UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FREDERICO MENINE SCHAF

ARQUITETURA PARA AMBIENTE DE ENSINO DE CONTROLE E AUTOMAÇÃO UTILIZANDO EXPERIMENTOS REMOTOS DE REALIDADE MISTA

Porto Alegre
2006

FREDERICO MENINE SCHAF

ARQUITETURA PARA AMBIENTE DE ENSINO DE CONTROLE E AUTOMAÇÃO UTILIZANDO EXPERIMENTOS REMOTOS DE REALIDADE MISTA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Automação e Instrumentação Eletro-Eletrônica

ORIENTADOR: Carlos Eduardo Pereira

Porto Alegre

2006

FREDERICO MENINE SCHAF

ARQUITETURA PARA AMBIENTE DE ENSINO DE CONTROLE E AUTOMAÇÃO UTILIZANDO EXPERIMENTOS REMOTOS DE REALIDADE MISTA

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Doutor pela Universidade de Stuttgart - Alemanha

Banca Examinadora:

Prof. Dr. João Bosco Motta Alves, UFSC

Doutor pela UFRJ – Rio de janeiro, Brasil

Prof. Dr. João Manoel Gomes da Silva Jr., UFRGS

Doutor pela Universidade Paul Sabatier – Toulouse, França

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS

Doutor pela UFMG – Belo Horizonte, Brasil

Prof. Dr. Walter Fetter Lages, UFRGS

Doutor pelo ITA – São José dos Campos, Brasil

Coordenador do PPGEE: _____

Prof. Dr. Marcelo Soares Lubaszewski

Porto Alegre, Dezembro de 2006.

DEDICATÓRIA

Dedico este trabalho aos meus pais, em especial pela dedicação e apoio em todos os momentos difíceis. Também dedico a todos os familiares que ofereceram apoio moral incondicional para a conclusão deste trabalho.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização de trabalhos em minha área de pesquisa.

Aos colegas do PPGEE e bolsistas de iniciação científica pelo seu auxílio nas tarefas e projetos desenvolvidos durante o curso.

A CAPES pela provisão da bolsa de mestrado e obtenção de acesso a periódicos da IEEE através da UFRGS.

Ao Carlos Eduardo Pereira, meu orientador, que proporcionou um direcionamento para a exposição de idéias presentes neste trabalho.

À FAURGS e à FEENG que custearam a participação no Congresso WIER 2005, de fundamental importância para o direcionamento deste trabalho.

Aos participantes do projeto RExNet que irão proporcionar uma continuação e aprimoração deste trabalho.

Aos integrantes do Artec Lab da Universidade de Bremen na Alemanha onde parte essencial deste trabalho foi desenvolvida e onde ferramentas de realidade mista como o *hyper-bond* foram concebidas.

À equipe de desenvolvimento deste mecanismo essencial para busca na Internet que é o Google e a Wikipedia que é uma ótima enciclopédia virtual.

À minha família que me acolheu durante o curso e que sempre me apoiou incondicionalmente.

RESUMO

Este trabalho aborda o uso de experimentação remota como forma de aprimorar a educação em sistemas de controle e automação. Para isto, uma arquitetura é proposta, onde experimentos reais podem ser combinados com partes simuladas (virtuais) criando um experimento de realidade mista. Um ambiente de ensino à distância, acessível na Internet, engloba o experimento remoto para proporcionar maiores recursos pedagógicos, guias e materiais educacionais teóricos aos estudantes. As diversas tecnologias, presentes no estado da arte, utilizadas para a elaboração da arquitetura são descritas e comparadas para facilitar o entendimento das escolhas realizadas. Trabalhos correlatos são apresentados para descrever o estado da arte dos laboratórios de experiências remotas de diversos tipos. Com base neste estudo, deficiências e vantagens são apontadas para a criação da arquitetura proposta. Para validar a arquitetura são propostos diversos estudos de caso com implementações utilizadas nos cursos de automação e controle da instituição na qual este trabalho está relacionado, ou ainda em cursos de educação à distância para atingir um público mais amplo. Esta dissertação está relacionada como resultado importante do consórcio internacional RExNet, financiado pelo projeto europeu Alfa, do qual a UFRGS faz parte.

Palavras Chaves: Laboratórios Remotos, Experimentação Remota, Ensino à Distância, Laboratórios baseados na Web, Experimentos de Realidade Mista.

ABSTRACT

This work aims to the use of remotely web-based experiments to improve the learning process of automation and control systems theory courses. An architecture combining virtual learning environments, remote experiments, students guide and experiments analysis is proposed based on a wide state of art study. The validation of the architecture uses state of art technologies and new simple developed programs to implement the case studies presented. All implementations presented in this work uses an Internet accessible virtual learning environment providing educational resources, guides and learning material to create a distance learning course associated with the remote mixed reality experiment. The experiment is integrated in the learning environment and support the mixture of real equipments combined with simulated ones (virtual). This mixture of equipments (components) is automatic controlled by the user and is a new developed technology called interchangeable components. The state of art section of this work presents, compare and describes several technologies used to implement the proposed architecture. Related work of remote experimentation and virtual learning environments is also presented. The architecture was elaborated based on appointed advantages and drawbacks of the related studies. This work is part of the RExNet consortium, supported by the European Alfa project.

Keywords: Remote Laboratories, Remote Experimentation, Distance Learning, Web-based Laboratories, Mixed-Reality Experiments.

SUMÁRIO

1 INTRODUÇÃO 18

1.1 MOTIVAÇÃO 18
1.2 CONTEXTUALIZAÇÃO 23
1.3 OBJETIVOS 24
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO 24

2 CONCEITOS TEÓRICOS 25

2.1 EXPERIMENTOS REMOTOS 25
2.1.1 Introdução 25
2.1.2 Simuladores Computacionais 25
2.1.3 Experimentos Reais 28
2.1.4 Experimentos de Realidade Mista ("Mixed-Reality") 35
2.1.5 Arquiteturas e Formas de Acesso 38
2.1.6 Características e Comparações 46
2.2 TREINAMENTOS BASEADOS EM COMPUTADOR (CBT) 48
2.2.1 Contextualização Histórica 50
2.2.2 Ferramentas Computacionais para Ensino à Distância 52
2.2.3 Blackboard 53
2.2.4 Claroline 54
2.2.5 MOODLE 54
2.2.6 WebCT 56
2.2.7 Cursos e Avaliadores 56

3 ESTADO DA ARTE 59

3.1 INTRODUÇÃO 59
3.2 EXPERIMENTOS EXISTENTES 59
3.2.1 OnlineLab (DUAN, 2003) 59
3.2.2 NetLab (ZHANG, 2004) 61
3.2.3 REL & VELO (AUER, 2003) 62
3.2.4 VLab (ALBU, 2004) 63
3.2.5 AIM-Lab (SHEN, 1999) 64
3.2.6 Projeto LabNet (DAVOLI, 2001) 65
3.2.7 Projeto PEARL (FERREIRA, 2002; COPPER, 2000; COOPER, 2002) 66
3.2.8 Projeto WGLN (WGLN, 2005) 70
3.2.9 Projeto MARVEL (MICHAELIDES, 2004; FERREIRA, 2004) 71
3.2.10 ACT (CASINI, 2003; CASINI, 2004) 78
3.2.11 RExNet (ALVES, 2005) 81
3.2.12 Outros 82
3.3 ANÁLISE DO ESTADO DA ARTE 84

4 PROPOSTA DE SISTEMA DE ENSINO PARA AUTOMAÇÃO E CONTROLE USANDO EXPERIMENTOS REMOTOS COM REALIDADE MISTA 86

4.1 ARQUITETURA PROPOSTA 86
4.2 DESCRIÇÃO DOS MÓDULOS DESENVOLVIDOS 89
4.2.1 Incorporação de Exp.Remotos em Ambientes Virtuais de Aprendizagem 89
4.2.2 Integração de Componentes Reais e Simulados 93
4.2.3 Proposta Preliminar de Sistema Tutorial 105

5 VALIDAÇÃO DA ARQUITETURA ATRAVÉS DE ESTUDOS DE CASO 110

5.1 INTRODUÇÃO 110
5.2 DESCRIÇÃO DOS SOFTWARES UTILIZADOS NOS ESTUDOS DE CASO 111
5.2.1 ISaGRAF 111
5.2.2 Elipse SCADA 113
5.2.3 Relés - SENAI 115
5.3 PLANTA FOUNDATION FIELDBUS 116
5.3.1 Equipamentos 117
5.3.2 Objetivo do Experimento 118
5.3.3 Arquitetura 119
5.3.4 Interface Remota 121
5.3.5 Incorporações da Arquitetura Proposta 122
5.4 PLANTA TÉRMICA 131
5.4.1 Equipamentos 132
5.4.2 Objetivo do Experimento 133
5.4.3 Arquitetura 133
5.4.4 Interface Remota 134
5.4.5 Incorporações da Arquitetura Proposta 135
5.5 EXPERIMENTO REMOTO DE MECATRÔNICA 138
5.5.1 Componentes do Experimento 138
5.5.2 Objetivo do Experimento 139
5.5.3 Arquitetura 139
5.5.4 Interface Remota 140
5.5.5 Incorporações da Arquitetura Proposta 141
5.6 PLANTA DE ENVASE DE GARRAFAS 144
5.6.1 Modelo da Planta de Envase de Garrafas 144
5.6.2 Objetivo do Experimento 146
5.6.3 Arquitetura 146
5.6.4 Interface Remota 148
5.6.5 Incorporações da Arquitetura Proposta 148
5.7 PROTÓTIPOS 149
5.7.1 Protótipo 1 151
5.7.2 Protótipo 2 152

6 CONCLUSÕES 155

REFERÊNCIAS 165

APÊNDICE – SIMULADORES 172

LISTA DE ILUSTRAÇÕES

Figura 1. Elementos básicos e relações do processo de modelagem e simulação. 26
Figura 2. Variedade de sistemas de aquisição de dados 30
Figura 3. Linha da Virtualidade (MILGRAM, 1995). 36
Figura 4. Experiência de realidade mista. 37
Figura 5. Comunicação OPC (ZHENG, 2002). 46
Figura 6. Navegador com interface de vídeo conferência. 49
Figura 7. Exemplo de interface do Blackboard 53
Figura 8. Claroline adotado na UFRGS pelo GCAR. 54
Figura 9. Plataforma MOODLE. 55
Figura 10. Experimentos do OnlineLab. 60
Figura 11. Experimento do VLab. 63
Figura 12. VLab – calibração virtual. 63
Figura 13. Experimento do AIM-Lab. 64
Figura 14. Arquitetura e dispositivos do LabNet. 65
Figura 15. Experimento da Open University. 67
Figura 16. Experimento do Trinity College. 68
Figura 17. Experimento de programação de microcontrolador 80C51. 69
Figura 18. Interface do experimento da FEUP. 70
Figura 19. Experimento da Universidade de Bremen. 72
Figura 20. Arquitetura do deriveSERVER. 74
Figura 21. Experimento do HTI. 75
Figura 22. Planta térmica solar de Delmenhorst. 76
Figura 23. Módulo de sistema de reservas. 77
Figura 24. Experimento da Zenon. 77
Figura 25. Experimento de levitação magnética do ACT. 80
Figura 26. Experimento de controle de azimute e elevação. 81
Figura 27. Experimento com mecanismos de force-feedback. 84
Figura 28. Diagrama de interação da arquitetura proposta. 87
Figura 29. Diagrama das interfaces de comunicação dos módulos da arquitetura. 88
Figura 30. Cenário somente com componentes simulados. 95
Figura 31. Gráficos do cenário com componentes simulados. 96
Figura 32. Cenário com controlador simulado e planta real. 97
Figura 33. Gráficos do cenário com planta real e controlador simulado. 98
Figura 34. Cenário com controlador real e planta simulada. 99
Figura 35. Gráficos do cenário com planta simulada e controlador real. 100
Figura 36. Cenário com controlador simulado e planta real. 101
Figura 37. Interações dos componentes intercambiáveis. 102
Figura 38. Arquitetura de comunicação de componentes intercambiáveis 104
Figura 39. Fluxo de dados para criação experiência segundo arquitetura 105
Figura 40. Fluxo de dados para o aconselhamento do sistema tutorial 106
Figura 41. Gráfico das métricas do analisador. 107

Figura 42. Idealização da Arquitetura Proposta. 109
Figura 43. Elipse SCADA "Organizer". 114
Figura 44. Interface do simulador Relés. 115
Figura 45. Foto da construção da atual Planta Didática FF da UFRGS. 117
Figura 46. Tecnologias de redes industriais. 118
Figura 47. Experiência multi-variável. 119
Figura 48. Arquitetura da Planta Didática FF da UFRGS. 120
Figura 49. Interface remota da Planta FF da UFRGS. 122
Figura 50. Curso de funcionamento da planta piloto. 123
Figura 51. Curso de controlador PID. 124
Figura 52. Curso de Foundation Fieldbus. 125
Figura 53. Página Inicial do MOODLE do GCAR na UFRGS. 126
Figura 54. Integração da interface do experimento no MOODLE. 127
Figura 55. Seleção de componentes intercambiáveis na planta FF. 128
Figura 56. Análise do experimento (relatório resumido) realizado. 131
Figura 57. Planta térmica. 132
Figura 58. Arquitetura da planta térmica. 134
Figura 59. Interface Remota do experimento da planta térmica 135
Figura 60. Seleção de componentes intercambiáveis na planta térmica. 137
Figura 61. Interface remota de um experimento de realidade mista 140
Figura 62. Instalação do MOODLE no SENAI em Caxias do Sul. 141
Figura 63. Modificações no deriveSERVER. 142
Figura 64. Interface gráfica da planta de envase. 145
Figura 65. Janelas do ISaGRAF da planta de envase. 146
Figura 66. Arquitetura remota da planta de envase simulada. 147
Figura 67. OPC unindo diferentes experimentos. 150
Figura 68. Protótipo 1. 151
Figura 69. Interações do Protótipo 2. 153
Figura 70. Estudo de caso 156
Figura 71. Metodologia de avaliação. 161
Figura 72. MatLab com Simulink. 173
Figura 73. LTI Viewer. 174
Figura 74. SoftScope. 175
Figura 75. Simulink com elementos de OPC. 177
Figura 76. Interfaces do MatLab Web Server. 177
Figura 77. MatLab RTW. 178
Figura 78. Demonstração de 4 linguagens da IEC 61131. 181
Figura 79. Diagramas seqüenciais SFC e FC. 182
Figura 80. Configuração de rede no ISaGRAF. 183
Figura 81. Arquitetura e comunicação entre elementos do HiBeam 185
Figura 82. Janelas do LabVIEW. 186
Figura 83. Diagrama de utilizações do LabVIEW. 189
Figura 84. Alvos da aplicação do LabVIEW. 191
Figura 85. Abrangência do LabVIEW. 192
Figura 86. CAD e Simulação no FluidSim. 193
Figura 87. Ajuda no FluidSim. 194
Figura 88. Modelos do 20-Sim. 195
Figura 89. Pacote de mecânica 3-D do 20-Sim. 197
Figura 90. Interface e simulação do COSIMIR. 198
Figura 91. Simulação no COSIMIR vs. realidade. 199

Figura 92. Aplicações do COSIMIR.....200
Figura 93. Interface OPC do COSIMIR.....202
Figura 94. Elipse SCADA "Organizer".....203
Figura 95. Visualização de vídeo e dados.....206
Figura 96. Janela do simulador "Relés".....207

LISTA DE TABELAS

TABELA 1. CLASSIFICAÇÃO DOS LABORATÓRIOS 20

TABELA 2. COMPARAÇÃO ENTRE LABORATÓRIOS 21

TABELA 3. EXPERIMENTOS REAIS VS. VIRTUAIS 47

TABELA 4. LINGUAGENS DE PROGRAMAÇÃO DA IEC 61131-3. 112

TABELA 5. PROTOCOLO DE 32 I/OS PELA PORTA PARALELA. 116

TABELA 6. CARACTERÍSTICAS DOS ESTUDOS DE CASO. 157

TABELA 7. CARACTERÍSTICAS DOS PROTÓTIPOS. 158

TABELA 8. TABELA COMPARATIVA DA ARQUITETURA PROPOSTA. 162

LISTA DE ABREVIATURAS

ADL	Advanced Distributed Learning
API	Application Programming Interface
ASP	Active Server Pages
ATE	Automatic Test Equipment
AVA	Ambiente Virtual de Aprendizagem
CAD	Computer-Aided Design
CBL	Computer Based Learning
CBT	Computer Based Training
CGI	Common Gateway Interface
CI	Circuito Integrado
CNC	Computer(ized) Numerical(ly) Control(led)
COM	Component Object Model
CSCL	Computer Supported Collaborative Learning
CSCW	Computer Supported Collaborative Work
DAC	Digital Acquisition Card
DCOM	Distributed Component Object Model
DCS	Distributed Control Systems
DDE	Direct Data Exchange
DDK	Driver Development Kit
DIN	Deutsches Institut für Normung (German Institute for Standardization)
DLL	Dynamic Link Libraries

DSC	Datalogging and Supervisory Control
DSP	Digital Signal Processor
EAI	External Authoring Interface
FPGA	Field-Programmable Gate Arrays
GPIB	General Purpose Instruments Bus
GPL	General Public License
GUI	Graphical User Interface
HIL	Hardware-In-the-Loop
HPIB	Hewlett-Packard Instruments Bus
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HW	Hardware
I/O	Input/Output
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Devices
IEEE	Institute of Electrical and Electronics Engineers
IHM	Interface Homem Máquina
IMS	Instructional Management System
ISO	International Organization for Standardization
IVI	Interface to Virtual Instrument
JMF	Java Media Framework
JRE	Java Runtime Environment
JVM	Java Virtual Machine
LOM	Learning Object Metadata
LTSC	Learning Technology Standard Committee

NVH	Noise, Vibration and Harshness
ODBC	Open Database Connectivity
ODE	Ordinary Differential Equation
OLE	Object Linking and Embedding
OPC	OLE Process Control
OPC-DA	OPC – Data Access
OSI	Open Systems Interconnection
PAC	Programmable Automation Controller
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
PLC	Programmable Logic Controller
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
PWM	Pulse Width Modulation
PXI	PCI eXtensions for Instrumentation
QoS	Quality of Service
RCP	Rapid Control Prototype
RMI	Remote Method Invocation
RSS	Really Simple Syndication
RT	Real Time
RTCP	Real-Time Control Protocols
RTU	Remote Terminal Unit
SBBT	Second Best of Being There

SCADA	Supervisory Control and Data Acquisition
SCORM	Sharable Content Object Reference Model
SCPI	Standard Commands for Programmable Instrument
SDK	Software Development Kit
SIL	Safety Integrity Level
SQL	Structured Query Language
SW	Software
TCP/IP	Transfer Control Protocol / Internet Poster
UCIT	Universal Constructive Instructional Theory
UDP	User Datagram Protocol
UGUI	Universal Graspable User Interface
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuit
VISA	Virtual Instrument Software Architecture
VLE	Virtual Learning Environment
VME	Versa Module Europa
VRML	Virtual Model Modeling Language
VXI	VME eXtensions for Instrumentation
WAVES	Web-based Audio/Video Education System
WBT	Web Based Training
WWW	World Wide Web
XML	eXtensible Markup Language

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

É de conhecimento geral que laboratórios possibilitam aplicação e testes de conhecimentos teóricos em situações práticas (AUER, 2003). O ensino de sistemas de controle e automação exige o contato de alunos com situações reais para permitir um confronto da teoria clássica e moderna com a prática. Do ponto de vista pedagógico, a utilização de laboratórios pelos alunos proporciona: um aprendizado ativo (*active learning*; WATSON, 1995; SMITH, 1989), um aprendizado distribuído (*distributed learning*; AUER, 2003) e um aprendizado de grupo (*team learning*; FALTIN, 2002). Por aprendizado ativo entende-se o "aprender fazendo" (*learning by doing*), enquanto no aprendizado distribuído tem-se uma separação de tarefas e responsabilidades entre alunos. No aprendizado de grupo, o trabalho em equipe e a comunicação entre alunos, para se chegar ao resultado esperado, são os principais conceitos explorados. O uso de laboratórios para aplicações práticas será referenciado ao longo deste trabalho como experimentação.

Experimentos reais instalados nos laboratórios das instituições de ensino são geralmente caros, isto é, a instituição além do custo de aquisição do equipamento necessita arcar com a manutenção, instalação e possível reposição dos materiais utilizados na experiência real. O aumento no número de estudantes demanda um aumento no número de laboratórios em universidades e outras instituições de ensino.

O crescimento da Internet, acessível a um número maior de estudantes, proporciona que novas tecnologias de educação sejam utilizadas, como o uso de laboratórios remotos, que são uma alternativa para atender à demanda por mais laboratórios (COOPER, 2000). Esta alternativa proporciona aos alunos "a segunda melhor alternativa de estar lá" (SBBT – *Second best of beeing there*; ATKAN, 1996). Laboratórios remotos são ferramentas poderosas para

ilustrar conceitos ministrados em aulas proporcionando que tecnologias de ponta sejam utilizadas também para ensino à distância, diminuindo assim a lacuna entre o ensino clássico e as práticas industriais.

Experimentos remotos, baseados na *Web* (*Web-based remote experiments*), possuem diversos aspectos positivos para a implantação, pois utilizam frequentemente equipamentos industriais caros, que requerem uma área específica (muitas vezes grande) para a instalação. Este tipo de experimento é acessível a um maior número de estudantes, pois possui tanto flexibilidade espacial (alunos podem estar fora da instituição de ensino), quanto temporal (os experimentos podem ser acessados 24 horas por dia, nos 7 dias na semana).

Embora os laboratórios remotos estejam aumentando o alcance dos laboratórios comuns, o uso deles para o ensino à distância deve estar ligado a materiais educacionais que expliquem o funcionamento do experimento, já que o professor ou instrutor não acompanhará a experiência remota. O desenvolvimento de ambientes virtuais de aprendizagem (*VLE – Virtual Learning Environment*), acessíveis na Internet, também possibilita que cursos atendam e estejam disponíveis a um número maior de alunos. Estes cursos, por sua vez, podem ser direcionados para o ensino da experiência usada no laboratório remoto. Os professores ligados a estes cursos nem sempre estão disponíveis, gerando a necessidade de desenvolvimento de ambientes virtuais de aprendizagem.

Laboratórios são basicamente classificados, segundo Auer (2003), quanto à localização dos alunos e tipo do experimento, conforme mostra a Tabela 1. Analisando esta tabela, nota-se um confronto evidente entre simulações, laboratórios virtuais e laboratórios reais, presentes tanto localmente como remotamente. Simulações realísticas representam bem situações reais de experimentos físicos, embora experimentos reais sejam melhores para o aprendizado. Laboratórios virtuais não existem fisicamente e geralmente utilizam todos os equipamentos simulados (virtuais). Experimentos reais são melhores educacionalmente que

simulações, pois apresentam características muitas vezes desprezadas pelos simuladores - como ruídos (perturbações), limites de controle e efeitos não lineares -, além de possuir uma dinâmica real do processo e possibilitar o uso de equipamentos empregados na indústria.

Tabela 1. Classificação dos laboratórios

		Aluno	
		Local	Remoto
Experimento	Real	Laboratório Tradicional	Laboratório Remoto
	Virtual	Simulação Local	Laboratório Virtual

A utilização de laboratórios reais produz freqüentemente diferenças entre resultados teóricos e resultados experimentais (implementação prática). Apesar de este aspecto levar, muitas vezes, os alunos à formulação do mito de que “a teoria na prática não funciona”, é importante frisar que a teoria, se bem aplicada, funciona resolvendo problemas do mundo real. Portanto, para se utilizarem tanto laboratórios remotos (reais) como simulações, é necessário demonstrar as diferenças entre as abstrações (modelos), os equipamentos reais e seus respectivos comportamentos em materiais educacionais.

A integração dos experimentos remotos (tanto reais quanto virtuais) com elementos educacionais interativos (tutoriais, exemplos e links) é simplificada, visto que o ambiente do experimento remoto já possui uma interface Web. O compartilhamento de equipamentos caros por diferentes grupos e instituições, usando implementações com tecnologias de Internet amplamente disponíveis e por custos baixos, é outra grande vantagem.

Conclui-se, portanto, que laboratórios reais acessíveis remotamente com suporte de ensino (*VLE*s automatizados) representam uma interessante alternativa para ensino de controle e automação. Métodos de segurança devem ser utilizados, como o acesso restrito aos

equipamentos, para que o custo de implantação compense. Todas estas tecnologias levam a um custo alto por estudante, que deve ser analisado cuidadosamente pela instituição de ensino, restringindo, assim, o número de entidades que podem arcar com este investimento. Autores como Atkan (ATKAN, 1996) afirmam que, embora os investimentos sejam altos, a utilização de laboratórios remotos para ensino se torna muito mais atrativa economicamente.

A Tabela 2 procura relacionar os tipos de laboratórios, descritos na Tabela 1, com características importantes para a escolha adequada da instituição de ensino. O termo 24-7 indica que o laboratório estará sempre disponível, ou seja 24 horas por dia, todos os sete dias da semana.

Tabela 2. Comparação entre laboratórios

	Custos	Didática	Acessibilidade
Laboratório Tradicional	Alto	Dependente do professor/instrutor	Dependente da instituição e do professor/instrutor
Laboratório Remoto	Alto	SBBT Requer VLE e material didático de boa qualidade	24-7 Requer agendador para distribuição de horários entre grupos/alunos
Simulação Local	Depende do custo do simulador e do modelo	Perda da ligação com realidade Dependente do professor/instrutor	Dependente da instituição e do professor/instrutor
Laboratório Virtual	Depende do custo do simulador e do modelo	Perda da ligação com realidade Requer VLE e material didático de boa qualidade	24-7

Uma solução, que busca combinar as vantagens e minimizar as desvantagens das alternativas acima descritas, combina laboratórios reais acessíveis remotamente com simulações, levando aos chamados “experimentos de realidade mista” ou *mixed-reality* (MR; MILGRAM, 1994). Experiências deste tipo usam elementos reais interagindo

bidirecionalmente com elementos simulados. As simulações utilizadas em aplicações de realidade mista são dotadas de grande realismo para que a experiência não perca a ligação com a realidade. Essa solução apresenta-se como uma alternativa de grande interesse para ensino de controle e automação e será explorada neste trabalho.

Experimentos de realidade mista constituem ótimas ferramentas de aprendizagem, embora didaticamente insuficientes, quando não associados aos materiais educacionais teóricos ou aos tutores virtuais. Portanto, é preciso desenvolver materiais e tutores para que o aprendizado seja eficiente e de qualidade. Os mecanismos de aprendizagem (*learning engines*), chamados assim na comunidade científica, podem ser integrados a experiências remotas e virtuais. Mecanismos de aprendizagem propiciam aos alunos tanto materiais educacionais quanto guias para estudo. Esses mecanismos ainda não são utilizados em larga escala nas instituições de ensino, mas há um crescente das universidades em difundí-los como reforço às aulas tradicionais (COOPER, 2002). Dessa forma, é possível transformar simples experiências de acesso remoto em cursos de ensino à distância. Utilizações de ambientes de aprendizagem à distância flexibilizam cursos e tornam mais dinâmicas e estimulantes as práticas de aprendizagem autodidáticas e colaborativas (entre alunos). Esses mecanismos não possuem ainda nenhuma forma de avaliar o que o aluno aprendeu e o que seria melhor reforçar. Existem, no entanto, algumas estratégias de avaliação do aprendizado, que são chamadas de mecanismos de avaliação (*evaluation engines*) (STERGIOULAS, 2002). Já há algumas propostas na comunidade científica para se criar um mecanismo automático de ensino e avaliação (BORTZ, 2002), mas deve-se avaliar muito mais a capacidade pedagógica do mecanismo do que a forma de sua constituição.

Neste trabalho será proposta uma arquitetura que utiliza um ambiente de aprendizagem ligado a experimentos remotos de realidade mista, presentes na pesquisa e ensino da Universidade Federal do Rio Grande do Sul.

1.2 CONTEXTUALIZAÇÃO

A presente dissertação visa dar continuidade ao trabalho desenvolvido por Rafael Zeilmann (ZEILMANN, 2002) no qual é proposta uma estratégia para controle e supervisão de processos via Internet. A estratégia foi validada com a implementação de uma planta-piloto de experiências remotas com equipamentos industriais que utilizam o protocolo de comunicação *Foundation Fieldbus*. A proposta desta dissertação estende o trabalho de Zeilmann, buscando a integração entre experimentos remotos com ambientes de aprendizagem e novas tecnologias para experimentação remota utilizada na educação.

Este trabalho desenvolvido também se insere no escopo do consórcio *RExNet* (ALVES, 2005), projeto europeu ALFA desenvolvido em parceria com universidades européias (Universidade de Bremen e Universidade Técnica de Berlim - Alemanha, Universidade do Porto e Instituto Superior de Engenharia do Porto - Portugal e Universidade de Dundee - Escócia) e latino-americanas (Instituto Técnico de Monterrey - México, Universidade Católica do Chile e Universidade Católica de Temuco - Chile), contando também com a participação de duas universidades brasileiras (as universidades federais de Santa Catarina – UFSC e do Rio Grande do Sul – UFRGS). O projeto versa sobre a criação de uma rede de experimentos remotos desenvolvidos pelas universidades participantes, buscando harmonizar o acesso aos experimentos através do uso de ferramentas de ensino padronizadas.

No âmbito do consórcio *RExNet*, estabeleceu-se uma parceria bastante efetiva com as Universidades de Bremen e Berlim da Alemanha, com as quais foi definido um outro projeto em parceria com o SENAI-RS, para o desenvolvimento de um laboratório real e virtual de eletro-pneumática para uso em mecatrônica, destinado ao ensino técnico no Centro de Mecatrônica do SENAI-RS, em Caxias do Sul, o qual também contribuiu para o desenvolvimento da presente dissertação.

1.3 OBJETIVOS

O objetivo principal do trabalho é propor uma arquitetura para um ambiente de ensino de automação e controle que integre os seguintes conceitos: (i) experimentos de realidade mista remotos, (ii) componentes intercambiáveis, (iii) ambiente virtual de aprendizagem, (iv) avaliação do aprendizado do aluno e (v) um guia de ensino que instrua o aluno que materiais educacionais devem ser revisados (*feedback*).

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação de mestrado encontra-se assim organizada: na seqüência a este capítulo com a motivação e introdução, tem-se uma descrição de conceitos teóricos que serão usados neste trabalho, abrangendo experimentos remotos e treinamentos baseados em computador. A seção subseqüente apresenta uma análise do estado da arte, na qual os principais trabalhos relacionados ao tema desta dissertação são apresentados e comparados. Os capítulos três e quatro descrevem a proposta desenvolvida no âmbito desta dissertação, sendo que no capítulo três é descrita a arquitetura proposta, no capítulo quatro são descritas as implementações realizadas e no capítulo cinco são apresentados resultados (validações) dos estudos de caso. Finalmente discussões de aplicabilidade e conclusões do trabalho são apresentadas no último capítulo.

Um apêndice somente sobre simuladores se encontra no final de forma a apresentar características envolvidas em diversos simuladores citados neste trabalho.

2 CONCEITOS TEÓRICOS

Esta seção visa facilitar a leitura e compreensão da presente dissertação e nela serão apresentados os principais conceitos teóricos utilizados ao longo do texto.

2.1 EXPERIMENTOS REMOTOS

2.1.1 Introdução

Conforme já discutido no capítulo anterior, experimentos remotos podem ser divididos em laboratórios remotos ou laboratórios virtuais (AUER, 2003). Experimentos remotos são acessíveis em servidores que são acessados por clientes através da *Web* (ou Internet) ou via uma rede local (LAN). Experimentos remotos podem ser tanto simulações quanto equipamentos reais, ou ainda experimentos de realidade mista.

A seguir serão descritos os tipos de experimentos referenciados neste trabalho: experimentos simulados (Seção 2.1.2), experimentos reais (Seção 2.1.3) e experimentos de realidade mista (Seção 2.1.4). Nas seções subseqüentes, formas de acesso remoto serão mostradas, assim como arquiteturas (Seção 2.1.5) e diagramas de interação do usuário com equipamentos remotos (Seção 2.1.6). Finalmente, um levantamento de características e comparações entre cada tipo de experimentos serão apresentados e discutidos (Seção 2.1.7).

2.1.2 Simuladores Computacionais

O termo “simulação” pode ser usado com diversos significados, desde encenações no futebol até previsões de catástrofes climáticas. Portanto vale a ressalva que a simulação referida neste trabalho é sempre computacional – segue regras calculadas por computadores – e experimental – baseada em experimentos reais. Em uma simulação experimental um processo físico (ou biológico) é “imitado” por outro processo (HARTMAN, 2005). Este outro

processo que imita o real é chamado geralmente de modelo computacional (no caso da simulação experimental computacional).

Modelos são abstrações da realidade, nos quais informações desnecessárias para a análise em questão são desprezadas e aspectos relevantes são descritos. Modelos descrevem normalmente a estrutura e comportamento de sistemas reais, podendo ser usados em lugar destes para diferentes propósitos. Na indústria, para se diminuir a quantidade de protótipos a serem construídos, simuladores são empregados de forma a diminuir custos e otimizar-se produtos. Na área educacional, são alternativas econômicas em razão dos altos custos de equipamentos reais.

Desta forma, para ser realizada uma simulação, é necessário construir um modelo computacional que corresponda à situação real que se deseja simular como o demonstrado na Figura 1.

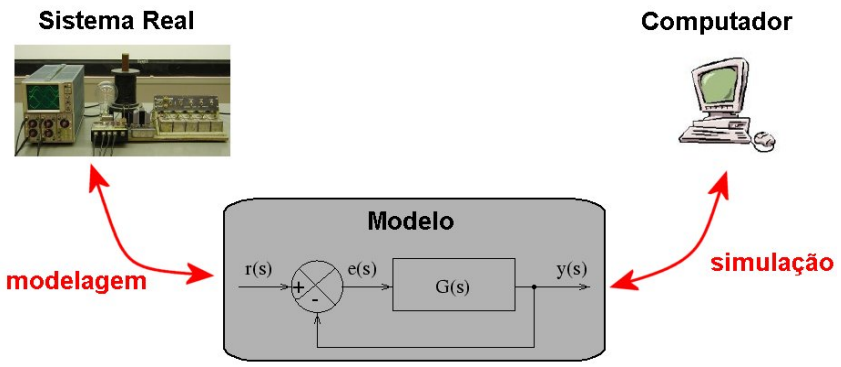

Figura 1. Elementos básicos e relações do processo de modelagem e simulação.

Simulações têm sido objeto de estudo, inclusive de filósofos, desde a Antigüidade até os dias de hoje. Um dos trabalhos filosóficos mais famosos sobre simulações foi escrito pelo sociólogo e filósofo francês Jean Baudrillard intitulado “Simulacra and Simulation” (BAUDRILLLARD, 1994) onde limites da simulação e realidade são discutidos. No mesmo

trabalho, foi também abordada a substituição da realidade por símbolos e sinais, assim criando uma realidade simulada (*simulacra*), mais real que a realidade (*hyperreal*).

Geralmente, simuladores representam sistemas através de modelos matemáticos estáticos ou dinâmicos dependendo do tipo de processo envolvido. Redes neurais e outras formas de representação matemática de sistemas também podem ser utilizados para modelos não convencionais.

2.1.2.1 Características

É importante ressaltar algumas características essenciais dos simuladores ou das simulações de sistemas dinâmicos:

- ✓ Capacidade de manipulação do tempo (ou variável independente):

  Pode-se acelerar ou frear o tempo para visualizarmos os processos e grandezas de interesse da dinâmica do processo de simulação

- ✓ Precisão:

  O número de iterações (*number of iterations*) é um parâmetro que geralmente permite configurar a precisão dos dados calculados (quanto maior o número de interações, maior a precisão). Outro parâmetro é o tipo de dado usado na simulação (booleano, inteiro, real, etc) e o número de casas decimais dos valores numéricos dos parâmetros (que impactam nos erros por arredondamento).

- ✓ Granularidade:

  O tamanho do passo (*step size*) é o parâmetro responsável pela granularidade da base de tempo isto é, a distância entre um ponto ao outro dentro do intervalo de simulação calculado pelo simulador.

- ✓ Limites da simulação:

  De forma a estabelecer limites para o cálculo e execução de simulações, parâmetros comuns são limites mínimos e máximos da variável independente em questão.

- ✓ Testar limites de uso de equipamentos sem a preocupação de danificá-los:

  Equipamentos simulados não podem ser destruídos. Alguns simuladores apresentam mensagens de erro de limites ou possuem animações para representação da destruição do equipamento modelado.

- ✓ Simulação passo a passo (*step by step simulation*):

  Para identificar erros, pode-se executar a simulação passo a passo, isto é, inserindo condições de parada em uma simulação para visualizarmos aquele momento (especialização da manipulação do tempo).

- ✓ Visualização e mudanças de parâmetros normalmente dificultados em experimentos reais:

  Parâmetros que normalmente não são medidos (devido a impossibilidades físicas) em um experimento real podem ser facilmente visualizados em simuladores.

- ✓ Múltiplas instâncias de uma mesma experiência acessíveis simultaneamente:

  Equipamentos virtuais são instanciáveis, isto é, múltiplas cópias podem ser utilizadas proporcionando a utilização simultânea de diversos experimentos simulados sem a preocupação com o número de equipamentos.

Alguns simuladores que estão relacionados ou que fazem parte do objeto de estudo deste trabalho são analisados mais detalhadamente no final deste trabalho no Apêndice – Simuladores.

2.1.3 Experimentos Reais

Experimentos reais se diferenciam de simuladores pela utilização de sistemas (equipamentos) reais no experimento. Vale a ressalva que, nos casos de experimentação remota, embora manipule-se equipamentos reais, os usuários não estão em contato físico com os equipamentos, pois a interação se dá através de uma interface remota. Desta forma, ainda

que simulações estejam em oposição com experimentos reais, simulações fidedignas do comportamento de um determinado sistema podem ser percebidas remotamente como reais.

De maneira a disponibilizar uma interface remota ao usuário, sistemas de aquisição e escrita de dados (interface com o processo) e servidores *Web* são comumente utilizados no servidor da experiência de forma a tornar experimentos reais remotamente acessíveis. Formas e *softwares* semelhantes aos usados nos simuladores são utilizados para disponibilizar dados de equipamentos reais aos usuários remotos em experimentos reais.

Experimentos reais usualmente incluem os seguintes módulos para permitir uma interação com usuários: sistema de aquisição (leitura e escrita) de dados e gerenciamento de experimentos. Estes serão detalhados nas próximas seções.

2.1.3.1 Aquisição de Dados

A aquisição de dados é geralmente feita por um *hardware* que pode medir as grandezas envolvidas na aquisição. Este *hardware* é controlado por um *software* que converterá os dados adquiridos em algum formato que o usuário entenda.

A aquisição pode ser digital ou analógica. Sinais digitais são quantizados (processo de aproximação de valores a limites inteiros) e discretos enquanto sinais analógicos são contínuos e não-quantizados. Os sinais são convertidos de analógicos para digitais e vice-versa por conversores. A conversão de sinais ou aquisição de dados analógicos para digitais não acontece sem perda de qualidade devido à discretização e quantização. Estas características estão presentes em conversores analógico-digitais por meio dos parâmetros: número de bits para a quantização e taxa de amostragem (*sampling rate*) para a discretização.

Computadores lidam com dados digitais (existem também computadores analógicos, mas estes não serão abordados neste trabalho), portanto para que as experiências estejam disponíveis remotamente os sinais analógicos presentes em qualquer experiência devem ser convertidos para digitais. Existem diversos dispositivos (*hardware*) que podem ser usados

para adquirir dados para o computador. Dados digitais podem ser adquiridos (lidos e escritos) até por simples portas paralelas (usadas por impressoras antigas) e seriais (RS-232) quando respeitados os limites do *hardware*, enquanto dados analógicos por placas de som (pela entrada do microfone). Sistemas mais sofisticados de medição utilizam placas de aquisição de dados (DAC) que podem adquirir tanto sinais analógicos como digitais. Instrumentos de medição também podem ser usados para adquirir dados e transmiti-los para o computador por portas paralelas, seriais ou usando protocolos de rede comunicação.

