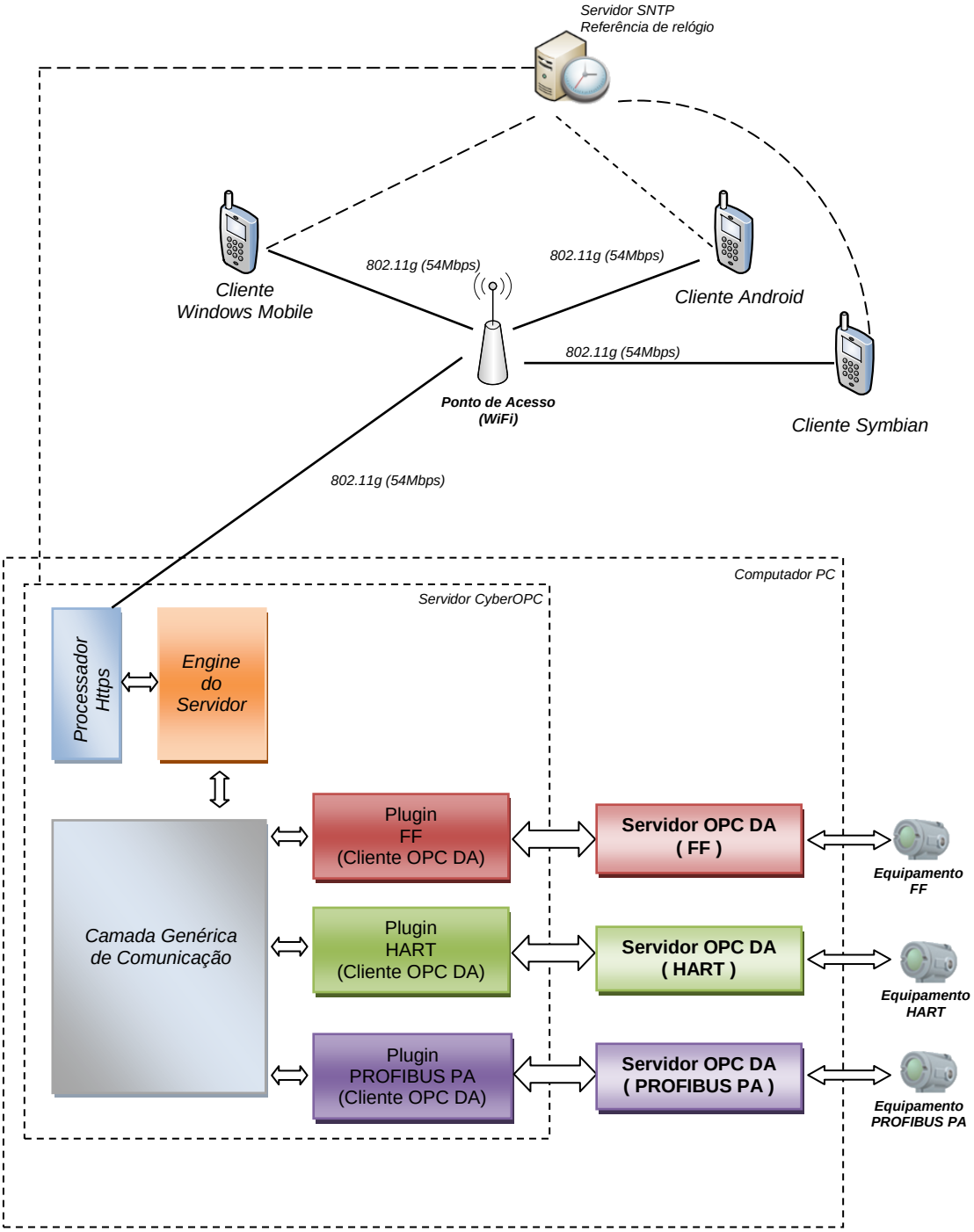

Figura 27 – Arquitetura em detalhes utilizada para os testes

4.2.1. A Temporização das Mensagens

A arquitetura contempla um servidor de relógio para sincronização de todos os elementos envolvidos.

O percurso da mensagem, envolve, basicamente 4 passos (de 1 a 4), e o ciclo é ilustrado na Figura 28, a seguir:

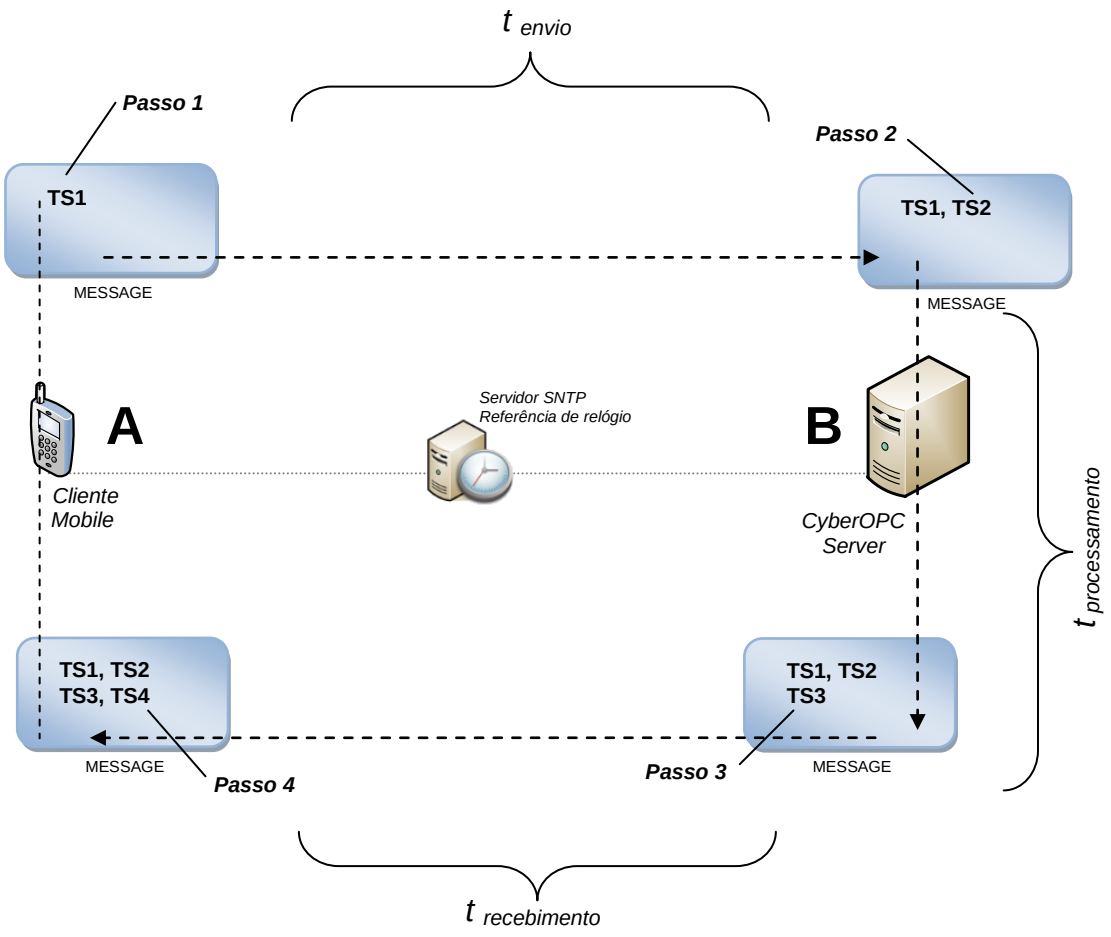

Figura 28 – Ciclo da mensagem de comunicação

A comunicação cliente/servidor é feita por meio do envio da mensagem do ponto ***A*** (dispositivo móvel cliente) para o ponto ***B*** (servidor), sendo processada em ***B*** durante algum tempo e retornando até ***A*** (cliente), conforme descrito na Figura 28.

Os pontos ***A*** e ***B*** são sincronizados com um servidor de relógio que usa o protocolo *Simple Network Time Protocol (SNTP)*, pois a mensagem percorre os passos de 1 a 4, e a arquitetura exige que todos os elementos que adicionem *timestamp* (estampa de tempo) estejam sincronizados entre si, para efeito de normalização da referência de tempo.

No momento da saída de A, um timestamp ***TS1*** é adicionado na mensagem pelo cliente A.

Após chegar em B, imediatamente antes do processamento, o servidor B adiciona outro timestamp ***TS2*** na mensagem. Ao sair de B, o timestamp ***TS3*** é adicionado.

Finalmente, ao receber novamente a mensagem em A, o cliente A adiciona o timestamp ***TS4***.

O tempo total de percurso da mensagem é dado pela equação 1:

$$t_{total} = t_{envio} + t_{processamentoCyberOPC} + t_{recebimento} \tag{1}$$

Com base nessa equação e fazendo-se a amostra dos processamentos das mensagens, é possível saber a capacidade de processamento de mensagens pela arquitetura.

Os valores de TS1 e TS4 são escritos pelo cliente, enquanto TS2 e TS3 são escritos pelo servidor.

Tem-se então:

$$t_{envio} = TS2 - TS1 \tag{2}$$

$$t_{processamentoCyberOPC} = TS3 - TS2 \tag{3}$$

$$t_{recebimento} = TS4 - TS3 \tag{4}$$

Substituindo-se 2, 3 e 4 em 1, tem-se:

$$t_{total} = (TS2 - TS1) + (TS3 - TS2) + (TS4 - TS3) \Rightarrow t_{total} = TS4 - TS1 \tag{5}$$

Os valores de TS1 e TS4 são armazenados pelo cliente tomando-se como base o ID da mensagem, ou seja, ao ser enviada, a mensagem possui um ID que é escrito, segundo o padrão CyberOPC. Ao receber a resposta, o cliente verifica o ID da mensagem recebida e consegue fazer a associação para inferir o valor de TS4, armazenando-se esses dados em um arquivo de log. Os valores de TS2 e TS3 são os próprios valores escritos pelo servidor CyberOPC e identificados como rcvTime e replyTime, conforme ilustrado na Figura 29.

```
POST /COMPUTER HTTP/1.1
User-Agent: CyberOPC Client Library
Host: COMPUTER
Accept: application/json
Connection: Keep-Alive

{
  "params":"null",
  "id":1,
  "method":"GetStatus"
}
```

ID

Mensagem Enviada

```
HTTP/1.1 200 OK
Connection:close
Content-Type: application/json
Server: CyberOPCServer/1.0

{
  "id":1,
  "result":
  {
    "getstatusresult":
    {
      "rcvTime":"1324299695005",
      "replyTime":"1324299695005",
      "serverState":"running"
    },
    "statusinfo":
    {
      "startTime":"2011-11-19T10:58:13.9024570-02:00",
      "productVersion":"1.0"
    },
    "vendorinfo":"RC",
    "specification":"1.0"
  }
}
```

TS2

TS3

Mensagem Recebida

Figura 29 – Mensagem GetStatus (enviada e recebida)

4.2.2. Escopo da implementação do protótipo servidor

A arquitetura do servidor foi divida em módulos para permitir a sua representação por meio de uma visão abstrata, cujo modelo segue o padrão de desenvolvimento orientado a objetos. Esse padrão tem como objetivo representar os módulos por meio de componentes de software, onde o desenvolvimento de cada módulo pode ser feito de maneira independente, facilitando-se, assim o processo de validação.