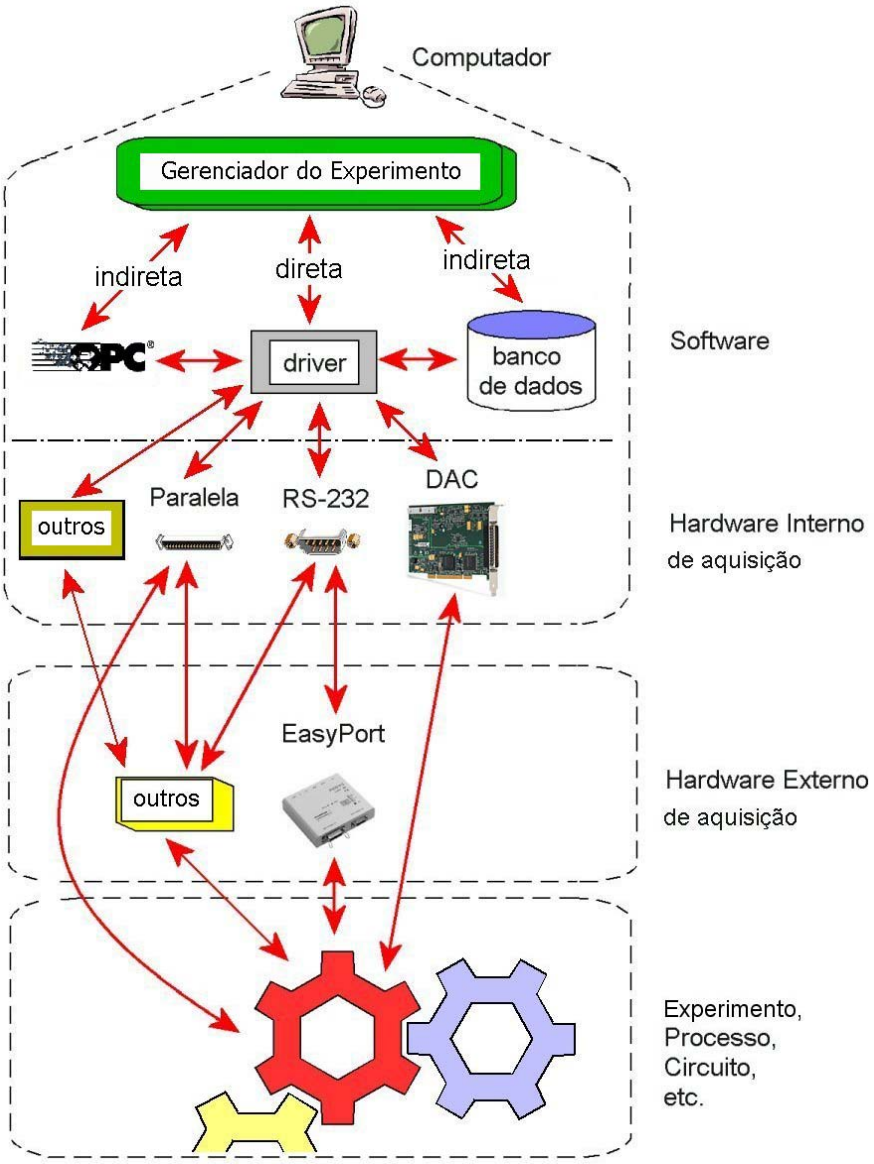

Figura 2. Variedade de sistemas de aquisição de dados

Programas simuladores mais sofisticados geralmente possuem blocos relacionados à aquisição de dados (condicionamento de sinais, etc), como o *LabVIEW* (que tecnicamente não é um simulador) e o *MatLab* (vide Apêndice – Simuladores). Aquisições "indiretas" de dados podem ser feitas utilizando-se protocolos de comunicação com bancos de dados e servidores OPC (*OLE for Process Control* – descrito melhor na Seção 2.1.5.10). Diz-se aquisição indireta quando o *software* responsável pela aquisição não necessariamente disponibiliza os dados adquiridos para um cliente específico (vide Figura 2). Por exemplo, uma placa de aquisição externa disponibiliza os dados medidos por meio do computador em um banco de dados. Este banco de dados pode ser acessado por qualquer cliente. Similar funcionalidade pode ser obtida usando-se um servidor OPC, que disponibiliza os dados de um medidor externo. Importante ressaltar que estes dados adquiridos indiretamente também podem ser virtuais, isto é, gerados por equipamentos que não existem na realidade.

Sistemas de aquisição de dados para controle supervisório (SCADA – *Supervisory Control and Data Acquisition Systems*) usualmente disponibilizam bibliotecas com funções de acesso (*drivers*) para diferentes interfaces de comunicação com diversos protocolos para ler dados de dispositivos de aquisição de dados e bancos de dados, assim como outros.

A aquisição de dados também pode ser configurada de acordo com necessidades específicas da interface, desenvolvendo-se programas (*softwares*) que utilizam *drivers* próprios, para a aplicação em questão.

2.1.3.2 Gerenciamento de Experimentos

Gerenciadores de experimentos são utilizados de forma a organizar os diversos dispositivos reais, presentes na experiência, e conectá-los (vide Figura 2) de forma a criar um experimento que pode ser manipulado via gerenciador. Os gerenciadores de experimentos também incluem interfaces gráficas com usuários, *GUI*s, que proporcionam aos usuários remotos a visualização e o controle do experimento através da Internet. Muitas vezes, esse

papel é executado por sistemas SCADA acessíveis remotamente. Assim como a aquisição de dados, os simuladores, muitas vezes, proporcionam maneiras de gerenciar experimentos remotos reais como o caso dos *softwares*: *ISaGRAF* com o *HiBeam*, *MatLab* com o *MWS*, *LabVIEW* com o *LabVIEW Internet*, e o *Elipse SCADA* com o *Elipse Web* (estes sistemas são apresentados em maiores detalhes no Apêndice – Simuladores).

A escolha entre sistemas de gerenciamento de experimentos muitas vezes é decidida pelo tipo de experiência e forma de apresentação dos dados. Programas como o LabVIEW e o MatLab possuem ferramentas "nativas" para o processamento matemático enquanto o ISaGRAF e o Elipse SCADA são mais voltados para sistemas de automação industrial sem grande processamento matemático. Preocupações com segurança do experimento também são cruciais na decisão do sistema a ser utilizado.

Obviamente, sistemas específicos para gerenciamento de experiências podem ser desenvolvidos usando-se linguagens de programação como C/C++ (SHEN, 1999), Java (JDDAC, 2006), Python (CHICULITA, 2002), etc.

2.1.3.3 Tipos de Experimentos Remotos em Laboratórios Reais

Existem diversos tipos de experimentos remotos, mas pode-se classificá-los em dois grandes grupos:

i. Experimentos que não necessitam transmissão de vídeo nem de áudio:

   Esses laboratórios não necessitam a transmissão de vídeo, pois o experimento em questão não irá produzir nenhuma alteração física (real) visual ou audível.

   Exemplo: Programação de microprocessadores (FERREIRA, 2002).

ii. Experimentos que possibilitam a visualização do experimento através da transmissão de vídeo/áudio:

   Apresentam alguma modificação visual física perceptível na experiência que é de grande importância para a realidade da experiência (ALBU, 2004).

> Geralmente, as transmissões de vídeo são de baixa qualidade produzidas por WebCams, isto se deve a falta de garantias de qualidade de serviço (*QoS*) e heterogeneidade das redes que dificultam a transmissão de vídeo de forma eficiente e de maneira flexível (WU, 2001).
>
> Exemplos: Experimentos de controle de nível de tanques (CASINI, 2003; ZEILMAN, 2002, DUAN, 2003); Experimentos de eletro-pneumática utilizados para comprovação de teorias de *hyper-bonds* em ambientes de realidade mista (BRUNS, 2004); etc.
>
> A transmissão de áudio é mais rara, mas ainda encontramos experimentos remotos que possibilitam mais esta característica. Exemplo: Instrumentação de óptica laser programada em Java (FALTIN, 2002); Motor para aplicação na tecnologia de informação (KIKUCHI, 1999) e planta de processos químicos (SRINAVASAGUPTA, 2003).

Para a transmissão de vídeo, algumas considerações devem ser feitas com relação à forma de transmissão e velocidade de conexão (COOPER, 2002). Existem várias maneiras de encapsular o vídeo para a transmissão na Internet. O *NetMeeting* da *Microsoft* lida com áudio e vídeo para conferências, e pode ser adaptado para transmitir áudio e vídeo de uma experiência remota real (SRINAVASAGUPTA, 2003). Outras maneiras mais portáveis utilizam *Java Media Framework* (JMF) para a transmissão de imagens (COOPER, 2002).

Gravar vídeos da experiência para a utilização a posteriori é opção bastante interessante que ajuda na visualização e identificação de processos lentos que não precisam de acompanhamento em tempo real (ZHANG, 2004).

As câmeras também podem ser adaptadas em bases controláveis pelo usuário (KO, 2001), fazendo com que ela se mova em qualquer direção, para melhor visualização de qualquer parte do experimento, assim como aproximações e panoramas.

2.1.3.4 Aspectos Importantes

Um conceito interessante envolvido na utilização dos experimentos remotos reais é o que chama-se de “segunda melhor alternativa de estar lá” (do inglês, *Second Best of Being There* – FALTIN, 2002; ATKAN, 1996). Esse conceito foi criado para retratar o esforço de desenvolvedores em proporcionar aos usuários remotos uma experiência o mais próxima possível de uma experimentação presencial na qual o aluno encontra-se no mesmo local dos equipamentos utilizados.

Diferentemente de experiências simuladas, experiências remotas realizadas em equipamentos reais apresentam restrições quanto ao acesso concorrente por mais de um usuário, uma vez que as simulações podem ser replicadas a um baixo custo, enquanto que a replicação de equipamentos reais implica em altos investimentos. Ou seja, apesar da experimentação remota disponibilizar os experimentos dos laboratórios reais a um número maior de alunos, equipamentos reais podem ser utilizados por apenas um aluno (cliente) ou grupo de alunos ao mesmo tempo. É importante ressaltar que esta restrição é quanto à escrita, isto é, a visualização (leitura de dados) do experimento não é restringida e sim a manipulação do experimento (escrita de dados). Do contrário, experiências de grupos diferentes interfeririam em seus experimentos, fazendo com que houvesse uma competição entre os usuários, quando o objetivo é a colaboração. Ferramentas, chamadas de sistemas de reserva (*booking systems*), são utilizadas de forma a organizar o acesso aos experimentos remotos (FERREIRA, 2002). O tipo mais comum de sistemas de reserva obedece a regra de *first-come first-served* (DUAN, 2003), isto é, o primeiro usuário a reservar um horário tem o direito de escolha entre todos os horários disponíveis, enquanto que os outros têm de escolher um horário não reservado. Sistemas mais complexos, como o da Universidade de Tóquio (MITSUI, 2004), procuram escalonar o acesso de experiências e materiais educacionais através de aproximações matemáticas que levam em conta a seqüência das aulas e a ordem

com que os recursos são utilizados. O horário reservado é chamado de *time slot*, representando a duração máxima permitida para um experimento antes que esse seja encerrado. A duração do *time slot* depende do tipo de experiência envolvida. As reservas são necessárias para limitar também o tempo de uso de um aluno/grupo, caso contrário o acesso a equipamentos da experiência poderia ficar bloqueado por tempo indeterminado. Geralmente, o *software* que realiza o controle de acesso pode ser alterado para proporcionar um sistema de reservas.

Aspectos de segurança são essenciais para garantir a integridade dos equipamentos envolvidos no experimento. Muitas vezes, o acesso aos equipamentos é restrito a usuários cadastrados por um sistema computacional que controla e registra todos os acessos em uma base de dados. Limites e alarmes são empregados para variáveis manipuladas no experimento remoto para evitar má utilização e prevenir situações inseguras.

Sistemas de operação remotos, assim como experiências remotas, utilizam-se das mesmas tecnologias. Portanto os experimentos remotos podem ser usados na indústria como base para a operação remota.

2.1.4 Experimentos de Realidade Mista (“Mixed-Reality”)

Experimentos de realidade mista, como o próprio nome sugere, são experimentos que incluem tanto objetos reais quanto virtuais. Esse tipo de experimento foi criado com o intuito de proporcionar vantagens oferecidas pela combinação entre os dois tipos de experimentos.

Na seqüência alguns importante conceitos de sistemas de realidade mista são discutidos, a fim de facilitar a compreensão pelos leitores dos capítulos subsequentes.

2.1.4.1 O que é realidade mista?

Realidade mista foi definida por Paul Milgram (MILGRAM, 1994) como “a combinação de mundos reais e virtuais em algum ponto da linha da virtualidade (*virtuality continuum*) que conecta completamente ambientes reais com virtuais”. O conceito de

realidade mista engloba: ambientes virtuais (*virtual environment*), virtualidade aumentada (*augmented virtuality*), realidade aumentada (*augmented reality*) e ambientes reais. Isso indica que a realidade mista pode ser considerada como uma especialização de realidade aumentada e mídia interativa (*interactive media*). Segundo Billinghurst e Kato (BILLINGHURST, 1999), realidade mista é comumente entendida como uma integração transparente ao usuário (*seamless*) entre os mundos real e virtual.

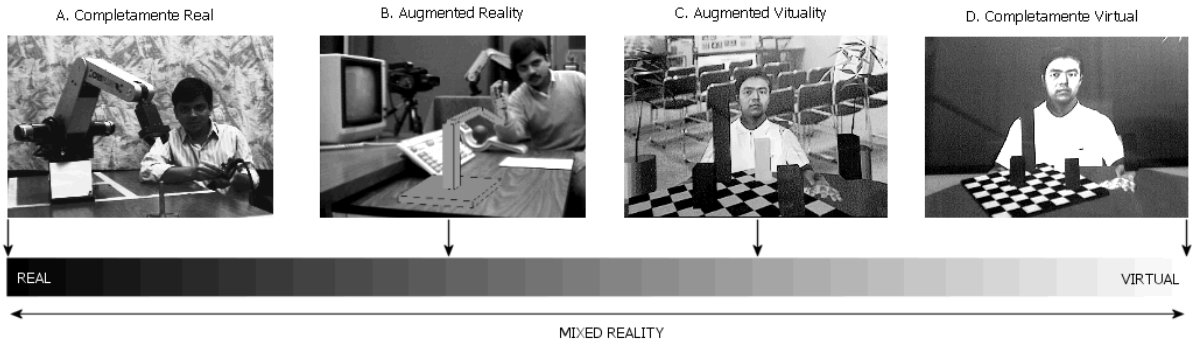

Figura 3. Linha da Virtualidade (MILGRAM, 1995).

A Figura 3 apresenta quatro situações em que o conteúdo real e virtual varia desde um sistema completamente real até um sistema completamente virtual. Na foto mais à esquerda tem-se um operador (real) interagindo com um robô real, na seqüência um sistema no qual o robô real é substituído por um sistema virtual (por exemplo, tal situação poderia ser utilizada em treinamentos de operadores/programadores de robôs, sem existir risco de danificar os equipamentos). Na terceira figura um cenário real (com sala e cadeiras e corpo do estudante reais) com um tabuleiro de xadrez virtual assim como o braço do manipulador, e por último uma situação completamente virtual.

É crescente o número de pesquisas e experimentos de realidade mista que são encontrados na literatura, em aplicações nas áreas de educação (BRUNS, 2004), entretenimento (jogos eletrônicos, filmes, etc.) (FAUST, 2006), indústria (projetos

arquitetônicos, simuladores, etc.) (SAT, 2006), cultural (turismo) (KRETSCHMER, 2001), e médica (BOCKHOLT, 2003), entre outras.

A implementação de um sistema de realidade mista se dá da mesma forma que os experimentos reais remotos com funcionalidades virtuais (por exemplo medidores virtuais). Uma vez que experimentos de realidade mista exigem a integração entre dispositivos reais e virtuais, propostas de integração entre sinais físicos reais e sinais "virtuais" (isto é, informação) são necessárias. Uma proposta de interface para integração entre sistemas reais e virtuais é o conceito de *hyper-bonds*.

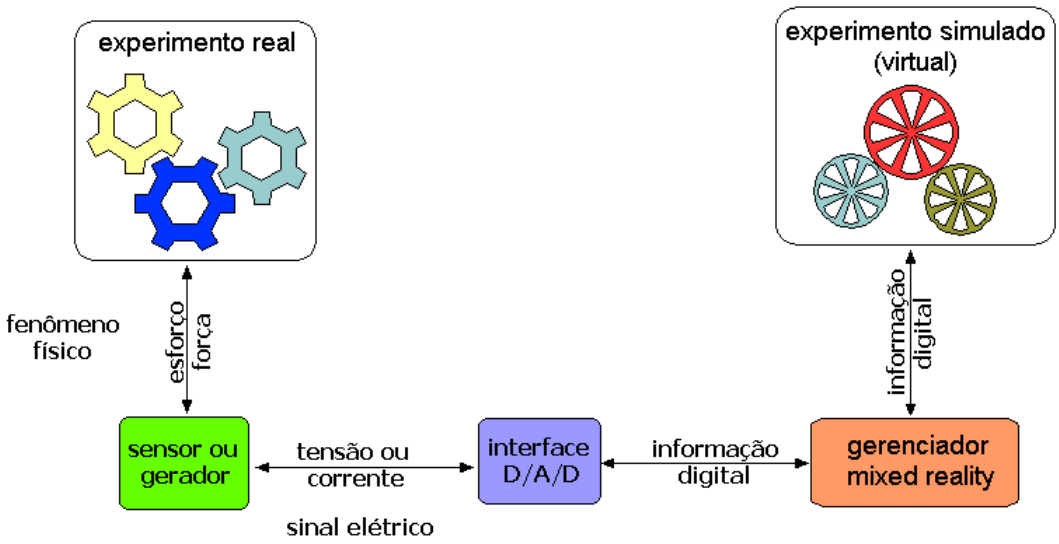

Figura 4. Experiência de realidade mista.

Os *hyper-bonds* combinam representações abstratas de sistemas unificados, usando gráficos de *Bond* [1], com a implementação do conceito de *hyper-connection* (superconexões). Essas conexões unem fenômenos físicos de representações externas ao ambiente computacional com estruturas lógicas de representações internas ao ambiente computacional, isto é, uma combinação de sistemas físicos com seus componentes virtuais (BRUNS, 2004).

[1] proposto por Paynter em 1961 e reformulado por Karnopp em 1990 (KARNOPP, 1990).

Em outras palavras, esta ferramenta possibilita sinais físicos (fenômenos reais) interagirem com sinais digitais (ambiente virtual) (MÜLLER, 2006).

Pode-se dividir o hyper-bond em duas partes: a parte de aquisição de dados (*hardware* – transdutor e condicionador de sinais), e a parte da comunicação com o gerenciador de experiência (*software*). Quando a aquisição de dados é indireta (vide Figura 2), dados de experimentos reais se confundem com dados de experimentos virtuais. Desta maneira o *hyper-bond* é transparente para o usuário que não diferencia variáveis reais de virtuais.

2.1.4.2 Vantagens

Como dito anteriormente, experimentos de realidade mista combinam vantagens de simuladores com experimentos reais. Vantagens clássicas dos simuladores (vide Apêndice – Simuladores), como o controle do tempo e execução passo a passo, são muito boas para os usuários dos sistemas (alunos). O realismo da experiência física também é apreciado (SBBT). Partes simuladas podem oferecer as mesmas vantagens dos simuladores com a capacidade de instanciar diversos modelos simultaneamente (SCHAF, 2006b).

2.1.5 Arquiteturas e Formas de Acesso

Existem duas arquiteturas básicas de servidores de experiências remotas: centralizadas e distribuídas. Nas arquiteturas centralizadas, os recursos computacionais para execução e interface dos experimentos encontram-se em um único computador que gerencia toda a experiência. Enquanto isso nas arquiteturas distribuídas, diversos computadores são responsáveis por gerenciar a experiência, cada computador pode estar conectado a diferentes equipamentos da experiência. As arquiteturas distribuídas podem ser constituídas de computadores que estão em instituições diferentes ligadas pela Internet. Deste modo um grupo de instituições pode disponibilizar uma experiência que utiliza recursos de todas.

As formas básicas de acesso podem ser divididas em dois grandes grupos: *thin-* e *fat-client*. Chamam-se clientes “magros” (*thin-client*) aqueles que exigem apenas um software de

navegação de Internet no computador a partir do qual se pretende acessar a interface da experiência remota. Exatamente o oposto ocorre com clientes “gordos” (*fat-client*) onde o usuário remoto necessita ter instalado em seu computador (cliente) softwares adicionais para poder interagir/visualizar a interface remota da experiência.

O conceito de *thin-client computing* (NIEH, 2000) foi criado para possibilitar que mais usuários remotos acessem recursos de um experimento remoto. Nele, clientes com poucos recursos de memória de processamento como *PDAs* e computadores industriais podem acessar recursos poderosos de controle e até de visualização que os servidores das experiências proporcionam.

A seguir algumas tecnologias envolvidas com as formas de acesso de clientes remotos serão apresentadas de forma a proporcionar maior conhecimento para futuras referências no texto.

2.1.5.1 HTML

A linguagem HTML é uma linguagem de hypertexto utilizada em páginas da *Web* para codificação de hypertextos (textos e gráficos estáticos) que serão decodificados no navegador (*Browser*). Hoje em dia, HTML é uma padronização ISO/IEC 15445:2000 especificada e mantida pelo consórcio da WWW (W3C).

Utilizada vastamente desde 1991 para representar hypertextos, essa linguagem não possui recursos de variáveis e processos. Todos os navegadores possuem interpretadores de HTML que são responsáveis por decodificar o código e mostrar o hypertexto da forma como foi escrito.

2.1.5.2 CGI

O CGI permite requisitar dados de um programa executado no servidor (*server-side*) pelo cliente através do navegador. Foi criado inicialmente por pesquisadores da NCSA (National Center for Supercomputing Applications), em 1993, usando variáveis de ambiente

no servidor para passar parâmetros, anterior ao desenvolvimento do processo separado chamado programa *CGI*.

Não é uma linguagem de programação, mas sim uma interface que facilita o uso de aplicações na *Web* em qualquer linguagem de programação com suporte de leitura de variáveis, processamento de dados e retornos de respostas, sendo independente de plataforma. A interface *CGI* pode ser descrita em diversas linguagens de programação de acordo com o sistema operacional usado no servidor. Por exemplo, nos sistemas operacionais Linux, usa-se *C/C++*, *Perl*, *Phyton*, *Java*, *shell* e outros, enquanto nos sistemas operacionais Windows usa-se *C/C++*, *Java*, *Visual Basic*, *Visual C/C++*, etc.

O CGI é uma interface definida de maneira a possibilitar a execução de programas (*gateways*) sob um servidor de informações. Desta forma, *gateways* são programas ou *scripts* (também chamados *cgi-bin*) que recebem requisições de informação, retornando um documento com os resultados correspondentes. Esse documento pode existir previamente, ou pode ser gerado pelo *script* especialmente para atender à requisição em questão.

Um programa CGI é um arquivo executável (compilado), executado pelo servidor, que pode agir como mediador entre o servidor HTTP e outros programas. Por ser executado no servidor *CGI*, é, muitas vezes, pouco seguro (ALVARÉZ, 2003) intrinsecamente. Medidas de segurança devem ser tomadas no desenvolvimento do script para remediar o problema.

Programas CGI estão sendo rapidamente substituídos por programas escritos em PHP (vide Seção 2.1.5.4).

2.1.5.3 JavaScript, JScript, Java Applets e Java Servlets

2.1.5.3.1 JavaScript

JavaScript é o nome dado à implementação do *Netscape* ao *ECMAScript* (Computer Manufacturers Association) baseado no conceito de protótipos (*prototypes*). A principal utilização desta linguagem de programação de scripts é em páginas da Web, mas também é

usada para possibilitar acesso por meio de scripts em objetos embarcados de outras aplicações. Essa linguagem é marca registrada da *Sun Microsystems* usada sobre licença de tecnologia inventada e implementada pela *Netscape*.

Apesar do nome, *JavaScript* é pouco relacionado à linguagem de programação *Java*, sendo a similaridade maior com o *C*. O JavaScript tem muito em comum com a linguagem de programação Self, que é orientada a objetos e baseada no conceito de protótipos. Diferentemente do *C*, o *JavaScript* não depende de bibliotecas de I/O e sim da máquina JavaScript do ambiente hospedeiro no qual está embarcado. A sua maior utilização está em funções escritas em páginas HTML para interagir com DOM (*Documento Object Model*) da página, funções estas que o HTML não desenvolve. Uma peculiaridade dos códigos em JavaScript é que nem sempre máquinas diferentes como o *Gecko* (*Netscape* e agora *Mozzila*) e o *Trident* (*Microsoft Internet Explorer*) possuem a mesma interface DOM, fazendo com o que o código funcione de maneiras diferentes.

2.1.5.3.2 JScript

O JScript é a implementação do *ECMAScript* desenvolvida pela *Microsoft* em 1996. A funcionalidade é exatamente a mesma do JavaScript. JScripts são escritos para serem usados na máquina *Trident* do Microsoft Internet Explorer.

2.1.5.3.3 Java Applets

Java Applets são aplicativos escritos em linguagem de programação Java, que são executados no contexto de outras aplicações. Por sua vez, estes aplicativos podem ser executados dentro de navegadores utilizando uma máquina virtual *Java* (JVM). Foi criado em 1995 pela *Sun* para proporcionar habilidades interativas para aplicações *Web* que não podem ser proporcionadas por simples *HTML*s. Por motivos de segurança estes Applets são executados em áreas restritas de memória chamadas de *sandboxes*. Java *bytecodes* são independentes de plataforma, portanto *Java Applets* podem ser executados em qualquer

navegador com suporte *Java* e em diversos sistemas operacionais como Windows, Linux, Unix e MacOS. Um episódio marcante na história pretende acabar com problemas de compatibilidade causados pelo desenvolvimento de duas máquinas virtuais Java: uma da *Sun* e outra da *Microsoft*. Neste episódio o desenvolvimento da máquina virtual Java da Microsoft foi suspenso e o navegador da Microsoft hoje em dia utiliza a máquina virtual Java da Sun.

Interessante notar que os Java bytecodes são “baixados” (*download*) para o cliente que executa a aplicação. A execução de partes de um mesmo código no servidor é possível através de um recurso bastante comum do Java, a invocação de métodos remotos (*RMI*), que é bastante segura (SOBH, 2002). Qualquer funcionalidade da linguagem orientada a objetos *Java* também pode ser utilizada nos Applets como: comunicação com *I/O*s, utilização de *sockets*, assim como funcionalidades de tempo real, etc.

2.1.5.3.4 Java Servlets

A interface de programação de aplicações (API) Java Servlets da Sun proporciona a geração dinâmica de conteúdos em um navegador da *Web* utilizando a plataforma Java. Possuem a mesma funcionalidade que tecnologias como o PHP, o ASP e o CGI.

2.1.5.4 PHP

O PHP é uma linguagem de programação de *scripts* usada para criação dinâmica de páginas da *Web* e aplicações *server-side*.

Amplamente utilizada por se tratar de uma linguagem reflectiva e de código aberto (*open source*), além de interagir com um grande número de sistemas de gerenciamento de bancos de dados relacionais como o *mySQL*, *Oracle*, *Microsoft SQL Server*, *PostgreSQL* e *SQLite* (PHP, 2006). O PHP pode ser utilizado em diversos sistemas operacionais incluindo Unix, Linux, Windows e MacOS interagindo com diversos servidores *Web*.

Criado em 1994 a partir de *scripts* em Perl e, mais tarde, aprimorado para códigos binários escritos em *C* para *CGI*, o PHP era inicialmente um conjunto de ferramentas de

páginas pessoais (**P**ersonal **H**ome **P**age Tools) para monitoramento de tráfego. Em 1998 o núcleo do PHP (*PHP Core*) foi reescrito para ser interpretado por uma máquina de *scripts* chamada de *Zend Engine*, criando assim a versão conhecida e utilizada mundialmente.

A atual versão é *Zend Engine II*, responsável pela interpretação de códigos PHP versão 5. O código fonte da máquina de scripts está livremente disponível desde 2001 sob licenças do tipo BSD (*Berkeley Software Distribution*).

O PHP pode ser visto como alternativa entre as linguagens de programação *ASP* (Microsoft), *Java* (Sun), *Zope/Python*, *ColdFusion* (Macromedia), *Perl* e, mais recentemente, do *Ruby on Rails*.

2.1.5.5 ASP

O ASP é uma tecnologia de geração dinâmica de páginas da Web no lado do servidor criada pela Microsoft e vendida como um recurso adicional (*add-on*) do IIS (*Internet Information Services*), que é o *software* servidor de conexões *HTTP* da *Microsoft*.

Páginas ASP são geralmente escritas em *Visual Basic Script* e outras linguagens que têm como objetivo a criação dinâmica de páginas da Web, como por exemplo *HTML*s. O ASP é composto de seis objetos separados chamados de *aplicação*, *ASPError*, *requisição*, *resposta*, *servidor* e *sessão*. Sua programação é bastante simples, pois utiliza objetos embutidos (*built-in*).

As funcionalidades são as mesmas de qualquer linguagem de script rodando no servidor, como o *CGI*, *JavaScript* e *PHP*.

2.1.5.6 MySQL

MySQL é um sistema gerenciador de bancos de dados (DBMS) *SQL* multiusuários e multi-tarefas (*multithreaded*). O *SQL* é uma linguagem computacional usada para criar, modificar e adquirir dados de sistemas de gerenciamento de bancos de dados relacionais. Esta linguagem é hoje uma padronização da ANSI/ISO desde 1987.

Através de associações com programas *PHP*, *CGI*, *Java* e outros, bancos de dados gerenciados pelo mySQL tornam-se acessíveis remotamente. Associações do sistema operacional Windows, Apache, MySQL e PHP chama-se de *WAMP* enquanto *LAMP* a associação dos mesmos programas utilizando o Linux. Ambas as associações, *LAMP* e *WAMP*, são muito utilizadas em aplicações da *Web* que envolvem bancos de dados.

A chave para o sucesso deste *software* está na disponibilidade (existem versões livres - *freeware*) e na portabilidade (pode ser usado em uma variedade muito grande de sistemas operacionais).

2.1.5.7 ODBC

Semelhantemente ao MySQL, o ODBC é uma interface com um sistema gerenciador de bancos de dados (DBMS). Usa o SQL como sua linguagem de acesso ao banco de dados para proporcionar uma interface de programação de aplicativos (API). Os aplicativos de banco de dados chamam funções na interface ODBC, que são implementadas nos módulos específicos de banco de dados denominados *drivers*. O uso de *drivers* isola os aplicativos das chamadas específicas de banco de dados. Como os *drivers* são carregados em tempo de execução, o usuário só tem que adicionar um novo *driver* para acessar um novo tipo de banco de dados; não é necessário recompilar ou fazer um novo *driver* do aplicativo para o novo banco de dados, mesmo se seu tipo for alterado.

2.1.5.8 XML

A linguagem estendida XML é recomendada pelo W3C como linguagem de propósitos generalizados, isto é, para qualquer propósito. Sua meta principal é o compartilhamento de dados entre sistemas, através da Internet, particularmente entre sistemas diferentes. É uma simplificação da linguagem SGML (*Standard Generalized Markup Language*).

Existem diversas linguagens baseadas no XML, como o XHTML, GML, RSS, e muitas outras são definidas de modo formal, possibilitando aos programas modificações e validações de documentos sem o conhecimento prévio de suas formas.

O XML é um texto que descreve e aplica estruturas de árvores a informações. Os tipos de caracteres utilizados em um documento são especificados para que cada sistema entenda o tipo de informação e a decodifique de forma correta.

2.1.5.9 OPC

OPC é uma padronização de interface criada em 1996 para sistemas de automação industrial com o intuito de controlar diversos dispositivos de diferentes fabricantes. A padronização original é baseada em tecnologias *OLE COM* e *DCOM* desenvolvidas pela *Microsoft* para sistemas operacionais *Windows*. Hoje em dia, essa padronização é mantida pela *OPC Foundation* (OPC, 2006) e foi renomeada para padronização OPC-DA (OPC de Acesso a Dados).

Foi desenvolvido para ligar aplicações do *Windows* com *software* e *hardware* de sistemas computacionais usados em aplicações de automação industrial e controle de processos. O OPC define métodos consistentes para se acessar dados de dispositivos de chão de fábrica, sendo que o método utilizado é o mesmo, independente do tipo e da fonte dos dados.

Servidores OPC, portanto proporcionam métodos para diferentes *softwares* (clientes OPC) acessarem dados de dispositivos de controle como *CLP*s e *DCS*s. Uma vez que um servidor OPC de dispositivo de controle é criado, não se necessita reescrever *softwares* com drivers que gerenciem este dispositivo, pois isso pode ser feito utilizando-se um cliente OPC conectado ao servidor do dispositivo de controle (vide Figura 5). Diversas aplicações diferentes, como *SCADA*s, *HMI*s, etc, podem se conectar ao servidor do dispositivo sem necessidade de reprogramação do mesmo.

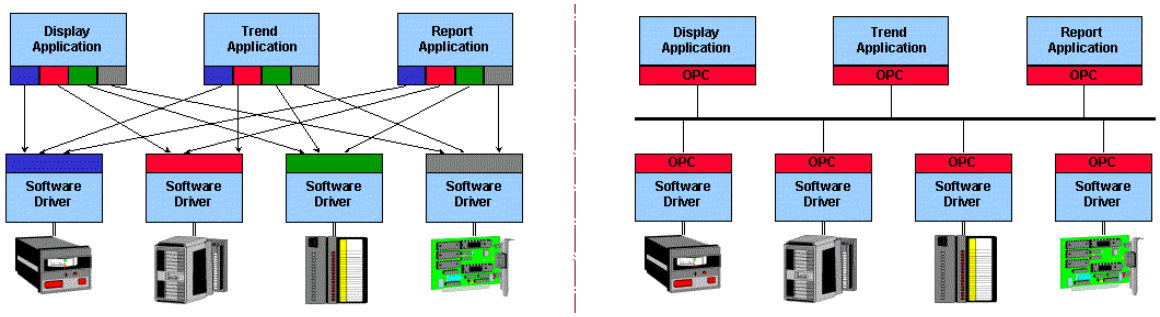

Figura 5. Comunicação OPC (ZHENG, 2002).

Além da simplicidade na conexão, dados disponibilizados pelo servidor OPC possuem parâmetros intrínsecos associados à segurança, tempo (*time stamping*), precisão, etc.

2.1.6 Características e Comparações

Os experimentos remotos, de todos os tipos discutidos anteriormente, são dotados de características marcantes por proporcionarem:

- ✓ maior disponibilidade;

  Experimentos podem ser acessados 24 horas por dia, não importando dia da semana, nem feriados.

- ✓ maior robustez:

  Se corretamente projetados, isto é, com sistema de segurança tais como sistemas de intertravamento, minimiza-se a possibilidade de uso incorreto dos dispositivos que levem a danificá-los.

- ✓ custo de utilização muito baixo.

  Se a experiência for de outra instituição, o custo se deve somente à conexão (caso concordância da instituição). Caso contrário, além do custo da conexão, custo de manutenção (eventual reposição de material) e gastos de energia. Sistemas de reservas dos usuários remotos podem miniminizar o consumo de energia dos experimentos.

- ✓ professor não precisa acompanhar a experiência.

  Acompanhamentos das experiências podem ser agendados com professores, caso haja necessidade.

Se formos comparar todas as vantagens oferecidas por cada tipo de experiência, iremos notar que experimentos de realidade mista agrupam essas vantagens. Pode-se projetar um experimento remoto do tipo *mixed-reality* de modo a englobar todas as vantagens proporcionadas por suas partes reais e virtuais (vide Tabela 3). Desta forma, experimentos remotos que usem conceitos de realidade mista apresentam-se como uma alternativa muito interessante, apresentando excelente relação de custo/benefício.

Tabela 3. Experimentos reais vs. Virtuais

	Vantagens	**Desvantagens**
Simulações (Experiências Virtuais)	✓ visualização de todas as variáveis envolvidas no modelo ✓ instanciação de equipamentos virtuais (uso de múltiplas cópias idênticas simultâneamente) ✓ execução passo a passo ✓ limites dos equipamentos são flexíveis ✓ ausência de manutenção ✓ geralmente de custo baixo	✓ didática ✓ restições dos modelos ✓ falta de perturbações ✓ funcionamento dos modelos é geralmente ideal ✓ perda de ligação com a realidade
Experimentos Reais	✓ emprego de equipamentos usados na indústria ✓ didática associada a equipamentos reais (*hands on*) ✓ dinâmica real do processo com eventos não lineares	✓ equipamento só pode ser usado por um aluno/grupo de cada vez ✓ manutenção freqüente ✓ custo alto

Uma arquitetura flexível pode configurar que partes devem ser simuladas e que partes devem ser reais, contemplando grande parte dos objetivos do experimento e das necessidades de aprendizagem através da experimentação.

É importante notar que uma experiência remota não terá a flexibilidade de manuseio de um experimento local (real) com relação aos equipamentos físicos, isto é, não pode-se modificar as ligações entre equipamentos em uma experiência remota, embora alguns melhoramentos, que serão discutidos nas próximas seções, possam amenizar esta desvantagem.

2.2 TREINAMENTOS BASEADOS EM COMPUTADOR (CBT)

Como foi discutido anteriormente, apesar da experimentação ser um ótimo recurso educacional, a disponibilização apenas do experimento, sem um material educacional que explique os conceitos a serem demonstrados, limita as possibilidades de aprendizagem pelos alunos. Desta forma, a integração da experiência executada pelos alunos em um ambiente computacional de aprendizado (*computer aided learning environment*) é essencial. Uma ferramenta chamada de mecanismo de aprendizagem (*learn engine*) associada ao experimento remoto, faria com que alunos utilizassem recursos da prática (*hands on*), em conjunto com o suporte teórico dos materiais educacionais.

Cooper em seu trabalho (COOPER, 2000) comprovou estatisticamente que conhecimentos prévios de experiências por alunos provêm na maioria de pesquisas na Internet, isto é, aprendizado baseado em computadores (*Computer Based Learning* – CBL). O trabalho de Poindexer (1999) cita que maneiras de tornar a Internet mais atraente para ambientes de aprendizado são feitas desde 1990.

O aprendizado do estudante é um processo ativo que depende de interação. Laurillard propõe um modelo de interação aluno/tutor/curso que facilita a análise de como a tecnologia educacional pode ser aplicada para suportar as diferentes interações que fazem parte do processo de aprendizado. Isso proporciona uma estrutura modelar para planejar soluções tecnológicas para o fornecimento de trabalhos práticos. As divisões de interações (ou conversações) são: 1. a transferência de conhecimento e discussão; 2. reflexão; 3. montagem

do experimento, e subseqüente adaptação; e 4. interações no nível de experiências reais (LAURILLARD, 1993).

Existem vários tipos de mecanismos de aprendizado, como apostilas, tutoriais, listas de exercícios, exercícios on-line, rastreadores de documentos acessados, etc. O mecanismo de aprendizado pode funcionar como uma aula teórica da disciplina em questão. O avaliador automático é a maneira como o mecanismo de aprendizagem classifica o aprendizado do aluno.

Experiências de disciplinas com experimentação remota também provam que o trabalho em grupo dos alunos beneficia a "fixação" do conhecimento. Para proporcionar esse ambiente de trabalho em grupo, alguns projetos desenvolveram interfaces de videoconferência para serem utilizados pelos alunos, proporcionando aprendizado colaborativo de suporte computacional (*Computer Support Collaborative Learning* – CSBL; FERRERIRA, 2004). O termo "aprendizagem colaborativa distribuída" (*distributed collaborative learning*) exemplifica o objetivo dessa interface, isto é, alunos devem estudar de maneira colaborativa, trocando informações entre si e auxiliando-se mutuamente no processo de aprendizagem.

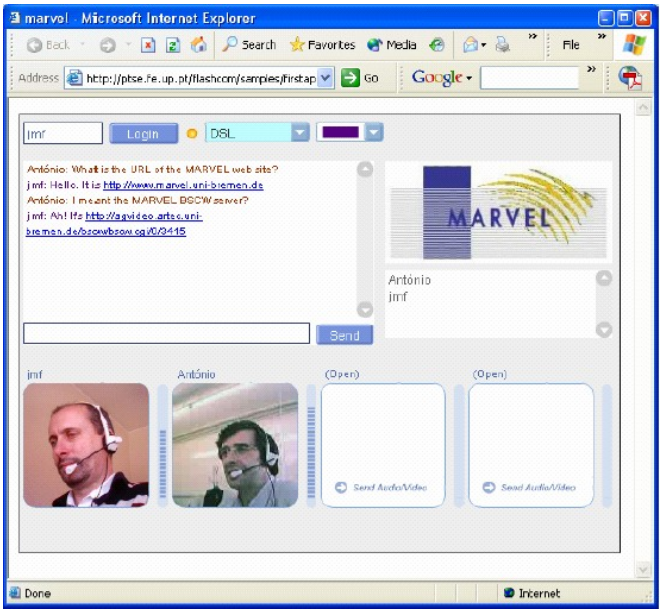

Figura 6. Navegador com interface de vídeo conferência.

Na Figura 6 ilustra-se um exemplo de uma interface de videoconferência integrada no navegador da Internet. *Softwares* que não necessitam supervisão de professores também podem ser utilizados, por exemplo: Microsoft MSN Messenger e NetMeeting. Existem também *softwares* livres de transmissão de áudio e vídeo de maior qualidade, como o Skype.

2.2.1 Contextualização Histórica

O projeto Telelab da Universidade do Leste da Austrália, lançado em 1993, foi um dos primeiros laboratórios (virtuais) com preocupação educacional até seu término em 1996 (TELELAB, 2005). Nesse laboratório, alunos tinham acesso, dentro da instituição, a simuladores desenvolvidos na própria instituição, assim como acompanhamento de um tutorial para guiar no processo de aprendizagem.

Diferente do Telelab, o projeto Learn-ED (*Learning and Educational Access using Remote Networks – Enabling the Disabled*) da Comunidade Européia, criado em 1995, tinha apoio tanto de instituições de ensino européias, quanto da indústria, para proporcionar ambientes que ajudassem pessoas com necessidades especiais a aprenderem utilizando o ensino remoto (HINE, 2005).

A partir desses projetos, vários consórcios entre instituições de ensino se sucederam, dando origem a diversos projetos com parcerias na indústria. O consórcio ASTEP (*Advanced Software for Teaching and Evaluation Processes*), criado pela Comissão Européia em 1998 (FOULK, 1998), está voltado para o desenvolvimento de *softwares* de ensino e processos de avaliação automática em todas as áreas de ensino. Já o ALLEGRO (*a Lifelong Education and Training Environment for Microelectronic*) da Comissão Européia (Projeto da Vinci), criado em 1999, é específico para ambientes de educação e treinamento em microeletrônica (ALLEGRO, 2005).

Uma proposta de uma rede global de ensino foi implementada no Projeto WGLN (*Wallenberg Global Learning Network*). Esse projeto ainda está em andamento como WGLN

II, ajudando principalmente estudantes de graduação nas universidades (Universiade de Stanford – EUA; Uppsala Universtiy, Royal Institute of Tecnology, Karolinksa Institute – Suécia; Universidade de Hannover, Universidade de Braunschweig e Escola de Artes de Braunschweig - Alemanha) (WGLN, 2005).

Voltado para a área de reabilitação de pessoas portadoras de necessidades especiais, o Projeto RESORT (*Remote Service of Rehabilitation Technology*) da União Européia, criado em 1998, que desenvolve, através de auxílios remotos via computadores, maneiras de tornar a vida das pessoas com necessidades especiais melhor (ZAGLER, 1997).

Cursos da Internet, não ligados a nenhuma instituição de ensino, apresentam apostilas e tutoriais em forma de documentos para os alunos, não se preocupando com o ambiente de estudo (tanto físico como virtual). Outros cursos, que geralmente são ligados a instituições de ensino, utilizam uma plataforma virtual de aprendizagem (*VLE*'s) que é responsável pelo direcionamento dos materiais educacionais.

De maneira a padronizar estes materiais educacionais, o Departamento de Defesa dos EUA criou em 1999 o modelo de referência de objetos de conteúdos compartilhados (*Sharable Content Object Reference Model* – SCORM) (SCORM, 2004) através da rede avançada de aprendizagem distribuída (*Advanced Distributed Learning* – ADL). É bom lembrar que esses cursos levam em conta a língua e a metodologia de ensino da instituição, isto é, é difícil padronizar um material educacional para instituições de outros países.

O padrão SCORM é baseado na padronização de objetos de aprendizagem da AICC (*Aviation Industry CBT Committee*) e tem a capacidade de ser utilizado por vários sistemas de gerenciamento de aprendizagem (*Learning Management System* – LMS). Um *SCO* (Objeto de Conteúdo Compartilhado) representa o nível mais baixo de conteúdo granular que pode ser rastreado por um *LMS* e deve ser independente do contexto de ensino para ser reutilizado em diferentes situações de aprendizagem (BOHL, 2002).

O Comitê de Padrões de Tecnologia de Aprendizagem (*Learning Technology Standard Committee* – LTSC) é um órgão regulamentador que busca a padronização de materiais de estudo para serem reutilizados em cursos diferentes, mas com conteúdos referenciados.

O MIT (*Massachussetts Institute of Technology*) dos EUA disponibiliza na Internet diversos cursos gratuitos na forma de apostilas segundo o formato *SCORM*. Cada curso contém os chamados meta-dados de objetos de aprendizagem (*Learning Object Metadata* – LOM) que irão ser gerenciados por um sistema instrucional de gerenciamento (*Instructional Management System* – IMS).

2.2.2 Ferramentas Computacionais para Ensino à Distância

Existem diversos *softwares* (ferramentas computacionais) que auxiliam a organização e gerenciamento de materiais educacionais, criando assim plataformas virtuais de aprendizagem que são chamados de mecanismos de aprendizagem. Essas plataformas podem não oferecer todas as ferramentas necessárias para se construir um curso à distância, mas agilizam o processo de desenvolvimento e organização de um curso. Nesta seção algumas dessas plataformas serão apresentadas e descritas.

Alguns *softwares* são de domínio público na Internet, tais como *MOODLE* (MOODLE, 2005) e *Claroline* (CLAROLINE, 2005). Existem outros que são *softwares* comerciais, como o *Blackboard* (BLACKBOARD, 2005) da empresa de mesmo nome, o *WebAssign*, da empresa Advanced Instructional Systems em conjunto com Universidade Estadual da Carolina do Norte (EUA) (WEBASSIGN, 2005), o *webAula* (WEBAULA, 2005) e o *WebCT* (WEBCT, 2006). Ainda há sistemas que não estão disponíveis fora da instituição de desenvolvimento, como o *EDUCA* (EDUCA, 2005), da Universidade Católica de Temuco, no Chile, o *Parla!* (PARLA, 2005), do SENAI, o *Instructional Design* (INSTRUCTIONAL DESIGN, 2005), da PUC-Rio, o *Nou-Rau*, da UniCamp (NOU-RAU, 2005), o *TelEduc*

(TELEDUC, 2006), o eProInfo (EPROINFO, 2006) recomendado pelo MEC, o ROODA (ROODA, 2006) desenvolvido na UFRGS e muitos outros.

A seguir, descreve-se as plataformas mais utilizadas atualmente pelas instituições de ensino e relacionadas neste trabalho.

2.2.3 Blackboard

O Blackboard (vide Figura 7) é um *software* comercial distribuído pela empresa de mesmo nome. Fundada em 1997, a empresa começou prestando consultoria contratada pelo consórcio não lucrativo IMS (*Instructional Management Systems*). Mais tarde se fundiu com uma pequena empresa de *software* CMS, da Universidade de Cornell, tornando-se a Blackboard Inc.

Possui a capacidade de importar *SCO*s, fazendo com que a produção de cursos seja simples e fácil. É compatível com a maioria dos padrões de ensino nas indústrias. Possui ferramentas que proporcionam aprendizagem colaborativa através de uma sala de aula virtual (*virtual classroom*). Possibilita a visualização de relatórios de aproveitamento dos alunos (rastreamento de documentos acessados), mas sem avaliação automática.

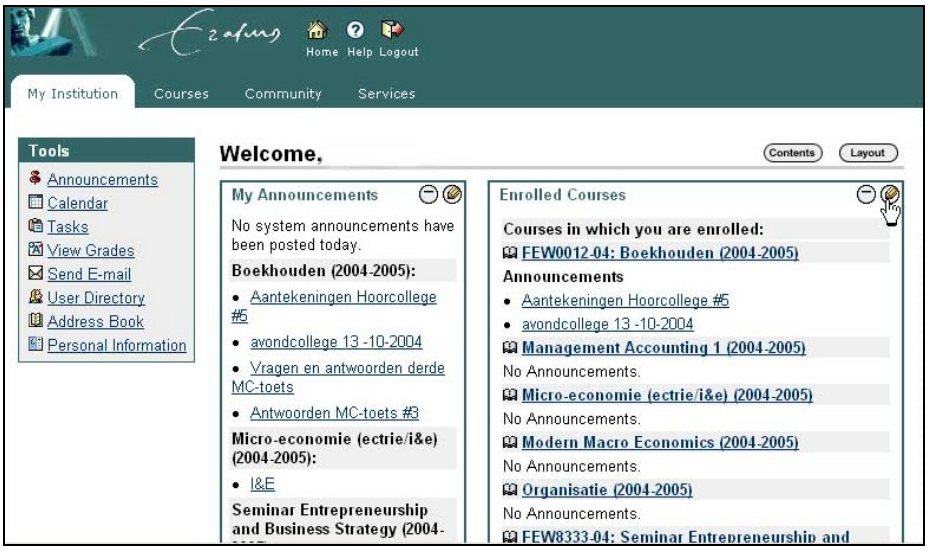

Figura 7. Exemplo de interface do Blackboard.

2.2.4 Claroline

A plataforma do Claroline é baseada em códigos PHP e bancos de dados MySQL, que proporciona um ambiente colaborativo de aprendizagem permitindo que professores ou instituições educacionais criem e administrem cursos na Internet. Essa plataforma disponibiliza gerenciamento de grupos, fóruns, repositório de documentos, calendário, chat, tarefas, links, administração de perfis de usuários, como o mostrado no exemplo da UFRGS na Figura 8. O Claroline foi traduzido em 28 línguas e é usado por centenas de instituições de ensino. O software foi produzido inicialmente pela universidade de Louvain, Bélgica, e agora está sob domínio público através da GPL. Como o ambiente utiliza banco de dados de acessos, pode-se integrar um sistema para monitorar o aluno rastreando os documentos acessados.

Figura 8. Claroline adotado na UFRGS pelo GCAR.

2.2.5 MOODLE

O *MOODLE* é um *software* bastante parecido com o Claroline, mas mais interativo e de fácil manipulação, isto é, recursos adicionais são facilmente incorporados. MOODLE

significa ambiente de aprendizagem dinâmico modular orientado a objetos (*Modular Object-Oriented Dynamic Learning Environment*).

Segundo seus desenvolvedores, o *software* segue uma particular linha filosófica de aprendizagem, a pedagogia construcionista social (*social constructionist pedagogy*). Esse *software* também é de domínio público e de código aberto (*open-source*), utilizando *PHP* e *MySQL* como banco de dados (pode-se usar outro gerente de bancos de dados *SQL*). O MOODLE conta com um grande número de cursos que podem ser referenciados no seu domínio. Basicamente, possui as mesmas funções do Claroline, isto é, gerenciamento de usuários, chats, fóruns, repositórios de documentos (arquivos), calendário.

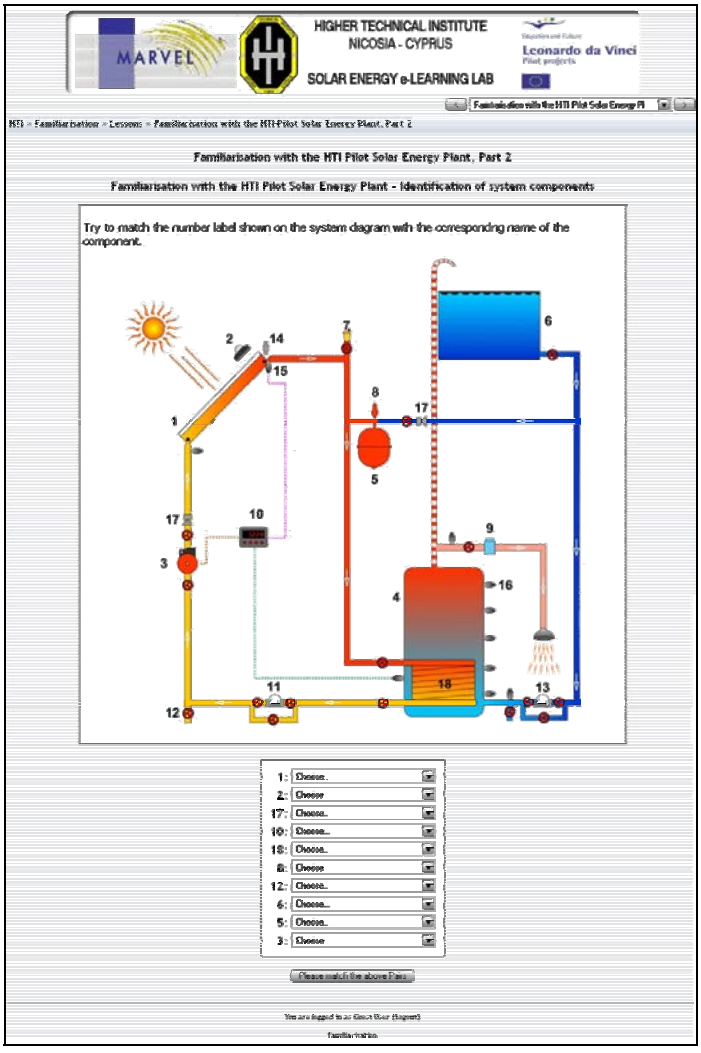

Figura 9. Plataforma MOODLE.

Na Figura 9 ilustra-se um exemplo da utilização do MOODLE como ambiente virtual de aprendizagem integrado no laboratório remoto de energia solar. Este ambiente conta com questionário de múltipla escolha sobre equipamentos usados no experimento.

A filosofia social construcionista do MOODLE evolui o construcionismo afirmando que o aprendizado ocorre melhor em um ambiente colaborativo onde alunos aprendem interagindo entre si. No MOODLE algumas opções permitem aos participantes interagirem tanto como alunos ou como professores, possibilitando compartilhamento de papéis (funções).

A vantagem dessa plataforma é interface com banco de dados (SQL). Pode-se rastrear os passos do usuário para futura manipulação de dados (por exemplo: avaliação). A plataforma Moodle é amplamente utilizada, sendo que à época de escrita desta dissertação indicava-se seu registro em mais de 10.000 *sites* em 147 países com aproximadamente 2,5 milhões de usuários em 242 mil cursos (MOODLE, 2006).

2.2.6 WebCT

O WebCT (*Web Course Tools*) foi originalmente desenvolvido pela University of British Columbia (WEBCT, 2006) em 1995. Em 1999, o *software* virou nome de empresa e passou a ser vendido comercialmente. Atualmente o WebCT foi adquirido por um de seus concorrentes no mercado, o Blackboard. Anteriormente, o WebCT foi muito utilizado em diversos campi universitários como ambiente virtual de aprendizagem (*VLE*) e chegou a ter 10 milhões de alunos em 80 países.

2.2.7 Cursos e Avaliadores

É claro que os cursos não são concebidos automaticamente, portanto os *softwares* citados somente proporcionam uma plataforma de desenvolvimento e gerência (LOM) de cursos. Nenhuma das ferramentas acima citadas proporciona mecanismos de avaliação automatizados, portanto este deve ser desenvolvido separadamente.

Segundo Bortz e Döring (BORTZ, 2002), pode-se usar um método comparativo para se avaliar o aprendizado do aluno, isto é, estratégias de ensino e resultados são avaliados comparando a aprendizagem autodidática (*self-directed learning* – BÖHNE, 2002), aprendizagem com auxílio do professor em aulas presenciais e aulas distribuídas usando a Internet (remotas) com auxílio de tutoriais (*tutorial assistance* – FALTIN, 2002). Esse método está de acordo com a UCIT (*Universal Constructive Instructional Theory*) de 1991 que divide o sistema de aprendizagem em quatro componentes: aluno, tarefa de aprendizagem, ambiente de aprendizagem e estrutura de referência (*framework*). Também separa o sistema de aprendizagem em três processos: uso de conhecimento, aquisição de conhecimento e armazenagem de conhecimento (TUTTAS, 2001).

Sistemas de inteligência da Web (*Web Intelligence* – WI) foram propostos para sistemas de educação adaptativos (*Adaptative Educational Systems* – AES) de instruções orientadas construcionistas em um laboratório remoto de sistemas de tutores/guias autônomos. Com o uso do sistema educacional de áudio/vídeo, baseados na *Web*, alunos mostram grande interesse em manusear experimentos remotos, mas ainda tem-se o problema de acompanhamento pelos professores. O sistema *WI* coleta informações e age como um guia para as experiências, não respondendo a perguntas com respostas padronizadas e, sim, respostas indiretas que fazem com que o aluno chegue sozinho à resposta. O mecanismo de resposta do *WI* é programado através de um banco de dados de perguntas e respostas formuladas pelo professor, que possuem pequenas variantes, isto é, perguntas semelhantes são respondidas da mesma forma. Uma lógica *fuzzy* é aplicada na decisão da resposta pelo sistema *WI* (LI, 2003).

No momento, mecanismos de avaliação (*evaluation engines*) ainda não são muito difundidos, pois dependem de vários fatores (modelo do aluno) que ainda não foram identificados para que sejam realmente utilizados e automatizados. O ideal seria ter uma

plataforma de aprendizagem que pudesse ser rastreada, isto é, informações sobre acessos dos alunos armazenados para futuras avaliações, por exemplo, por um sistema automático de avaliação. O *MOODLE* e o *Claroline* possuem interfaces com bancos de dados e estes dados podem ser rastreados e se encaixariam perfeitamente no cenário de desenvolvimento de um sistema automático de avaliação. É evidente que não basta somente rastrear documentos acessados pelos alunos, e, sim, em conjunto com isso, impor testes que demonstrem o potencial desses alunos, assim como experiências remotas, que também são gravadas em bancos de dados para serem comparadas e avaliadas pelo professor.

3 ESTADO DA ARTE

3.1 INTRODUÇÃO

Abastecendo a crescente demanda por cursos cada vez mais interativos na área de automação industrial, laboratórios remotos têm sido desenvolvidos para proporcionar um ensino baseado no uso de experimentos remotos. Mecanismos de aprendizagem estão disponíveis e podem ser integrados aos experimentos remotos para acomodar guias educacionais aos conceitos teóricos envolvidos na experiência, assim como oferecer apoio prático a questões relevantes.

Essa seção apresentará uma série de trabalhos de instituições de ensino que utilizam experimentos remotos que de alguma forma têm caráter educacional. Experimentos remotos, dos tipos anteriormente citados, e de diversas aplicações diferentes, serão analisados e comparados.

3.2 EXPERIMENTOS EXISTENTES

3.2.1 OnlineLab (DUAN, 2003)

O *OnlineLab*, desenvolvido pela Universidade Tecnológica de Nyang, Singapura, possui dois experimentos: pêndulo invertido e sistema de tanques acoplados, ambos acessíveis no endereço `http://onlinelab.ntu.edu.sg/onlinelab.htm`

3.2.1.1 Pêndulo Invertido

Foi desenvolvido para proporcionar aos alunos um ambiente para aprendizado de técnicas de projeto de controladores estudados em teoria de sistemas de controle, desde controle clássico até controladores utilizando lógica *fuzzy* e redes neurais. A interface utilizada foi desenvolvida especificamente para a experiência, nenhum *software* comercial foi

utilizado no desenvolvimento dos sistemas de aquisição de dados, gerenciamento e interface *Web*.

Os equipamentos utilizados são: um microcontrolador e portas seriais. O servidor/gerenciador da experiência se comunica por meio da porta serial com o controlador. O ângulo é medido por um potenciômetro, enquanto que um motor mede a velocidade através de um *encoder*.

O desafio proposto aos alunos é girar o pêndulo (vide Figura 10) da posição normal (virado para baixo) e balançá-lo na posição invertida (virado para cima) controlando a potência do motor (saída PWM do controlador) através das medidas de ângulo (potenciômetro passando pelo conversor A/D) e da velocidade com o *encoder*. O aluno deve passar por duas fases antes de testar o projeto do controlador: primeiramente deve realizar a identificação do sistema a ser controlado e posteriormente realizar o projeto de controlador com realimentação de estados.

3.2.1.2 Tanques Acoplados

Este experimento tem a construção bastante compacta com dois tanques de acrílico tipo torre acoplados a um reservatório de água montado logo abaixo dos tanques. Duas bombas da água independentes enchem cada tanque do topo. O *LabVIEW* foi utilizado para gerenciamento do *hardware* e geração de interface para *Web*. Uma placa de aquisição de dados serve como *hardware* de medição e controle.

Figura 10. Experimentos do OnlineLab.

Diversas estratégias de controle podem ser utilizadas, como: manual, *ON/OFF* e PID. Depois de finalizado o experimento, os dados da experiência podem ser copiados pelo usuário remoto. Os experimentos estão ilustrados na Figura 10, onde à esquerda estão os tanques acoplados e à direita o pêndulo invertido.

3.2.2 NetLab (ZHANG, 2004)

Desenvolvido pela Universidade de Zhejiang na China este laboratório remoto oferece mais de trinta experimentos remotos divididos em quatro categorias:

i. Eletroeletrônica:

   Circuitos analógicos e digitais.

ii. Eletrônica de potência:

   Retificadores e inversores.

iii. Controle automático:

   Elementos clássicos de controle, resposta em freqüência, sistemas servo motores, pêndulo inverso, ventoinha e disco, controle de motor de passo, tanques acoplados, controle de elevadores.

iv. Motores elétricos:

   Operação de motores elétricos.

Para acessar e utilizar esse laboratório o aluno necessita de um *software* específico desenvolvido para prover acesso ao NetLab e compilar códigos *C* para *DLL*s, que serão executadas no servidor para controlar os experimentos.

Alguns recursos interessantes usados nesta proposta são a gravação de vídeos para visualização pós-experiência e controle de segurança do código enviado (*DLL*). Para garantir o funcionamento dos equipamentos, mesmo depois de uma pane (queda de energia), todos os experimentos são *self-reset*, isto é, reiniciam automaticamente ficando prontos para utilização.

Apesar dos múltiplos experimentos utilizados, nenhuma citação sobre mecanismos educacionais foi apresentada no trabalho.

3.2.3 REL & VELO (AUER, 2003)

REL é a sigla para “Remote Electronics Laboratory”, que disponibiliza um conjunto de *softwares* acessíveis remotamente através de uma ferramenta chamada *Citrix Application Server*, que gera arquivos *HTML* como interfaces do simulador para que o usuário remoto possa interagir com os simuladores no servidor. Os *softwares* utilizados são o *MatLab* (simulador), *OrCAD* e *PAC-Designer*.

Os cursos são criados pelos próprios professores escritos em arquivos *XML* para melhor modularização e estruturação. Tutoriais com os conhecimentos teóricos são organizados de forma a introduzir o aluno na experiência e nos objetivos desejados. Simuladores são utilizados para propor aos alunos tarefas que serão postas sob prova em questionários de múltipla escolha. Os experimentos são utilizados na Universidade Técnica de Carinthia em Villach na Áustria.

Já o *VELO* é uma sigla para *Virtual Electronics Laboratory*, que é um experimento remoto real de testes de circuitos que usa o *LabVIEW* como gerenciador e gerador da interface *Web*. O experimento está montado (parte *hardware*) na Universidade da Transilvânia em Brasov na Romênia. Tutoriais e *HTML*s didáticos estão distribuídos entre a universidade austríaca e romena.

Em ambos os casos a topologia usada por alunos é *thin-client*. A preocupação com a transferência do conhecimento para os alunos é explícita no trabalho que apresenta diversas vantagens dos experimentos remotos reais, que, ainda assim, não possibilitam a montagem do experimento (conexões e medições dinâmicas). Testes efetuados pelas universidades deste projeto comprovaram que alunos se preparam melhor para experiências quando submetidos a materiais educacionais de qualidade e a aprendizagem colaborativa.

3.2.4 VLab (ALBU, 2004)

O VLab, *Virtual Laboratory*, é um laboratório real de experimentação remota que é acessível via *Web* pela interface criada no *LabVIEW*. O experimento é utilizado em aulas de sistemas de potência, pois envolve um circuito trifásico com três chaves contactoras e diversas cargas elétricas.

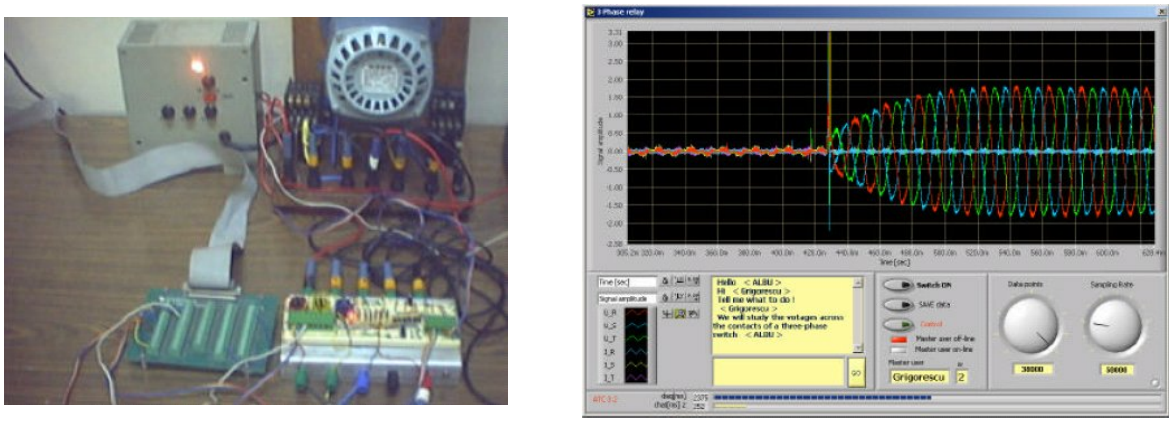

Figura 11. Experimento do VLab.

A Figura 11 ilustra o experimento real (à esquerda) e a interface do instrumento virtual (à direita). O objetivo é demonstrar os efeitos da ligação sem sincronia de cargas trifásicas.

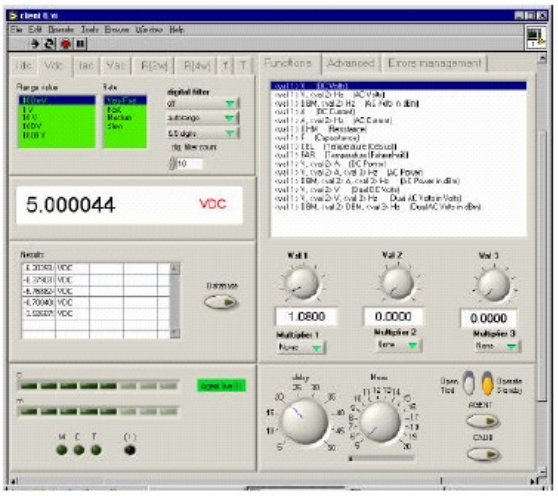

Figura 12. VLab – calibração virtual.

A calibração de equipamentos virtuais, ilustrada na Figura 12, é uma peculiaridade deste trabalho. O experimento também está distribuído entre a Universidade Estadual do Arizona e Universidade de Bucareste que utilizam o experimento em caráter colaborativo.

3.2.5 AIM-Lab (SHEN, 1999)

O AIM-Lab, ***A****utomated* ***Internet*** ***M****easurement* ***Lab****oratory*, da Universidade Norueguesa de Ciência e Tecnologia em conjunto com a o Instituto Politécnico Rensselaer usa a medição remota para a caracterização de dispositivos semicondutores pelos seus alunos em um curso de microeletrônica. Neste laboratório de medição remota, como é chamada, alunos podem levantar curvas características de polarização de transistores *CMOS*, que são testados num *chip* de *CMOS* variando-se a tensão entre dreno e fonte, correntes de dreno e tensão de porta (*gate*).

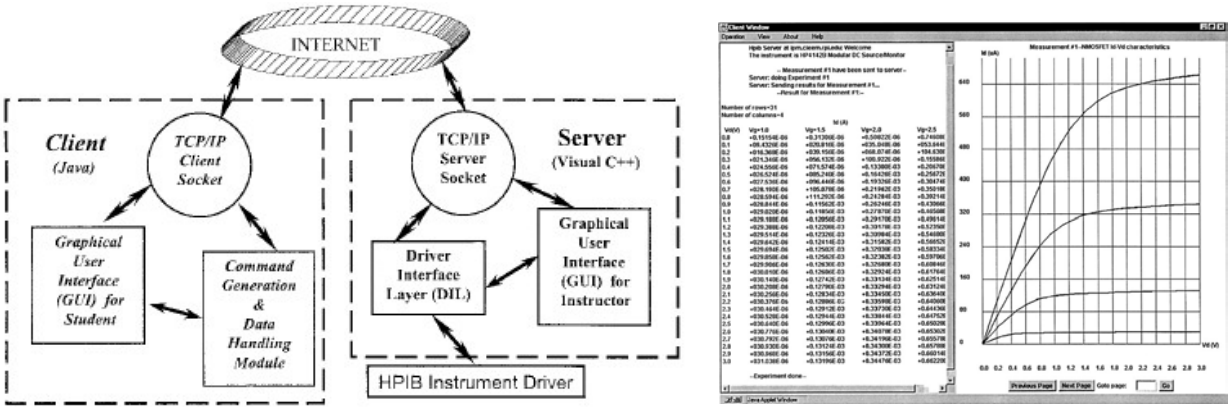

Figura 13. Experimento do AIM-Lab.

A Figura 13 ilustra a configuração do *AIM-Lab* com a arquitetura de acesso à direita e os resultados de uma experiência num transistor *NMOS* à esquerda.

O interfaceamento do experimento real é gerenciado por um programa desenvolvido especialmente para esta aplicação em código *Visual C++*. A aquisição de dados é conduzida por equipamentos compatíveis com o protocolo de comunicação serial HPIB IEEE 488

(GPIB). O gerenciador também é um servidor da experiência que se comunica com o cliente por meio de *sockets* TCP/IP. No lado do cliente uma simples *applet Java* estabelece conexões via *sockets* com o servidor. O trabalho de Shen (1999) faz ressalvas à utilização de *softwares* proprietários e placas de aquisição de dados comerciais, comentando que isto provoca dependência do sistema a versões específicas de *hardware* e *software*.

3.2.6 Projeto LabNet (DAVOLI, 2001)

Criado em Abril de 2000 pelo Ministério de Pesquisa e Desenvolvimento da Itália, o projeto ***Lab****oratory* ***Net****work* inclui as seguintes organizações: CNIT (Consórcio Italiano de Telecomunicações) e CINI (Consórcio Italiano de Ciência da Computação).

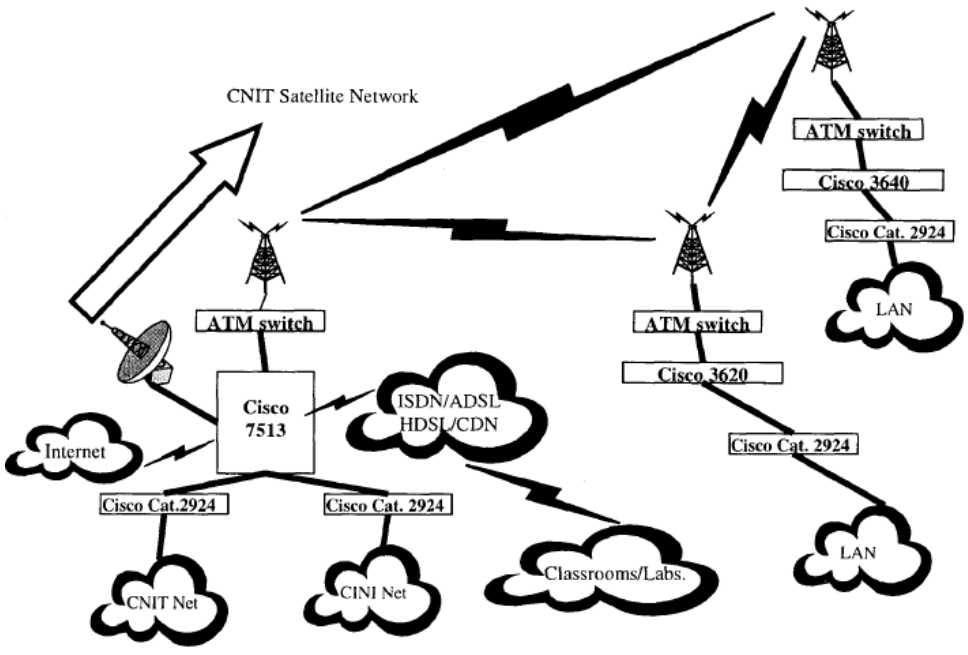

Figura 14. Arquitetura e dispositivos do LabNet.

A arquitetura (vide Figura 14) foi montada na região de Nápoles e com derivações em Savona e Gênova. O laboratório remoto oferece uma completa análise de redes para testes com vários tipos de velocidades de conexões de Internet (WAN, ISDN, LAN, T1, etc.) e equipamentos como switches e hubs. Análises de novos equipamentos e protocolos, que

utilizam qualidade de serviço (*QoS*), podem ser testados em uma rede fechada. Os acessos aos equipamentos são os mais variados possíveis, desde comunicação serial até SNMP (*Simple Network Management Protocol*).

3.2.7 Projeto PEARL (FERREIRA, 2002; COPPER, 2000; COOPER, 2002)

O Projeto PEARL foi criado pela *European Commission's of Information Society Technologies* em março de 2000 e terminado em fevereiro de 2003. A sigla significa experimentação prática por acesso educacional remoto (*Practical Experimentation by Accessible Remote Learning*). A página do projeto está acessível no endereço:

`http://iet.open.ac.uk/pearl`

O projeto foi desenvolvido por um consórcio de instituições de ensino: Open University (Reino Unido), Trinity College (Irlanda), Faculdade de Engenharia da Universidade do Porto – FEUP (Portugal) e University of Dundee (Escócia), e, como parceira industrial, a empresa Zenon da Grécia. Cada instituição envolvida possuia um experimento remoto distinto, abrangendo as seguintes áreas: bioquímica, física fundamental, inspeção visual e eletrônica digital. Todos os experimentos do PEARL utilizam a topologia *thin-client*, e, como plataforma de aprendizagem, o *WebCT* (já apresentado na Seção 2.2.6).

3.2.7.1 Espectrômetro Ótico Motorizado

O espectrômetro foi desenvolvido pela Open Universisty em conjunto com a empresa Zenon. Ele tem por finalidade demonstrar as diferentes características espectrais de cada elemento químico (vide Figura 15). O experimento começa com uma calibração do equipamento através da utilização de uma lâmpada de sódio como referência. Depois de calibrado, os alunos podem observar as medições de comprimentos de onda espectrais quando quatro diferentes elementos químicos são introduzidos no bico de Bunsen.

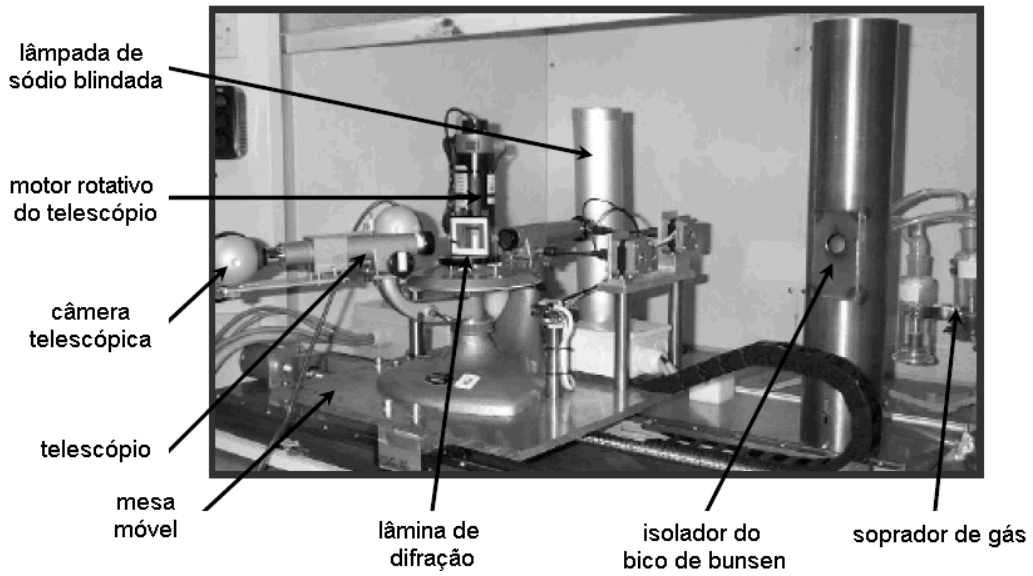

Figura 15. Experimento da Open University.

O experimento é gerenciado por *software* próprio e a interface Java do cliente se comunica com o gerenciador por pontes *CORBA* (*CORBA Bridges*). A transmissão de vídeo é encapsulada em pacotes *RTCP* (Real-Time Control Protocols) pelo *Java Media Framework* (JMF). O acesso ao *hardware* do experimento utiliza a tecnologia *ActiveX*.

3.2.7.2 Sistema de Processamento de Imagens (“Machine Vision”)

O equipamento foi produzido pelo Trinity College Dublin. O experimento analisa, identifica e procura defeitos em circuitos impressos através de processamento digital de imagens (vide Figura 16). Os circuitos impressos estão acomodados em uma base motorizada com liberdade para mover em duas dimensões, isto é, *XY*. O circuito é analisado pela sistema de visão, o qual conta com uma calibração inicial da câmera através de marcas na base motorizada.

Diversos experimentos podem ser montados com estes equipamentos. Alunos podem calcular a distância focal da câmera através de um tabuleiro de xadrez de dimensões conhecidas, que se encontra na base motorizável. A iluminação e o tempo de exposição, assim

como o *zoom* da câmera, podem ser controlados para determinar características do sistema de visão.

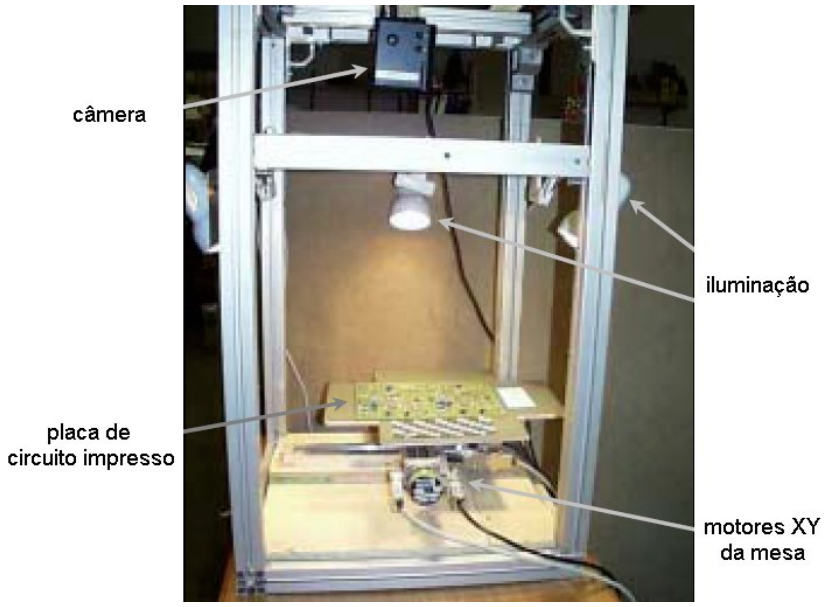

Figura 16. Experimento do Trinity College.