Os módulos, por sua vez foram organizados para permitir a aplicação da arquitetura de software *Model-View-Controller* (*MVC*), na qual é possível separar a lógica de negócio da lógica de apresentação, tornando viável a independência das camadas envolvidas.

As principais camadas identificadas foram: a) camada de apresentação com o usuário; b) camada de regras de negócios; c) camada de comunicação; d) camada de interface CyberOPC; e e) camada do *driver* de comunicação acoplado ao CyberOPC – esse *driver* tem 3 versões distintas, aplicadas aos protocolos *HART*, *PROFIBUS PA* e *Foundation Fieldbus*.

Para a execução dos testes, implementou-se um servidor CyberOPC com suporte aos protocolos: a) HART; b) FF e c) PROFIBUS PA.

Para cada um dos protocolos descritos foi também realizada a implementação de um servidor OPC DA (*Data Access*).

O desenvolvimento foi concebido utilizando-se o padrão proposto pelo projeto acadêmico CyberOPC (TORRISI et al, 2007), usando-se a linguagem C# .NET, com os seguintes métodos: *GetStatus*, *Browse*, *Read* e *Write*. Apesar de o padrão CyberOPC apresentar suporte a outros métodos, como por exemplo, streaming e AJAX reverso, os mesmos não foram utilizados, uma vez que apenas os métodos supra citados foram suficientes para essa aplicação.

O Servidor CyberOPC, por sua vez, foi ligado a servidores OPC DA, que foram desenvolvidos utilizando-se a linguagem C#, por meio do toolkit de desenvolvimento (ADVOSOL, 2010).

Para cada servidor OPC DA de cada um dos protocolos: HART, FF e PROFIBUS PA, foram também desenvolvidos drivers de comunicação, os quais foram usados para emulação da comunicação com equipamentos físicos.

Os valores dos itens para os servidores OPC DA dos protocolos FF, HART e PROFIBUS PA foram escritos em arquivos no padrão *eXtensible Markup Language* (*XML*) e foram definidos conforme ilustrado nas Figuras 30, 31 e 32, a seguir:

```xml
<?xmlversion="1.0"encoding="utf-8" ?>
<deviceprotocol="FF">
<parameters>
<parameteritemId="AO_MODE_BLK_ACTUAL"value="0"/>
<parameteritemId="AO_MODE_BLK_TARGET"value="1"/>
<parameteritemId="AO_OUT"value="10"/>
<parameteritemId="AO_READBACK"value="50.0"/>
<parameteritemId="AO_SP"value="55.0"/>
<parameteritemId="AO_XD_SCALE_EU_0"value="0.01"/>
<parameteritemId="AO_XD_SCALE_EU_100"value="99.09"/>
<parameteritemId="TRD_BLOCK_ERR"value="0"/>
<parameteritemId="TRD_CAL_MIN_SPAN"value="1"/>
<parameteritemId="TRD_CAL_POINT_HI"value="50"/>
<parameteritemId="TRD_CAL_POINT_LO"value="10"/>
<parameteritemId="TRD_MODE_BLK_ACTUAL"value="0"/>
<parameteritemId="TRD_MODE_BLK_TARGET"value="0"/>
<parameteritemId="TRD_PRIMARY_VALUE_STATUS"value="32"/>
<parameteritemId="TRD_PRIMARY_VALUE_VALUE"value="35.0"/>
<parameteritemId="TRD_SECONDARY_VALUE_STATUS"value="24"/>
<parameteritemId="TRD_SECONDARY_VALUE_VALUE"value="18.0"/>
<parameteritemId="DIAGTRD_BLOCK_ERR"value="0"/>
<parameteritemId="DIAGTRD_CAL_POINT_HI"value="11.0"/>
<parameteritemId="DIAGTRD_CAL_POINT_LO"value="0.23"/>
<parameteritemId="DIAGTRD_CAL_TEMPERATURE"value="25.5"/>
<parameteritemId="DIAGTRD_DEV_SN"value="0003453234"/>
<parameteritemId="DIAGTRD_FIRMWARE_REV"value="1.23"/>
<parameteritemId="DIAGTRD_HW_REV"value="2.45"/>
<parameteritemId="DIAGTRD_MAIN_BOARD_SN"value="09F303AF0A"/>
<parameteritemId="DIAGTRD_ST_REV"value="1"/>
<parameteritemId="AI_MOD_BLK_ACTUAL"value="2"/>
<parameteritemId="AI_MOD_BLK_TARGET"value="2"/>
<parameteritemId="AI_OUT_VALUE"value="30.3"/>
<parameteritemId="DSP_MNEMONIC_1"value="TAG"/>
<parameteritemId="DSP_ALPHA_NUM_1"value="ABC"/>
</parameters>
</device
```


Figura 30 –FF_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo FF

```xml
<?xmlversion="1.0"encoding="utf-8" ?>
<deviceprotocol="HART">
<parameters>
<parameteritemId="COMM_FAILURE"value="0"/>
<parameteritemId="MANUFACTURER_CODE"value="123"/>
<parameteritemId="MANUFACTURER_NAME"value="HART VENDOR"/>
<parameteritemId="DEVICE_CODE"value="54321"/>
<parameteritemId="DEVICE_NAME"value="GENERIC SIMULATOR"/>
<parameteritemId="DEVICE_REVISION"value="1"/>
<parameteritemId="DEVICE_VARIABLE"value="5"/>
<parameteritemId="CONFIG_CHANGE_COUNT"value="1"/>
<parameteritemId="PV_PERC_CHANGE"value="29.93"/>
<parameteritemId="LOOP_CURRENT"value="20.1"/>
<parameteritemId="PV_OUT_OF_LIMITS"value="0"/>
<parameteritemId="PV_ANALOG_OUT_SATURED"value="1"/>
<parameteritemId="PV_ANALOG_OUT_FIXED"value="0"/>
<parameteritemId="MORE_STATUS_AVAILABLE"value="0"/>
<parameteritemId="CONFIGURATION_CHANGED"value="0"/>
<parameteritemId="DEVICE_MALFUNCTION"value="0"/>
<parameteritemId="PV"value="830.2345"/>
<parameteritemId="PV_UNITS"value="5"/>
<parameteritemId="PV_UNITSTRING"value="mm Hg"/>
<parameteritemId="SV"value="99.3432"/>
<parameteritemId="SV_UNITS"value="32"/>
<parameteritemId="SV_UNITSTRING"value="deg C"/>
<parameteritemId="TV"value="240.2345"/>
<parameteritemId="TV_UNITS"value="170"/>
<parameteritemId="TV_UNITSTRING"value="ppm"/>
<parameteritemId="QV"value="234.34"/>
<parameteritemId="QV_UNITS"value="169"/>
<parameteritemId="QV_UNITSTRING"value="m3"/>
<parameteritemId="POOL_ADDRESS"value="0"/>
<parameteritemId="LOOP_CURRENT_MODE"value="Enabled"/>
<parameteritemId="PV_CLASS_STRING"value="Pressure"/>
<parameteritemId="SV_CLASS_STRING"value="Temperature"/>
<parameteritemId="TV_CLASS_STRING"value="Concentration"/>
<parameteritemId="QV_CLASS_STRING"value="Volume"/>
<parameteritemId="MAINTENANCE_REQUIRED"value="0"/>
<parameteritemId="DEVICE_VARIABLE_ALERT"value="0"/>
<parameteritemId="TAG"value="TAG"/>
<parameteritemId="DESCRIPTOR"value="DESCRIPTOR"/>
<parameteritemId="MORE_STATUS_STRING"value="00000000"/>
</parameters>
</device>
```


Figura 31 –HART_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo HART

```xml
<?xmlversion="1.0"encoding="utf-8" ?>
<deviceprotocol="PROFIBUS PA">
<parameters>
<parameteritemId="PHY_DEVICE_ID"value="2000"/>
<parameteritemId="PHY_DEVICE_MAN_ID"value="654"/>
<parameteritemId="PHY_DEVICE_SER_NUM"value="010203"/>
<parameteritemId="PHY_DIAGNOSIS"value="00F10000"/>
<parameteritemId="PHY_DIAGNOSIS_EXTENSION"value="000000000000"/>
<parameteritemId="PHY_DIAGNOSIS_MASK"value="000000"/>
<parameteritemId="PHY_IDENT_NUMBER_SELECTOR"value="1"/>
<parameteritemId="PHY_SOFTWARE_REVISION"value="1.2"/>
<parameteritemId="PHY_TAG_DESC"value="TAG"/>
<parameteritemId="PHY_DESCRIPTOR"value="DESC"/>
<parameteritemId="PHY_DEVICE_MESSAGE"value="MESSAGE"/>
<parameteritemId="TRD_CAL_POINT_HI"value="99.08"/>
<parameteritemId="TRD_CAL_POINT_LO"value="0.03"/>
<parameteritemId="TRD_CAL_TYPE"value="1"/>
<parameteritemId="TRD_MAX_SENSOR_VALUE"value="200.00"/>
<parameteritemId="TRD_MIN_SENSOR_VALUE"value="0.08"/>
<parameteritemId="TRD_MAX_TEMPERATURE"value="400.00"/>
<parameteritemId="TRD_MIN_TEMPERATURE"value="0.00"/>
<parameteritemId="TRD_PRIMARY_VALUE"value="53.65"/>
<parameteritemId="TRD_PRIMARY_VALUE_UNIT"value="deg C"/>
<parameteritemId="TRD_SECONDARY_VALUE"value="50.00"/>
<parameteritemId="TRD_SECONDARY_VALUE_UNIT"value="psi"/>
<parameteritemId="AO_SP"value="80.0"/>
<parameteritemId="AO_READBACK"value="78.3"/>
</parameters>
</device>
```

Figura 32 –PROFIBUS_SIMULATOR_DeviceParameters.xml - Arquivo XML usado para emulação de um Servidor OPC DA do Protocolo PROFIBUS PA

Em modo emulado, o driver de comunicação faz a busca dos itens do servidor OPC DA em arquivos XML, permitindo as operações de Browse, Read e Write.

A justificativa da escolha de um driver em modo emulado está relacionada à simplificação do ambiente de testes, uma vez que não exige a presença de equipamentos físicos.

Além do ambiente emulado, foi escolhido o protocolo PROFIBUS PA, para uso com um equipamento físico, que será detalhado adiante.

A ferramenta utilizada para as implementações de todos os módulos do servidor foi o Visual Studio 2008 da Microsoft. A Figura 33, a seguir, ilustra esse ambiente de desenvolvimento.