O *MatLab* (vide MatLab no Apêndice – Simuladores) pode ser usado com seu pacote de processamento de imagens para identificar componentes no circuito impresso com a imagem adquirida pelo experimento através da técnica de reconhecimento de padrões. Outra utilidade do experimento está na determinação de linearidade da câmera para variações de iluminação.

O gerenciador da experiência também foi desenvolvido no Trinity College e está escrito em linguagens *C/C++* e *Java*. Os equipamentos do experimento são controlados por portas seriais RS-232. A interface remota do experimento utiliza *CORBA* para se comunicar com o gerenciador que dispara *servlets* no servidor *Web Tomcat.*

3.2.7.3 Experimentos de Lógica Digital

O experimento foi desenvolvido pela Faculdade de Engenharia da Universidade do Porto (FEUP). Ele possui três variações: programação de microcontroladores 80C51, projeto

de lógica em FPGA e projetos de testes utilizando as padronizações IEEE 1149.1 e IEEE 1149.4.

No experimento de programação de microcontroladores 80C51, os alunos enviam o código hexadecimal do controlador, que é executado no servidor, demonstrando todo o tipo de resposta do programa desenvolvido (vide Figura 17).

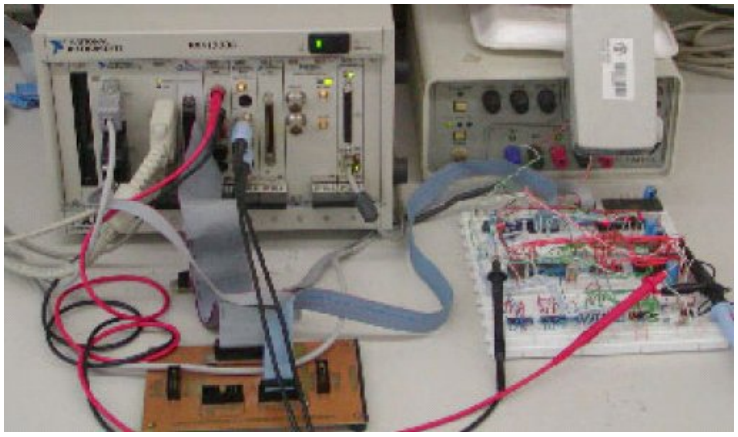

Figura 17. Experimento de programação de microcontrolador 80C51.

Para se demonstrar o projeto de lógica de circuitos *hardwired* , uma placa com FPGAs da empresa Xilinx é utilizada com uma interface quase idêntica à do experimento de programação do microconrolador 80C51. De fato, a única modificação é que o *hardware* utilizado não suporta sinais analógicos, portanto osciloscópios são utilizados somente para visualização de formas de onda digitais.

Na última experiência desenvolvida, uma placa da IEEE 1149.4 é utilizada para demonstrar aos alunos técnicas de testes de circuitos digitais (IEEE 1149.1 e IEEE 1149.4) e analógicos (IEEE 1149.4). Nos testes da IEEE 1149.1 são usados procedimentos de teste padrão e varredura de limites da arquitetura, enquanto na IEEE 1149.4 testes padrões para barramento de sinais mistos (digitais e analógicos).

O experimento utiliza o *LabVIEW* (vide LabVIEW no Apêndice – Simuladores) para gerenciar a experiência e a aquisição de dados (sistema PXI). O servidor, responsável por gerar a interface remota do experimento, é uma ferramenta comercial chamada *AppletVIEW* (NACIMIENTO, 2006) que disponibiliza uma janela de chat, uma janela de videoconferência (*CuSeeMe*) e uma interface *Java* do experimento remoto (vide Figura 18). O programa de videoconferência *CuSeeMe* foi desenvolvido pela Universidade de Dundee para ser usado em aulas e teleconferências.

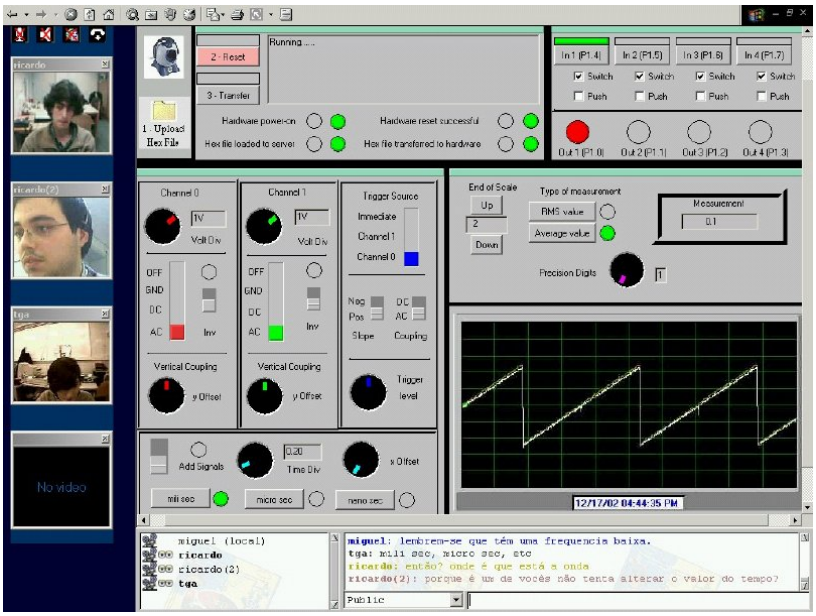

Figura 18. Interface do experimento da FEUP.

3.2.8 Projeto WGLN (WGLN, 2005)

A primeira tentativa de uma rede global de ensino foi proposta no projeto ***W**allenberg **G**lobal **L**earning **N**etwork*, criado em 1999 pela Swedish Learning Lab (*SweLL*) com os seguintes participantes: Uppsala University (Suécia), Royal Institute of Technology (Reino Unido), Karolinska Institute (Suécia) e a Universidade de Stanford (EUA). Aderiram ao

projeto em 2000 a Universidade de Hannover, Universidade de Braunschweig e a Braunschweig School of Arts (todas alemãs).

Mais de 30 projetos estão associados aos pesquisadores ligados à *WGLN*. Atualmente estuda-se a implementação de cursos dos seguintes projetos:

- Visualização 3D para o Aprendizado Avançado de Anatomia e Patofisiologia;
- Exames medicinais baseados em sintomas de pacientes simulados (*web-based simulation of patients*);
- Simulador de Perspectivas Visuais para Operações no Cérebro;
- Treinos Cognitivos Computadorizados (CCT);
- Sala de emergência simulada para otimização de trabalhos em grupo (SimTech);
- Incorporação e Caracterização de Equipamentos Avançados em Pesquisas e Educação de Nanotecnologia;
- Integração de Dispositivos Móveis e Espaços de Trabalhos Interativos para Projetos;
- Simulação de Ambientes para Radiologia;
- REALSIMPLE: Combinações de Realidade Física com Simulações em Experimentos de Laboratório.

3.2.9 Projeto MARVEL (MICHAELIDES, 2004; FERREIRA, 2004)

O laboratório virtual para aprendizado eletrônico e acesso remoto em mecatrônica (*Virtual Laboratory in **M**echatronics **A**ccess to **R**emote and **V**irtual **E**-**L**earning*) foi um projeto criado em 2002 como parte do programa Leonardo da Vinci II, da Comunidade Européia. Faziam parte do projeto as universidades do Porto (UP – Portugal) e de Bremen (UB – Alemanha), o colégio técnico de Delmenhorst (DEL – Alemanha), o Instituto Técnico do Chipre (HIT – Chipre), o West Lothian College (WLC – Escócia), as empresas Zenon

(Grécia), FESTO e BMW (Alemanha), a autoridade de qualificação Scottish Qualifying Authority (Escócia), e os parceiros associados Faculdade de Engenharia do Porto (FEUP – Portugal) e Haute École Valaisanne (HEV – Suíça). A página do projeto está acessível no endereço:

`http://www.marvel.uni-bremen.de/`

Assim como o PEARL, o MARVEL também utiliza um ambiente virtual de aprendizagem (VLE) específico em todas as suas experiências remotas para auxiliar a aprendizagem através dos experimentos. Todos os experimentos estão integrados no VLE *MOODLE* (vide Seção 2.2.5).

3.2.9.1 deriveSERVER

O projeto da Universidade de Bremen de servidor de ambiente real e virtual distribuído de aprendizagem de mecatrônica e serviço teleoperado (*Distributed Real and Virtual Learning Environment for Mechatronics and Teleservice*), chamado de *deriveSERVER*, oferece um ambiente completo para experimentação realidade mista (vide Figura 19). O experimento está acessível no endereço:

`http://lab.artec.uni-bremen.de`

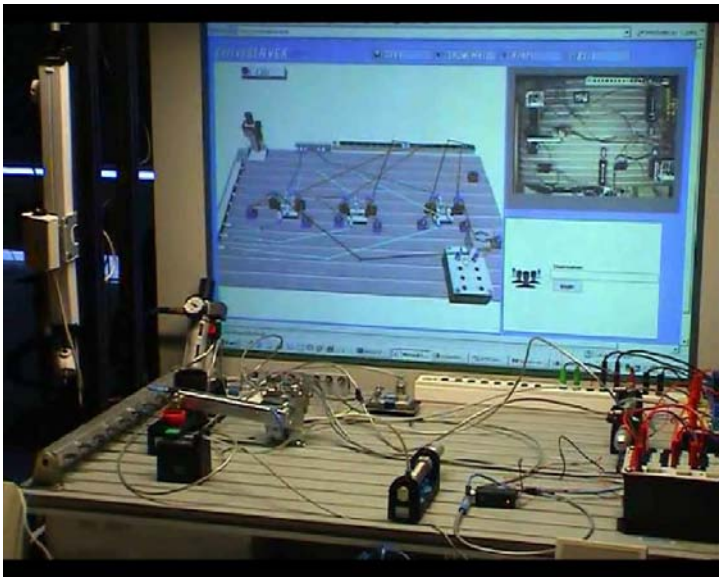

Figura 19. Experimento da Universidade de Bremen.

Equipamentos eletro-pneumáticos são utilizados tanto em simulações quanto em experimentos reais. Uma ferramenta chamada *hyper-bond* (vide Seção 2.1.4.2) permite que equipamentos virtuais (simulados) interajam com equipamentos reais e vice-versa. Desta forma temos uma perfeita combinação de experimentos, isto é, realidade mista.

experimento tem o objetivo de introduzir aos alunos noções básicas de eletro pneumática. Na bancada virtual podem ser montados diversos circuitos eletro pneumáticos usando a biblioteca de equipamentos (modelos VRML) fornecida na interface remota virtual. Alunos podem interagir remotamente com equipamentos reais acionando (conectando o ar comprimido ou alimentação de tensão) às saídas do *hyper-bond*. A bancada real também pode intergir com a bancada virtual (ou seja a interação é bi-direcional) mas o acesso aos equipamentos reais neste caso deve ocorrer localmente (no laboratório real).

O *software* chamado ROMAN (*Real Object Manager*) desenvolvido em *Visual C++* é o gerenciador da experiência, pois centraliza todas as ações e comportamentos dos equipamentos, tanto virtuais como reais. Para se ilustrar a utilização do conceito de *hyper-bond* foi desenvolvido um *software*, que leva o mesmo nome do conceito, que é responsável pelo interfaceamento do *hardware*, isto é, ele é responsável em transmitir sinais (esforços) dos equipamentos virtuais aos reais e também dos reais aos virtuais. O *hardware* utilizado para a geração/aquisição de sinais utiliza o *EasyPort*, da empresa FESTO, e sensores e relés de eletropneumática. O *VCK* (*Virtual Construction Kit*) é uma aplicação Java que carrega a interface virtual do experimento. Os equipamentos virtuais utilizados são descritos em uma linguagem de programação de modelos de realidade virtual chamada de *VRML*. *Java Applets* e *scripts* são utilizados para que as ações (entradas do mouse e teclado), efetuadas no manuseio dos modelos *VRML*, possam ser capturadas e transmitidas ao *ROMAN*. A interação com os modelos virtuais é possível utilizando-se o EAI 2.0 (*External Authoring Interface*) do *Plug-in VRML Cortona*, da empresa *Parallel Graphics*. O *ROMAN* gerencia a cena do

experimento virtual e, através de protocolo de comunicação próprio, envia a descrição *VRML* da cena ao VCK. Uma câmera de vídeo (pode ser uma WebCam) disponibiliza imagens, usando o *JMF*, para visualizar os equipamentos reais conectados à experiência.

A arquitetura de *softwares* da experiência é baseada na topologia cliente-servidor, onde o ROMAN é o servidor e os outros (VCK e hyper-bond) são clientes, facilitando assim a distribuição e modularização do sistema (vide Figura 20).

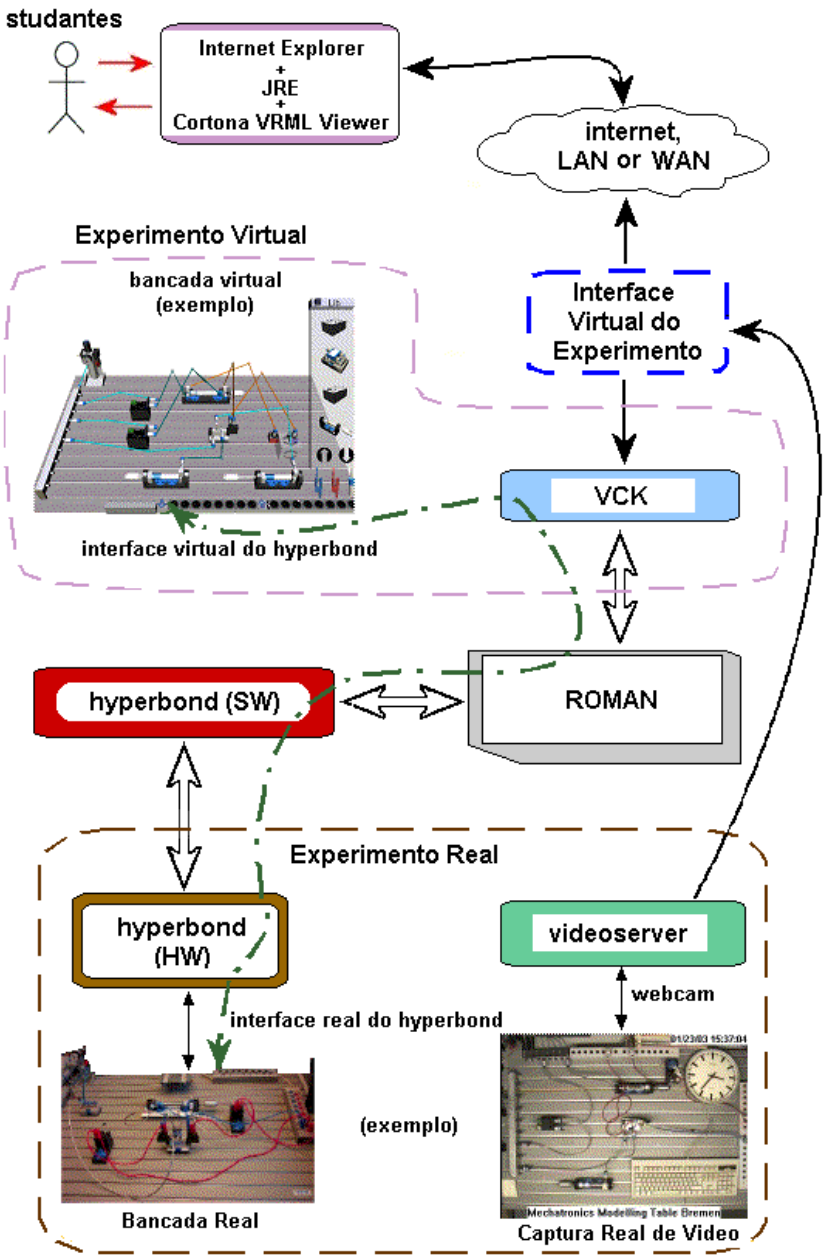

Figura 20. Arquitetura do deriveSERVER.

Apesar de possuir uma integração com o *MOODLE*, nenhum curso sobre a utilização do experimento foi formulado. O maior objetivo do experimento é demonstrar a capacidade de implementação dos *hyper-bonds*, que são usados em diversos projetos de realidade mista no *ArtecLab* na Universidade de Bremen.

3.2.9.2 Planta de Energia Solar

Criado pelo Instituto Superior de Tecnologia do Chipre (HTI), o experimento conta com painéis de energia solar para geração de energia elétrica e também para aquecimento de um sistema de água (MICHAELIDES, 2004).

Cursos criados dentro do *MOODLE* são chamados de *livros virtuais* que possibilitam aos alunos conhecimento do experimento e até questionários de identificação dos equipamentos envolvidos no laboratório e de processos relativos à geração de energia térmica e elétrica. A Figura 21 mostra as instalações do laboratório à esquerda, e à direita está arquitetura do experimento (equipamentos).

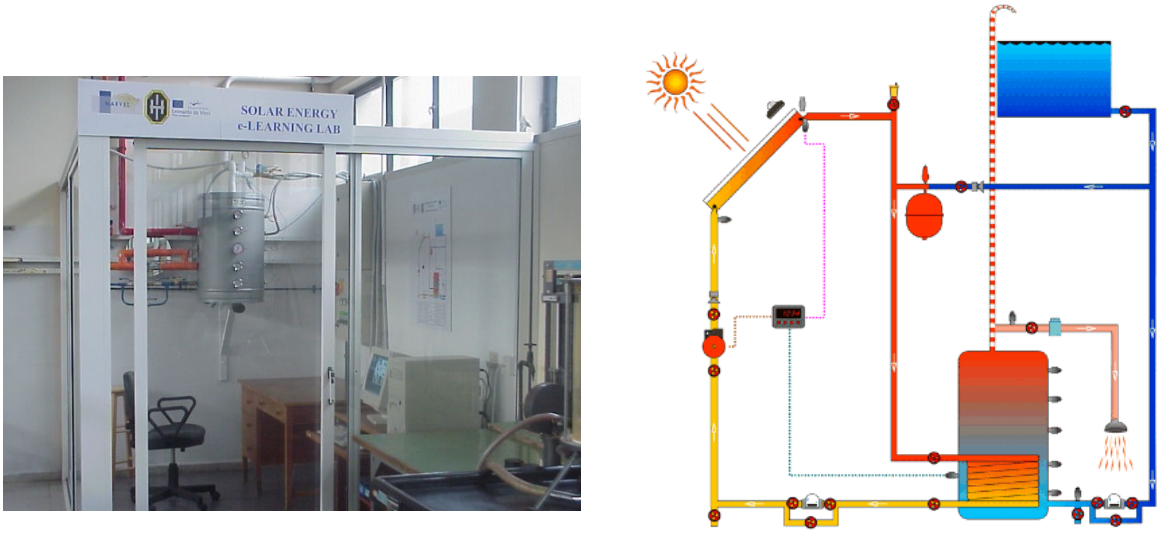

Figura 21. Experimento do HTI.

A Figura 9, anteriormente comentada na Seção 2.2.5, ilustra a instalação do MOODLE no HIT e sua implementação com perguntas de múltipla escolha.

3.2.9.3 Planta de Aquecimento Solar

O experimento foi desenvolvido e implementado pelo Colégio Técnico de Delmenhorst. Ele consiste de uma planta térmica solar. O experimento oferece pouca interatividade com o usuário remoto. O sistema supre um aquecimento de água de 1200 litros por dia através de quatro painéis solares instalados no teto do colégio. Um *software* SCADA, o *WinCC* da Siemens, é utilizado para a supervisão do processo. Lógicas de processos são controladas por um CLP.

A Figura 22 ilustra à esquerda instalações do experimento e à direita o supervisório utilizado para controlar a experiência.

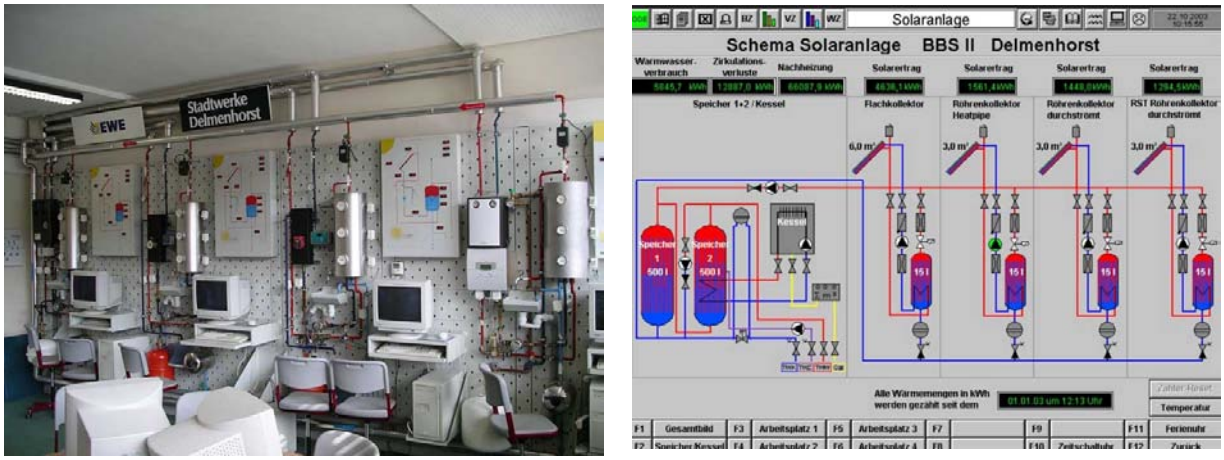

Figura 22. Planta térmica solar de Delmenhorst.

3.2.9.4 Lógica Digital

O experimento criado pela FEUP é o mesmo utilizado no projeto PEARL pela instituição (vide Seção 3.2.7.3). Materiais educacionais anteriormente integrados no *WebCT* agora foram disponibilizados através do *MOODLE*. Algumas funcionalidades foram adicionadas como a utilização da videoconferência integrada no *MOODLE* através do uso do *FlashCom Server* da *Macromedia* (atualmente *Flash Media Server* da *Adobe*) e o uso de sistema de reservas de horários (*booking systems*). O sistema de reservas foi desenvolvido como módulo do próprio *MOODLE* (vide Figura 23).

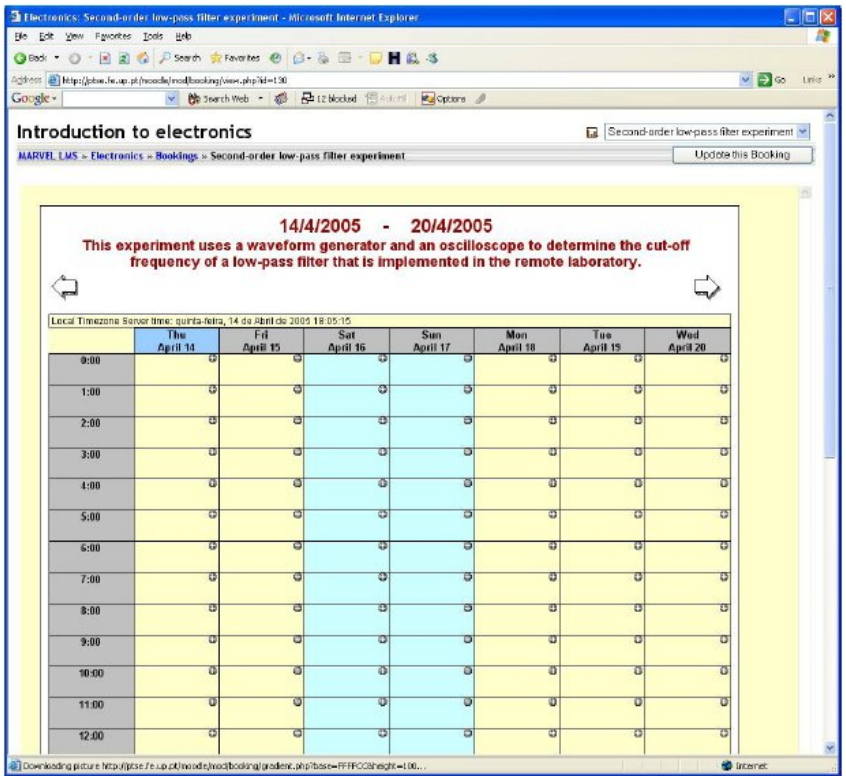

Figura 23. Módulo de sistema de reservas.

3.2.9.5 Treinamento de Robôs

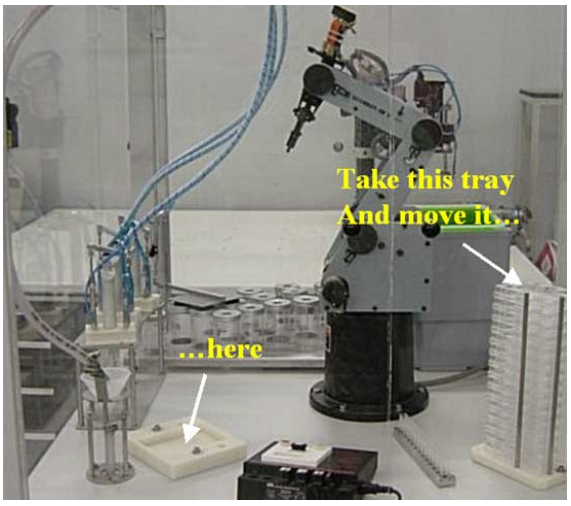

Figura 24. Experimento da Zenon.

Desenvolvido pela Zenon este experimento remoto de treinamento de robôs tem por objetivo propiciar aos usuários remotos a possibilidade de utilizar robôs para efetuar

exercícios relacionados ao treinamento de robôs (vide Figura 24). Cursos desenvolvidos para treinamento de *trainees* em áreas de segurança e saúde, familiarização com o equipamento, programação básica e programação industrial são utilizados para capacitar trabalhadores para o *learning in the job* (aprendizado no trabalho).

3.2.9.6 Experimentos Diversos de Mecatrônica

Desenvolvido pelo WLC consiste de dois sistemas digitais: uma caixa de controle da *Deltronics* e placa de programação do micro controlador *PIC*, e um sistema eletro pneumático.

3.2.10 ACT (CASINI, 2003; CASINI, 2004)

O ***A****utomatic* ***C****ontrol* ***T****elelab* é um laboratório com diversos experimentos de controle. Está acessível livremente no endereço:

`http://www.dii.unisi.it/~control/act/`

Existem seis experimentos acessíveis para projetos de controladores em plantas reais. O controlador é gerado por *software* enquanto a planta é real. Todos os experimentos podem ser controlados tanto com controladores predefinidos (geralmente PID), isto é, ajustando parâmetros, quanto por controladores definidos pelo usuário.

Os controladores desenvolvidos pelos usuários têm que seguir regras de um modelo (*template*) do *Simulink*. Antes de iniciar o experimento alunos carregam o modelo do controlador definido pelo usuário para que o controlador possa ser testado. Para que se possa utilizar este recurso o pacote RTW (*Real Time Workshop*) do *MatLab* (vide MatLab no Apêndice – Simuladores) foi utilizado. Nele o modelo do controlador é compilado e “linkado” com as bibliotecas que possuem funções de comunicação com os instrumentos ligados ao experimento real, produzindo um programa executável que irá rodar no servidor para controlar o experimento.

A interface do usuário utiliza a topologia *thin-client* com a utilização de somente aplicativos *Java*. A transmissão de vídeo é de baixa qualidade através de *Java Applets*.

3.2.10.1 Controle de Posição Angular

Usando um motor de corrente contínua (CC) conectado a um medidor de ângulo (foi utilizado um simples potenciômetro) pode-se realizar experimentos de controle de posição angular. Variando-se a entrada do sistema (sinal de tensão elétrica) é possível visualizar a resposta do controlador projetado (saída – posição angular). Diversos gráficos são utilizados para visualização das variáveis envolvidas no processo. As entradas podem ser sinais do tipo: salto, rampa, senoidal ou quadrada.

3.2.10.2 Controle de Velocidade

Da mesma forma que para o controle angular o experimento pode ser utilizado para controlar a velocidade do motor *CC*. A velocidade do motor é medida por um dínamo tacométrico. A interface é exatamente a mesma do experimento anterior com exceção de um modo de identificação do sistema, isto é, pode-se executar a experiência para identificar os parâmetros do sistema.

3.2.10.3 Controle de Nível

O experimento consiste em controlar o nível de um tanque de água de 3 litros que possui um escape de água no fundo conectado ao reservatório. Um sensor de pressão é utilizado para medir o nível da água no tanque. Além do controlador predefinido PID convencional pode-se utilizar controladores PID com *anti-windup* e o controle por resposta linearizada (*feedback linearization controller*).

3.2.10.4 Controle de Fluxo

A montagem da experiência é idêntica à anterior. Para este experimento um transdutor de pressão é utilizado para medir o fluxo de água na tubulação que enche o tanque de água. Neste experimento o controlador predefinido utilizado é um PID convencional

3.2.10.5 Levitação Magnética

O experimento consiste de um sistema de suspensão de uma esfera magnética (vide Figura 25). O objetivo da experiência é controlar a altura de suspensão da esfera ajustando a corrente elétrica de um eletro-imã através da aplicação de tensão elétrica no mesmo. A posição vertical da esfera é medida por um sensor ótico que somente faz a leitura dos valores verticais, desprezando os valores horizontais. A levitação máxima admitida é *3 cm*. A força magnética utilizada só pode ser atrativa, portanto comandos de corrente negativa são desligados, isto é, não se admite repulsão magnética. Um sistema de segurança desliga o eletro imã se a corrente passar de *3 A*.

Figura 25. Experimento de levitação magnética do ACT.

3.2.10.6 Simulador de Helicóptero

O mais complexo experimento do ACT consiste de um modelo de helicóptero (Humusoft CE 150) conectado a uma base sólida (vide Figura 26). O modelo do helicóptero conta com duas hélices que controlarão o azimute e a elevação do experimento. Os motores CC que giram as hélices são controlados por PWM. Os ângulos de rotação são medidos por sensores IRC.

Figura 26. Experimento de controle de azimute e elevação.

É possível mudar o centro de gravidade do helicóptero através de um peso colocado no eixo horizontal que pode ser movido por um servo motor. Além do controle predefinido de controladores PID, temos controladores LQR chaveados (*Switching LQR Controller*) e neurocontrole preditivo inverso (*Predictive Inverse Neurocontrol*).

3.2.11 RExNet (ALVES, 2005)

RExNet – "Remote Experimentation Network" é um projeto de pesquisa desenvolvido no escopo de um projeto europeu Alfa, o qual visa criar uma rede de experimentação remota. Participam do consórcio RExNet dez instituições, sendo cinco latino americanas e cinco européias. São elas: Universidades Federais de Santa Catarina (UFSC) e do Rio Grande do Sul (UFRGS), do Brasil, Universidade Católica de Temuco (UCT) e PUC do Chile (PUCC), do Chile, Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), do México, Universidade do Porto (UP) e Instituto Politécnico do Porto (IPP), de Portugal, Universidade de Dundee (UD), da Escócia, Universidade de Bremen (UB) e Universidade Técnica de Berlim (TUB), da Alemanha.

Entre os experimentos envolvidos estão alguns já citados como:

- experimento de eletro-pneumática (UB) que utiliza conceitos de *hyper-bond* para combinar experimentos reais com virtuais;
- lógica digital (FEUP) que usa programação de microcontroladores para projetos de filtros digitais e outras aplicações.

E os ainda não citados como:

- experimento de controle de nível que utiliza a planta *Foundation Fieldbus* didática da UFRGS (que será visto na Seção 4.2.1.1) ao qual o este trabalho se fundamenta e o experimento de controle de temperatura (que será visto na Seção 4.2.1.3)
- microservidor *Web* da UFSC utilizado em um experimento que envolve um microcontrolador 80C51 para monitoramento de temperatura remota de baixo custo e com servidor *Web* (acessível na Internet).
- experimentos virtuais para o controle de robôs móveis do ITESM que utiliza linguagem de programação Java para efetuar o controle de trajetória.

O IPP entra em conjunto com a FEUP no experimento de lógica digital. As instituições UCT e PUCC ainda não possuem experimentos remotos e estão mais interessados na tecnologia de ambientes de aprendizagem.

Como forma de cooperação entre os institutos de ensino do consórcio foi feito um estudo de viabilidade de plataformas de aprendizagem e foi estipulada a adequação de todos os institutos à plataforma *MOODLE*.

3.2.12 Outros

O trabalho de Overstreet e Tzes da Universidade Politécnica de Nova Iorque (OVERSTREET, 1999) discute as topologias de acesso dos experimentos remotos (vide Seção 2.1.5) e defende a topologia *fat-client*, pois permite a liberdade de projeto do

controlador. Embora o projeto do controlador da experiência (planta) seja livre a utilização de um *software* específico é necessária, isto é, o usuário remoto projeta (e executa) o controlador em um *software* instalado no cliente. Diferentemente, topologias *thin-client* geralmente limitam o projeto do controlador, que são executadas no servidor da experiência, a possibilidades pré-programadas, onde somente podem-se alterar parâmetros. Mesmo sendo a opção mais flexível do ponto de vista do desenvolvedor/aluno, isso traz alguns problemas como o de licenças para os alunos e restrições de acessos a quem não tem os *softwares* utilizados. Várias publicações defendem a utilização de Java, que já está incorporado na maioria dos navegadores da *Web*, pois é independente de plataforma e sistema operacional (DAVOLI, 2001).

Alternativas para o acesso de clientes a *softwares* do servidor através do Microsoft NetMeeting são propostas pela universidade do Texas (SWAMY, 2002). Por meio deste *software* podem-se compartilhar programas instalados no servidor, isto é, o aluno pode utilizar o programa como se estivesse no seu computador para fazer suas experiências. O controle de acesso se dá através do cadastro de usuários e suas respectivas senhas. Esta solução se prende a plataformas e versões de *software*, que também apresenta fraquezas de segurança, já que dá muita liberdade para o aluno utilizar os *softwares* do servidor. A opção de limitar o acesso, por questões de segurança, também inviabiliza o acesso a usuários que não estão cadastrados, gerando a necessidade de um *software* adicional para cadastro ou de um responsável.

Há também a possibilidade de se enviarem códigos pré-compilados no cliente através de *softwares* que não necessitam de licenças. É o caso do trabalho do Instituto IPN Cinestav, no México, onde alunos descrevem controladores PID e PD para o movimento de robôs em Java e enviam para o servidor, onde esses são simulados em um sistema chamado de Serviço da Web de Robô (*Robot Web Service* – RWS; SALAZAR-SILVA, 1999).

O trabalho de Bruns (2004) também propicia a conversão de esforços virtuais em esforços reais, fazendo com que o experimento se torne bidirecional. Com esta tecnologia pode-se transportar esforços físicos através da Internet com mecanismos de *force feedback* (vide Figura 27) utilizando *hyper-bonds*. Desta maneira o experimento proporciona um trabalho colaborativo distribuído real.

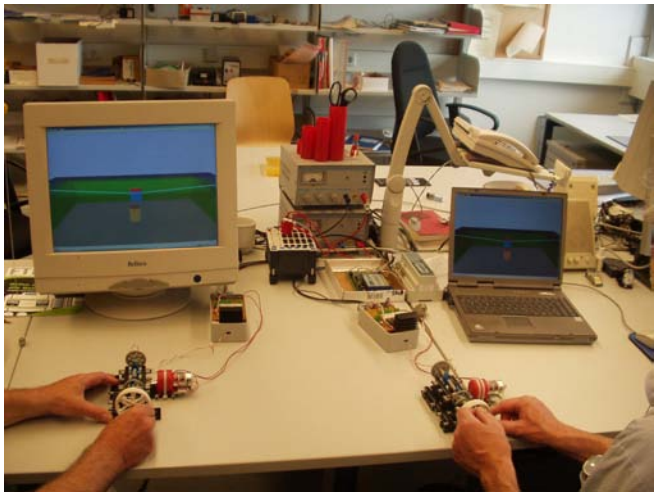

Figura 27. Experimento com mecanismos de force-feedback.

3.3 ANÁLISE DO ESTADO DA ARTE

Como se pode notar nos experimentos citados, são poucas as instituições que buscam integrar aos experimentos remotos com ambientes virtuais de aprendizagem. Muitas vezes isto se deve ao fato das instituições utilizarem os experimentos somente para reforçar aulas teóricas ministradas dentro da Universidade (aula presencial).

Também é notável que a maioria dos experimentos remotos apresentados nesta seção necessita de programas específicos, ou *plug-ins* para o navegador da *Web*, de forma que o usuário visualize e controle o experimento (*fat-client*). Normalmente estes programas têm suporte somente para plataformas *Windows* da *Microsoft*, que é um produto comercial. Há casos das interfaces ainda possuirem dependência do navegador da *Web* além da plataforma

Windows, isto é, necessitam do uso do *Internet Explorer* que também é um produto comercial da *Microsoft* (que são os casos descritos nas seções 3.2.9.1 e 3.2.9.4).

Estas duas características, citadas nos parágrafos anteriores, restringem o uso dos experimentos, isto é, o aluno não possui nenhum embasamento teórico fora da aula presencial (experimento fica restrito ao uso dos alunos presenciais) e precisam se adequar à plataforma e navegador no qual a interface oferece suporte.

A utilização da realidade mista em experiências, com um nível de interatividade grande, como o caso do experimento da Universidade de Bremen (vide Seção 3.2.9.1), representa um progresso nas tradicionais experiências remotas em que o nível de interatividade é reduzido.

Interfaces de experimentos mais sofisticadas ainda comprovam que a utilização de ferramentas coloborativas ligadas à experimentação ampliam a capacidade de aprendizagem em grupo dos alunos, como é o caso da experimento de lógica digital da FEUP (vide Seção 3.2.7.3 e Figura 18).

A gama de experiências demonstradas comprova o fato de que diversas experiências físicas podem ser adaptadas para serem controladas remotamente, desde experiência com elementos químicos (vide Seção 3.2.1.7) passando por sistemas de potência (vide Seção 3.2.4) até redes de comunicação (vide Seção 3.2.6).

Dados os fatos é evidente que a experimentação remota está se encaminhado para criar ambientes mais interativos e colaborativos que tenham materiais educacionais associados às experiências remotas. Tal fato tornaria simples experimentos remotos em cursos que podem ser usados no ensino à distância (EAD) e também como suporte efetivo de aulas presenciais.

4 PROPOSTA DE SISTEMA DE ENSINO PARA AUTOMAÇÃO E CONTROLE USANDO EXPERIMENTOS REMOTOS COM REALIDADE MISTA

4.1 ARQUITETURA PROPOSTA

Com base na análise do estado da arte e nos objetivos descritos neste trabalho, apresentados anteriormente, propõem-se um sistema de ensino para automação e controle, o qual incorpora os seguintes conceitos e qualidades:

- ✓ Experimentação remota (a exemplo do experimento de Zeilmann (2002));
- ✓ Realidade mista (a exemplo do experimento de Bruns (2004));
- ✓ Ambiente computacional para aprendizagem (a exemplo da integração do experimento de Michaelides (2004));
- ✓ Flexibilidade de uso de componentes reais e simulados;
- ✓ Ferramenta para proposta preliminar de sistema tutorial para acompanhamento dos alunos na realização de experimentos e guia de seqüenciamento de material educacional.

Esta arquitetura proposta, portanto, proporcionará um ambiente colaborativo onde o aluno encontrará materiais educacionais organizados, desafios práticos em experiências remotas, futuras análises automáticas do experimento realizado e guia automático de revisão de material educacional, caso os objetivos estabelecidos pelo professor para o experimento não sejam alcançados pelo aluno.

O ambiente computacional de aprendizagem colaborativo é o centro da arquitetura porposta, isto é, todos os outros componentes estão integrados a este, pois o objetivo principal da arquitetura é criar um sistema de ensino. A flexibilidade de usar componentes reais e/ou simulados está diretamente ligada à utilização do experimento, ou seja, a interface integrada do experimento remoto de realidade mista. O controle sobre a configuração da experiência,

entretanto, é repassado ao ambiente computacional de aprendizagem que também pode ser chamado de ambiente virtual de aprendizagem (AVA do inglês *Virtual Learning Environment* – VLE). A configuração da experiência deve ser tal que permita descrever diversos cenários da experiência de realidade mista por meio de parâmetros facilmente controláveis pelo AVA. O AVA também deve possuir controle de acesso para que os alunos, que estejam inscritos nos cursos associados aos experimentos remotos, possam ser identificados. Esta identificação é de extrema importância, pois somente alunos inscritos nos cursos terão direito a executar experiências remotas, devido à unificação do controle de acesso do experimento ao AVA.

Ferramentas de análise de experiências e sistema tutor também estarão diretamente ligadas ao AVA. Esta conexão deve ser tal que os dados do analisador e do sistema tutorial estejam sempre disponíveis ao AVA que os gerenciará e manipulará de acordo com o objetivo da experiência.

Devido a utilização de diversos conceitos o desenvolvimento da arquitetura se divide em módulos, cada um associado a um conceito. A Figura 28 mostra a interação entre os módulos da arquitetura proposta e a interação do aluno com o experimento através do ambiente virtual de aprendizado.

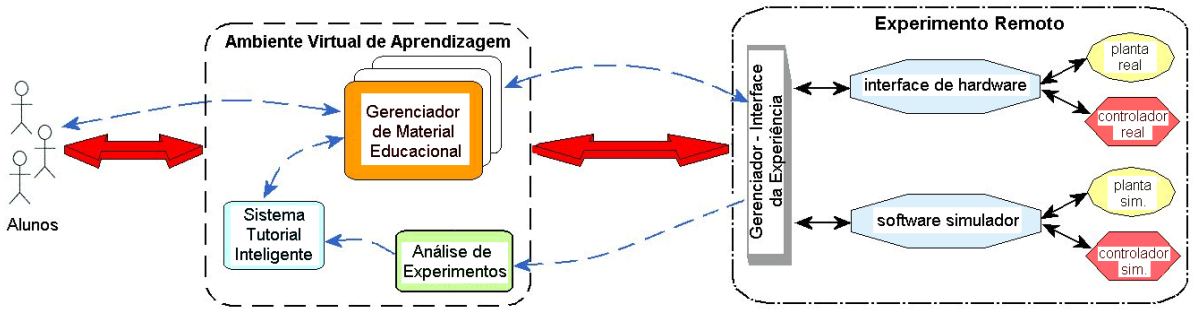

Figura 28. Diagrama de interação da arquitetura proposta.

Como pode-se notar na Figura 28, os alunos têm acesso ao experimento através do AVA que comporta o gerenciador de material educacional, o sistema tutorial e o analisador de

experimentos. Todos os módulos componentes da arquitetura possuem interface com o banco de dados central localizado no servidor (vide Figura 29). Assim sendo, variáveis do banco de dados comum são responsáveis pela transmissão de comandos e parâmetros de um módulo para o outro. A Figura 29 mostra as interfaces de comunicação usadas na interface *Web* da arquitetura proposta, o acesso dos alunos é feito através da Internet (também por LAN ou WAN) por um navegador *Web* com suporte ao JRE da *Sun Microsystems*. Através do AVA, que usa o banco de dados central como interface de comunicação com os outros módulos, é possivel criar uma configuração de experiência remota. *Applets Java* que se comunicam com o gerenciador da experiência são responsáveis pela representação de elementos gráficos na interface *Web* da experiência remota, como gráficos e demais mostradores (*displays*) de variáveis.

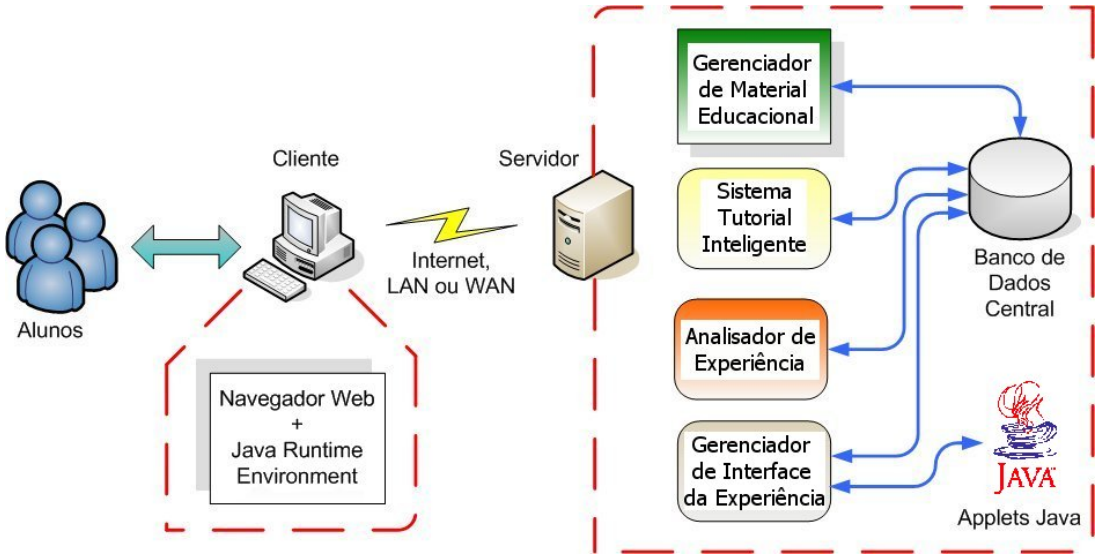

Figura 29. Diagrama das interfaces de comunicação dos módulos da arquitetura.

Alguns aspectos da arquitetura são extensões de projetos anteriores da UFRGS. O consórcio *RExNet* também é responsável por estudos descritos neste trabalho, realizados por parceiros, visando aprimorar tecnologicamente e pedagogicamente todos os experimentos do consórcio de forma a proporcionar uma utilização colaborativa dos mesmos.

A seguir uma descrição detalhada da especificação de cada módulo desenvolvido no âmbito da arquitetura será apresentada.

4.2 DESCRIÇÃO DOS MÓDULOS DESENVOLVIDOS

Para esclarecer melhor os aspectos que foram desenvolvidos na arquitetura proposta, analisaremos cada um separadamente.

4.2.1 Incorporação de Experimentos Remotos em Ambientes Virtuais de Aprendizagem

A incorporação de experimentos remotos em *AVA*s ainda não se tornou popular (vide Seção 3.2), apesar dos experimentos remotos possuírem em sua maioria interfaces baseadas na *Web* e destas interfaces facilitarem a integração em qualquer tipo de AVA. Esta impopularidade é causada pelo fato dos experimentos remotos desenvolvidos servirem apenas de reforço a aulas teóricas presenciais e não estarem inseridas em cursos completos de educação à distância.

No âmbito deste trabalho serão utilizados diversos experimentos remotos que estão divididos em dois grupos: experimentos discretos e contínuos, mais especificamente experimentos com variáveis puramente *booleanas* (0 ou 1) e variáveis do domínio real. Os objetivos dos experimentos estão ligados à linha de pesquisa ao qual este trabalho está relacionado, isto é, sistemas de automação e controle. Portanto os experimentos buscam fixar a teoria através do projeto prático de controladores para automatizar sistemas. Nos sistemas contínuos são utilizados controladores PID (Proporcional Integral Derivativo) enquanto nos discretos (com valores *booleanos*) controladores baseados em lógicas para *CLP* (Controlador Lógico Programável). Vale lembrar que em *CLP*s também pode-se manipular valores reais discretizados e por este motivo implementar controladores PID dentro de *CLP*s.

Os experimentos remotos, demonstrados no estado da arte deste trabalho, propiciam diversas vantagens que motivam o desenvolvimento e a implantação em instituições de

ensino. Entre estas vantagens pode-se citar: a maior disponibilidade devido a flexibilidade temporal e espacial, a disponibilidade de equipamentos caros (muitas vezes industriais) aos alunos e maior segurança dos equipamentos (devido à restrição de uso e implementação de limites para variáveis manipuladas e de processo).

Embora a utilização remota propicie aos alunos a SBBT, estes experimentos ainda não possuem o mesmo peso educacional que laboratórios presenciais. Portanto para a utilização de experimentos remotos no ensino é necessário que estes estejam associados a materiais educacionais cuidadosamente desenvolvidos para surtir o melhor efeito de aprendizagem possível. O aprendizado é um processo ativo que depende de interação e este fato deve ser aproveitado nos materiais educacionais desenvolvidos. No modelo de interação aluno/tutor/curso de Laurillard (1993) existem quatro processos de interações: (i) a transferência de conhecimento e discussão; (ii) reflexão; (iii) montagem do experimento, e subseqüente adaptação; e (iv) interações no nível de experiências reais. Desta maneira os cursos desenvolvidos devem respeitar estes processos de construção do conhecimento e proporcionar as interações mencionadas de cada processo em um ambiente virtual colaborativo de aprendizagem.

Além de proporcionar uma plataforma de desenvolvimento de cursos de ensino à distância, os *AVA*s incentivam o aprendizado colaborativo, ativo e autodidata. Estes conceitos de aprendizado são de extrema importância quando se possibilita flexibilidade temporal e espacial, já que os tutores (professores ou não) não estarão acompanhando os alunos da mesma forma que nos cursos presenciais tradicionais (disciplinas).

A incorporação de experimentos remotos em ambientes virtuais de aprendizagem é altamente benéfica para o presente trabalho devido às diversas afirmações citadas anteriormente. Para o desenvolvimento de *AVA*s optamos por uma ferramenta flexível e, ao mesmo tempo, livre de licenças comerciais a partir de um estudo das funcionalidades e

capacidades dos ambientes computacionais de aprendizagem (e também mencionados como *AVA*s) existentes (vide seção 2.2). O *MOODLE* (vide Seção 2.2.5) e o *Claroline* (vide Seção 2.2.4) cumprem esses requisitos. O *MOODLE*, além desses requisitos, é pedagogicamente específico com diversos trabalhos relacionados para aumentar sua capacidade de transferência de conhecimento. O consórcio *RExNet*, do qual a UFRGS faz parte, também analisou características das plataformas de aprendizagem (ALVES, 2005) e optou pelo *MOODLE*.

Assim, todo o material educacional referente ao experimento remoto deve estar organizado no interior do ambiente *MOODLE* para que os alunos não tenham dificuldade em buscar fundamento teórico para a experiência prática a ser realizada. Cursos separados serão criados de acordo com seu propósito de ensino. Inicialmente, pode-se definir quatro cursos básicos válidos para os experimentos referentes a linha de pesquisa deste trabalho (automação e sistemas de controle):

- Utilização dos Experimentos:

  Curso que ilustra os passos (estilo tutorial) necessários para se efetuar o experimento remoto em questão. Por exemplo: (i) como iniciar o experimento; (ii) tipos de experimentos existentes; (iii) objetivos dos experimentos; etc.

- Funcionamento dos Experimentos:

  Curso que descreve de forma clara e didática como os experimentos funcionam e quais equipamentos, materiais e tecnologias foram utilizadas para sua construção.

- Controladores PID:

  Curso contendo materiais educacionais referentes aos controladores proporcionais, integrais e derivativos com tópicos de conceitos básicos, construções e métodos de ajuste. Este material é importante, pois os controladores mais utilizados para processos dinâmicos são do tipo PID.

- Norma IEC 61131-3:

  Curso contendo materiais educacionais referentes à norma de padronização de linguagens de programação para sistemas de automação internacional IEC 61131-3. Neste curso serão apresentados as cinco linguagens de programação utilizadas em dispositivos industriais de automação como *CLP*s.

O *MOODLE* possui módulos pré-configurados de fórums (repositório de mensagens), questionários e *Wiki*s (*What I Know Is*) o que facilita a interação entre alunos, tutores e professores, assim como ajudará o aperfeiçoamento dos materiais educacionais publicados no ambiente. Os *Wiki*s são bases de conhecimento construídas (publicadas) colaborativamente de fácil utilização, isto é, cada integrante do curso pode publicar conhecimentos referentes ao curso em questão.

O ambiente virtual selecionado ainda possui código aberto (escrito em PHP) o que facilita eventuais alterações que venham a acrescentar mais "customização" e futuras integrações de módulos externos. A linguagem de programação na qual o ambiente está escrito é versátil permitindo interação e execução de *softwares* externos no lado do servidor, isto não limita o desenvolvimento de novos programas escritos em outras linguagens de programação. A visualização de recursos da *Web* externos ao AVA também é possível dentro do AVA utilizando *frames*.

Toda a base de dados do *MOODLE* está armazenada em bancos de dados o que também é muito importante para fornecer uma interface comum para incorporação com os experimentos remotos, que também possuem esta interface. Este *software* também possui controle de acesso e registro de atividades de usuários (*logs*). Desta maneira é possível rastrear toda a navegação do usuário (aluno) e identificar que materiais educacionais foram visitados e possíveis interações com estes materiais, como envio de mensagens, respotas a questionários, atividades em *Wikis*, etc. Sistemas de controle de acesso das experiências

remotas incorporadas (através de direcionamento de endereços da *Web* externos) podem ser integrados utilizando somente o cadastro do usuário no *MOODLE*. Como a interface dos experimentos remotos utilizados nesta arquitetura se baseia na *Web* a incorporação é de fácil implementação.

4.2.2 Integração de Componentes Reais e Simulados

Geralmente, segundo Auer (2003) os experimentos são: somente simulados, no caso de laboratórios virtuais, ou reais, no caso de laboratórios remotos (vide Tabela 1). Uma combinação ou mistura entre os dois tipos é chamada de laboratórios (ou experimentos) de realidade mista. Portanto, basta termos equipamentos reais interagindo com virtuais (simulados) em uma mesma experiência para o experimento seja classificado com de realidade mista (vide Seção 2.1.4).

A integração de componentes reais e simulados, proposta nesta arquitetura, possibilitará a criação de "novos experimentos" (novas configurações de experimentos) que possibilitam o que pode-se chamar de "componentes intercambiáveis" (*interchangeable components*). Este intercâmbio de componentes deve ser desenvolvido de forma que fique transparente para os usuários, aumentando assim a flexibilidade para criação de experimentos didáticos a partir de diversos componentes diferentes de experimentos separados.

4.2.2.1 Componentes Intercambiáveis

Os componentes intercambiáveis estão diretamente associados aos experimentos de realidade mista (vide Seção 2.1.4) dando mais flexibilidade ("configurabilidade") às combinações de simulações (equipamentos virtuais) com equipamentos reais. A flexibilidade pode aproveitar vantagens clássicas dos simuladores (vide Seção 2.1.2.1), como o controle do tempo de simulação para execução passo a passo, que possuem grande valor didático para alunos iniciantes, e do realismo da experiência física (SBBT). Experimentos com realidade mista proporcionam diversas vantagens na aprendizagem e na acessibilidade de usuários

remotos e os componentes intercambiáveis aumentam ainda mais a didática das experiências através da criação de cenários de diferentes configurações de experimentos.

Os componentes de um experimento, que faz parte da linha de pesquisa de sistemas de automação e controle, são chamados de planta e controlador. A planta é o sistema ou processo que se deseja automatizar e o controlador é parte do sistema de automação responsável pelo controle dos atuadores do processo. Como exemplo simples, pode-se ter uma estação de tratamento de água como planta e como controlador um *CLP* responsável pelo acionamento de diversas bombas e válvulas de fluxo (atuadores), que estão ligadas a diferentes tanques de reagentes químicos da estação de tratamento. O controle de acionamento dos atuadores segue lógicas programadas no controlador com base em leituras de sensores do processo (planta). As grandezas físicas captadas (lidas) pelos sensores (transdutores) são comumente chamadas de variáveis do processo (PV – *Process Variable*) enquanto as grandezas que estão relacionadas a atuadores são chamadas de variáveis manipuladas (MV – *Manipulated Variable*). Chama-se um processo de automatizado (controlado) aquele que possui um controlador associado a uma planta.

De forma a esclarecer os diversos cenários que podem ser criados com as diferentes combinações dos componentes intercambiáveis, serão apresentados quatro cenários distintos que podem ser desenvolvidos a partir de um único experimento hipotético.

1º. Cenário

Ambos os componentes (controlador e planta) são simulados por modelos computacionais que representam os comportamentos de interesse dos componentes envolvidos (vide Figura 30). Este cenário tem o intuito de demonstrar características que não podem ser concebidas em experiências com equipamentos reais. O aluno pode tanto observar leis de controle passo a passo como visualizar mudanças também passo a passo de variáveis

de processo na planta. A partir desta observação teorias de controle podem ser mais facilmente entendidas por estudantes iniciantes do curso.

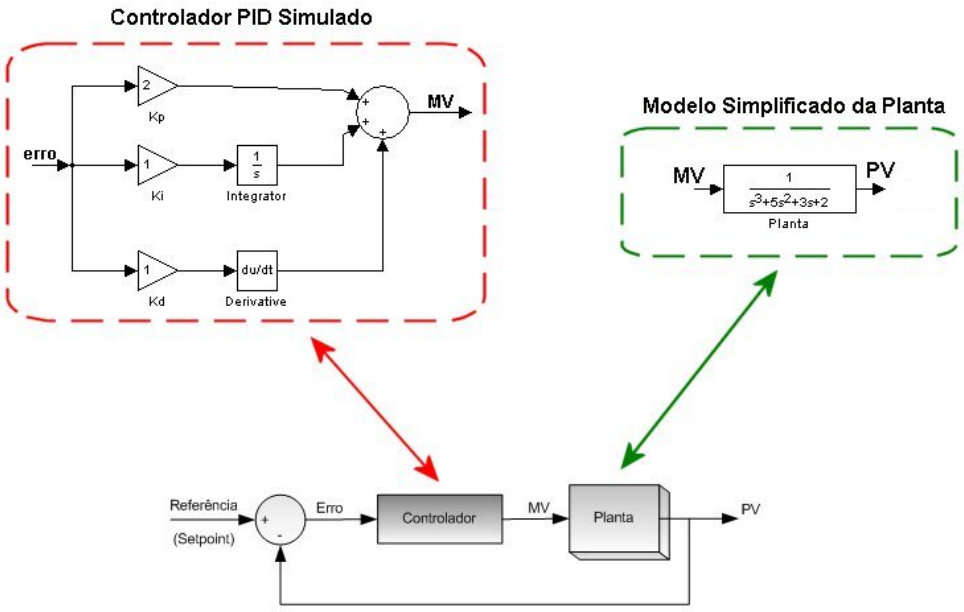

Figura 30. Cenário somente com componentes simulados.

Na Figura 30 temos um exemplo hipotético de um sistema representado pelo diagrama de blocos de controle que utiliza um controlador PID (com $K_p = 2$, $K_i = 1$ e $K_d = 1$) e uma planta de comportamento bastante simplificado.

Este cenário simplifica o experimento idealizando comportamentos tanto do controlador como da planta, isto é, os modelos utilizados na simulação dos componentes não são afetados por perturbações de nenhuma natureza (são idealizados), fato este que não ocorre em equipamentos reais. É evidente que seria plenamente possível incorporarmos modelos mais detalhados e mais realistas (fidedignos) e simulá-los, pois a utilização de simulações não necessariamente implica no uso de modelos idealizados, mas este não é o objetivo da utilização de simulações neste cenário. Não linearidades da planta e atrasos provenientes do tempo de resposta do controlador são exemplos de perturbações desprezadas (ignoradas) nos modelos computacionais usados neste cenário. É importante ressaltar que esta simplificação

somente é benéfica para alunos iniciantes fazendo com que estes não percam o interesse. Para alunos que já passaram por esta fase inicial experimentos mais sofisticados como o segundo, terceiro e quarto cenário são mais indicados e trarão mais resultados educacionais para os alunos.

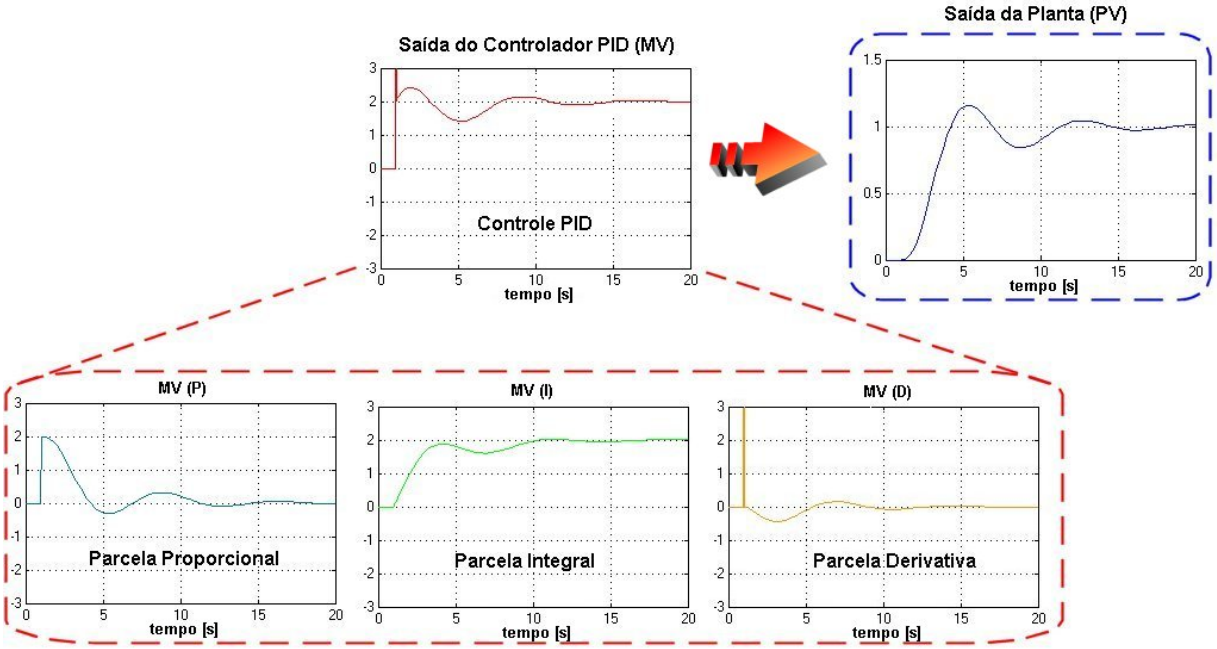

Figura 31. Gráficos do cenário com componentes simulados.

Nos gráficos da Figura 31 pode-se visualizar a resposta ao salto (*step response*) do controlador (vide Figura 30) e também as contribuições de cada parcela do PID (K_p, K_i e K_d) para controlar a variável manipulada do sistema (MV). A saída resultante do controle (saída da planta – PV) esta à direita. Nota-se que salto ocorre instantaneamente no tempo 1 (eixo das abscissas) e por este motivo a parcela derivativa neste instante é um impulso infinito. O impulso infinito só pode ser concebido em ambientes simulados.

Evidentemente que no modelo de controlador PID pode-se também ter combinações como controladores somente proporcionais, somente integrais, somente derivativos, proporcionais integrais, proporcionais derivativos e integrais derivativos. Para se simular estas

combinações do PID, basta fazer com que os parâmetros das características P, I e D associadas à combinação desejada representem em valores a ação de cada parcela. Por exemplo, se quisermos um controlador PI basta atribuirmos valores não-nulos aos parâmetros K_p e K_i e valor nulo (zero) ao parâmetro K_d. Deste modo a ação derivativa fica anulada.

2º. Cenário

O componente controlador é simulado e a planta é real (vide Figura 32). Este cenário tem o intuito de demonstrar como um controlador simulado ideal interagiria com uma planta real que possui diversas características inerentes de sua construção e dos materiais e equipamentos utilizados. A partir deste cenário os estudantes podem perceber a dificuldade de controlar processos reais e os cuidados que devem ser tomados na configuração do controlador e no tratamento dos sinais dos sensores (reais). Para exemplificar o cenário usamos uma planta real hipotética de um motor de corrente contínua.

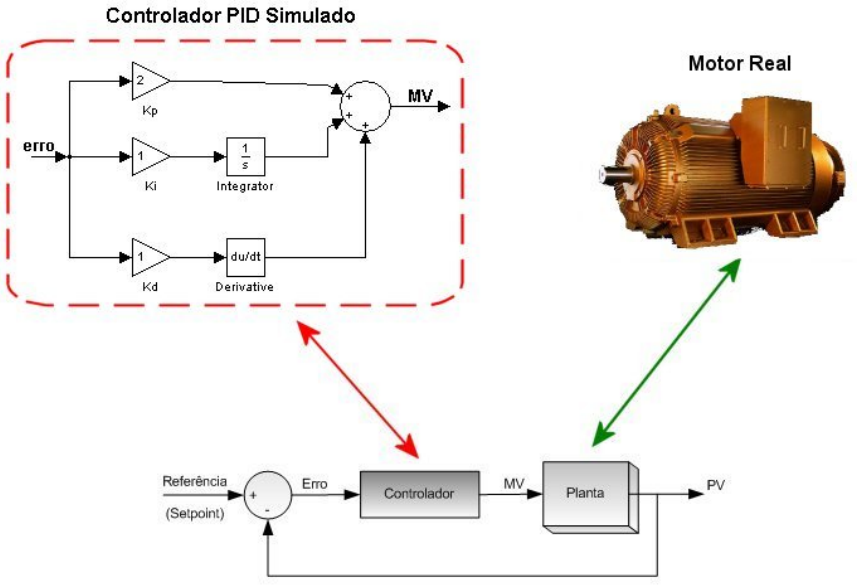

Figura 32. Cenário com controlador simulado e planta real.

Como pode-se ver na Figura 32 o diagrama de blocos de controle utiliza um controlador PID ideal (como o do primeiro cenário) e a planta tem um comportamento real. O

motor CC possui comportamentos influenciados por inércia e atrito da carga como também a resistência e indutância do rotor. Estes fatores do comportamento real criam um cenário ideal para o estudo de plantas reais utilizadas na indústria. A variável manipulada do controlador é a tensão aplicada nos terminais do rotor do motor CC e a variável do processo é a rotação do eixo do motor (velocidade) medida através de um tacômetro. Com o aumento da tensão aplicada o eixo do motor começa a se movimentar e as dinâmicas do processo se alteram levemente, pois não temos mais influência do atrito estático e sim do atrito dinâmico.

Este cenário é utilizado por alunos que já passaram pelo experimento inicial do primeiro cenário. No decorrer do experimento os alunos notarão a diferença entre uma planta real e uma simulada, pois os sinais dos sensores provocarão muito mais oscilações (impulsos) nas parcelas derivativas do controlador.

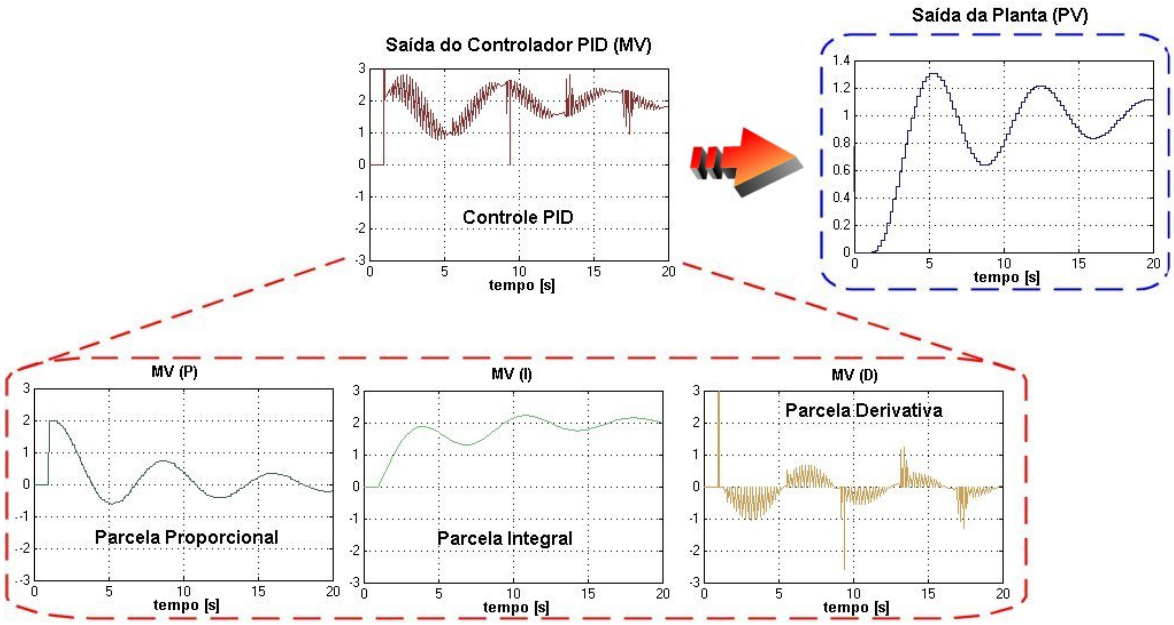

Figura 33. Gráficos do cenário com planta real e controlador simulado.

A Figura 33 ilustra os gráficos deste cenário. Pode-se notar que todas as variáveis envolvidas no processo parecem estar “discretizadas”, embora a saída do controlador PID seja contínua ela efetua a lei de controle baseada na entrada (PV) que é “discretizada”. A variável

do processo é “discretizada” devido a construção do sistema de aquisição da planta, isto é, embora a variável PV seja contínua o sistema de aquisição (com conversor/amostrador A/D) ligado ao gerenciador da experiência disponibiliza os dados (PV) ao controlador discretamente. As parcelas proporcionas e integrais sofrem pouca “interferência”. O gráfico da saída da planta também ilustra que o sistema tem um tempo de estabilização maior que no primeiro cenário com os mesmos parâmetros PID.

3º. Cenário

O componente controlador é real e a planta é simulada (vide Figura 34). Este cenário tem o intuito de demonstrar como um controlador real interagiria com uma planta simulada. A partir deste cenário os estudantes podem perceber a dificuldade de se projetar controladores e os cuidados que devem ser tomados na configuração do controlador e no tratamento dos sinais dos sensores.

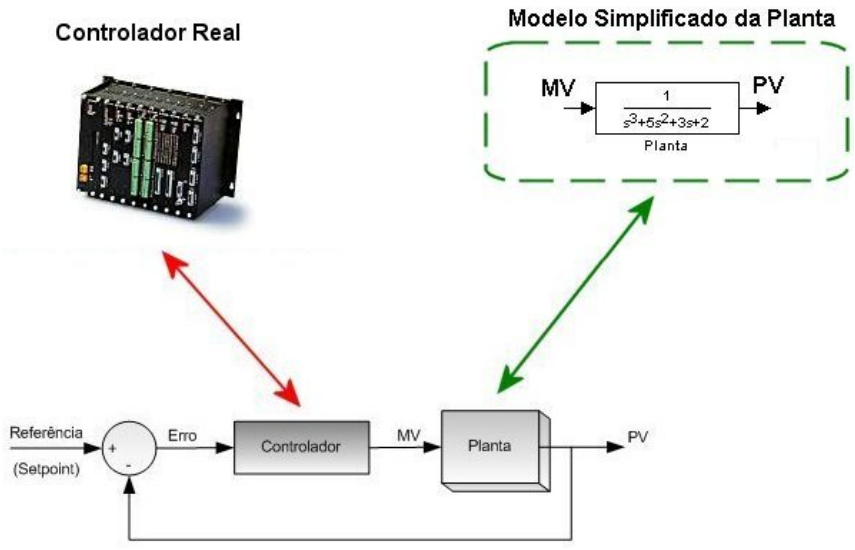

Figura 34. Cenário com controlador real e planta simulada.

Apesar da configuração da experiência parecer simples a implementação de rotinas para integração e derivação de sinais em controladores reais é bastante complexa e os parâmetros usados nas simulações dos cenários anteriores geralmente apresentam diversas

diferenças visíveis quando usados em controladores reais (podem perder a estabilidade). Resta ao aluno identificar estas diferenças e com base nos materiais educacionais adaptar os parâmetros para serem usados em implementações reais de controladores.

A Figura 35 ilustra os gráficos de resposta ao salto para controladores PI. Nota-se que o sistema tem um tempo de estabilização bem maior que os cenários anteriores e que característica derivativa não foi usada.

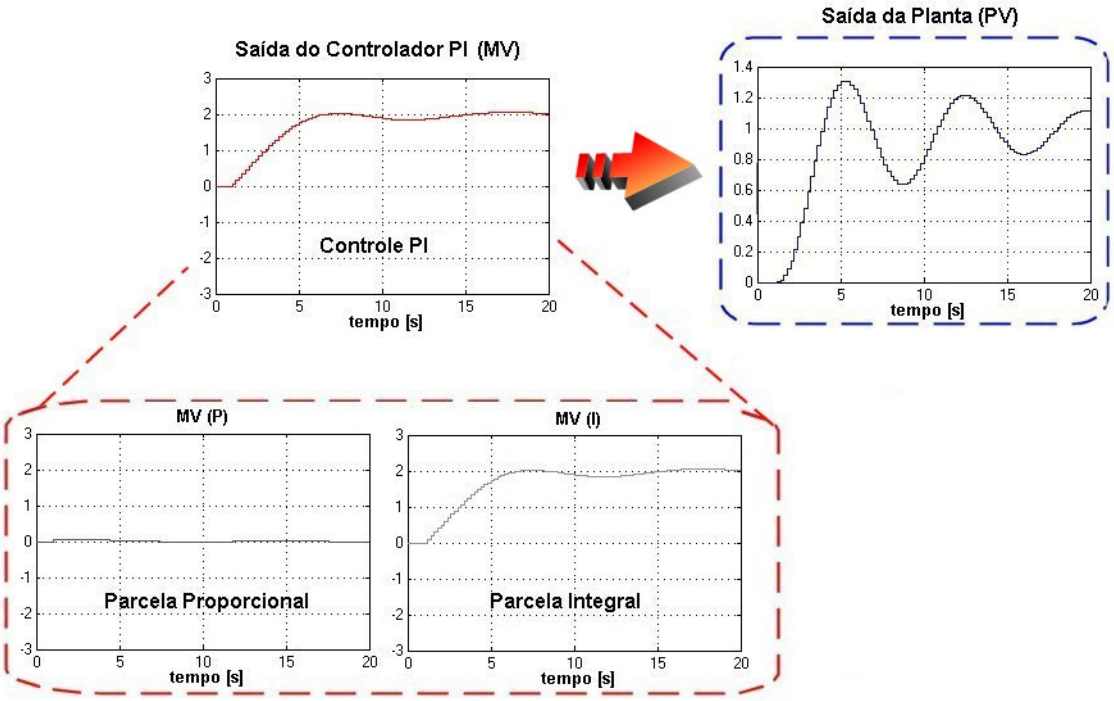

Figura 35. Gráficos do cenário com planta simulada e controlador real.

4º. Cenário

Ambos os componentes (controlador e planta) são reais (vide Figura 36) e, portanto, o experimento é real. Este cenário tem o intuito de demonstrar como um sistema real funciona quando em ambos os componentes estão presentes perturbações, não linearidades, atrasos, histereses, etc, resultantes da arquitetura do experimento real.

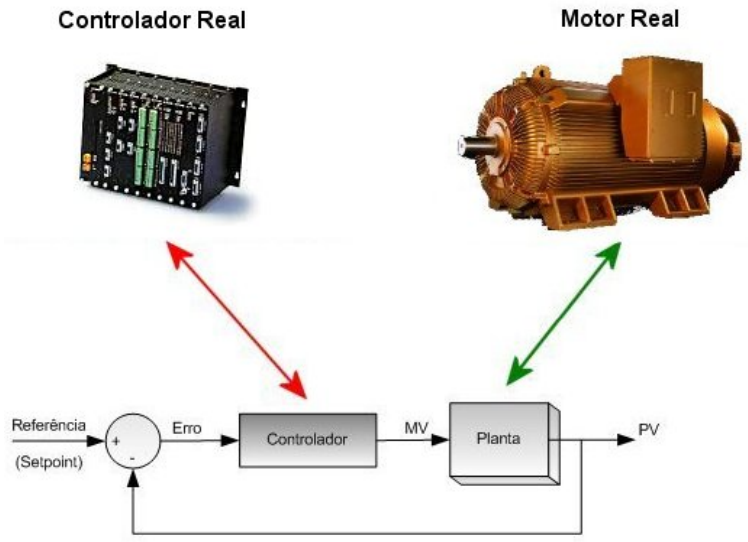

Figura 36. Cenário com controlador simulado e planta real.

A Figura 36 ilustra a configuração do cenário onde temos ambos os componentes reais utilizados nos cenários anteriores. Esta configuração possibilita aos estudantes usufruir de equipamentos usados na indústria através de uma interface SBBT e com isso percebem a dificuldade (complexidade) de controlar processos reais via *Web* e os cuidados que devem ser tomados para a segurança dos equipamentos envolvidos.

Os alunos que podem executar este tipo de experiência já passaram por todos os outros cenários e estão aptos a executar uma experiência real. A experiência real é o passo final na construção do conhecimento do aluno sobre o experimento, e por conseqüência sobre a teoria de automação e controle na qual o experimento está inserido.

4.2.2.2 Idealização da Arquitetura dos Componentes Intercambiáveis

Embora o tenhamos citado, anteriormente, nos quatro cenários o controlador PID, o tipo de controle pode ser o mais variado possível desde que possamos ter tanto uma implementação simulada quanto uma real para desta forma oferecer aos alunos os mesmos cenários apresentados anteriormente. Controles como o *Bang-bang* (com relés para acionamento *ON – OFF*) também são de grande valia educacional, pois estes também são largamente utilizados na indústria.

A Figura 37 ilustra as interações demonstradas nos quatro cenários apresentados de em um único diagrama. Vale ressaltar que o sistema de automação, referido na figura, também foi referido como controlador anteriormente.

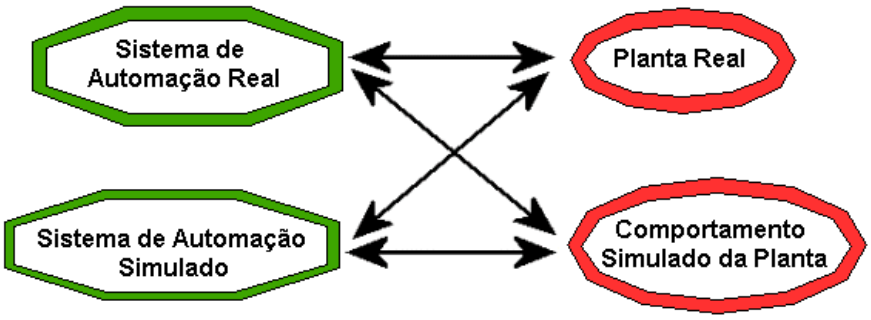

Figura 37. Interações dos componentes intercambiáveis.

Combinações entre os cenários também são possíveis. Pode-se usar, por exemplo, uma implementação de um controlador derivativo simulado e uma implementação de um controlador proporcional integral formando um controlador misto (real e simulado) interagindo com uma planta qualquer em um cenário alternativo.