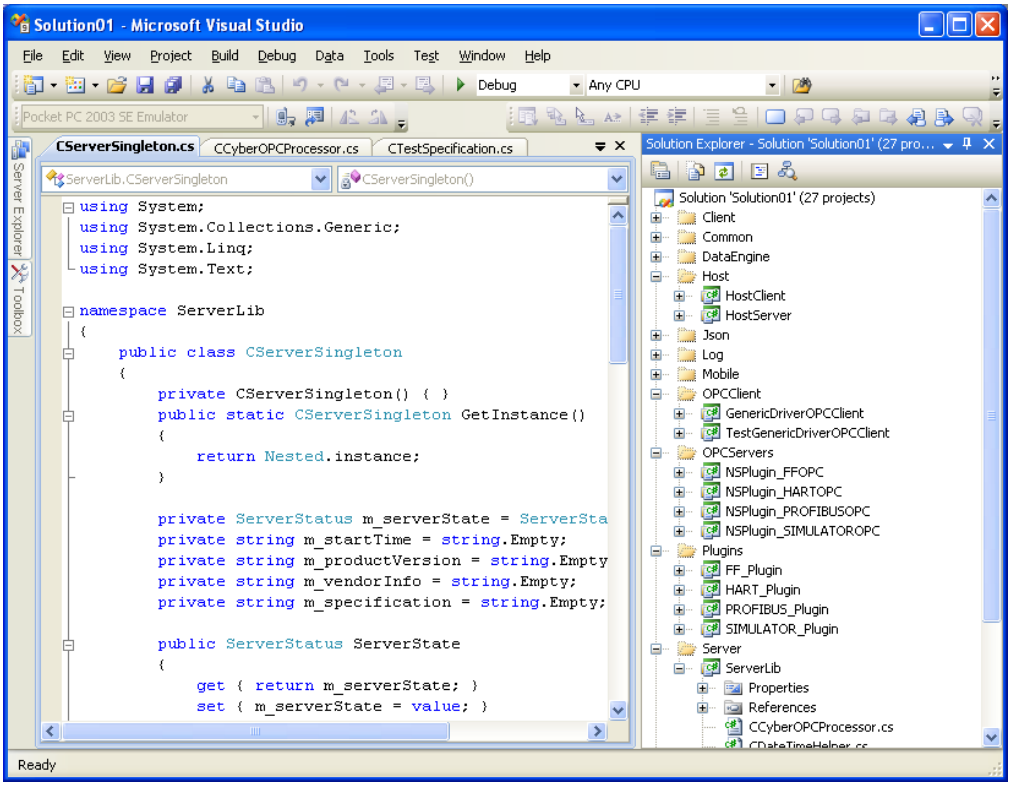

Figura 33 – Visual Studio 2008 – Ferramenta utilizada para o desenvolvimento do servidor

4.2.3. Escopo da implementação do protótipo cliente

Para implementação do protótipo cliente foram escolhidas 2 abordagens a serem utilizadas como elementos de comparação: a) implementação do cliente em plataforma Windows (Computador Desktop); e b) implementação do cliente em plataformas de dispositivos móveis.

Os dados coletados a partir do cliente Windows Desktop são confrontados com os dados coletados a partir de clientes em plataformas de dispositivos móveis.

Dentre as plataformas móveis apresentadas no capítulo 3, foram escolhidas três para o desenvolvimento do software cliente:

a) Windows Mobile;

b) Android;

c) Symbian.

A justificativa dessas três plataformas móveis teve como base o uso desses sistemas operacionais apresentado segundo Canalys (2010) e Gartner (2011).

A escolha do sistema operacional *Windows Mobile* está relacionada ao uso da plataforma Microsoft Windows na grande maioria dos sistemas ligados à automação industrial. Além disso, o *Windows Mobile* possui suporte para desenvolvimento em *.NET* por meio do *.NET Compact Framework* e da Ferramenta Visual Studio, facilitando a implementação de um cliente para acesso aos recursos do instrumento via padrão *CyberOPC*.

O sistema operacional Symbian, segundo Canalys (2010), foi responsável pela maioria dos smartphones utilizados no mundo até 2010.

O sistema operacional Android, por sua vez, apresentou um crescimento bastante acentuado nos últimos anos (CANALYS, 2010) e (GARTNER, 2011), e hoje é responsável por praticamente metade de todos os dispositivos móveis celulares do tipo smartphone.

Com relação ao Android e Symbian, os mesmos suportam desenvolvimento em linguagem Java (GRONLI; HANSEN e GHINEA, 2010). Por esse motivo, a linguagem escolhida para ambas as plataformas foi o Java e a ferramenta de desenvolvimento foi o Eclipse (ECLIPSE FOUNDATION, 2011).

O fato da possibilidade do uso da linguagem Java permitiu, portanto, o reaproveitamento de bibliotecas tanto para o Android quanto para o Symbian.

O software cliente Symbian é um MIDlet Java (uma aplicação Java que é executada em um dispositivo móvel).

Já o software cliente Android, apesar de ser escrito em Java, não é executado em uma Máquina Virtual Java Pura, mas sim em uma máquina virtual Dalvik, padrão nativo do ambiente Android (MURPHY, 2010).

O dispositivo móvel, por sua vez, faz acesso às redes *HART*, *Profibus PA* e *Foundation Fieldbus*, onde pode realizar operações típicas de leitura e escrita de parâmetros para fins de uso na atividade de gerenciamento de ativos.

As figuras 34, 35 e 35 ilustram as telas dos softwares para as plataformas Windows Mobile, Symbian e Android, respectivamente.

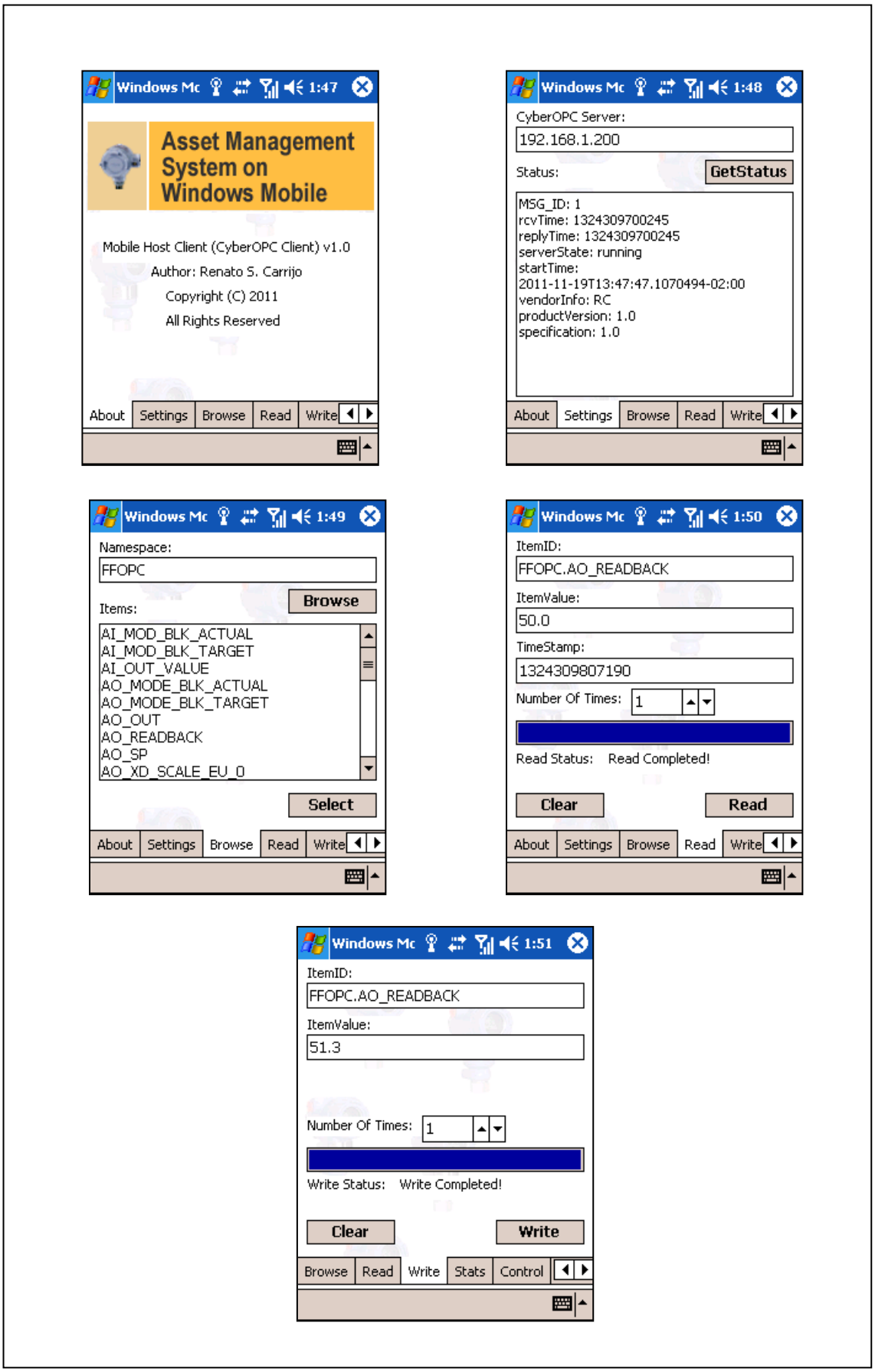

Figura 34 – Cliente CyberOPC (Windows Mobile)

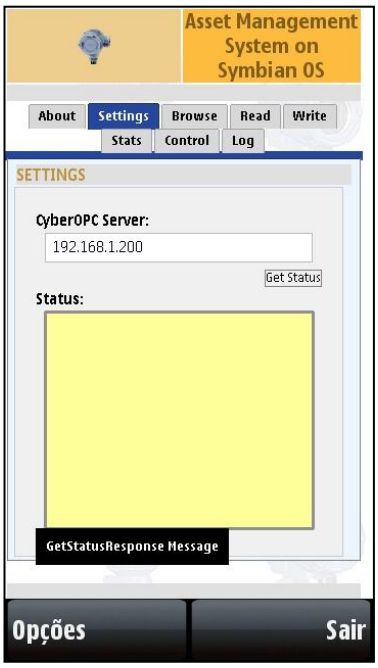

Figura 35 – Client CyberOPC (Symbian)

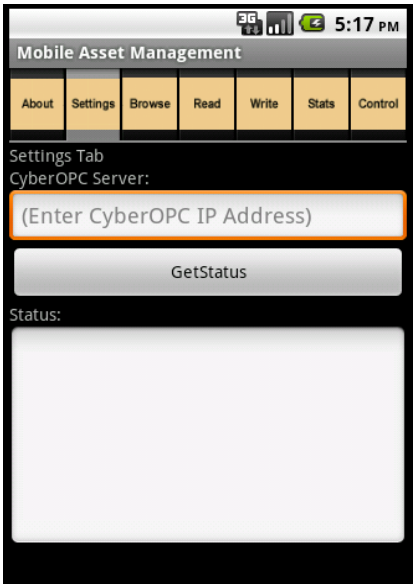

Figura 36 – Client CyberOPC (Android)

4.2.4. Escopo dos testes

Para efeito de levantamento de limites de utilização foram feitos testes usando-se 7 experimentos distintos, utilizando-se como meio de transmissão uma rede wireless do tipo 802.11g (54 Mbps):

a) Servidor CyberOPC com Driver HART Emulado e Cliente Windows Mobile (Pocket PC 2003);
b) Servidor CyberOPC com Driver FF Emulado e Cliente Windows Mobile (Pocket PC 2003);
c) Servidor CyberOPC com Driver Profibus Emulado e Cliente Windows Mobile (Pocket PC 2003);
d) Servidor CyberOPC com Driver Profibus Emulado e Cliente Symbian;
e) Servidor CyberOPC com Driver Profibus Emulado e Cliente Android;
f) Servidor CyberOPC com Driver Profibus Emulado e Cliente Windows Desktop;
g) Servidor CyberOPC com Driver Profibus Real e Cliente Windows Mobile;

Os experimentos **a**, **b** e **c** apresentam como elemento comum o Cliente Windows Mobile aplicado aos protocolos HART, FF e Profibus PA em modo emulado, sendo a escolha de um driver emulado justificada pela possibilidade de simulação sem a necessidade dos equipamentos de campo físicos.