Assim, resta desenvolver uma interface simples para comunicação entre os componentes intercambiáveis que seja ao mesmo tempo flexível e que abranja grande parte da dos simuladores existentes com tecnologia de comunicação entre *softwares*. A partir de um estudo minucioso de diversos simuladores existentes no mercado (vide Apêndice – Simuladores) e das tecnologias de comunicação entre diferentes *softwares* optou-se pela interface OPC. A interface de comunicação OPC é simples, segue a arquitetura cliente servidor onde o servidor publica os valores do processo a que está associado para qualquer cliente interessado, e é largamente difundida na indústria tanto de simuladores quanto de sistemas SCADA. Apesar desta interface estar intimamente ligada às plataformas *Windows* da *Microsoft* há um novo padrão em desenvolvimento, o *OPC-XML* (*OPC Foundation*), que possibilitará interação com outros sistemas operacionais.

O intercâmbio entre os componentes deve acontecer de forma dinâmica, isto é, um componente real pode ser trocado por um de comportamento simulado, sem necessidade de reprojeto ou modificação do sistema real. Apesar chamarmos a flexibilidade de “dinâmica” não há sentido em modificar uma experiência durante o seu andamento, isto é, uma vez configurada a experiência para usar um sistema simulado não há porque trocá-lo por um sistema real “dinamicamente”. Este fato, embora interessante, geraria problemas de transporte de variáveis de sistemas reais para simulados. Portanto a flexibilidade dinâmica referida neste trabalho refere-se à configuração inicial do cenário do experimento. Por exemplo, caso o aluno deseje testar um sistema de automação simulado conectado a uma planta real, nenhuma modificação na construção do experimento é necessária, apenas uma seleção (endereçamento) de entradas e saídas deve ser efetuada no gerenciador da experiência. Mesmo assim durante a experiência o aluno não poderá mudar os componentes envolvidos, isto é, não poderá trocar uma planta real por uma simulada muito menos um sistema de automação simulado para um real. Depois de encerrada a experiência outras, com diferentes componentes, poderão ser iniciadas somente mudando-se a seleção das entradas e saídas.

Para gerenciar a experiência utiliza-se um *software SCADA* que geralmente possui uma infinidade de interfaces de comunicação, entre elas as mais comuns são o *OPC* e os bancos de dados *ODBC* e *MySQL*. A gerência da experiência inclui a configuração dos cenários, portanto o sistema *SCADA* será o *software* que ligará o *AVA* à experiência remota. A Figura 38 ilustra as interações entre os componentes mostrados nos cenários através da interface *OPC* e as interações entre o *AVA* e o sistema *SCADA*.

A implementação do gerenciador de experiência (SCADA) deve ser tal que o ambiente virtual de aprendizagem consiga iniciar e executar uma experiência através da simples passagem de parâmetros (configuração da experiência) via banco de dados que serão responsáveis pela seleção dos componentes utilizados. A configuração da experiência é

transparente ao usuário que não terá conhecimento do interfaceamento dos componentes nem das conexões dos equipamentos reais e simulados usados no cenário.

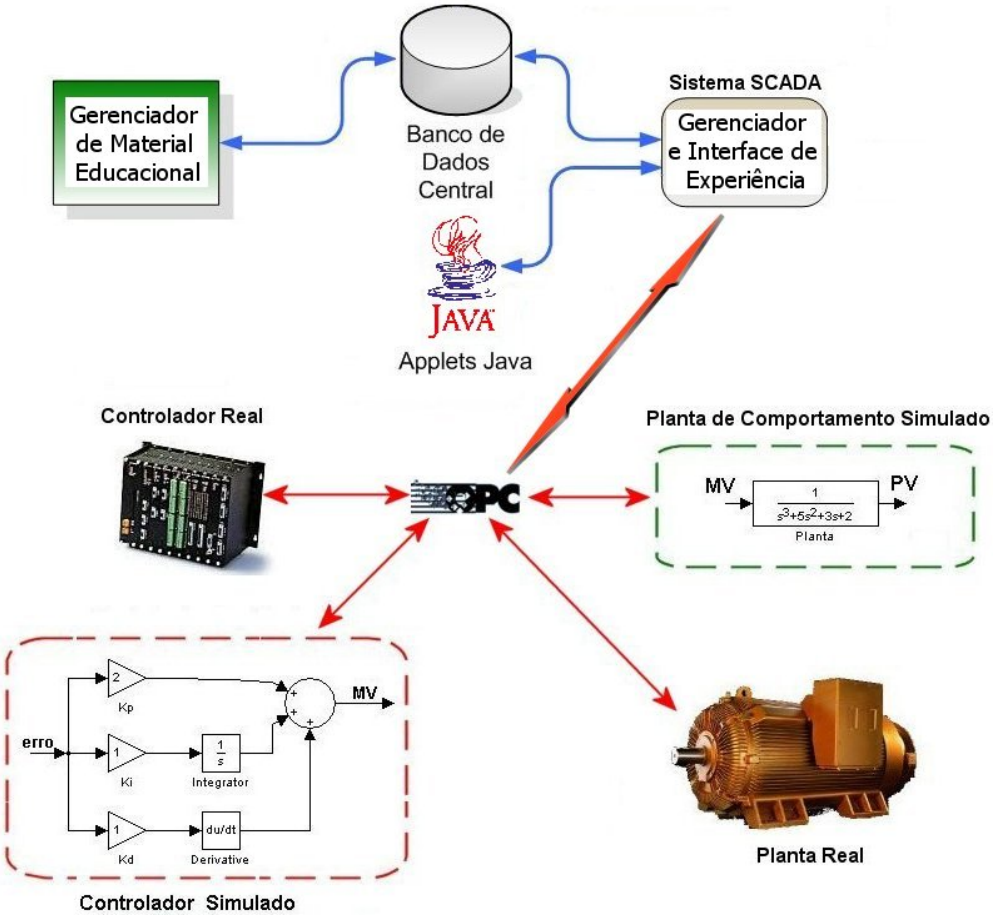

Figura 38. Arquitetura de comunicação de componentes intercambiáveis

A Figura 39 ilustra o fluxo de dados entre o aluno, o AVA o banco de dados e o gerenciador de experiência para que a integração dos componentes reais e simulados seja possível. Nota-se que a seleção dos componentes é feita através da passagem de parâmetros do AVA para o gerenciador da experiência (SCADA). O SCADA por sua vez possui uma programação (scripts internos) que funcionam de forma a selecionar o(s) componente(s) que deve(m) ser conectado(s) ao experimento ligando diferentes variáveis de diferentes componentes conectados ao SCADA através da interface OPC.

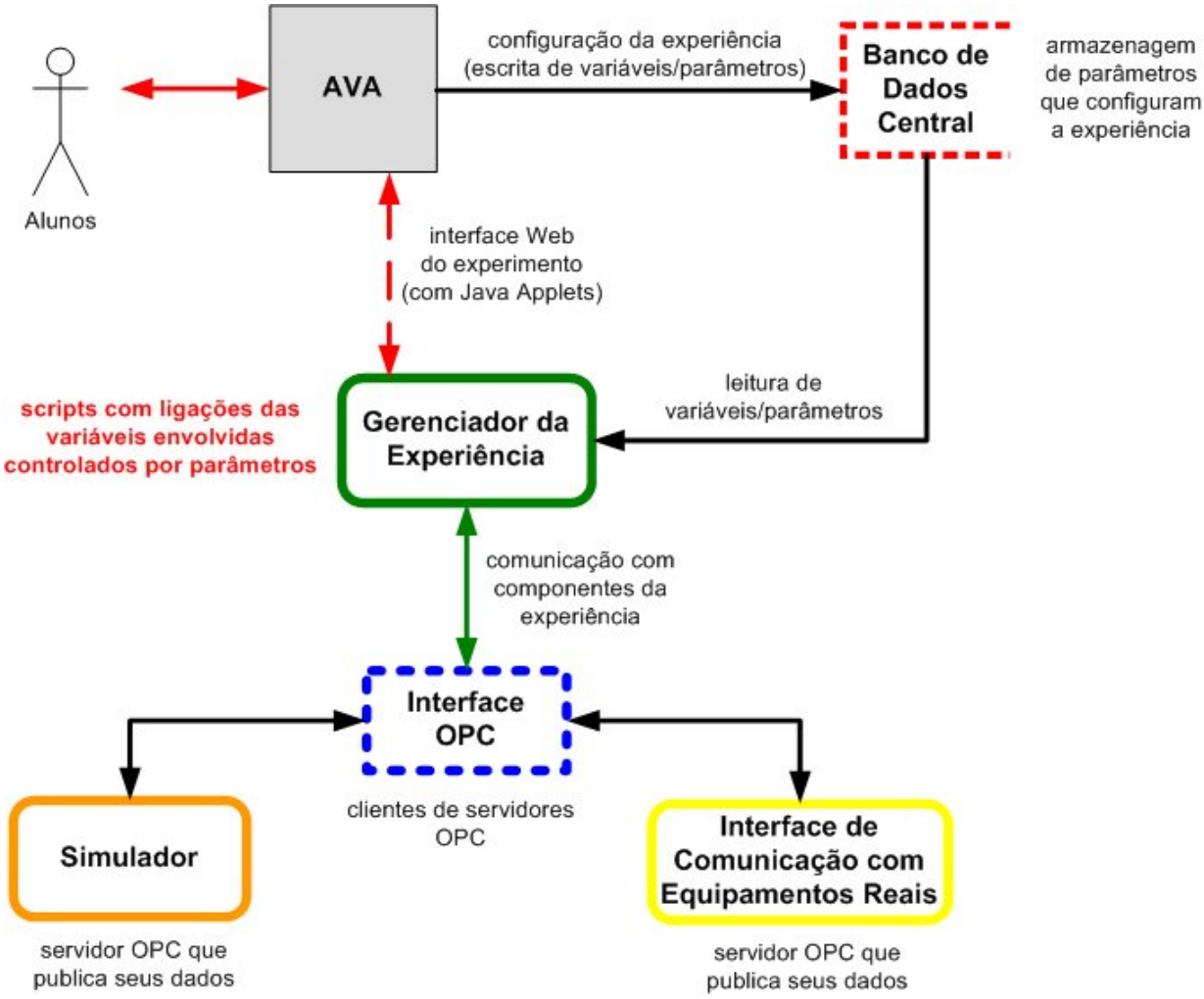

Figura 39. Fluxo de dados para criação experiência segundo arquitetura.

4.2.3 Proposta Preliminar de Sistema Tutorial

Precisamos projetar um sistema integrado no ambiente virtual de aprendizagem *MOODLE* que analise os experimentos realizados pelos alunos e verifique se os resultados obtidos atendem às especificações propostas. Em caso negativo, o sistema deverá indicar ao aluno quais os conceitos teóricos devem ser revisados, a fim de sanarem-se os problemas (aprendizagem falha) encontrados. No caso dos experimentos de controle com sistemas contínuos, a análise se resume a métricas como sobrepasso ou sobresinal (“overshoot”), tempo de subida (“rise time”) e tempo de acomodação (“settling time”). Em casos de sistemas discretos apenas com variáveis *booleanas* onde geralmente a seqüência dos processos é

essencial, um simples analisador de seqüência e controle de tempo é suficiente para analisar a experiência.

Com a intenção de incorporar estas análises na arquitetura, um *software* que se comunique através da interface de banco de dados será desenvolvido. Para guiar o aluno, um sistema tutorial também se faz necessário. Resultados da experiência (relatório) podem ser interpretados pelo analisador da experiência que possui informações necessárias para auxiliar o aluno a conduzir a experiência ao seu objetivo principal com especificações previamente atribuídas pelos tutores.

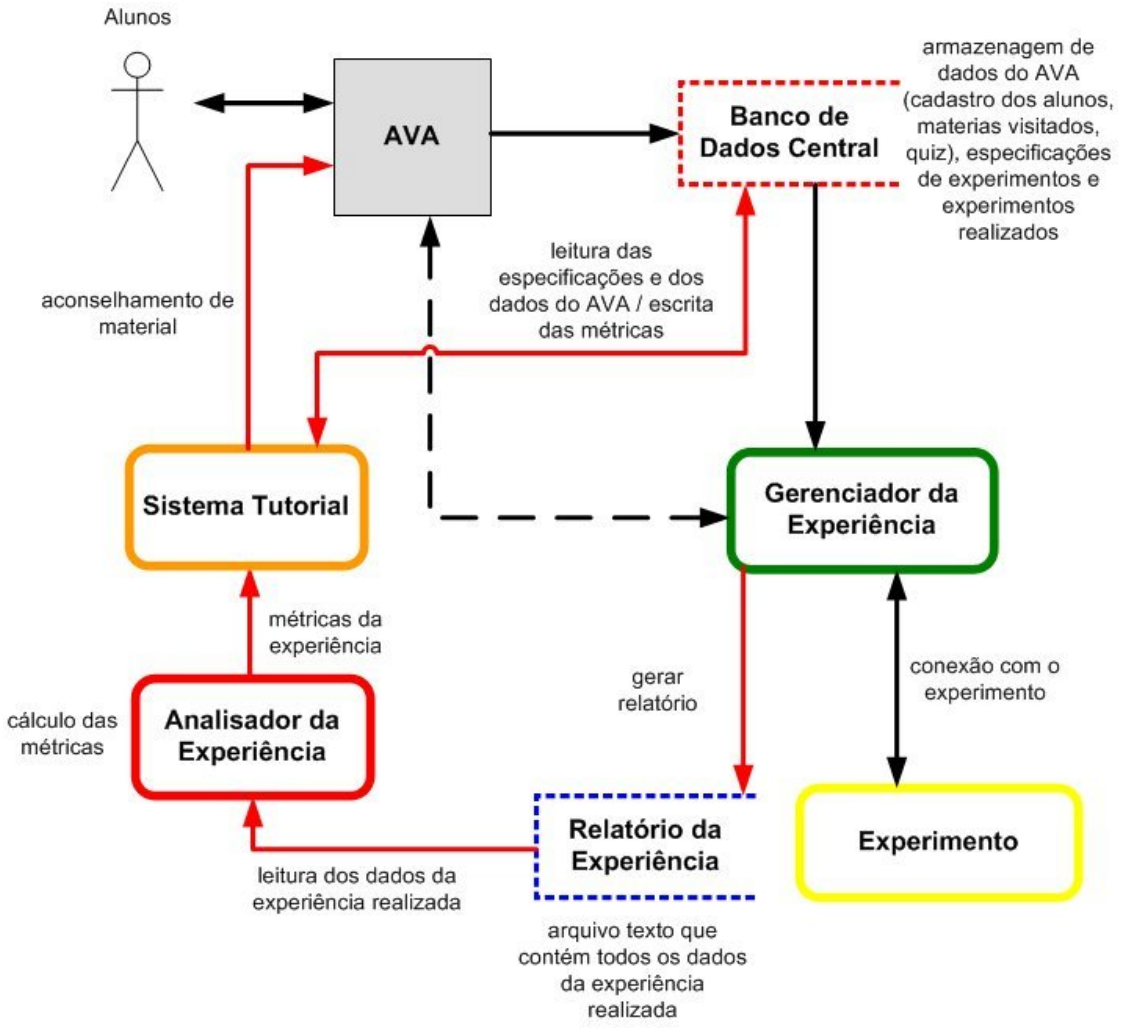

Figura 40. Fluxo de dados para o aconselhamento do sistema tutorial

A Figura 40 ilustra o fluxo de dados entre o AVA, o analisador de experiência, o sistema tutorial e o gerenciador de experiência para que a incorporação do sistema tutorial seja possível. Nota-se que é preciso que o SCADA gere um relatório da experiência e com base neste e nos dados armazenados no banco de dados central o sistema tutorial aconselha o aluno a visitas novos materiais dentro do AVA.

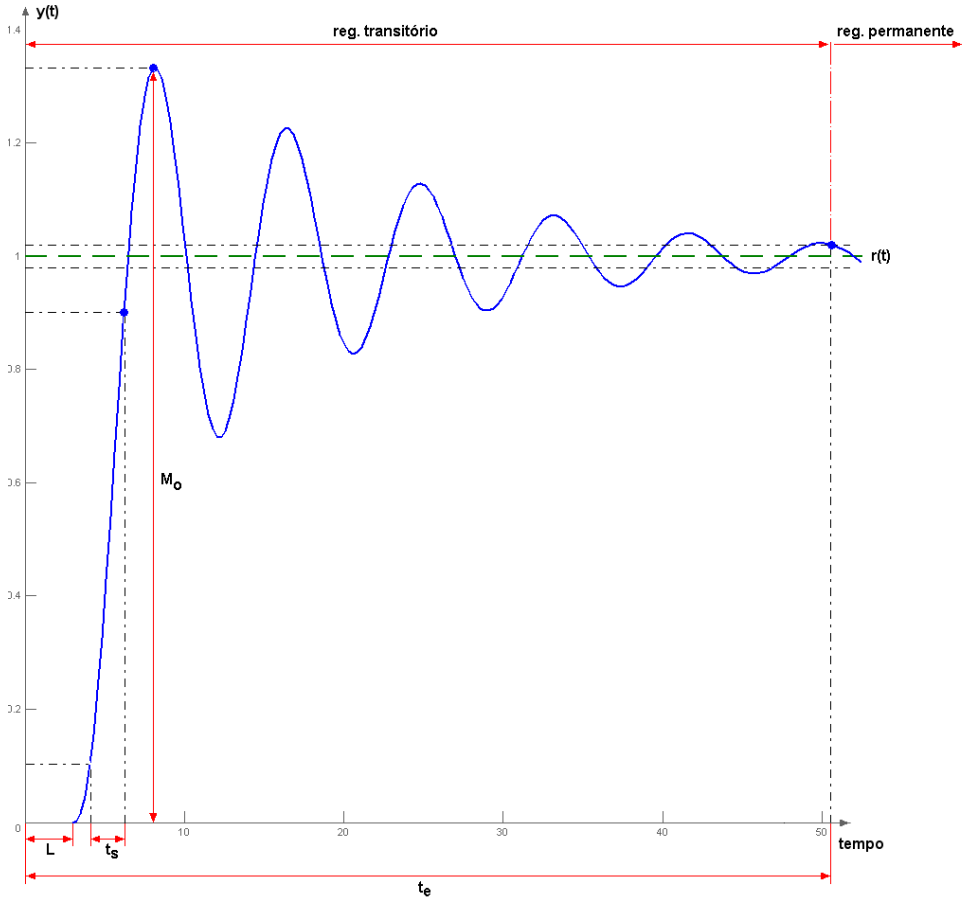

Figura 41. Gráfico das métricas do analisador.

A Figura 41 ilustra as regras para cálculo das métricas do analisador de experiência para sistemas contínuos onde, r(t) é a referência (*setpoint*, em verde tracejado), y(t) a saída do sistema (PV, em azul), M_o o sobrepasso máximo, t_s o tempo de subida e t_e o tempo de estabilização ou acomodação. O tempo de subida é calculado através da diferença entre o tempo em que o sistema atingiu 90% do valor do *setpoint* e o tempo em que atingiu 10%. Já o

tempo de estabilização é o tempo decorrido desde o início do experimento até sua entrada em regime permanente. Diz-se que o sistema está em regime permanente quando este oscila dentro de limites seguros (tipicamente de 2 a 5% do sinal de referência). Os limites de sobrepasso máximo, tempo de subida e tempo de estabilização podem ser configurados pelos professores/tutores do experimento segundo seus objetivos educacionais. A Figura 41 também ilustra o atraso de transporte, L. que é o tempo transcorrido até que o sinal de referência (*setpoint* do controlador) reflita diretamente na variável do processo (PV).

O *software* analisador de experiência proposto calculará as métricas automaticamente e estas serão analisadas e comparadas com limites pré-estabelecidos pelos professores/tutores, que estarão armazenados no banco de dados central, por outro *software* chamado de guia tutorial inteligente. Este *software* também terá como função rastrear (pesquisar) os materiais educacionais visitados pelo aluno (no banco de dados do AVA) e aconselhar (sugestão) a leitura de outros materiais baseados nos resultados da experiência e na pesquisa efetuada. Este aconselhamento é o que chama-se de *feedback*, ou resposta do sistema ao experimento realizado. O AVA escolhido possui registro de diversas interações inclusive de questionários (*quiz*) o que facilita a identificação das falhas na aprendizagem. Caso o aluno não tenha respondido nenhum questionário o sistema tutorial primeiro aconselhará o aluno a responder questões referentes ao experimento. As informações deste questionário “cruzadas” com as métricas do analisador de experiência indicarão que material educacional deve ser sugerido ao aluno. O material educacional sugerido, que não está nos materiais normais do curso, é elaborado de forma que aborde as falhas detectadas e indique maneiras de melhorar o rendimento do experimento.

A Figura 42 ilustra todas as interações dos módulos da arquitetura proposta e a demonstra o que é disponibilizado ao aluno através do AVA.

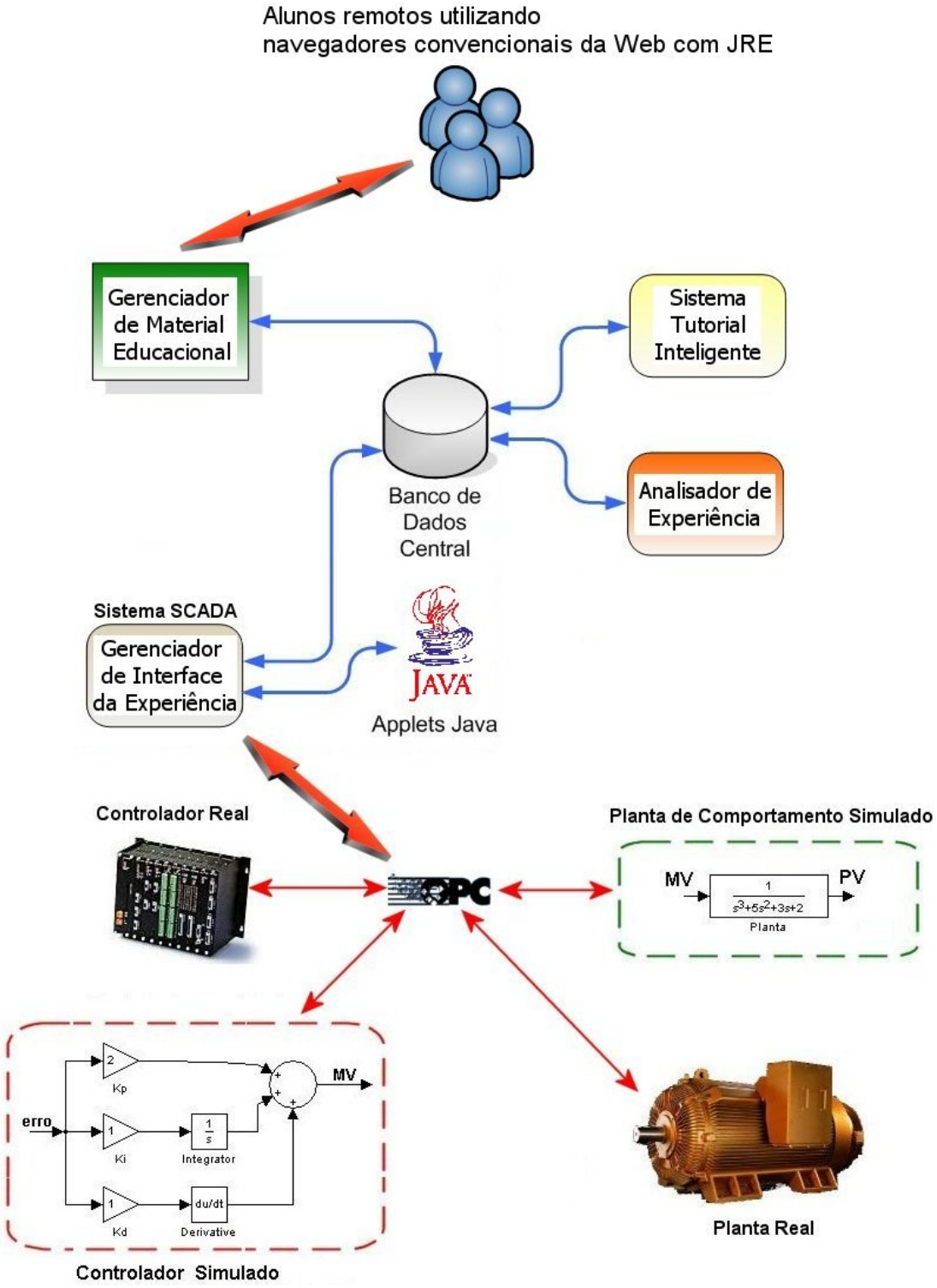

Figura 42. Idealização da Arquitetura Proposta.

5 VALIDAÇÃO DA ARQUITETURA ATRAVÉS DE ESTUDOS DE CASO

5.1 INTRODUÇÃO

A validação será dividida em estudos de caso que demonstram a arquitetura utilizada. Serão citadas referências à arquitetura e às tecnologias envolvidas para interligar o trabalho aos conceitos teóricos previamente analisados. Para a construção dos estudos de caso foram utilizados diversos trabalhos e tecnologias existentes, entretanto, a contribuição desta dissertação não se resume meramente ao emprego destes trabalhos e tecnologias.

Os estudos de caso apresentados estão acessíveis no LASCAR (Laboratório de Sistemas de Controle, Automação e Robótica) da UFRGS. São eles: Planta Térmica, Planta Foundation Fieldbus, Experimento Remoto de Mecatrônica e Planta de Envasilhamento de Garrafas. A seguir estes experimentos serão descritos assim como as particularidades de cada um analisadas de acordo com a arquitetura proposta.

Uma característica importante da composição destes experimentos está na possibilidade de serem combinados de acordo com a flexibilidade proposta pelos componentes intercambiáveis. Por este motivo, após ser descrito cada experimento, serão apresentados protótipos (combinações entre experimentos) que são implementações da arquitetura proposta neste trabalho.

Testes e estudos de caso usando sistemas simulados e sistemas SCADA foram projetados usando três *softwares* comerciais: o *ISaGRAF* (ISAGRAF, 2006) da empresa *ICS Triplex*, o *Elipse SCADA* da empresa brasileira Elipse (ELIPSE, 2006) e o *Relés* desenvolvido pelo Núcleo de Educação à Distância (NEAD) do SENAI-RS. Para facilitar o entendimento dos estudos de caso uma seção de descrição destes *softwares* precederá a apresentação dos estudos de caso.

5.2 DESCRIÇÃO DOS SOFTWARES UTILIZADOS NOS ESTUDOS DE CASO

5.2.1 ISaGRAF

O ISaGRAF (ISAGRAF, 2006) é um *software* de simulação baseado nas linguagens de programação da norma IEC 61131 para automação de sistemas. As lógicas de programação são simuladas como se estivessem sendo executadas em *CLP*s (Controlador Lógico Programável). O *software* também possui recursos para simular a execução das lógicas de programação como se estivesse sendo executado em um *CLP* real (*hardware*), este recurso é chamado de *softPLC*, ou seja, um *CLP* em *software*. O *ISaGRAF* é um produto comercial da *ICS Triplex* que além de simular sistemas pode ser usado como um *software* supervisório (SCADA) em sistemas de controle.

Este simulador utiliza seis linguagens de programação: cinco abrangidas na IEC 61131-3 (vide Tabela 4) e o *Flow Chart* (diagrama de fluxo). Por este motivo é bastante utilizado na simulação e programação de *CLP*s para controles locais ou distribuídos (*DCS*). A idéia básica de utilizar um simulador que utilize seis linguagens de programação é combiná-las produzindo uma lógica que possibilite tirar vantagens de cada linguagem. É obvio também que a padronização de linguagens é muito importante no desenvolvimento de um projeto que segue normas internacionais.

As linguagens de programação vão desde simples linguagem de relés (*Ladder*) até texto estruturado que é uma linguagem que se assemelha ao Pascal. Programações concorrentes também podem ser utilizadas usando-se o *SFC*. Blocos de funções mais elaboradas como funções integrais e derivativas são usadas no *FBD*. Todas as linguagens usam as mesmas variáveis fazendo com que linguagens diferentes executem em conjunto uma lógica de programação.

Tabela 4. Linguagens de programação da IEC 61131-3.

Linguagem	Abreviação	Tipos de Aplicações
Sequential Function Chart (diagrama de funções seqüencial)	SFC	processos seqüenciais
Function Block Diagram (diagrama de blocos funcionais)	FBD	fluxos de processos
Ladder Diagram (diagrama de relés)	LD	fluxos elétricos
Structured Text (texto estruturado)	ST	matemáticas e textuais
Instruction List (lista de instruções)	IL	Booleanas, simples e textuais

Entre as funcionalidades mais desejadas do *software* estão o servidor OPC-DA, a integração com bancos de dados e comunicações com dispositivos externos. Estas funcionalidades dão ao *ISaGRAF* qualificações para ser usado como supervisórios *SCADA* ou *softDCS* (*Software Distributed Control Systems*) e não como meros simuladores de CLP ou *softPLC*s. Uma descrição mais aprofundada do *software* se encontra no Apêndice de Simuladores neste trabalho.

5.2.1.1 CLP Reais, Simulados e SoftPLCs

Como o próprio nome sugere, o *CLP* real é um sistema físico que é empregado na indústria como forma de automatizar processos industriais. Ele conta com diversas *I/O*s, controladas (lidas e escritas) pelo programa em execução no seu microprocessador. Trabalha com variáveis discretas (digitais) e foi projetado para substituir circuitos de controle compostos por diversos relés, contactoras e temporizadores. Medições analógicas são digitalizadas por conversores para serem utilizadas nos controladores.

*CLP*s reais são soluções ideais para controlar processos industriais flexíveis que envolvem grande complexidade apesar de seu preço elevado. Pode efetuar técnicas de

controle mais avançadas como controladores PID e alguns até trabalhar com processamento de imagens (visão de máquina).

A lógica programada em um *CLP* pode ser simulada (emulada) por qualquer *software* simulador simples que possa reproduzir a programação, embora esta lógica destoe da execução de *CLP*s reais. A diferença básica entre um *CLP* simulado e um *softPLC* é sua fidelidade na simulação do equipamento real. Um *softPLC* é um *CLP* em *software*, que desempenha, porém, exatamente os mesmos comportamentos de um *hardware* (no caso o *CLP* real). Isso somente é possível em ambientes com restrições temporais rígidas, como em sistemas operacionais de tempo real (RTOS). O *ISaGRAF* possibilita geração de alvos de execução tempo real para *RTOS*, ou, ainda, admite execução de *softPLC* através de seu módulo de *Runtime* (vide ISaGRAF no Apêndice – Simuladores).

5.2.2 Elipse SCADA

O *Elipse* SCADA (ELIPSE, 2006) não é tecnicamente um *software* simulador, mas pode ser programado para simular alguns processos, assim como interagir com simuladores externos. Desenvolvido para suprir todos os requisitos de um *software* supervisório SCADA, o *Elipse* combina uma ferramenta gráfica de edição com uma linguagem de programação bastante simples baseada em *scripts*.

A linguagem de programação utilizada é orientada a eventos (*event driven*), chamada de *Elipse Basic* (baseada no *Basic*). Entradas de dados como “cliques de mouse” e mudanças de variáveis ativam *scripts* que são programados nos objetos da ação.

Como pode ser visto na Figura 43, a janela *Organizer* do *Elipse SCADA* é que gerencia todos os elementos envolvidos no supervisório. As variáveis internas são chamadas de “tags” e podem interagir com outras variáveis externas de bancos de dados (*databases*), servidores OPC (*OPCServers*), e variáveis descritas em *drivers* para dispositivos de automação. No *Organizer* clientes de servidores OPC e bancos de dados são configurados

para serem usados nos *scripts* do *Elipse* como variáveis para serem manipuladas. O *software* permite a monitoração de sistemas através de recursos de captura, registro e transmissão digital de imagens (*Watcher*).

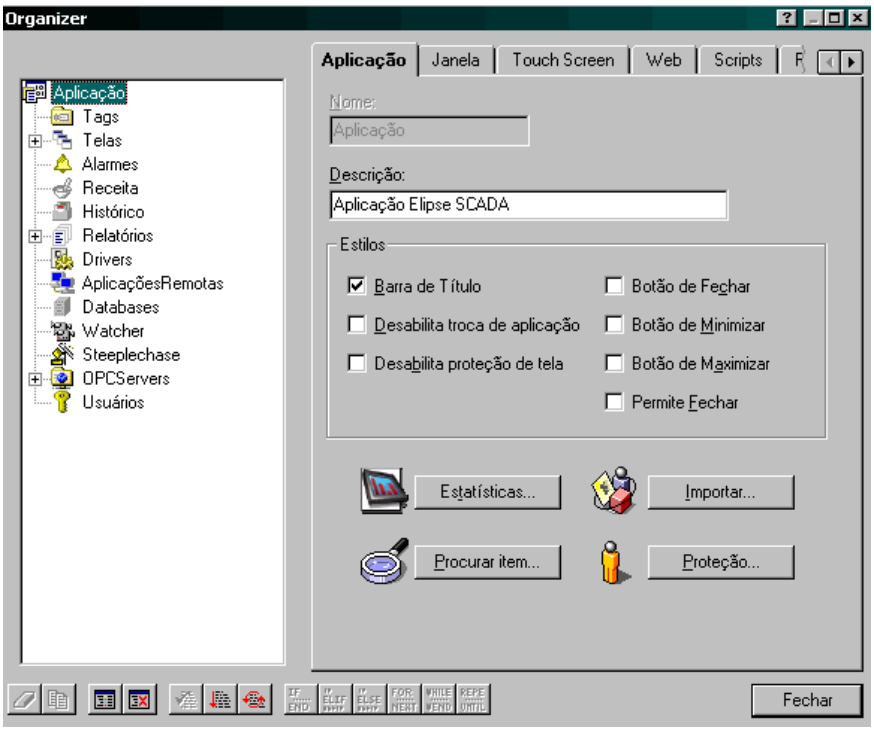

Figura 43. Elipse SCADA “Organizer”.

Outro recurso muito utilizado em casos de interfaces remotas é chamado de *Web* onde a supervisão de processos através da *Internet* é possível. Neste recurso é gerado um *Applet Java* incorporado em uma página HTML que disponibiliza a visualização da interface do supervisório com os dados locais para usuários remotos. Desta forma basta que o computador onde está rodando o supervisório possua um *software* servidor (exemplo Apache) e conexão com a Internet para que usuários remotos possam se conectar-se a uma estação de supervisão, utilizando qualquer navegador com suporte de execução de aplicativos Java (JRE). A interface (Applet Java) gerada não aceita entradas dados, assim para passamos dados ao supervisório necessitamos de um banco de dados acessível remotamente. Desta forma

repassamos os dados para o supervisório indiretamente através do banco de dados que é escrito remotamente.

A monitoração é característica dos softwares supervisórios, portanto a geração de relatórios e históricos é essencial para guardarmos informações sobre um ensaio realizado.

5.2.3 Relés - SENAI

O *software Relés* foi desenvolvido pelo Núcleo de Educação à Distância (NEAD) do SENAI-RS. Este simulador bastante simples busca demonstrar aos alunos a linguagem de relé (*Ladder*) para programação em *CLP*s, que é a mais difundida entre os técnicos eletricistas. Os elementos de programação são simples contatos, bobinas, memórias e temporizadores. O objetivo principal do uso do simulador é demonstrar a funcionalidade da linguagem relé.

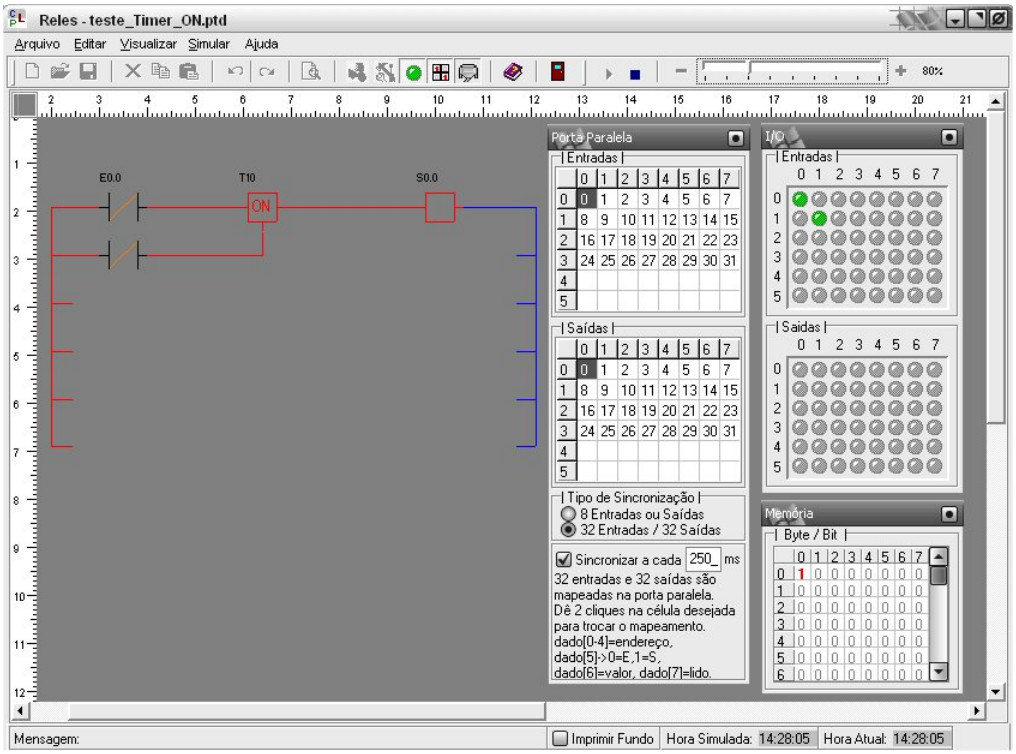

Figura 44. Interface do simulador Relés.

A partir do projeto de Laboratórios Remotos do SENAI algumas modificações no simulador foram propostas para proporcionar uma interface com *hardware*. A interface criada possibilita a comunicação com equipamentos reais através da porta paralela de um computador do tipo PC, isto é, as entradas e saídas do simulador podem ser direcionadas diretamente para os 8 bits de dados da porta paralela. Um protocolo de comunicação bastante simples possibilita o controle de 32 I/Os através de combinações dos 8 bits de dados da porta paralela (vide Tabela 5).

Tabela 5. Protocolo de 32 I/Os pela porta paralela.

Bits de Dados							
D0	D1	D2	D3	D4	D5	D6	D7
Endereço da entrada ou saída (de 0 a 31)					Seletor de Entrada ou Saída	Valor da Entrada ou Saída	ACK

O protocolo criado utiliza os cinco primeiros bits para endereçamento das 32 possíveis entradas e saídas. É importante notar que temos 32 entradas e 32 saídas endereçadas. O bit 5 sinaliza se o dado é uma entrada ou saída e o bit 6 indica o valor deste dado. O último bit é de controle irá sinalizar se o dado foi compreendido (lido) ou se o dado é novo. Desta forma é bastante simples desenvolver uma interagir (lendo e escrevendo) com *hardware* ou *software*.

5.3 PLANTA FOUNDATION FIELDBUS

Este experimento foi desenvolvido no âmbito da dissertação de mestrado de Rafael Zeilmann (ZEILMANN, 2001; ZEILMANN, 2002; ZEILMANN, 2003) no PPGEE da UFRGS, e será estendido de forma a se adequar a um estudo de caso da arquitetura para validação desta dissertação.

De forma a tornar mais clara a descrição deste estudo de caso esta seção será dividida em tópicos como: equipamentos, objetivo do experimento, arquitetura, interface remota e logo

após as incorporações desenvolvidas. Os outros estudos de caso também seguirão este modelo de apresentação.

5.3.1 Equipamentos

O experimento consiste em uma planta de controle de nível multi-variável composta por dois tanques de água, três moto-bombas e diversos sensores de pressão e válvulas posicionadoras pneumáticas (vide Figura 45). Todos os equipamentos utilizados possuem interface de comunicação *Foundation Fieldbus* (*FF*; Foundation Fieldbus, 2006). As válvulas (PY302), sensores de pressão (LD302), conversores 4-20mA (FI302) e interface de controle de processos para PC (PCI302) usados no experimento foram fabricados pela empresa SMAR (SMAR, 2006), uma empresa brasileira. Esta montagem típica de equipamentos é muitas vezes chamada de planta piloto didática *Foundation Fieldbus*.