Os experimentos **c**, **d** e **e** apresentam como elemento comum o Driver Profibus Emulado, permitindo a observação do comportamento de três plataformas distintas (Symbian, Android e Windows Mobile) aplicadas a um mesmo protocolo.

Já o experimento **f** fornece dados para observação dos limites de utilização de um cliente sendo executado em um computador do tipo *PC desktop*.

Finalmente, o experimento **g** apresenta dados para a observação dos limites de utilização em Cliente Windows Mobile para acesso a instrumentos de campo físicos (não emulados).

Para cada um dos testes foram enviadas 100 mensagens do tipo Read, armazenando-se os tempos TS1, TS2, TS3 e TS4 descritos no item 4.2.1.

Com base nesses dados, utilizando-se as equações (6) e (7),

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{6}$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2} \tag{7}$$

foram calculados a média (6) e o desvio padrão (7) para as 100 mensagens em cada um dos cenários.

Adicionalmente, foi levantada a capacidade de processamento de mensagens em função do tempo, de acordo com a equação (8).

$$C_{processamento} = \frac{n_{mensagens\ processadas}}{t_{total}} \tag{8}$$

4.2.4.1. **Teste A**: Servidor CyberOPC com Driver HART Emulado e Cliente Windows Mobile (Pocket PC 2003)

O cenário do teste é apresentado na Figura 37.

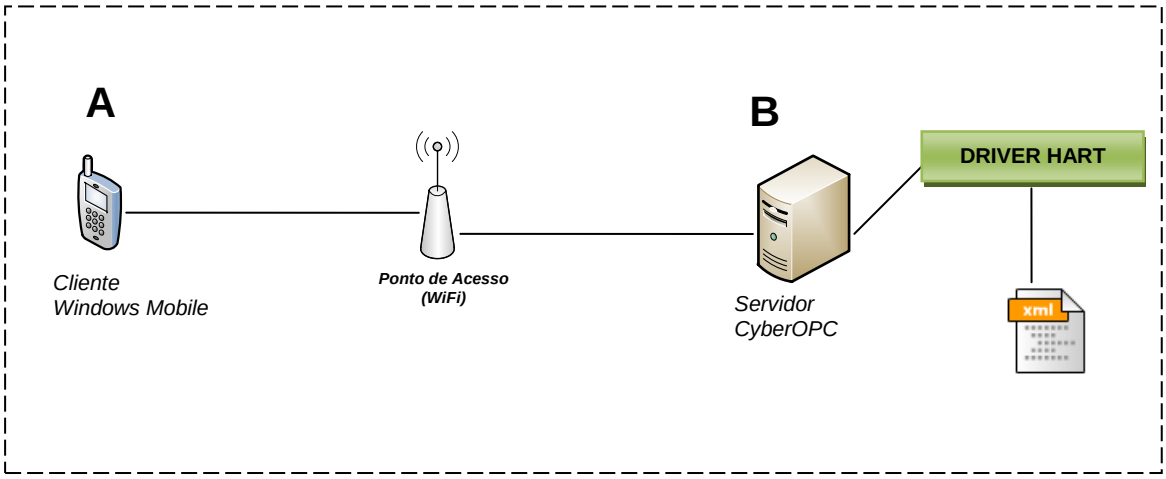

Figura 37 – Servidor CyberOPC com Driver HART Emulado e Cliente Windows Mobile

Resultados Obtidos:

Tabela 5 – Resultados estatísticos obtidos do Teste A

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	72138	721,38	379,63
T processamento (ms)	103951	1039,51	85,73
T recebimento (ms)	49791	497,91	303,69

Tempo total de processamento: 225880 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,44 mensagens/segundo
Capacidade (msg/min): 26,56 mensagens/minuto
Tempo médio de uma mensagem: 2,26 segundos

4.2.4.2. **Teste B**: Servidor CyberOPC com Driver FF Emulado e Cliente Windows Mobile (Pocket PC 2003)

O cenário do teste é apresentado na Figura 38.

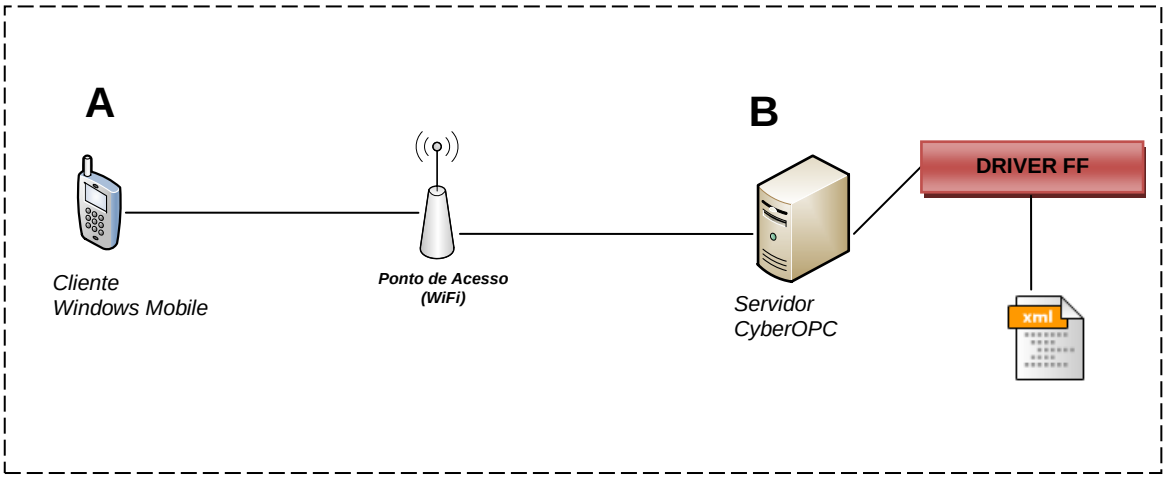

Figura 38 – Servidor CyberOPC com Driver FF Emulado e Cliente Windows Mobile

Resultados Obtidos:

Tabela 6 – Resultados estatísticos obtidos do Teste B

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	43878	438,78	625,61
T processamento (ms)	102342	1023,42	43,94
T recebimento (ms)	50394	503,94	770,07

Tempo total de processamento: 196614 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,51 mensagens/segundo
Capacidade (msg/min): 30,52 mensagens/minuto
Tempo médio de uma mensagem: 1,97 segundos

4.2.4.3. **Teste C**: Servidor CyberOPC com Driver Profibus Emulado e Cliente Windows Mobile (Pocket PC 2003)

O cenário do teste é apresentado na Figura 39.

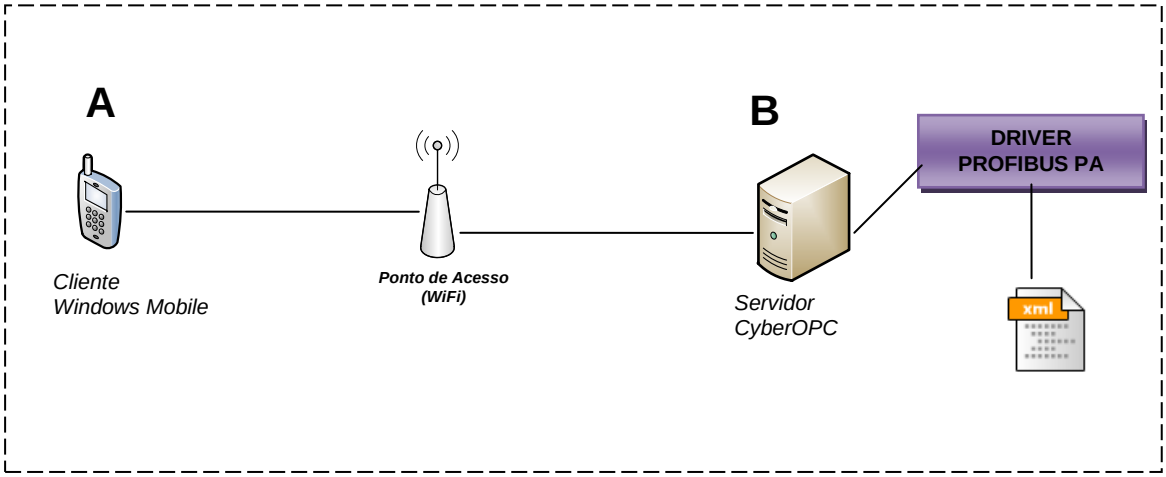

Figura 39 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Windows Mobile

Resultados Obtidos:

Tabela 7 – Resultados estatísticos obtidos do Teste C

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	47946	479,46	599,74
T processamento (ms)	102342	1023,42	43,94
T recebimento (ms)	62572	625,72	780,57

Tempo total de processamento: 212860 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,47 mensagens/segundo
Capacidade (msg/min): 28,18 mensagens/minuto
Tempo médio de uma mensagem: 2,13 segundos

4.2.4.4. **Teste D**: Servidor CyberOPC com Driver Profibus Emulado e Cliente Symbian (Nokia 5800)

O cenário do teste é apresentado na Figura 40.

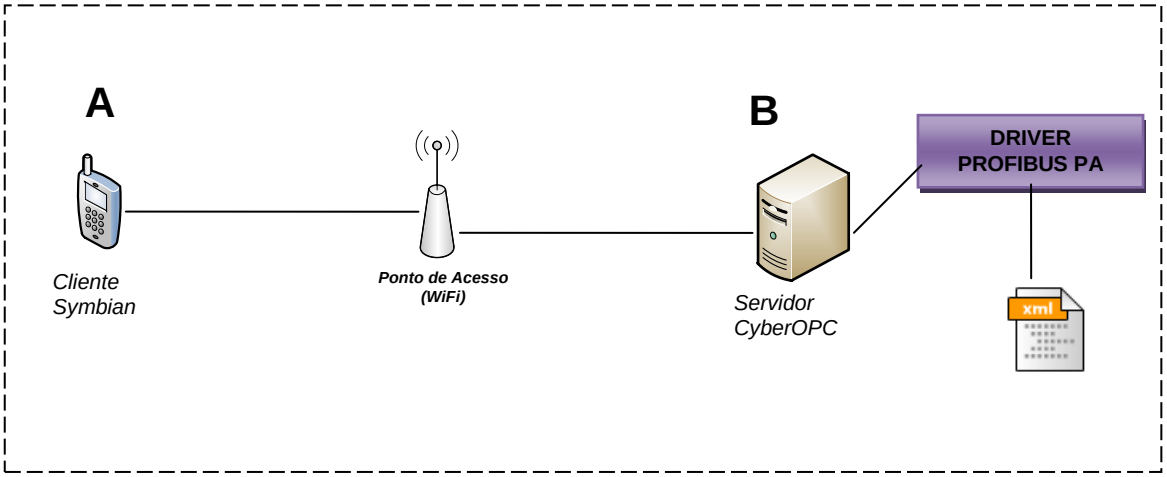

Figura 40 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Symbian

Resultados Obtidos:

Tabela 8 – Resultados estatísticos obtidos do Teste D

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	79247	792,47	529,81
T processamento (ms)	104232	1042,32	42,74
T recebimento (ms)	47713	477,13	290,11

Tempo total de processamento: 231192 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,43 mensagens/segundo
Capacidade (msg/min): 25,95 mensagens/minuto
Tempo médio de uma mensagem: 2,31 segundos

4.2.4.5. **Teste E**: Servidor CyberOPC com Driver Profibus Emulado e Cliente Android (Motorola Atrix)

O cenário do teste é apresentado na Figura 41.

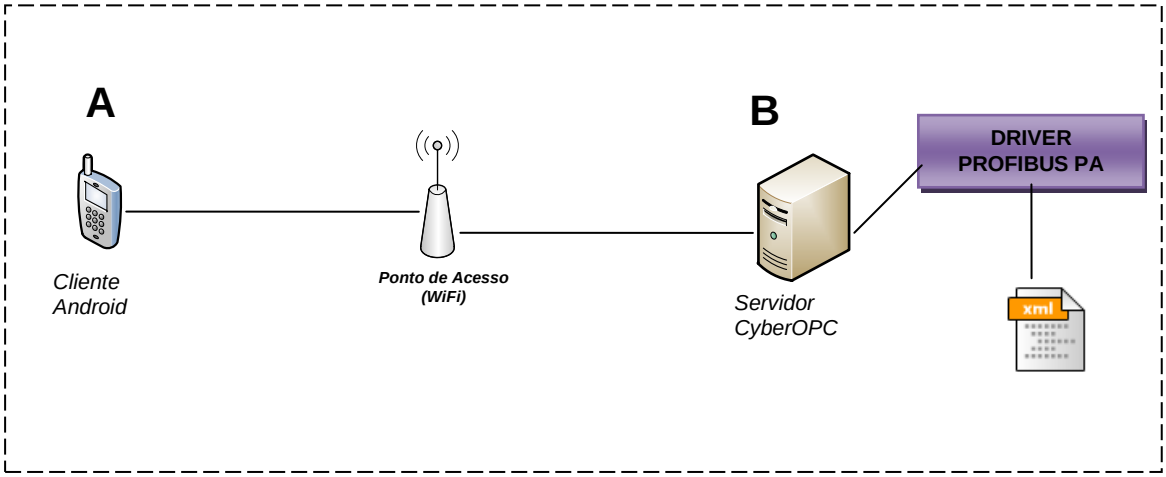

Figura 41 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Android

Resultados Obtidos:

Tabela 9 – Resultados estatísticos obtidos do Teste E

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	57274	572,74	437,99
T processamento (ms)	102479	1024,79	109,58
T recebimento (ms)	42465	424,65	281,82

Tempo total de processamento: 202218 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,45 mensagens/segundo
Capacidade (msg/min): 29,67 Mensagens/minuto
Tempo médio de uma mensagem: 2,02 segundos

4.2.4.6. **Teste F**: Servidor CyberOPC com Driver Profibus Emulado e Cliente Windows Desktop (Windows XP)

O cenário do teste é apresentado na Figura 42.

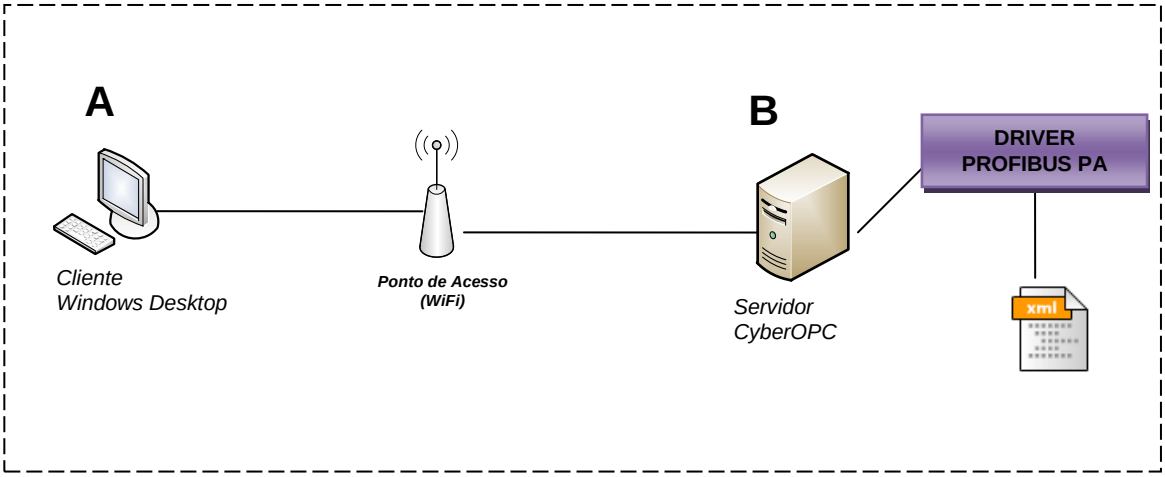

Figura 42 – Servidor CyberOPC com Driver PROFIBUS Emulado e Cliente Windows Desktop

Resultados Obtidos:

Tabela 10 – Resultados estatísticos obtidos do Teste F

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	16828	168,28	46,00
T processamento (ms)	147488	1474,88	15,36
T recebimento (ms)	20079	200,79	3,46

Tempo total de processamento: 184395 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,54 mensagens/segundo
Capacidade (msg/min): 33,54 mensagens/minuto
Tempo médio de uma mensagem: 1,84 segundos

4.2.4.7. **Teste G**: Servidor CyberOPC com Driver Profibus Real e Cliente Windows Mobile (Pocket PC 2003)

O cenário do teste é apresentado na Figura 43.

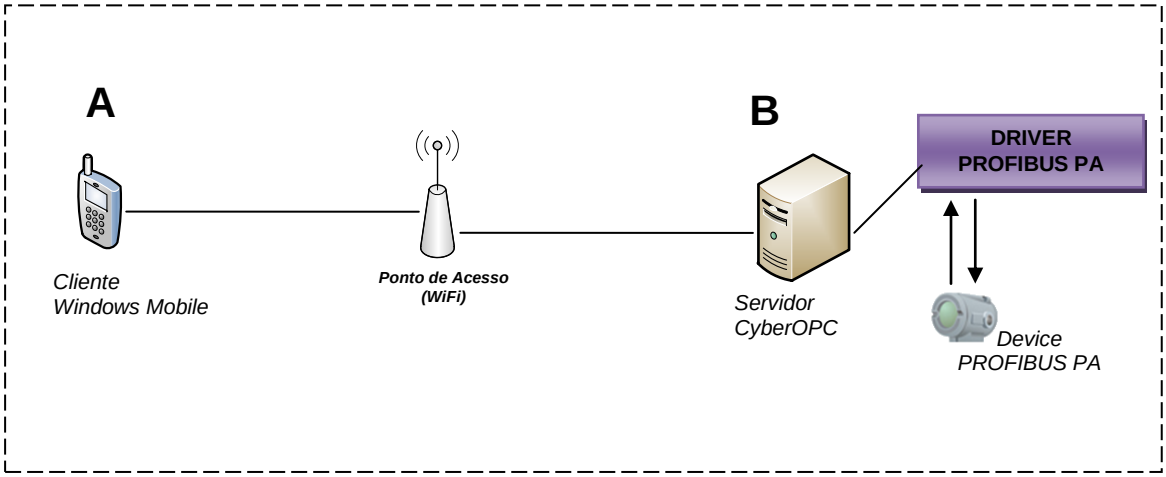

Figura 43 – Servidor CyberOPC com Driver PROFIBUS Real e Cliente Windows Mobile

Resultados Obtidos:

Tabela 11 – Resultados estatísticos obtidos do Teste G

	Tempo Total (ms)	Média	Desvio Padrão
T envio (ms)	76182	761,82	281,04
T processamento (ms)	151338	1513,38	30,38
T recebimento (ms)	22615	226,15	98,21

Tempo total de processamento: 250135 ms
Número de mensagens: 100
Capacidade(msg/seg): 0,40 mensagens/segundo
Capacidade (msg/min): 23,99 mensagens/minuto
Tempo médio de uma mensagem: 2,5 segundos

4.3. Análise e Validação dos Resultados

A partir do desenvolvimento da aplicação para o dispositivo móvel, foram realizados os ensaios de desempenho. Conforme verificado, os limites de utilização estão mais associados com o processamento de dados no servidor do que na aplicação do dispositivo móvel.

A Tabela 12 , a seguir verifica em números, a capacidade de processamento de mensagens por minuto para cada uma das plataformas testadas:

Tabela 12 – Comparação entre o Sistema Operacional, Protocolo e Número de mensagens processadas por minuto

CLIENTE	HART (emulado)	FF (emulado)	PROFIBUS PA (emulado)	PROFIBUS PA (real)
Client Windows Desktop	-	-	33,54	-
Client Windows Mobile	26,56	30,52	28,18	23,99
Symbian	-	-	25,95	-
Android	-	-	29,67	-

As diferenças entre os clientes Windows Desktop e Windows Mobile são significativas, uma vez que o Windows Desktop conseguiu fazer um processamento de 33,54 mensagens por minuto enquanto o Cliente Windows Mobile processou apenas 28,18 , comparando-se o mesmo protocolo PROFIBUS PA emulado.

Comparando-se as plataformas móveis, conseguiu-se um melhor desempenho no Android (29,67 mensagens/minuto), seguido pelo Windows Mobile (28,18 mensagens/minuto) e por fim o Symbian (25,95 mensagens/minuto).

É interessante notar que o processamento real ficou inferior ao emulado, uma vez que dados reais tiveram que ser colocados na rede industrial. Nesse caso, o Windows Mobile conseguiu fazer um processamento de aproximadamente 24 mensagens/minuto.

Apesar de pequenas diferenças, todos eles estão na mesma ordem de grandeza, inclusive o Windows Desktop.

A variação entre o processamento de uma mensagem está na ordem de alguns segundos. Isso assegura, portanto, a possibilidade de uso na área de gerenciamento de ativos.

É possível verificar, portanto, que essa ordem de grandeza (segundos) também coincide com os limites geralmente obtidos pelos padrões OPC DA (*Data Access*) e OPC AE (*Alarms and Events*), que são comumente usados em softwares de gerenciamento de ativos.

Consequentemente, a arquitetura proposta para o uso de dispositivos móveis por meio do *CyberOPC* está validada, sendo capaz de ser utilizada da mesma forma o projeto acadêmico propõe inicialmente.

Os dados coletados por meio dos experimentos estão detalhados no apêndice A.

4.4. Estudo de Caso aplicado a uma operação típica de gerenciamento de ativos

O protótipo desenvolvido foi usado para resolver um problema típico da área de manutenção em uma usina de açúcar e álcool: a manutenção "*in loco"*.

O chão de fábrica do ambiente industrial de uma usina de cana-de-açúcar, na maioria das vezes, não possui o conforto de uma sala de controle onde fica localizado, geralmente, o software de acesso ao sistema de gestão de ativos da planta. Quando um operador necessita efetuar uma calibração de um instrumento de campo, ele se desloca até o local onde o instrumento se encontra em operação e executa a manutenção por meio da comunicação via rádio ou *walk-talks* com outra pessoa localizada na sala de controle, que faz a operação do software de *asset management*.

A possibilidade do processo de calibração via software de gestão de ativos poder ser feito "*in loco"*, a partir de um dispositivo portátil e sem fios, torna-se, portanto, um elemento facilitador para o processo de manutenção industrial.