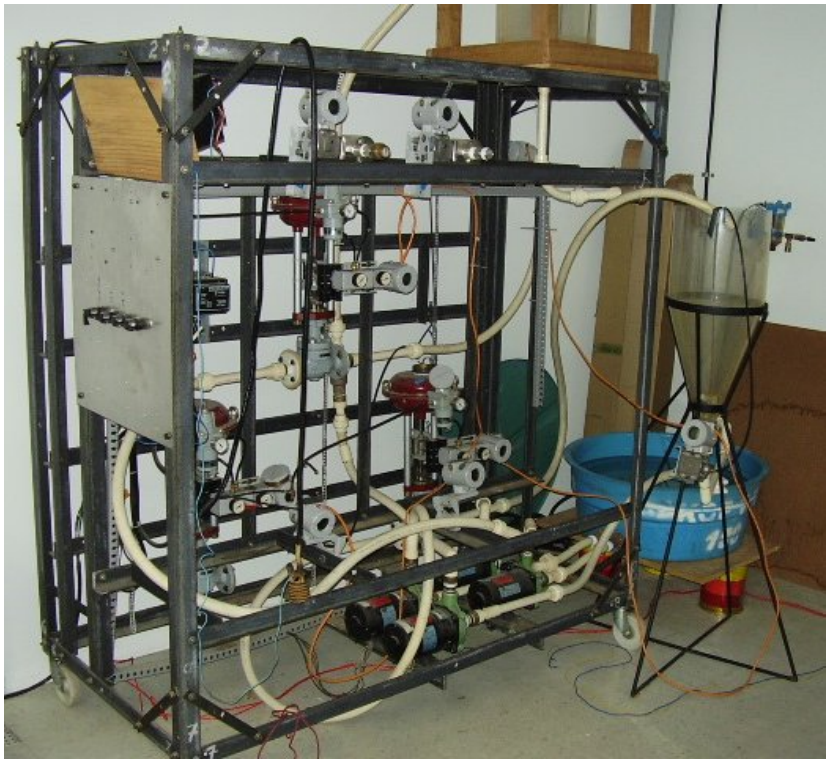

Figura 45. Foto da construção da atual Planta Didática FF da UFRGS.

Equipamentos FF são dotados de “inteligência”, isto é, podem efetuar controle sem a necessidade de dispositivos para controle (*CLP*s ou outros controladores industriais) ou de um *software* de controle externo (SCADA). Desta forma, a rede de dispositivos se torna uma FCS (*Fieldbus Control Systems*) em que a inteligência do sistema está distribuída entre seus dispositivos (vide Figura 46).

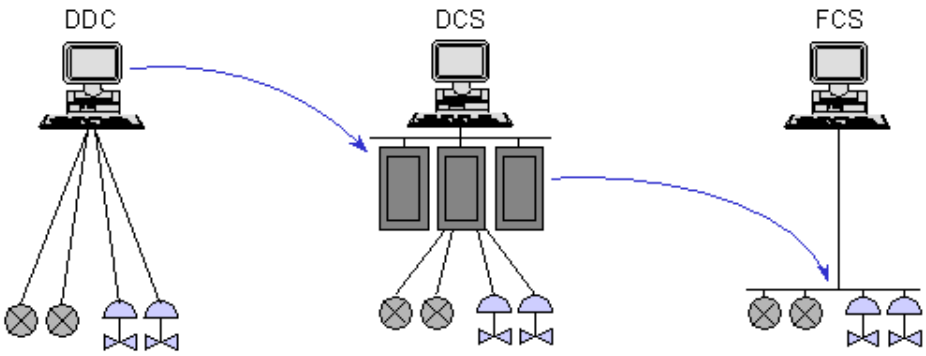

Figura 46. Tecnologias de redes industriais.

5.3.2 Objetivo do Experimento

O objetivo do experimento é controlar o nível de um ou dois tanques de água através da leitura de sensores de pressão (usados para determinar o nível de água) e da escrita em válvulas posicionadoras (atuadores). O tanque 1 está montado em um ponto mais elevado que o tanque 2, e ambos estão conectados (interligados), como mostra a Figura 47. As moto-bombas enchem os tanques independentemente, sendo que o tanque 1 enche também o tanque 2 pela ação da gravidade.

No experimento definido, um controlador PID é usado para executar o sistema de controle (automação do processo). A tarefa a ser desenvolvida pelos alunos consiste na parametrização do controlador PID e observação do comportamento do sistema de controle em malha fechada. O ajuste do controlador se dá através dos parâmetros proporcionais (P),

integrais (I) e derivativos (D). Estes parâmetros são enviados aos dispositivos que efetuam a lógica de controle automaticamente.

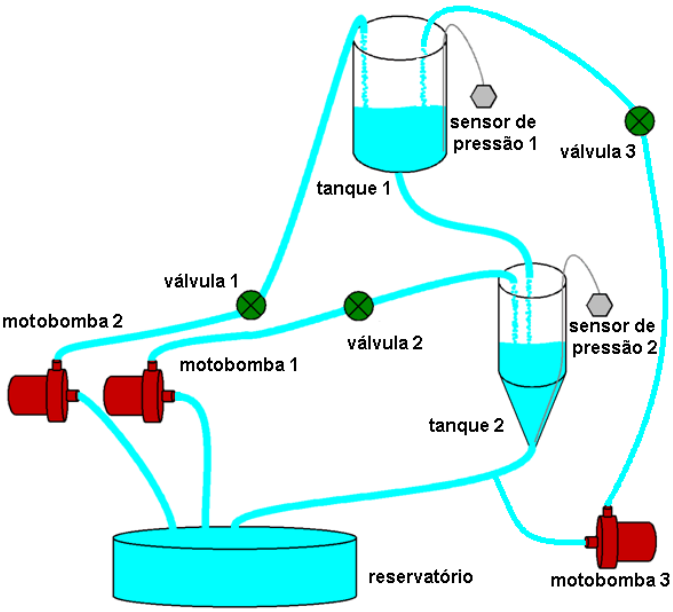

Figura 47. Experiência multi-variável.

O experimento pode ser usado tanto para controle multi-variável (dois tanques) quanto para controle mono-variável (apenas o tanque 2) O mono- e multi-variável não está associado ao número de tanques, mas sim ao número de variáveis controladas (por exemplo, além do nível poderíamos controlar temperatura, vazão, etc. O tanque 2 possui forma cônica, apresentando desta maneira uma variação não linear do seu volume (vide tanque 2 na Figura 47). Isso possibilita variar experimentos utilizados em ajustes de *PID*s através de diferentes níveis de água, já que o projeto do controlador PID não funciona de forma satisfatória para todos os níveis de água com os mesmos parâmetros (SILVA, 2004).

5.3.3 Arquitetura

A arquitetura do experimento está descrita na Figura 48, onde se destacam: (i) o sistema SCADA (*Elipse SCADA*), responsável pela monitoração e envio de dados ao configurador dos dispositivos FF, além disponibilizar um servidor *Web* e de vídeo ao usuário;

(ii) o banco de dados *MySQL*, responsável pelo registro de dados da experiência; o configurador de rede FF, sistema da SMAR com um servidor OPC para a PCI302, responsável pela interface com o barramento FF (*software* chamado de *Syscon*); (iii) os demais equipamentos e conexões com o usuário. Nessa figura são também descritos três blocos (AI, AO e PID), que podem ser configurados nos dispositivos FF (por exemplo, o bloco AI executando no sensor de nível e os blocos AO e PID no atuador da válvula) de forma a controlar o nível de água através de um controlador PID, que é executado na válvula *PY302*. O bloco AI (*Analog Input*) representa uma entrada analógica do sensor de nível no sistema de controle (bloco PID), desempenhada pela leitura do nível no sensor de pressão LD302. A saída do controle (atuação da válvula) é representada pelo bloco AO (*Analog Output*), ligada à saída do bloco PID. O bloco PID pode ser executado em qualquer dispositivo FF, inclusive no PCI302.

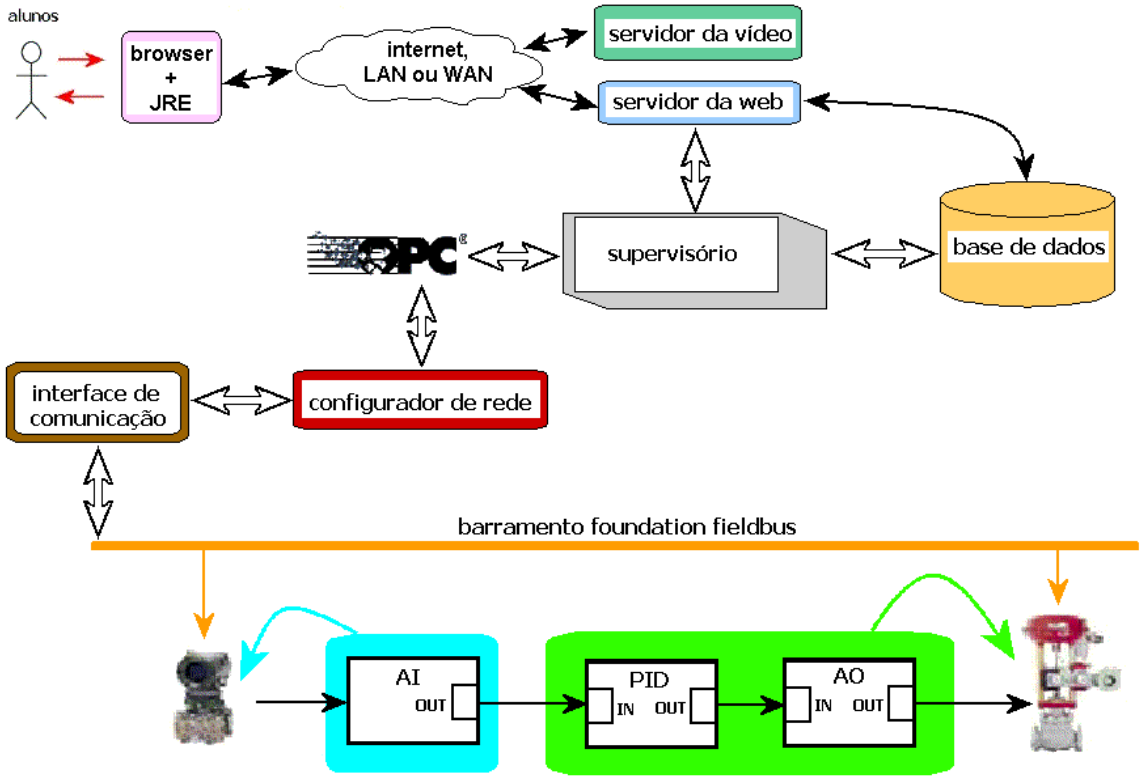

Figura 48. Arquitetura da Planta Didática FF da UFRGS.

De forma a corrigir alguns problemas de incompatibilidade de navegadores da *Web*, da implementação anterior (ZEILMANN, 2002), o servidor de vídeo do *Elipse SCADA* não está sendo mais utilizado. Em lugar dele um outro *software* se encarrega somente da transmissão de vídeo, chamado de *WebCam 2000* (WEBCAM2000, 2006) e distribuído livremente, funciona independente do sistema SCADA como servidor de imagens que são atualizadas a cada segundo. Assim o *Applet Java* gerado pelo *Elipse SCADA* para visualização das variáveis do processo não necessita gerar uma interface para transmissão de vídeo, ficando "mais leve".

5.3.4 Interface Remota

Seguindo a topologia *thin-client*, os usuários remotos (alunos) acessam o experimento através de uma interface simples utilizando páginas PHP com recursos de *Java Applets* gerados pelo *Elipse SCADA* (vide Elipse SCADA no Apêndice – Simuladores). O *Apache* é utilizado como *software* servidor *Web* de páginas *PHP*. Utilizando recursos de acesso de bancos de dados remotos do PHP, pode-se enviar parâmetros da experiência para o banco de dados *MySQL*. Assim, usuários remotos podem enviar comandos via bancos de dados ao gerenciador da experiência, isto é, o *Elipse SCADA*.

A atuação no experimento segue os seguintes passos: 1. O aluno escreve no banco de dados utilizando métodos de acesso do *PHP*; 2. O sistema *SCADA* lê o banco de dados; 3. O sistema *SCADA* envia dados lidos do banco de dados via OPC ao configurador de rede (SW); 4. O configurador de rede escreve dados no barramento *FF* (parâmetros PID, liga/desliga bombas, abertura de válvula, etc.) via PCI302 (HW); 5. O configurador de rede lê parâmetros do barramento FF; 6. O servidor OPC do configurador de rede (SW) atualiza parâmetros; 7. O sistema SCADA (cliente OPC) lê novos parâmetros e disponibiliza novos valores através da interface *Java Applet* ao usuário remoto. A transmissão de vídeo ocorre paralelamente aos

processos de envio/recebimento de dados através do servidor de vídeo do *Elipse SCADA* integrado no *Java Applet* e visualizado pelo usuário remoto.

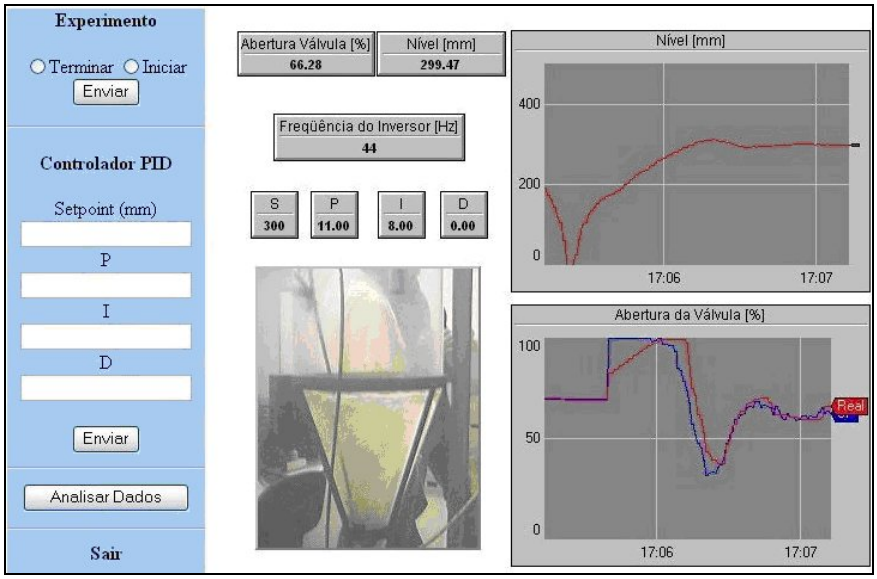

Figura 49. Interface remota da Planta FF da UFRGS.

A Figura 50 ilustra a interface com a entrada de parâmetros à esquerda e a visualização de variáveis e vídeo à direita. Nota-se também uma diferença entre o *setpoint* da abertura da válvula (saída do controlador em azul no gráfico) e a abertura real (em vermelho), pois o mecanismo de operação da válvula pneumática tem uma dinâmica específica que não acompanha a variação rápida do *setpoint*. Por exemplo, variações de abertura da válvula de 0 a 100% não ocorrem instantaneamente e sim demoram algum certo tempo.

5.3.5 Incorporações da Arquitetura Proposta

De acordo com a proposta de arquitetura deste trabalho uma incorporação com ambientes virtuais de aprendizagem foi desenvolvida assim como a flexibilização dos componentes intercambiáveis e um analisador de experiência com sistema tutorial para guia

de material educacional. Cada uma destas incorporações será descrita em diferentes subseções para melhor entendimento do trabalho.

5.3.5.1 Integração com Ambiente Virtual de Aprendizagem

Como apresentado anteriormente, o *MOODLE* será usado como plataforma de aprendizagem contendo material educacional referente aos experimentos remotos da UFRGS. Os materiais educacionais teóricos ligados ao experimento da planta FF estão organizados em quatro cursos:

i. Experimentos Remotos

   Curso que mostra passo a passo como usar todos os experimentos remotos da UFRGS. O primeiro tópico do curso é dedicado ao experimento com a planta FF no qual detalha como iniciar o experimento e como interagir com a interface para efetuar o controle de nível remotamente.

ii. Funcionamento dos Experimentos Remotos

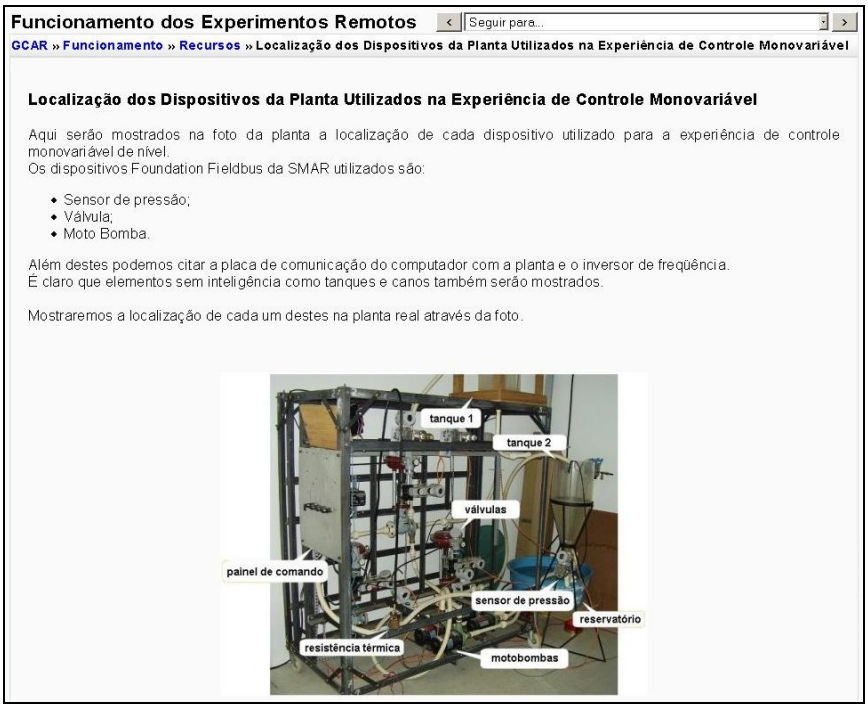

Figura 50. Curso de funcionamento da planta piloto.

Nesse curso, material educacional referente ao funcionamento dos experimentos foi desenvolvido. O primeiro tópico do curso descreve a planta piloto da UFRGS (vide Figura 50). Alunos podem visualizar desde a configuração dos equipamentos utilizados até a lógica de controle. Algumas considerações sobre os equipamentos FF também é descrita neste curso.

iii. Controladores PID

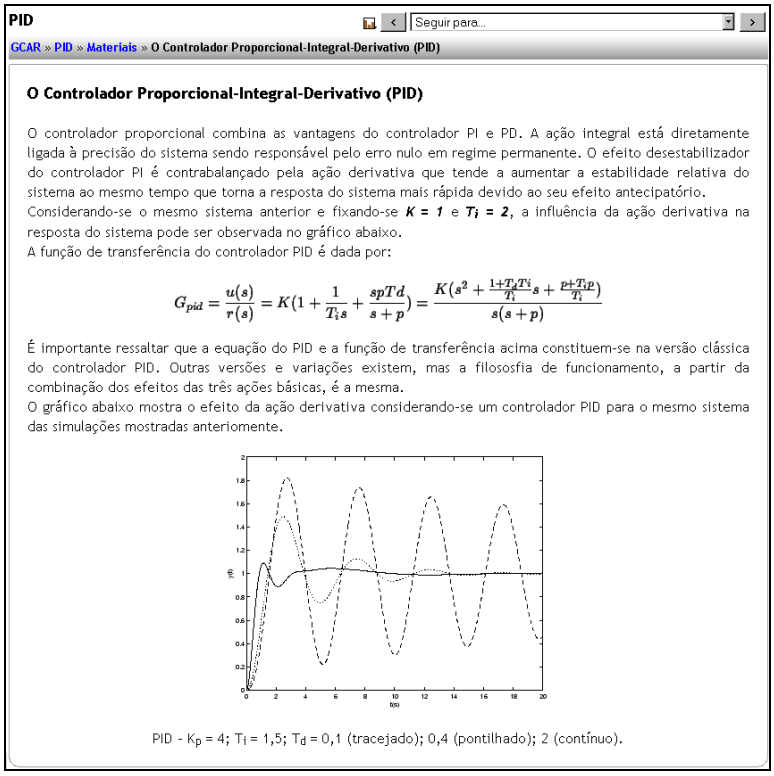

Figura 51. Curso de controlador PID.

Já que os objetivos das experiências são o aprendizado do funcionamento e do ajuste de controladores PID (na maioria), esse curso é o mais importante para servir de apoio teórico educacional do ambiente de aprendizado (vide Figura 51). Nele são descritas as considerações sobre sistemas de controle, assim como os métodos de ajuste. O curso se subdivide em três partes: conceitos básicos, controladores e métodos de ajuste de controladores PID. Este material

é baseado em estudos e apostilas de professores da UFRGS da disciplina de sistemas de controle (BAZANELLA, 2005).

Os métodos de ajuste, contidos no curso, são: período crítico, resposta ao salto e alocação de pólos. Uma alternativa a esses métodos é o ajuste manual em que o aluno usa o método de “tentativa e erro” para chegar ao resultado esperado (*trial and error*) o que é didaticamente desaconselhado.

iv. Protocolo Foundation Fieldbus

Nesse curso, o protocolo de comunicação FF é descrito de maneira didática (vide Figura 52) assim como sua configuração, instalação e as camadas ISO/OSI. Do ponto de vista acadêmico, esse curso é direcionado a alunos da pós-graduação.

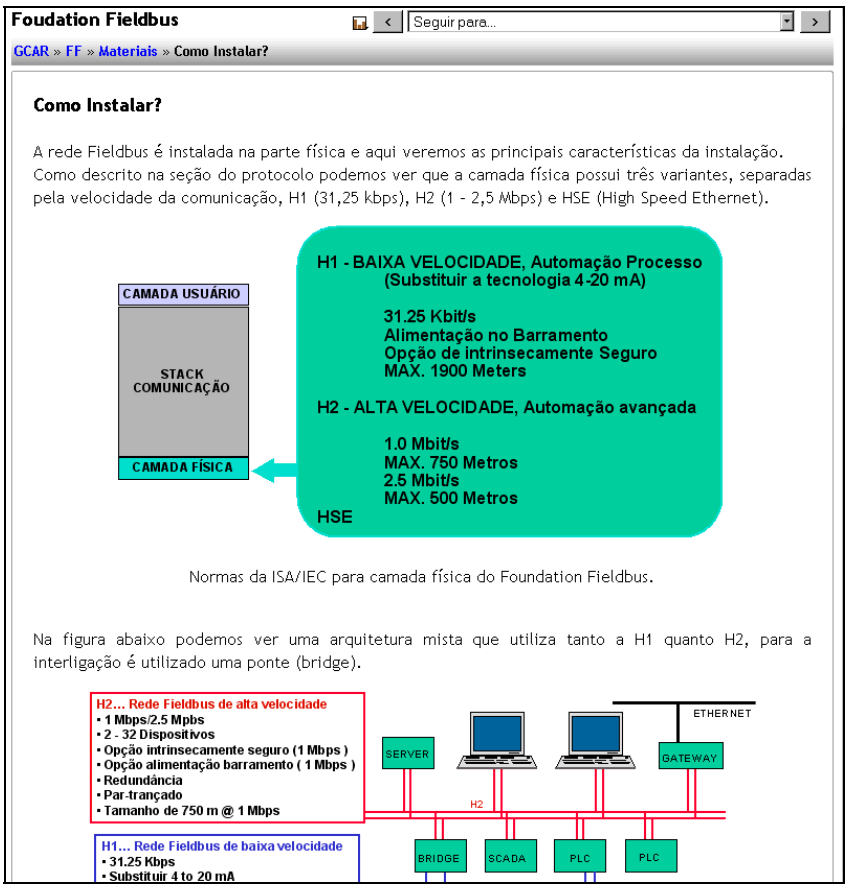

Figura 52. Curso de Foundation Fieldbus.

Atualmente, os cursos oferecidos pelo *MOODLE* na UFRGS estão disponíveis abertamente, isto é, qualquer usuário não cadastrado (*guest* ou convidado) pode visualizar seu conteúdo (vide Figura 53). O sistema de gerenciamento remoto da experiência está integrado no *MOODLE*, assim como o sistema de reservas (*booking system*). Inicialmente, um sistema próprio de reservas foi utilizado. Recentemente, utilizamos o sistema de reservas desenvolvido pela FEUP no Projeto MARVEL (vide Seção 3.2.9.4), que funciona como um módulo instalado no próprio *MOODLE*, mas este módulo apresentou alguns erros e incompatibilidade com navegadores *Web*. Por este motivo, o sistema reservas antigo desenvolvido pela UFRGS foi remodelado e funciona integrado ao MOODLE. O cadastro de usuários está integrado ao gerenciamento da experiência, isto é, apenas usuários cadastrados no *MOODLE* têm acesso remoto à experiência da UFRGS.

Figura 53. Página Inicial do MOODLE do GCAR na UFRGS.

Na Figura 54, está demonstrada a integração da experiência com a plataforma de aprendizagem *MOODLE*. Acima está situado o cabeçalho do ambiente virtual de aprendizagem MOODLE. Abaixo dele está a interface remota da experiência. À esquerda está

ilustrado a entrada de dados do usuário com o botão de Liga/Desliga e a entrada de parâmetros do controlador PID. A interface gráfica gerada pelo *Elipse SCADA* (*Applet Java*) está no centro. Pode-se notar que a curva de controle PID da abertura da válvula está ilustrada no gráfico acima, e no gráfico, logo abaixo desse, está a curva nível da água no tanque 2 para controle mono-variável. À esquerda da figura está ilustrada a transmissão de vídeo que foi retirada do *Applet Java* gerado pelo supervisório. Nota-se que esta interface (Figura 54) é diferente da interface desenvolvida anteriormente (vide Figura 49).

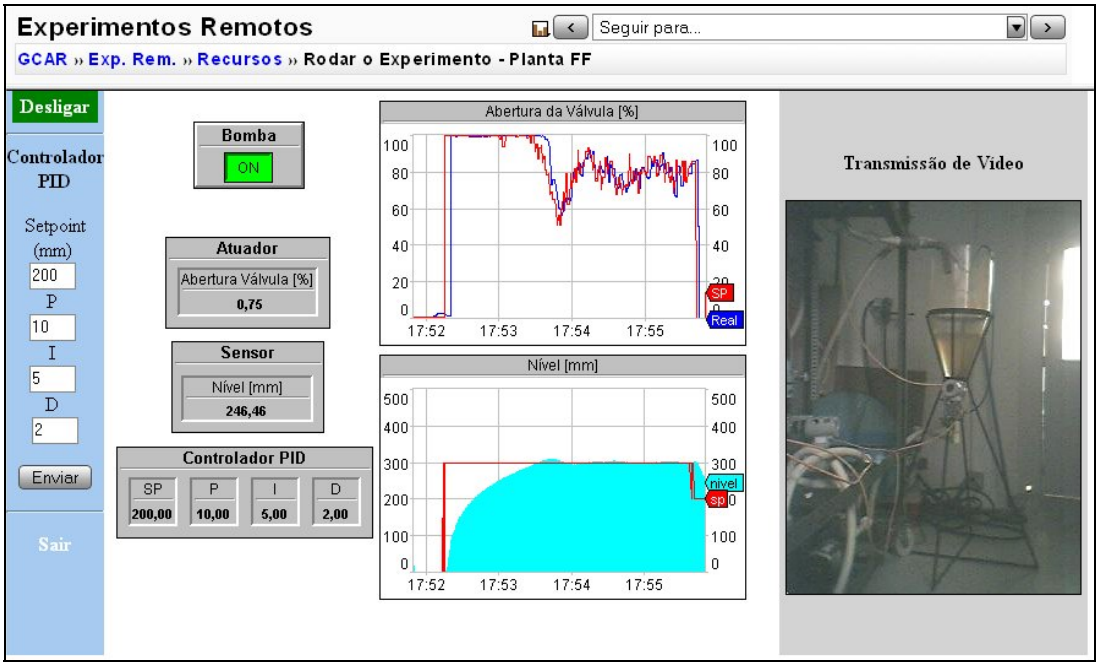

Figura 54. Integração da interface do experimento no MOODLE.

5.3.5.2 Desenvolvimento dos Componentes Intercambiáveis

Acrescentaremos à planta FF a interação com dispositivos virtuais de controle (simuladores e *softPLC*s) através da já usada comunicação OPC. Utilizando o *Elipse SCADA*, pode-se direcionar o controle para um *software* externo, que manipula as variáveis do experimento (abertura das válvulas e potência das moto-bombas) através de variáveis OPC. A ligação entre diferentes *softwares* é controlada pelo supervisório que funciona como cliente

OPC em ambos os *softwares* (da SMAR e simulador de controle), transmitindo os valores das variáveis de um servidor para outro (funcionamento *bridge*).

A Figura 55 ilustra como pode-se controlar a planta didática da UFRGS através de um controlador simulado no *software ISaGRAF*. Uma variável no banco de dados, chamada de *switch_opc*, é responsável pela seleção dos componentes (servidor(es) OPC) que farão parte do experimento. A ligação entre os servidores escolhidos é desempenhada pelo *Elipse SCADA*, que é cliente OPC de ambos os aplicativos (configurador da planta real e *ISaGRAF*). Assim, fazendo *switch_opc = 1*, pode-se utilizar um controlador simulado para controlar tanto o inversor de freqüência (controle da moto-bomba) como a abertura da válvula. A experiência funciona de forma tradicional, sem a utilização de simuladores, deixando *switch_opc = 0*.

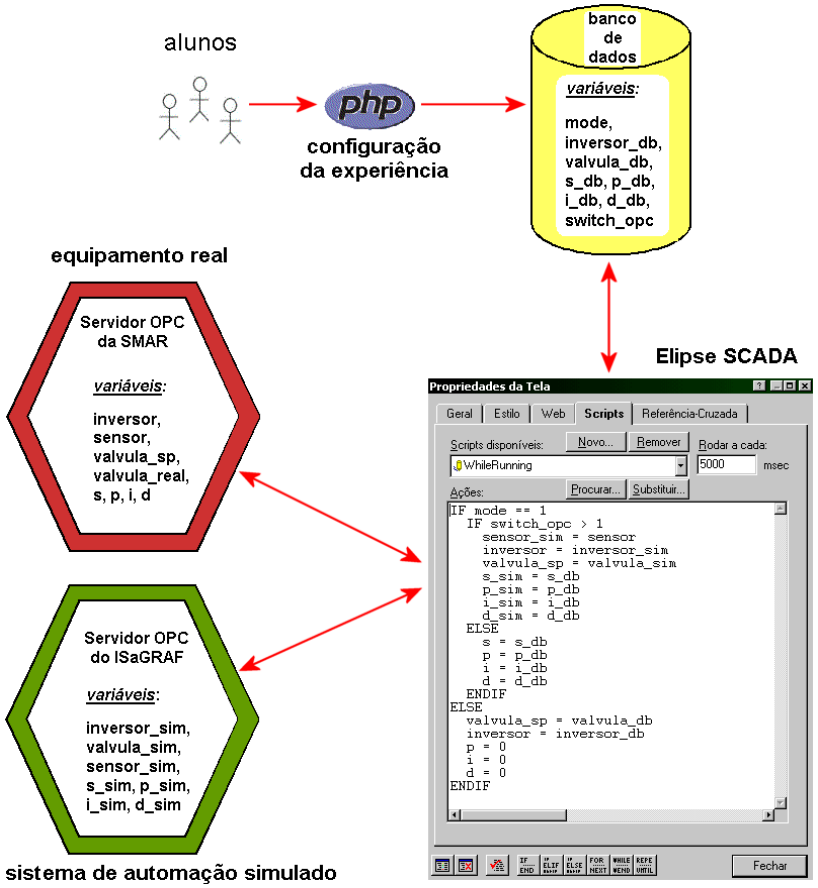

Figura 55. Seleção de componentes intercambiáveis na planta FF.

Também na Figura 55 observa-se que outra variável, chamada de *mode*, é responsável pela seleção do funcionamento do sistema em malha fechada com controlador PID, ou em malha aberta, quando o aluno configura seus próprios valores para a abertura da válvula e a freqüência do inversor. É importante notar que o simulador do sistema de automação (*ISaGRAF*) pode ser configurado para utilizar outros tipos de controladores, e não somente o PID. Ainda na Figura 55 nota-se que as variáveis *sensor* e *sensor_sim* não são armazenadas no banco de dados. Isto ocorre porque estas variáveis são apenas visualizadas na interface *Web* e não podem ser modificadas (escritas) pelos alunos. A presença da variável *valvula_real* que não acompanha instantaneamente a *valvula_sp* na planta real serve para ilustrar que equipamentos reais tem uma dinâmica própria.

Embora possamos selecionar o uso de um controlador simulado o usuário remoto pode apenas alterar parâmetros do controlador e não projetá-lo livremente, pois o *software ISaGRAF* não permite acesso/desenvolvimento remoto. O projeto do controlador simulado é estritamente local (servidor). Localmente, pode-se configurar o sistema de automação simulado para obedecer a qualquer lógica de controle desejada.

Outras combinações são obtidas de maneira idêntica mudando-se os componentes simulados e os reais, isto é, com um modelo dinâmico simulado da planta FF (no *ISaGRAF*) pode-se obter uma experiência totalmente simulada. Com a utilização de um *CLP* real teremos um sistema de automação real com uma planta simulada. Assim todas as possibilidades dos componentes intercambiáveis são cobertas utilizando a mesma técnica de chaveamento e os mesmos *softwares* simuladores.

5.3.5.3 Desenvolvimento do Analisador de Experiência e Sistema Tutorial

O analisador da experiência está integrado no *MOODLE*, assim como o experimento, de acordo com a arquitetura proposta. Para desenvolver estas tarefas, é utilizado um simples programa PHP, executado no servidor, que lê o relatório associado ao experimento realizado.

A análise gera e calcula algumas métricas associadas ao experimento de controle. A experiência é empregada para projetos de controladores PID, portanto, as métricas relevantes são: sobrepasso (*overshoot*), tempo de subida (*rise time*) e tempo de acomodação (*settling time*) do sistema.

O programa PHP, que executa o algoritmo para cálculo das métricas, é bastante simples. A partir do relatório da experiência criado pelo *Elipse SCADA*, o programa lê o arquivo-texto e retira dos dados do relatório as informações pertinentes (valores de nível e *setpoint* de controle).

O sistema tutorial, associado ao programa analisador, também é simples, pois basta analisar e comparar as métricas aos limites preestabelecidos para verificar se alguma das métricas está fora dos limites. Caso as métricas estejam fora dos limites, o direcionamento de material educacional mais elaborado (com exemplos práticos) é sugerido através de um link (botão). Por exemplo, pode-se especificar um projeto de controlador PID que possibilite ao sistema um *overshoot* máximo de 10 % e um tempo de subida máximo de 50 s. Se o aluno não atingir a especificação, será sugerido o material teórico sobre ajustes de controladores PID e mais especificamente sobre os problemas encontrados no experimento através da análise das métricas.

Na Figura 56, estamos ilustrando o relatório resumido de um ensaio de controle de nível, com seus respectivos resultados calculados pelo analisador. Como pode-se ver, o tempo de amostragem é bastante alto (um segundo), pois a dinâmica do processo não exige uma taxa de amostragem maior. O relatório completo mostra todos os valores armazenados no relatório, a cada segundo do ensaio realizado (em média 240 segundos), por esta razão é muito mais extenso.

Caso o aluno aceite a sugestão oferecida pelo sistema tutorial (clique no botão) um material didático com exemplos será apresentado de acordo com a mensagem apresentada. No

caso da Figura 56 uma sugestão de como diminuir o sobrepasso causado pelo controlador PID será apresentada. Caso os parâmetros projetados pelo aluno para o controlador torne o sistema instável (detectado pelo tempo de acomodação) o material educacional sugerido conterá teoria de estabilidade de sistemas. Todos os dados contidos no relatório também podem ser "baixados" para serem analisados em um *software* como o *MatLab* da *Mathworks* ou ainda o *Excel* da *Microsoft*. Ambos estes softwares podem reproduzir as curvas mostradas na interface gráfica do experimento. O *MatLab* ainda possui ferramentas de identificação de sistemas e muitos outros recursos utilizados em sistemas de controle (vide MatLab no Apêndice – Simuladores).

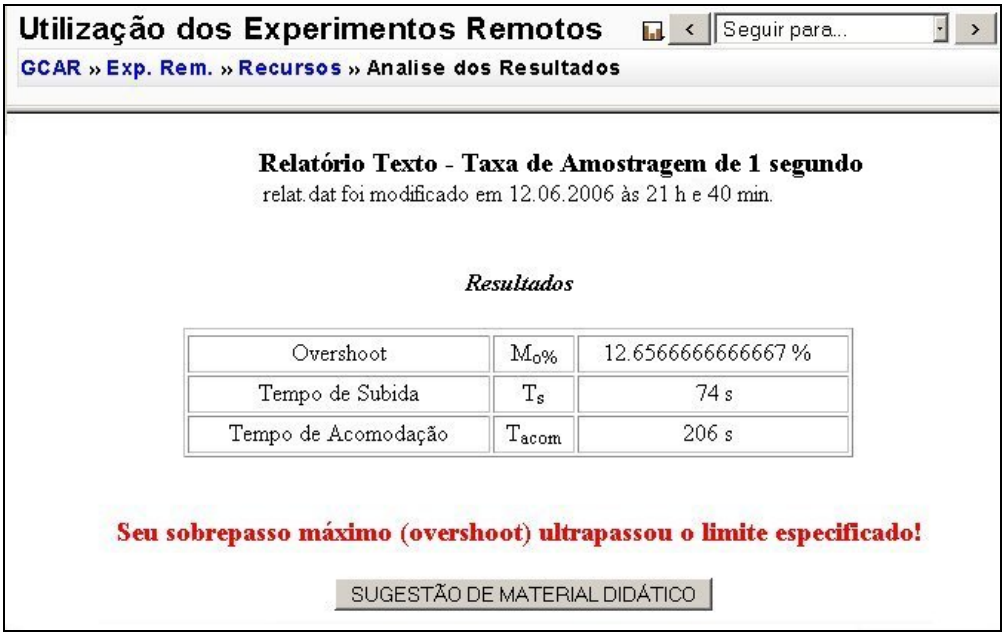

Figura 56. Análise do experimento (relatório resumido) realizado.

5.4 PLANTA TÉRMICA

Esta planta ilustra problemas de controle de temperatura em um experimento simples e de construção compacta para aulas da disciplina de sistemas de controle ministradas no curso de graduação da Engenharia Elétrica da UFRGS (SILVA, 2002).

5.4.1 Equipamentos

A planta térmica é um conjunto de equipamentos acomodados em uma caixa plástica preta (vide Figura 57). O experimento consiste dos seguintes equipamentos: uma resistência elétrica, montada dentro de um cilindro metálico; um termopar tipo K; e um controlador PID industrial. A alimentação da resistência é feita pelo chaveamento da tensão da rede elétrica, modulação de largura de pulso (PWM), através de um relé de potência localizado fisicamente no controlador PID industrial.

Figura 57. Planta térmica.

O controlador PID industrial do é tipo "single loop" modelo N1100 produzido pela empresa NOVUS (NOVUS, 2006). As características deste controlador de maior interesse para os experimentos são as seguintes:

- ✓ Controlador com microprocessador, permitindo uma ampla programação de parâmetros e modos de operação, via teclas do próprio equipamento ou remotamente via interface serial RS-485 com protocolo MODBUS ASCII e RTU;
- ✓ Entrada universal multi-sensor. Em particular, para sensores de temperatura (termopares e PT100) uma linearização interna via software é feita;

- ✓ Saída em PWM através de um relé de potência (3 A / 250 V);
- ✓ Display permitindo a visualização de, entre outros: valores da variável manipulada (MV) e da variável de processo (PV) em tempo real, valores dos parâmetros programados;
- ✓ Modos de operação manual (controle em malha aberta) e automático (controle em malha fechada com controlador PID).