Para ilustrar o uso da arquitetura proposta foi usado o software sendo executado em um Pocket PC (Windows Mobile) para execução de um *autosetup* de um posicionador de válvula.

O posicionador é o FY303 da Smar, que utiliza o protocolo de comunicação PROFIBUS PA, demonstrado na Figura 44.

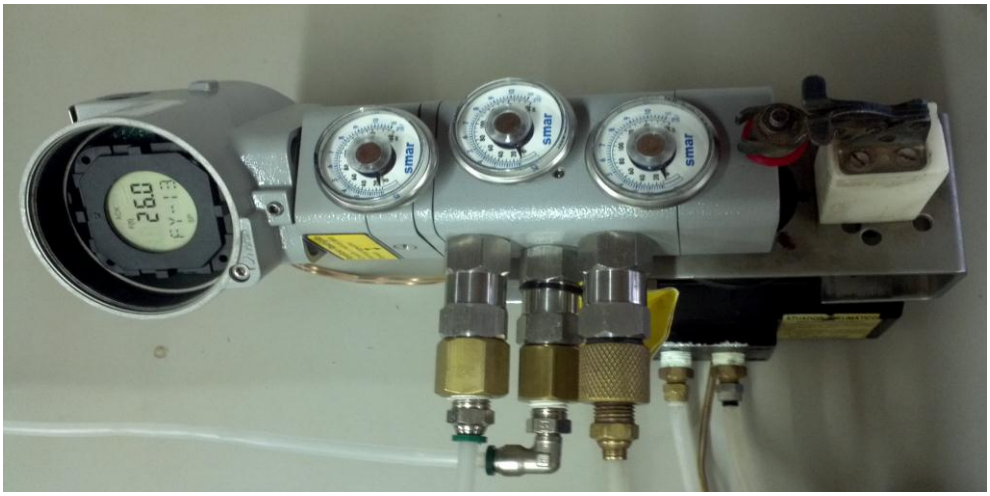

Figura 44 – FY303, Posicionador de Válvula da Smar (Protocolo PROFIBUS PA)
Fonte: Smar (2011)

A tela de operação do aplicativo é demonstrada na Figura 45, a seguir:

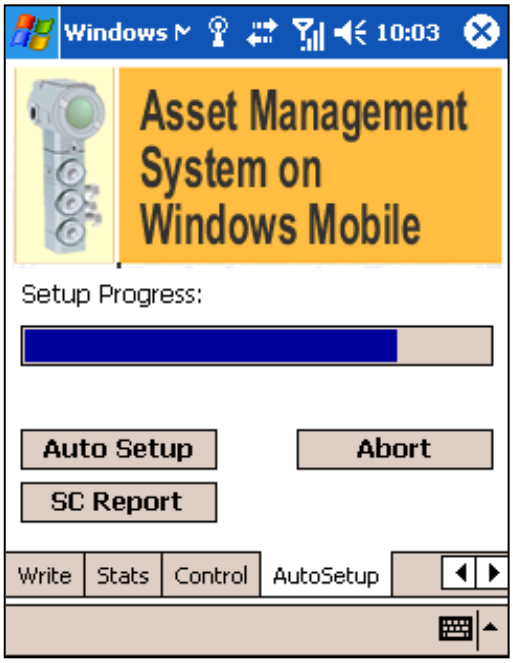

Figura 45 – Cliente CyberOPC no Windows Mobile, executando um *autosetup* do FY303

A operação de execução de um autosetup é um característica do posicionador de válvula FY303 e é bastante simples do ponto de vista de execução.

São exigidos como pré-requisito: os blocos PHYSICAL, TRANSDUCER e AO do FY303 tem que estar no modo automático.

A operação consiste em se escrever um valor na variável *AutoSetup* e logo após fazer-se monitoração do *Transducer Pressure Status*.

O posicionador de válvula posiciona a válvula no começo de curso e realiza uma excursão completa, fazendo-se a calibração dos pontos onde a válvula está totalmente aberta e totalmente fechada, perfazendo-se de 0 a 100 %.

A operação demorou 2 minutos e 58 segundos e o equipamento foi, portanto, calibrado.

5. CONSIDERAÇÕES FINAIS E CONCLUSÃO

A análise dos padrões *HART*, *PROFIBUS PA* e *Foundation Fieldbus* permite concluir que eles são bastante adequados para o uso na aplicação de gerenciamento de ativos, uma vez que possibilitam a exploração das informações de diagnósticos presentes nos instrumentos de campo.

Por meio dos testes executados, é possível concluir que o uso do padrão *CyberOPC* permite um mecanismo simples e eficiente para a troca de dados em um cliente móvel e um servidor de dados que suporte o padrão *OPC*.

Praticamente não foram observadas diferenças entre os drivers *HART*, *FF* e *PROFIBUS PA* quando em modo emulado, uma vez que todos eles compartilham a estrutura de simulação.

A tarefa mais dispendiosa na arquitetura, que exige maior processamento e, por consequência, consome mais tempo está localizada no processamento do *CyberOPC*, no momento em que ele faz a ligação com outro servidor *OPC DA* qualquer. Essa característica apresenta um grande benefício do ponto de vista de interoperabilidade, entretanto, agrega o custo de se trabalhar com interfaces *OPC* gerando um consequente aumento da latência na comunicação.

O uso de plataformas distintas (*Windows Mobile*, *Android* e *Symbian*) não teve o foco da escolha da plataforma mais adequada, mas sim da verificação de independência de plataforma proposta na arquitetura, sendo essa independência validada com sucesso.

O protótipo desenvolvido e usado para os testes permitiu, portanto, a verificação e validação da viabilidade técnica da arquitetura proposta, cumprindo o objetivo proposto por esse trabalho.

Trabalhos futuros poderão estender a aplicação a outros tipos de rede sem fio, como, por exemplo, redes de telefonia celular de terceira e quarta gerações (3G e 4G).

Outras melhorias poderão ser incorporadas ao padrão *CyberOPC*, como, por exemplo, o suporte a outros tipos de codificação que suportem compactação de dados.

O presente trabalho poderá no futuro integrar-se com o projeto acadêmico FIBUSIM (BRANDÃO, 2005), tornando mais rica a experiência de simulação de uma rede industrial.

A possibilidade do uso de plataformas móveis associadas a aplicações diretamente no chão de fábrica abre novos horizontes para solução de antigos problemas e fomenta novas pesquisas no sentido do crescimento do uso da gestão de ativos em um ambiente industrial.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ADVOSOL (2010). Advosol: Advanced OPC Solutions, disponível em:<http://www.advosol.com>, Acesso em: 5 Mar 2010.

AKESSON, I.N. (1999). Plant Auditing – The Core of Asset Management, INTERKAMA, Düsseldorf, ISA. 1999.

APPLE (2010). iOSDeveloper Center. Disponível em: <http://developer.apple.com >, Acesso em: 22 Dez 2010.

BANGEMANN, T.; HAHNICHE, J.; NEUMANN, P. (1998). Integration of fieldbus system in computer-aided facility management. In: INDUSTRIAL ELECTRONICS SOCIET CONFERENCE, 1998. Proceedings.. IEEE V3 p. 1835-1840.

BERGE J. (2002). Fieldbus for Process Control: Engineering, Operation and Maintenance. Research Triangle Park: ISA Books.

BIXLER, T. S. (2008). Remote monitoring and expert diagnostic support for the pulp and paper industry. In: Pulp and Paper Industry Technical Conference, 2008. p. 181-191. (ISSN : 0190-2172).

BRANDÃO, D. (2005). Ferramenta de simulação para projeto, avaliação e ensino de redes *fieldbus*. 151f. Tese (Doutorado em Engenharia Mecânica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005.

CAMPBELL,. (2003). Collaborative Process Automation Systems. Research Triangle Park: ISA Books, 2010.

CANALYS (2009). Canalys research release 2009/112. Disponível em: <http://www.canalys.com/static/press_release/2009/r2009112.pdf>, Acesso em: 22 Dez 2010.

_____. (2010). Canalys research release 2010/111. Disponível em: <http://www.canalys.com/static/press_release/2010/r2010111.pdf>, Acesso em: 22 Dez 2010.

COFFIN, E. H. (2010) - ScientificApparatusMakersAssociation, Disponível em: <http://www.geeintl.com/sama_symbols.aspx>, Acesso em: 23 Dez 2010.

COMER, D. E.; DROMS, R. (2004). Computer Networks and Internets with Internet Applications, 4th ed. Prentice Hall, 2004.

ECLIPSE FOUNDATION (2011). The Eclipse Foundation open source community website. Disponível em: < http://www.eclipse.org>, Acesso em: 11 Ago 2011.

ENDRESS+HAUSER (2008). Automation solutions for Asset Management.Disponível em: <http://www.endress.com>, Acesso em: 23 Nov 2010.

FDT GROUP (2007). FDT Technical Description. Disponível em: <http://www.fdtgroup.org>, Acesso em: 20 Nov 2007.

FELSER, M.; SAUTER, T. (2002). The fieldbus war: history or short break between battles? In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS, 4., 2002, Vasteras. Proceedings… New York: IEEE. p. 73-80.

FIELDBUS FOUNDATION (1999). Foundation specification 31.25 kbit/s physical layer profile: FF-816-1.4. Austin.

_____. (1999a). Foundation specification system architecture: FF-800-1.4. Austin.

_____. (1999b). Foundation specification common file format: FF-103-1.4. Austin.

_____. (1999c). Foundation specification function block application process, part 1. Austin.

FOROUZAN, B. A. (2004). Comunicação de Dados e Redes de Computadores. 3ª ed. New York: McGraw Hill, 2004.

FORSTNER, B.; LENGYEL, L.; KELENYI, I.; LEVENDOVSZKY, T.; CHARAF, H.; (2005).Supporting Rapid Application Development on Symbian Platform. In: Computer as a Tool, 2005. EUROCON 2005.The International Conference on, vol. 1, pp.72, 21-24 Nov 2005.

GARTNER (2011). Gartner report “Market Share: Mobile Communication Devices by Region and Country, 3Q11”. Disponível em: http://www.gartner.com/it/page.jsp?id=1848514>, Acesso em: 2 Dez 2011.

GRONLI, T.M.; HANSEN, J.; GHINEA, G. (2010). Android, Java ME and Windows Mobile Interplay: The Case of a Context-Aware Meeting Room. In: Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, pp 920, 20-23 April 2010.

HART COMMUNICATION FOUNDATION (2010). Disponível em: <http://www.hartcomm.org>. Acesso em: 23 nov 2010.

_____. (1996) HART Communicationn Foundation Document Number: HCF_SPEC-81. Austin.

_____. (1997) HART Communicationn Foundation Document Number: HCF_SPEC-99. Austin.

HOLLENDER, M. (2010). Collaborative Process Automation Systems. Research Triangle Park: ISA Books, 2010.

INTERNATIONAL ELECTROTECHNICAL COMMISION (2000). IEC 61158: Digital data communications for measurement and control – fieldbus for use in industrial control systems. Suíça. CD-ROM.

LARYSZ, J.; NEMEC, M.; FASUGA, R. (2011). User Interfaces and Usability Issues From Mobile Applications. In: ICDICP INTERNATIONAL CONFERENCE ON DIGITAL INFORMATION PROCESSING AND COMMUNICATIONS, 2011. Proceedings.., Part II, p. 30-43.

LI, Q.; LIU, F. (2011). The future of the device integration: Field device integration. In: IEEE 2nd INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND SERVICE SCIENCE (ICSESS), 2011.

LIU, Q.; REN, P. (2007). Design and Implementation of MS/TP in Embedded System. In: Industrial Electronics and Applications, 2007.2nd IEEE Conference on. Proceedings… p 2475-2478.

MAHALIK, N. P. J. (2003). Fieldbus Technology: Industrial Network Standards for Real-Time Distributed Control. Springer. Berlim, 2003.

MICROSOFT (2010). Windows CE Embedded. Disponível em: <http://www.microsoft.com/windowsembedded>, Acesso em: 15 Dez 2010.

MITSCHKE, S. (2004).Demystifying a protocol.ISA 2004.Disponível em: <http://www.isa.org/Template.cfm?Section=InTech&template=/ContentManagement/Contentdisplay.cfm&ContentID=36848>. Acessoem: 29 de out. 2009.

MOHSENI, M. (2003). What does asset management mean to you? In: TRANSMISSION AND DISTRIBUTION CONFERENCE AND EXPOSITION, 2003. IEEE Vol. 3, pp 962-964.

MULLER, J.; UECKER, F.; WOLLSCHLAEGER, M.; DIEDRICH, C.; EPPLE, U.; .(2001). The Asset Management Box – making information about intelligent field devices accessible without configuration. In: Intelligent and Self-Validating Instrumentos – Sensor and Actuators, IEEE Seminar on, pp 9/1 – 9/6, 14 Dec 2001.

MULLER, J.; EPPLE, U.; WOLLSCHLAEGER, M.; DIEDRICH, C. (2003).An Efficient Information Server for Advanced Plant Asset Management.Proceedings of the IEEE.Vol 2, pp 66-73, 2003.

MURPHY, M.(2010). Beginning Android 2. Springer, New York, 2010. ISBN: 978-1-4302-2630-7.

NAGASHIMA, M.; SUZUKI, K; KAI, H; NAKANO, N. (2008). Emerging FDI technologies/requirements and study reporto n tackling it as the solution for multi technology issues of FA open system integration. In: SICE Annual Conference 2008, Japan, Aug. 20-22, 2008.

NAMUR (2001). User Association of Process Control Engineering in Chemical and Pharmaceutical Industries), NAMUR-Recommendation NE91: Requirements for Online Plant Asset Management Systems, First Issue: 11/2001.

NOKIA (2010).Web RunTime (WRT) Forum Nokia.Disponível em: <http://wiki.forum.nokia.com/index.php/Category:Web_Runtime_(WRT)>, Accesso em: 8 Mar 2010.

OPC FOUNDATION (2010). Especificação OPC. Disponível em: <http://www.opcfoundation.org>. Acesso em: 11 Out 2010.

PANTONI, R.P. (2006). Desenvolvimento e implementação de uma descrição de dispositivos aberta e não-proprietária para equipamentos Foundation Fieldbus Baseada em XML. (Dissertação de Mestrado) – Escola de Engenharia de São Carlos, USP.

PROFIBUS INTERNATIONAL (2010). Disponível em:<http://www.profibus.org>, Acesso em: 5 Mar 2010.

_____ (2009). Specification PROFIBUS PA Profile for Process Control Devices V3.02 , 2009.

REGH, J.A.; SWAIN, W.H.; YANGULA, B.P. (1999). Fieldbus in the process control laboratory - its time has come. In: Frontiers in Education Conference, 1999. 29th FIE'99, vol. 3, pp.13B4/12 – 13B4/17, 1999.

SCHULT, W.; POLZE, A. (2002). Aspect-oriented programming with C# and .NET. In: Object-Oriented Real-Time Distributed Computing, 2002 (ISORC 2002). Proceedings. Fifth IEEE International Symposium on Digital Object Identifier, p 241-248.

SYMBIAN FOUNDATION (2010): Licensing. Disponível em: <http://licensing.symbian.org/>. Acesso em 2 Out 2010.

SMAR EQUIPAMENTOS INDUSTRIAIS (2011). Disponível em: <http://www.smar.com>, Acesso em: 23 Nov 2010.

TANEMBAUM, A.S. (1997). Redes de Computadores, Rio de Janeiro: Campus 1997.

THEURICH, S.; LEHMANN, R.; WOLLSCHLAEGER, M. (2010) .Network Asset Models in Intelligent Field Devices. In: Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on, pp. 161-164, 18-21 May 2010.

THOMESSE, J.P. (1998). A Review of the Fieldbuses.Annual Reviews in Control, 1998.

THOMPSON, C. W. (2005). Smart devices and soft controllers. In: IEEE INTERNATIONAL CONFERENCE ON INTERNET COMPUTING, 2005. Proceedings.. IEEE. V.9, p 82-85.

TIPSUWAN, Y.; CHOW, M. (2003). Control methodologies in networked control systems. Control engineering practice, Amsterdam, v. 11, n. 10, p. 1099-1111, Oct. 2003.

TORRISI, N. M.; BRANDÃO, D.; PANTONI, R.P.; OLIVEIRA, J.F.G. (2007).Design of a communication system for integration of industrial networks over public IP networks. In: Industrial Informatics, 2007 5th IEEE Internacional Conference on, vol. 1, no., pp.201-206, 23-27 June 2007.

ZAYKOVSKIY, D.; SCHMITT, A (2008). Java vs. Symbian: A comparison of software-based DSR implementations on mobile phones. In: Intelligent Environments, 2008 IET 4th International Conference on, pp. 1-6, 21-22 July 2008.

ZHANG, P. (2008). Industrial Control Technology: A Handbook for Engineers and Researchers. William Andrew. New York, 2008.

Apêndice A

Dados de Validação e Resultado dos Testes

Tabela A.13 – Dados de tempo de resposta para as mensagens no Teste A

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	1030	1041	71
2	602	1052	654
3	1114	1052	166
4	646	1032	678
5	129	1031	160
6	531	1031	562
7	953	1031	16
8	475	1021	496
9	977	1021	20
10	369	1041	410
11	911	1022	933
12	433	1022	455
13	946	1011	957
14	448	1071	519
15	1050	1031	81
16	532	1042	574
17	1044	1042	86
18	476	1032	508
19	1029	1051	80
20	551	1051	602
21	1003	1031	34
22	535	1012	547
23	1107	1073	20
24	471	1041	512
25	913	1001	914
26	405	1051	456
27	837	1031	868
28	369	1052	421
29	921	1032	953
30	434	1041	475
31	930	1002	932
32	422	1022	444
33	934	1022	956

Continua na próxima página

Tabela A.13 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
34	417	1021	438
35	919	1021	940
36	451	991	442
37	2656	1041	697
38	1098	1021	119
39	500	1041	541
40	922	991	913
41	334	1041	375
42	866	1031	897
43	388	991	379
44	870	1032	902
45	392	1042	434
46	834	1042	876
47	257	1021	278
48	789	1021	810
49	321	1051	372
50	787	1052	839
51	320	1041	361
52	852	1031	883
53	284	1021	305
54	676	1001	677
55	1088	1051	139
56	640	1012	652
57	1172	1042	786
58	466	1021	487
59	978	1011	11
60	420	1011	431
61	832	1041	873
62	374	1042	416
63	916	1012	928
64	418	1002	420
65	891	1051	942
66	343	1041	384
67	905	1021	926
68	397	1032	429
69	939	1012	951
70	451	1022	473
71	863	1052	915
72	296	1051	347
73	848	1051	899
74	410	1041	451

Continua na próxima página

Tabela A.13 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
75	942	1042	16
76	384	1052	436
77	937	1031	968
78	469	1021	490
79	1001	1011	12
80	433	1051	484
81	985	1032	17
82	2220	1021	241
83	642	1041	683
84	1194	1022	216
85	696	1052	748
86	138	1032	170
87	590	1032	622
88	1123	1031	154
89	635	1041	676
90	1087	1011	98
91	619	1022	641
92	1131	1012	143
93	653	1062	715
94	236	1041	277
95	758	1031	789
96	41	1022	63
97	493	1032	525
98	1046	1031	77
99	558	1051	609
100	1020	1021	41

Tabela A.14 – Dados de tempo de resposta para as mensagens no Teste B

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	40	1041	1
2	163	1031	132
3	143	1031	826
4	421	1002	419
5	124	1031	93
6	312	1032	280
7	509	1041	468
8	84	1031	53
9	32	1011	43
10	146	1022	124
11	353	1061	292
12	830	1042	788
13	135	1031	834
14	229	1032	739
15	374	1031	343
16	12	1001	13
17	155	1113	42
18	183	1124	59
19	195	1113	4082
20	988	1011	1
21	802	991	793
22	586	1012	598
23	956	936	108
24	606	919	525
25	406	910	496
26	291	1001	290
27	505	1001	504
28	650	1011	639
29	1255	1158	1097
30	360	1054	1306
31	356	1068	288
32	424	1101	323
33	370	1080	550
34	188	1021	167
35	18	1032	50
36	139	1011	128
37	337	1002	335
38	446	1041	513
39	73	1041	114
40	75	992	83
41	282	1031	251

Continua na próxima página

Tabela A.14 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	303	1011	686
43	106	1011	117
44	316	1021	663
45	248	1042	206
46	415	1001	414
47	287	982	731
48	80	981	61
49	127	1021	106
50	431	1012	419
51	362	1011	627
52	154	1012	166
53	561	1002	559
54	232	1031	737
55	124	991	133
56	3386	1002	612
57	199	1031	770
58	375	1002	373
59	89	992	97
60	398	1021	581
61	276	1022	254
62	1670	1021	309
63	24	1001	23
64	548	1032	516
65	295	1032	673
66	858	1011	847
67	228	1011	761
68	1993	1022	15
69	540	1021	519
70	48	1113	161
71	884	1113	3
72	472	1002	474
73	267	1011	278
74	847	1021	868
75	29	928	101
76	604	910	486
77	15	910	1105
78	382	1001	1381
79	616	1011	605
80	464	1011	6453
81	528	1101	427

Continua na próxima página

Tabela A.14 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	563	1058	505
83	376	1021	603
84	288	1011	277
85	72	1021	51
86	516	1022	494
87	99	1011	88
88	287	1012	275
89	484	1021	463
90	319	1021	660
91	445	1031	414
92	141	1012	847
93	163	1021	142
94	131	1042	827
95	25	982	7
96	150	1001	849
97	47	1041	6
98	4815	1132	53
99	252	1032	220
100	371	1123	506

Tabela A.15 – Dados de tempo de resposta para as mensagens no Teste C

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	195	1041	154
2	318	1031	287
3	12	1031	981
4	576	1002	574
5	279	1031	248
6	467	1032	435
7	664	1041	623
8	239	1031	208
9	123	1011	112
10	301	1022	279
11	508	1061	447
12	985	1042	943
13	20	1031	989
14	74	1032	894
15	529	1031	498
16	143	1001	142
17	310	1113	197
18	338	1124	214
19	350	1113	4237
20	833	1011	156
21	647	991	638
22	431	1012	443
23	801	936	263
24	451	919	370
25	561	910	651
26	446	1001	445
27	660	1001	659
28	805	1011	794
29	1410	1158	1252
30	515	1054	1461
31	511	1068	443
32	579	1101	478
33	215	1080	705
34	343	1021	322
35	137	1032	105
36	294	1011	283
37	492	1002	490
38	291	1041	668
39	82	1041	41
40	230	992	238
41	437	1031	406

Continua na próxima página

Tabela A.15 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	148	1011	841
43	49	1011	38
44	161	1021	818
45	403	1042	361
46	570	1001	569
47	132	982	886
48	75	981	94
49	282	1021	261
50	586	1012	574
51	207	1011	782
52	10	1012	11
53	716	1002	714
54	77	1031	892
55	279	991	288
56	3231	1002	767
57	44	1031	925
58	530	1002	528
59	244	992	252
60	243	1021	736
61	431	1022	409
62	1515	1021	464
63	179	1001	178
64	703	1032	671
65	140	1032	828
66	1013	1011	1002
67	73	1011	916
68	1838	1022	140
69	695	1021	674
70	107	1113	6
71	729	1113	158
72	317	1002	319
73	112	1011	123
74	692	1021	713
75	184	928	256
76	449	910	641
77	170	910	1260
78	537	1001	1536
79	771	1011	760
80	619	1011	6608
81	683	1101	582

Continua na próxima página

Tabela A.15 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	718	1058	660
83	221	1021	758
84	443	1011	432
85	227	1021	206
86	671	1022	649
87	254	1011	243
88	442	1012	430
89	639	1021	618
90	164	1021	815
91	600	1031	569
92	14	1012	1002
93	318	1021	297
94	24	1042	982
95	130	982	148
96	50	1001	1004
97	202	1041	161
98	4660	1132	208
99	407	1032	375
100	216	1123	661

Tabela A.16 – Dados de tempo de resposta para as mensagens no Teste D

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	189	1031	220
2	841	1042	883
3	343	1032	375
4	876	1041	917
5	418	1061	479
6	920	1051	971
7	462	1042	504
8	3027	1032	59
9	519	1022	541
10	941	1032	973
11	444	1041	485
12	1006	1041	47
13	438	1031	469
14	850	1031	881
15	402	1042	444
16	934	1042	976
17	366	1042	408
18	909	1031	940
19	451	1031	482
20	873	1041	914
21	2218	1031	249
22	770	1041	811
23	192	1031	223
24	614	1032	646
25	1136	1032	168
26	658	1042	300
27	461	1032	507
28	1025	1041	66
29	567	1051	618
30	1099	1032	131
31	531	1032	563
32	1073	1052	125
33	606	1051	657
34	1048	1021	69
35	580	1041	621
36	1122	1032	154
37	634	1042	676
38	1136	1052	188
39	599	1041	640
40	3154	1051	205
41	576	1031	607

Continua na próxima página

Tabela A.16 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	1008	1041	49
43	550	1031	581
44	1082	1032	114
45	534	1103	637
46	1177	1167	344
47	774	1135	909
48	327	1135	462
49	1035	1134	169
50	632	1135	767
51	207	1146	353
52	889	1012	901
53	2439	1021	460
54	946	1001	947
55	472	1031	503
56	890	1021	911
57	417	954	371
58	808	919	727
59	1093	928	21
60	413	955	368
61	680	955	635
62	1099	928	973
63	419	984	403
64	786	1021	807
65	180	1021	201
66	577	1021	598
67	1108	1041	149
68	644	1123	767
69	188	1123	311
70	894	1144	38
71	567	1113	680
72	426	1122	548
73	959	1052	11
74	531	1042	573
75	963	1052	15
76	526	1061	587
77	1128	1031	159
78	650	1032	682
79	1182	1042	224
80	725	1031	756
81	287	1081	368

Continua na próxima página

Tabela A.16 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	859	1022	881
83	401	1042	443
84	953	1042	5
85	456	1041	497
86	1048	1021	69
87	610	1042	652
88	1032	1032	64
89	524	1042	566
90	1097	1031	128
91	2692	1041	733
92	214	1062	276
93	696	1042	738
94	248	1022	270
95	761	1041	802
96	184	1021	205
97	606	1062	332
98	320	1051	371
99	882	1042	924
100	444	1042	486

Tabela A.17 – Dados de tempo de resposta para as mensagens no Teste E

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	203	1041	244
2	819	1021	840
3	167	1016	183
4	594	892	486
5	914	928	842
6	769	919	688
7	81	919	0
8	437	940	377
9	799	1011	810
10	222	1030	252
11	694	1021	285
12	2296	1021	317
13	641	1021	662
14	139	1090	229
15	758	1112	870
16	345	1102	447
17	814	1101	85
18	314	1112	426
19	837	1133	30
20	445	1112	557
21	937	1042	21
22	349	1032	381
23	842	1001	843
24	334	1125	661
25	2739	1135	874
26	241	1021	262
27	733	1031	764
28	225	1022	247
29	727	1052	779
30	249	1032	281
31	711	1032	1257
32	375	1042	417
33	857	1052	91
34	360	1041	401
35	892	1031	77
36	404	1041	445
37	906	1042	52
38	388	1032	420
39	910	1042	48
40	443	1041	484
41	945	1051	4

Continua na próxima página

Tabela A.17 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	377	1051	428
43	879	1032	89
44	2434	1032	466
45	926	1022	52
46	438	1062	500
47	311	1032	343
48	693	1042	735
49	105	1042	147
50	537	1012	549
51	899	1012	89
52	371	1052	423
53	813	1032	845
54	316	1001	317
55	798	1051	849
56	230	1031	261
57	716	1032	748
58	188	1012	200
59	560	1052	612
60	143	1041	184
61	645	1041	686
62	137	1041	178
63	593	1022	615
64	5	1032	27
65	487	1022	509
66	990	1041	31
67	412	1041	453
68	804	1051	855
69	286	1041	327
70	708	1071	779
71	140	1041	181
72	532	1041	573
73	934	1021	45
74	346	1021	367
75	738	1061	799
76	270	1062	332
77	742	1052	794
78	274	1062	336
79	807	1031	838
80	329	1061	390
81	761	1041	802

Continua na próxima página

Tabela A.17 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	247	1032	279
83	760	1011	771
84	192	1041	233
85	664	1001	665
86	146	1031	177
87	668	1022	690
88	160	1052	212
89	692	1012	704
90	195	1041	236
91	627	1041	668
92	39	1051	90
93	531	1061	408
94	104	1032	136
95	587	1021	608
96	79	1041	120
97	621	1051	672
98	63	1012	75
99	439	1022	461
100	931	1002	67

Tabela A.18 – Dados de tempo de resposta para as mensagens no Teste F

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	106	1500	200
2	107	1485	200
3	91	1469	200
4	90	1468	200
5	106	1484	200
6	200	1484	200
7	200	1485	200
8	200	1484	200
9	91	1469	200
10	200	1484	200
11	200	1469	216
12	200	1484	200
13	200	1469	200
14	200	1500	200
15	200	1469	200
16	200	1469	200
17	200	1468	200
18	200	1468	216
19	200	1469	200
20	90	1468	200
21	200	1484	200
22	200	1500	200
23	200	1469	216
24	106	1484	200
25	200	1469	200
26	106	1484	200
27	200	1469	200
28	184	1469	200
29	91	1469	200
30	200	1485	200
31	200	1469	200
32	91	1469	200
33	90	1484	200
34	200	1469	200
35	200	1469	200
36	200	1484	200
37	200	1500	200
38	200	1390	200
39	200	1469	200
40	184	1469	200
41	200	1469	200

Continua na próxima página

Tabela A.18 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	184	1484	200
43	106	1484	200
44	91	1469	200
45	91	1485	200
46	91	1469	200
47	184	1469	200
48	106	1484	200
49	200	1469	200
50	200	1391	200
51	106	1484	200
52	106	1484	200
53	107	1485	200
54	200	1469	200
55	91	1469	200
56	184	1469	200
57	106	1469	200
58	200	1469	200
59	184	1469	200
60	200	1469	200
61	184	1468	200
62	200	1484	200
63	200	1469	200
64	200	1485	200
65	106	1484	200
66	200	1469	200
67	90	1468	200
68	185	1469	200
69	200	1485	200
70	106	1484	200
71	200	1469	200
72	200	1469	200
73	200	1469	200
74	200	1469	200
75	200	1468	200
76	91	1484	200
77	200	1500	200
78	200	1484	200
79	200	1485	200
80	200	1468	200
81	200	1468	200

Continua na próxima página

Tabela A.18 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	185	1469	200
83	200	1485	200
84	200	1469	200
85	200	1484	200
86	107	1485	200
87	200	1484	200
88	107	1485	200
89	91	1469	200
90	200	1500	200
91	200	1484	200
92	200	1485	200
93	200	1469	215
94	106	1469	216
95	200	1469	200
96	200	1469	200
97	200	1469	200
98	200	1485	200
99	200	1469	200
100	200	1484	200

Tabela A.19 – Dados de tempo de resposta para as mensagens no Teste G

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
1	720	1547	216
2	691	1484	178
3	672	1516	196
4	701	1531	211
5	710	1500	184
6	690	1500	293
7	1715	1469	211
8	704	1484	225
9	686	1547	601
10	689	1485	600
11	686	1484	173
12	679	1578	187
13	685	1515	193
14	707	1547	190
15	697	1531	205
16	708	1515	201
17	804	1484	175
18	688	1516	911
19	692	1515	196
20	741	1484	208
21	681	1485	188
22	683	1516	194
23	797	1468	199
24	2943	1703	521
25	703	1516	185
26	692	1500	191
27	706	1500	198
28	790	1484	249
29	699	1547	188
30	700	1531	182
31	700	1515	189
32	703	1516	214
33	688	1563	183
34	683	1531	219
35	706	1469	258
36	691	1515	188
37	722	1515	196
38	805	1484	175
39	785	1515	183
40	714	1500	270
41	788	1484	242

Continua na próxima página

Tabela A.19 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
42	723	1469	221
43	686	1516	192
44	835	1500	189
45	717	1515	211
46	671	1547	206
47	806	1485	232
48	701	1484	188
49	698	1531	185
50	689	1516	178
51	699	1500	174
52	680	1516	187
53	689	1515	220
54	916	1485	273
55	671	1516	276
56	676	1578	189
57	693	1531	200
58	684	1516	203
59	699	1516	209
60	816	1469	288
61	788	1484	192
62	695	1500	198
63	694	1500	179
64	702	1516	206
65	717	1531	195
66	702	1516	206
67	690	1516	187
68	703	1515	205
69	772	1531	181
70	597	1531	195
71	682	1531	200
72	683	1500	241
73	781	1484	289
74	788	1500	195
75	715	1515	214
76	1762	1484	179
77	697	1515	191
78	691	1531	252
79	693	1515	205
80	678	1500	195
81	686	1532	196

Continua na próxima página

Tabela A.19 – Continuação da página anterior

ID mensagem	T msg (ar) Env. (ms)	Tempo de Processamento (ms)	T msg (ar) Rec. (ms)
82	696	1516	191
83	716	1500	197
84	680	1516	208
85	706	1500	237
86	786	1500	187
87	831	1485	218
88	696	1515	212
89	701	1516	187
90	705	1516	202
91	694	1563	287
92	549	1484	210
93	1557	1500	288
94	540	1515	239
95	679	1516	188
96	791	1578	185
97	677	1515	201
98	681	1516	287
99	699	1531	173
100	689	1515	230