5.4.2 Objetivo do Experimento

O objetivo é controlar a temperatura da resistência térmica que é aquecida pela circulação de corrente elétrica através do circuito. A corrente elétrica é manipulada pelo controlador industrial PID que aciona a saída PWM com valores de 0 a 100 %. A medição de temperatura é efetuada por um termopar próximo à resistência que está ligado ao controlador PID na entrada universal multi-sensor. A temperatura deve ser limitada em 200^{o}C para não danificar a resistência (atuador) nem o termopar (sensor).

O experimento também possibilita o controle de uma ventoinha que pode ser utilizada para refrigerar a resistência mais rapidamente ou diminuir a taxa de aquecimento da resistência.

5.4.3 Arquitetura

A arquitetura previamente implementada utiliza a porta serial do PC e um *driver* (Modbus RTU) fornecido pela NOVUS para conectar o sistema *SCADA* ao controlador PID industrial. Desta forma a arquitetura para acesso remoto se torna muito simples (vide Figura 58), onde o *Elipse SCADA* disponibiliza o acesso à interface remota (*Java Applet*) e variáveis do processo são escritas via banco de dados por páginas PHP a exemplo da Planta FF apresentada anteriormente.

Como pode-se notar o servidor de vídeo não é utilizado, já que o experimento não apresenta nenhuma modificação visual perceptível. O controlador PID industrial é responsável pelo controle do circuito e se conecta tanto com os seus atuadores: resistência e ventoinha; quanto com seu sensor: o termopar.

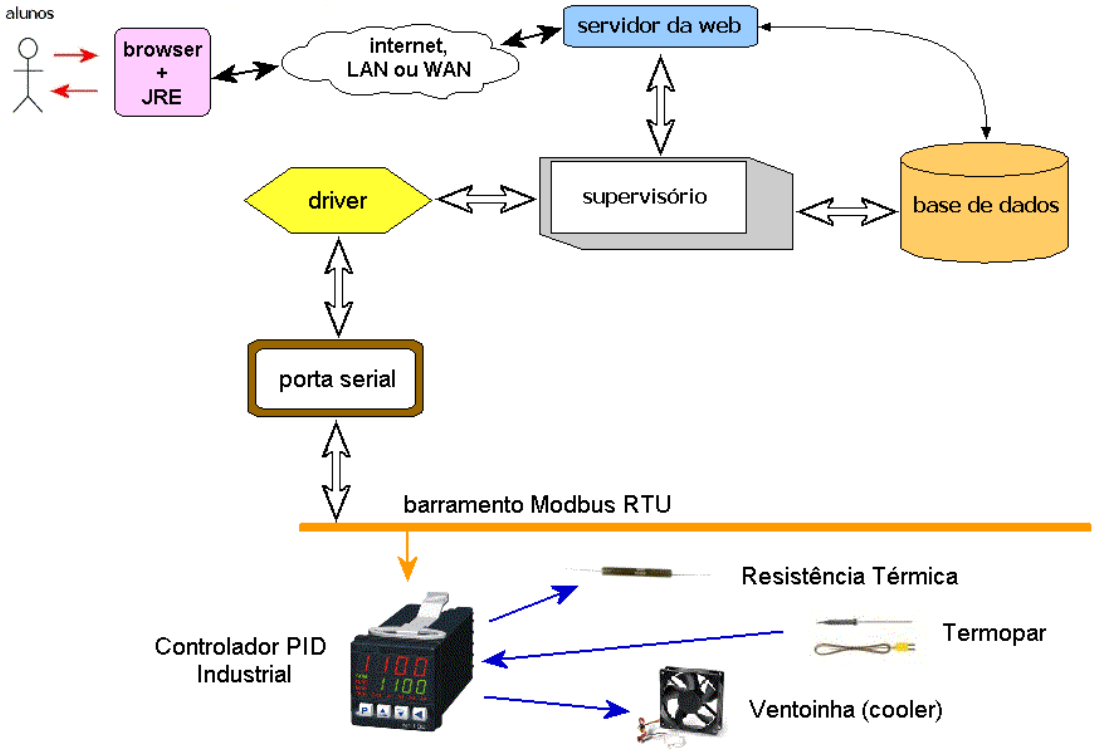

Figura 58. Arquitetura da planta térmica.

5.4.4 Interface Remota

A planta térmica a exemplo da planta FF também possui uma topologia *thin-client* e a interface do experimento é similar assim como a forma de acesso. Diferentemente da Planta FF a passagem de parâmetros não utiliza a comunicação OPC. O supervisório retransmite os dados escritos no banco de dados para o dispositivo da NOVUS.

A dinâmica do processo de temperatura da planta térmica é mais rápida que o da planta didática FF (controle de temperatura nos tanques), já que utiliza uma simples resistência térmica ao ar livre, diferentemente do experimento da planta didática FF onde a

resistência fica imersa em tanques de água, que possui uma inércia (calor específico) muito maior.

A Figura 59 ilustra a interface para um ensaio em malha aberta, isto é, a potência da resistência é controlada manualmente e não por um controlador PID. Este tipo de ensaio é utilizado para identificar as constantes que regem a dinâmica da planta.

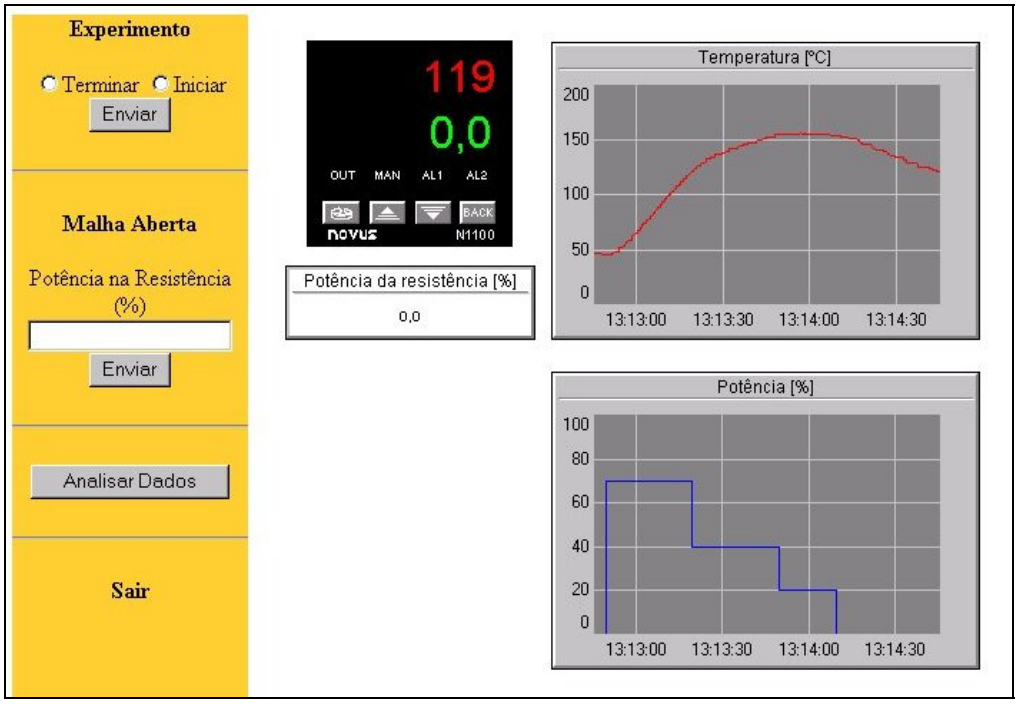

Figura 59. Interface Remota do experimento da planta térmica

5.4.5 Incorporações da Arquitetura Proposta

As incorporações deste estudo de caso seguem o mesmo padrão das do experimento anterior (planta FF), portanto somente algumas diferenças serão apontadas.

5.4.5.1 Integração com Ambiente Virtual de Aprendizagem

Materiais educacionais referentes a este experimento foram desenvolvidos e armazenados no ambiente de aprendizagem *MOODLE*. Os cursos em que este experimento se insere são os mesmos da planta FF com exceção do curso de *Foundation Fieldbus*, pois

ambos utilizam da mesma técnica de controle e praticamente a mesma arquitetura. Apenas tópicos diferentes dentro de cada curso são específicos para a planta térmica. Um novo curso do protocolo de comunicação *Modbus* será desenvolvido futuramente. A interface do experimento está totalmente integrada no MOODLE a exemplo do estudo de caso anterior.

5.4.5.2 Desenvolvimento dos Componentes Intercambiáveis

Diferentemente do experimento da planta FF não será selecionado qual servidor OPC deve ser utilizado, mas sim se o controlador industrial manipulará a potência aplicada na resistência térmica (visto como modo automático para o controlador) ou se a potência será manipulada de fora do controlador (visto como modo manual para o controlador). Por exemplo, estaremos utilizando um sistema de automação simulado e uma planta real se configurarmos o controlador industrial para funcionar no modo manual e manipularmos a potência em um *software* simulador. Para automatizar este arranjo uma variável do banco de dados será responsável pela seleção do componente de sistema de automação (reflete diretamente no modo de operação do controlador industrial) e um *script* do *Elipse SCADA* direcionará os valores de potência da resistência calculados pelo controlador simulado para o experimento (com controlador PID no modo manual) caso este utilize um sistema de automação simulado.

Outras combinações são obtidas de modo idêntico ao estudo de caso anterior (planta FF), onde somente necessita-se do modelo de simulação da planta térmica projetado no *ISaGRAF*. A interação entre diferentes experimentos também pode ser obtida usando-se a mesma técnica, isto é, pode-se controlar tanto o nível da planta FF quanto a temperatura da planta térmica num mesmo experimento apenas programando o supervisório para conectar as diferentes plantas em uma única interface.

A Figura 60 ilustra a montagem necessária para se automatizar a ligação do componente de sistema de automação. O controlador simulado será novamente modelado no

ISaGRAF e a conexão do sistema SCADA com o simulador é via OPC. A flecha verde tracejada ilustra o trajeto percorrido pelos dados do controlador simulado (no *ISaGRAF*) até a planta real, passando pelo supervisório (que será a ponte entre eles) e o controlador real (que estará no modo manual para aceitar controle externo).

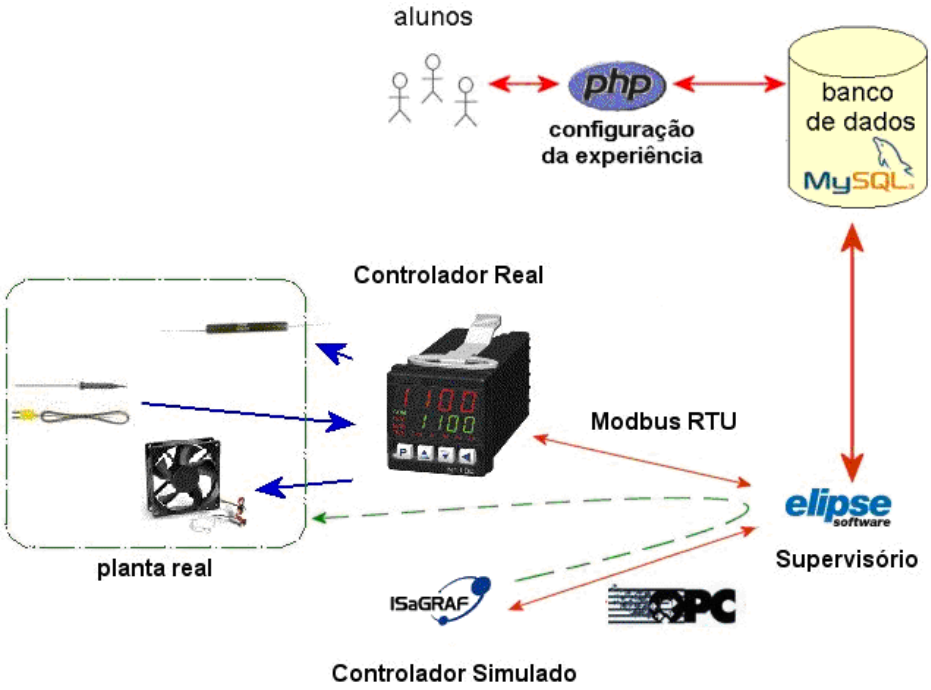

Figura 60. Seleção de componentes intercambiáveis na planta térmica.

5.4.5.3 Desenvolvimento do Analisador de Experiência e Sistema Tutorial

Apesar de este experimento utilizar grandezas físicas (potência e temperatura) diferentes do estudo de caso anterior (abertura da válvula e nível), o relatório gerado pelo *Elipse SCADA* é praticamente o mesmo, portanto quase nenhuma modificação precisa ser desenvolvida no programa analisador de experiência. O sistema tutorial agora deve comparar com especificações diferentes de tempo de subida e tempo de estabilização, pois a dinâmica deste processo é diferente. A sugestão de material educacional do sistema tutorial é a mesma do estudo de caso anterior, pois a técnica de controle utilizada (PID) é idêntica.

5.5 EXPERIMENTO REMOTO DE MECATRÔNICA

O SENAI-RS com grande interesse em experimentos remotos para serem utilizados no ensino elaborou um projeto, chamado de laboratório remoto de mecatrônica, em cooperação com as Universidades alemãs de Bremen e técnica de Berlim, a UFRGS e o Centro de Excelência de Tecnologia Avançada (CETA-RS). Este projeto visa a construção de experimentos baseados no sistema *deriveSERVER* da Universidade de Bremen (vide Seção 3.2.9.1). O experimento de mecatrônica da Universidade de Bremen foi construído com o intuito de comprovar teorias de *hyper-bonds* (vide Seção 2.1.4.2) utilizados para conectar componentes virtuais aos reais e vice-versa que é a essência da realidade mista. O centro tecnológico de mecatrônica do SENAI se propôs a estender o sistema original para utilizá-lo em cursos técnicos de Mecatrônica em Caxias do Sul.

5.5.1 Componentes do Experimento

O experimento conta com diversos módulos responsáveis pelo processamento distribuído de diferentes partes. O módulo central executado no servidor, é chamado de ROMAN (*Real Object Manager*) e gerencia todos os outros módulos do sistema *deriveSERVER* (vide 3.2.1.9). O módulo VCK (*Virtual Construction Kit*) é responsável pela interface gráfica do experimento e pela representação dos equipamentos simulados, isto é, a bancada virtual. No VCK também são construídas as ligações entre os equipamentos virtuais e os reais e vice-versa. O módulo chamado de *hyper-bond* é responsável pelo interfaceamento de dispositivos reais (ou externos) ao VCK.

Portanto os componentes do experimento são: equipamentos virtuais e eventuais equipamentos reais e externos de eletro-pneumática. É importante ressaltar que com exceção do ROMAN os outros módulos podem ser executados em clientes diferentes.

5.5.2 Objetivo do Experimento

O objetivo da experiência é demonstrar o funcionamento e a lógica associada aos circuitos eletro-pneumáticos. Os experimentos são bastante flexíveis e uma infinita gama de arranjos de equipamentos tanto reais (dependendo da disponibilidade do usuário) quanto virtuais é possível.

5.5.3 Arquitetura

A arquitetura do experimento é a mesma do projeto original alemão (vide Figura 20). Uma característica muito útil do experimento alemão é a capacidade de distribuir seu *hardware* e *software*. O cliente pode interagir com a experiência remota a partir de seus próprios *hardware* e *software*. Por exemplo, um kit de microprocessador instalado no cliente pode controlar a experiência remota através da interface *hyper-bond* executada no cliente. O mesmo serve para *softwares* ligados à experiência através de um *hyper-bond*. A distribuição no experimento do *hardware* e do *software* é possibilitada pela arquitetura cliente-servidor de seus módulos, isto é, o *software hyper-bond* pode ser executado no cliente, que se comunica com o ROMAN, executado no servidor.

Inicialmente, o projeto alemão do *deriveSERVER* suporta *I/O*s somente através de um sistema de aquisição de dados chamado *EasyPort*, da fabricante alemã FESTO. Com as modificações implementadas no Projeto de Laboratórios Remotos do SENAI, pode-se utilizar um *hardware* de aquisição muito mais simples, ligado à porta paralela do PC. O *hardware* inicial testado para interagir com a porta paralela suporta 32 entradas e 32 saídas digitais e foi construído com *CI*s como *shift registers* e multiplexadores ligados a bits de controle da porta paralela do PC. A partir desta idéia foi desenvolvido um protocolo de comunicação simples para interfacear 32 *I/O*s através da porta paralela. Este protocolo é o mesmo proposto na modificação do simulador *Relés* do SENAI (vide Seção 5.2.3 e Tabela 5). Através da porta

serial, pode-se até usar algum outro dispositivo que utilize o mesmo protocolo de comunicação do *EasyPort*.

Infelizmente, o experimento montado no sistema *deriveSERVER* não possui topologia *thin-client*, o que é uma característica indesejável, pois temos de usar *softwares* específicos para poder visualizar a bancada virtual e a real. O aluno também necessita de uma cópia do *software* do *hyper-bond*, se desejar interagir com o experimento remoto através de seu próprio *software/hardware*.

5.5.4 Interface Remota

A Figura 61 demonstra uma configuração típica da interface remota de realidade mista do experimento.

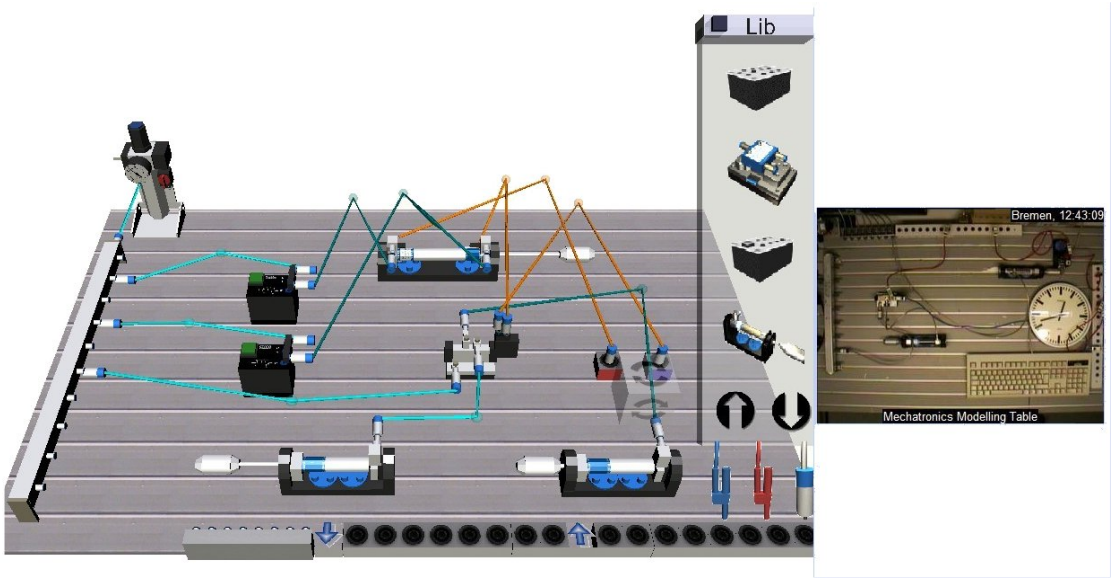

Figura 61. Interface remota de um experimento de realidade mista

A interface remota se resume a bancada virtual (módulo VCK) que é carregada a partir de *Applets* e *Scripts Java* interpretados pelo navegador *Internet Explorer* da *Microsoft*. Estes *Applets* são responsáveis por carregar (“baixar”) modelos VRML que são interpretados pelo

plug-in Cortona da *Parallel Graphics* para visualização dos componentes tridimensionais virtuais. Assim, a bancada virtual possibilita a visualização da interface remota do experimento que pode ainda incluir a transmissão de vídeo dos equipamentos reais conectados ao ROMAN no servidor. Modelos VRML de conectores do *hyper-bond* são responsáveis pela conexão dos equipamentos reais à bancada virtual.

5.5.5 Incorporações da Arquitetura Proposta

5.5.5.1 Integração com Ambiente Virtual de Aprendizagem

Este estudo de caso ainda está em fase de incorporação no ambiente virtual de aprendizagem MOODLE. Alguns materiais educacionais como: manuais dos dispositivos e o tutorial de uso do sistema já foram desenvolvidos e estão sendo hospedados na instalação do MOODLE no CT de Mecatrônica em Caxias do Sul, onde será extensivamente utilizado. Outros cursos como o de eletro-pneumática básica e o de automação de sistemas eletro-pneumáticos estão sendo desenvolvidos pelo pessoal especializado do NEAD do SENAI. A Figura 62 ilustra como se encontra atualmente o ambiente virtual de ensino do SENAI.

Figura 62. Instalação do MOODLE no SENAI em Caxias do Sul.

Como forma cooperação no projeto o SENAI se propôs a construir os cursos também em inglês para serem utilizados futuramente nas disciplinas ministradas em Bremen e em Berlim. O MOODLE oferece a seleção de linguagem do conteúdo facilmente não sendo necessária a formulação de cursos em línguas diferentes.

5.5.5.2 Desenvolvimento dos Componentes Intercambiáveis

Para comprovar a capacidade de intercambiar experimentos, além do mostrado na Figura 61, desenvolvemos modificações no *hyper-bond* do sistema original. O experimento, agora, possui interfaces de comunicação OPC (cliente), o que possibilita comniações com qualquer servidor OPC, além do funcionamento tradicional (*mixed-reality*).

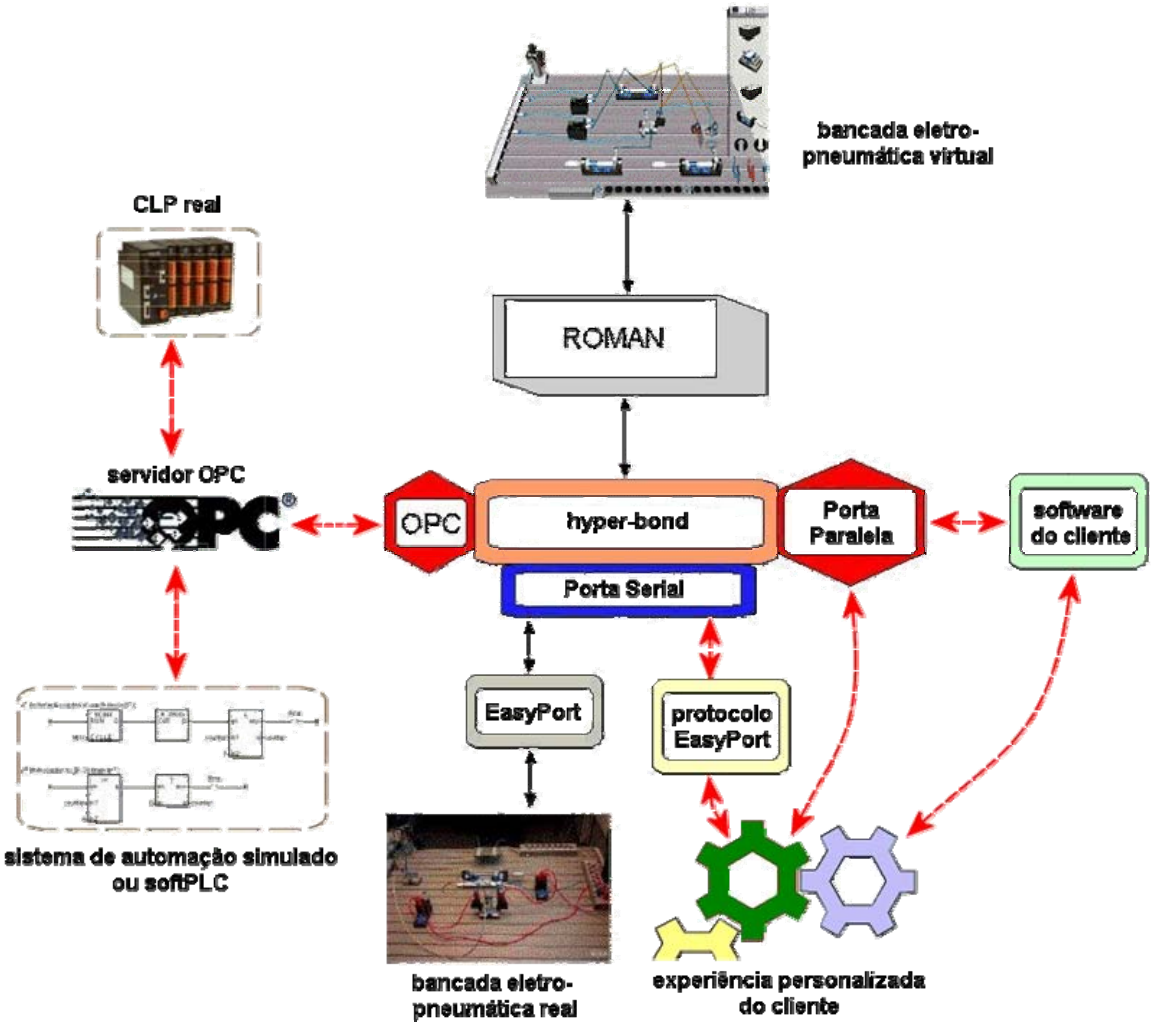

Figura 63. Modificações no deriveSERVER.

Pode-se interagir com uma gama muito grande de *softwares* e experiências personalizadas em razão das modificações efetuadas no *deriveSERVER*, como demonstrado na Figura 63. As flechas tracejadas (e maiores) bidirecionais na figura ilustram interações novas adicionadas ao projeto original.

Com estas novas interações desenvolvidas nas interfaces *hyper-bond* pode-se conectar o simulador *Relés* a bancada virtual e controlar tanto dispositivos virtuais quanto reais da experiência. O mesmo serve para o simulador *ISaGRAF* que também pode interagir na bancada virtual através da nova interface OPC. Desta maneira, pode-se construir um experimento completamente distribuído que pode ser executado em diversos centros do SENAI e conectados através de uma única bancada virtual. Nesta montagem o experimento possuiria várias instâncias da interface *hyper-bond* em diferentes clientes conectados virtualmente.

Mais adiante neste trabalho, uma seção de protótipos (vide Seção 5.7.1) possui diversas implementações deste sistema como forma de integrar vários tipos de experiência diferentes tanto simulados quanto reais em um único experimento.

5.5.5.3 Desenvolvimento do Analisador de Experiência e Sistema Tutorial

O analisador de experiência e o sistema tutorial não foram desenvolvidos para este estudo de caso, pois ainda não há definição das experiências que deverão ser executados pelos alunos utilizando o sistema. A partir desta definição será criado um sistema que monitore e identifique se a experiência foi executada de forma correta. O sistema manipula somente dados discretos devido a construção do sistema alemão. Portanto o analisador de experiência detectaria somente seqüências de dados discretos que apontariam a execução correta do experimento. O sistema tutorial ofereceria ao aluno material educacional referente a possíveis falhas previamente configuradas.

5.6 PLANTA DE ENVASE DE GARRAFAS

A planta de envase (envasilhamento) de garrafas, ou de enchimento de garrafas, é uma simulação, criada durante a disciplina de estágio em docência, na qual alunos aprendem a projetar sistemas de automação baseados em plantas de sistemas industriais hipotéticos. A planta foi desenvolvida (projetada) no *software ISaGRAF*, que possui linguagens de programação padronizadas pela norma IEC 61131. Os alunos, além de aprenderem a projetar sistemas de automação industrial, são introduzidos a cinco linguagens de programação, amplamente usadas na indústria.

5.6.1 Modelo da Planta de Envase de Garrafas

A planta conta com três estações de trabalho distintas e um reservatório, onde o líquido é estocado. Cada estação é responsável por uma etapa do processo de envase: enchimento, prensa para colocar a tampa e rotuladora. Sensores detectam a presença da garrafa em qualquer uma das estações, isto é, *sensor 1* na estação de enchimento, *sensor 2* na prensa e *sensor 3* na rotuladora. Outros dois sensores efetuam as medições no reservatório: um sensor de nível de líquido e um sensor de temperatura do líquido.

De posse dos cinco sensores distintos, controlar-se a planta através de seis atuadores:

i. Esteira: responsável pelo transporte das garrafas através das estações.
ii. Válvula de enchimento: enche de líquido a garrafa;
iii. Prensa: coloca e prensa a tampa da garrafa;
iv. Rotuladora: coloca o rótulo na garrafa;
v. Válvula principal de escoamento do líquido: enche o reservatório;
vi. Resistência elétrica: aquece o líquido no reservatório.

Apesar da construção simples, o modelo de envase de garrafas possibilita uma infinidade de variações através do ajuste de parâmetros. Os alunos se valem de

temporizadores para regular o tempo necessário de atuação em cada estágio do processo industrial hipotético.

Na Figura 64, pode-se visualizar a simulação com um sistema de automação projetado corretamente. A garrafa, elemento central do modelo da planta, se encontra na estação de enchimento e por este motivo o *sensor 1* está ativo. O sistema de automação projetado desligou a *esteira* e abriu a *válvula* iniciando o enchimento da garrafa. Já que nenhum erro na produção de garrafas foi detectado o modelo sinaliza com a mensagem: “nenhum erro”. Pode-se notar que cada atuador possui um parâmetro associado, podendo assim ser criadas diversas versões diferentes da planta para a experimentação.

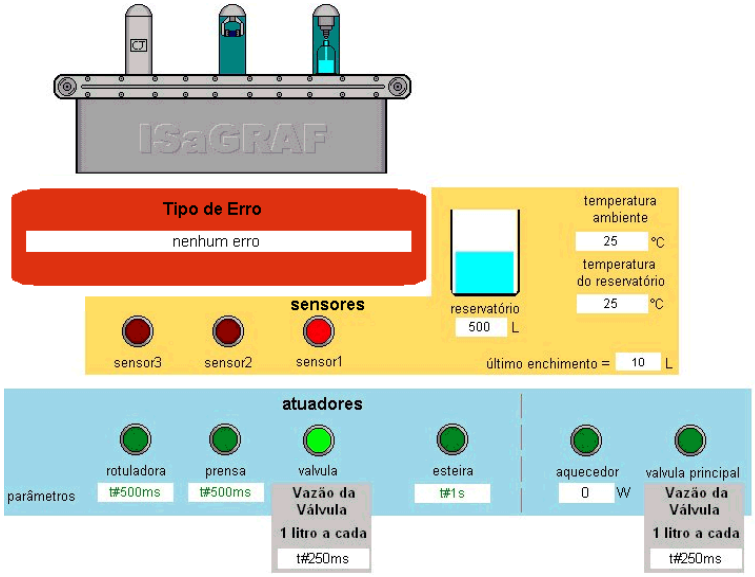

Figura 64. Interface gráfica da planta de envase.

A Figura 65 demonstra a interface de programação do *ISaGRAF* para projetar o controle do modelo simulado. Nota-se que a linguagem utilizada é o *Ladder* (lógica de relés) para o controle da esteira e que o modelo é composto de diversos módulos.

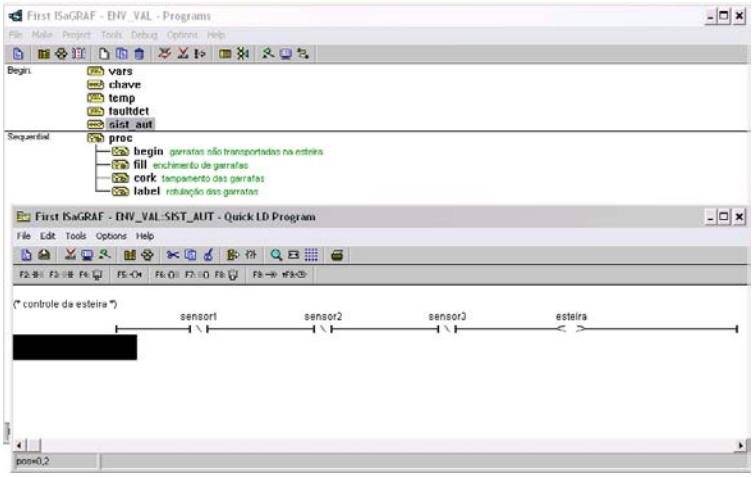

Figura 65. Janelas do ISaGRAF da planta de envase.

5.6.2 Objetivo do Experimento

Com base nas variáveis simuladas de *I/O* (sensores e atuadores) da planta, os alunos podem projetar seu próprio sistema de automação, valendo-se apenas de seis atuadores previamente descritos. O sistema projetado para a automação fará uso das linguagens de programação padronizadas na norma IEC 61131-3. Portanto, com este objetivo, os alunos indiretamente aprendem todas as linguagens de programação associadas ao modelo e ainda se defrontam com problemas comuns na automação de sistemas industriais.

A planta também pode servir para ensinamentos de otimização de processos industriais, já que o sistema de automação proporciona esta vantagem diretamente. Usando este mesmo modelo ainda podem ser propostos desafios extras de velocidade e desempenho na produção de garrafas.

5.6.3 Arquitetura

Para este estudo de caso foram criadas duas arquiteturas distintas: uma local e outra remota. A arquitetura local utiliza somente o simulador *ISaGRAF* versão 3.3 (estudantil) que deve ser copiado pelo aluno assim como o modelo de simulação da planta (vide Figuras 64 e

65). A arquitetura remota utiliza a versão 5 (comercial) do *ISaGRAF* e a interface OPC para comunicação com o *Elipse SCADA* que oferece a interface gráfica através *Applets Java* que são disponibilizados na *Web* (vide Figura 66).

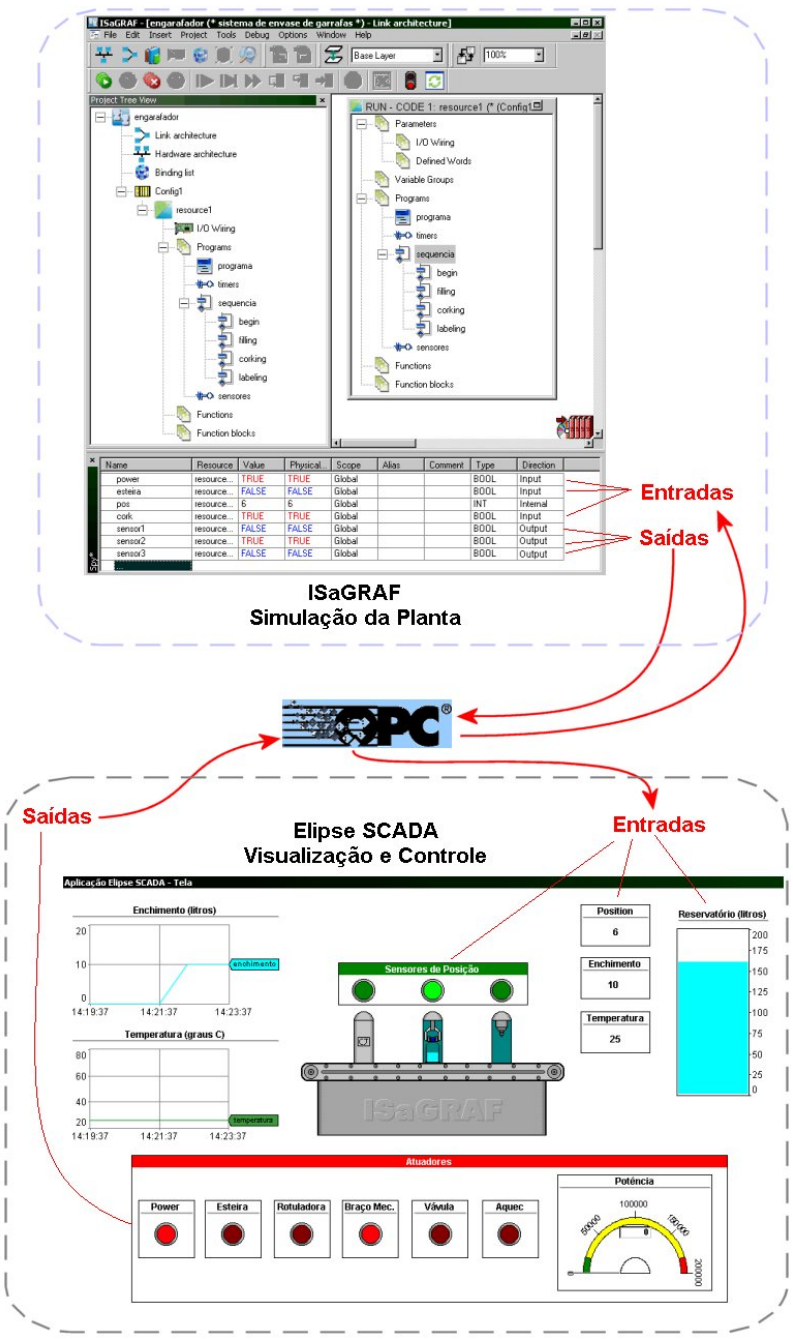

Figura 66. Arquitetura remota da planta de envase simulada.

5.6.4 Interface Remota

A interface é semelhante a da Figura 64 apresentada anteriormente. O *Elipse SCADA* gera *Applets Java* que se comunicam com o supervisório e este por sua vez com o *ISaGRAF* por meio da interface OPC. Da mesma forma que os estudos de caso anteriores (Planta FF e Planta térmica) o usuário remoto controla o experimento por meio de variáveis do banco de dados do servidor. A interface remota ainda não oferece a interação necessária para controle automático por parte do aluno. Por este motivo alguns protótipos serão apresentados de maneira a solucionar este problema.

5.6.5 Incorporações da Arquitetura Proposta

5.6.5.1 Integração com Ambiente Virtual de Aprendizagem

Materiais educacionais sobre a norma IEC 61131 e sobre o como utilizar o modelo para simulação local (arquitetura local) foram desenvolvidos no curso chamado de Norma IEC61131-3 (vide Figura 53). Neste curso estão características das linguagens englobadas pela IEC61131-3 assim como um tutorial de como usar a planta de envase simulada. Material referente ao funcionamento da planta simulada também está disponível neste curso.

Para utilizar a arquitetura remota foi desenvolvido dentro do curso de utilização de experimentos remotos um tópico especial sobre implementações da planta de envase, inclusive com interações com outros experimentos como a planta térmica.

5.6.5.2 Desenvolvimento dos Componentes Intercambiáveis

Os componentes intercambiáveis podem ser selecionados através de *scripts* analogamente aos outros estudos de caso que utilizaram o *Elipse SCADA* como supervisório. A planta é simulada e suporta diversas interações com componentes diferentes por este motivo combnações diferentes serão propostas na seção de protótipos (vide Seção 5.7). A

construção de uma planta real para este processo ainda está em estudo, portanto nenhuma combinação com um componente da planta real será apresentado.

5.6.5.3 Desenvolvimento do Analisador de Experiência e Sistema Tutorial

O analisador da experiência foi projetado no próprio modelo, isto é, na própria simulação da planta existe uma programação que detecta possíveis operações que resultem em erros. Estes erros são mostrados na interface, que alerta com o tipo de erro, assim que detectados. O modelo está programado para detectar erros ocasionados por má temporização dos estágios de produção assim como erros de intertravamento. Por exemplo, a simulação sinaliza: erro de temporização da válvula se o sistema projetado encher a garrafa além do limite; ou com erro de temporização da prensa se a prensa (braço da prensa) estiver abaixada e a esteira for acionada.

O sistema tutorial para este estudo de caso foi desenvolvido com base nas mensagens de erro geradas pelo modelo de simulação que são armazenadas pelo supervisório assim como todo o relatório das variáveis da simulação. Desta forma, o sistema tutorial irá apontar um material educacional referente ao tipo de erro e possíveis causas deste para que o aluno efetue a correção.

5.7 PROTÓTIPOS

A seguir, será apresentado possíveis protótipos de implementações com combinações dos estudos de caso descritos. Estes protótipos aumentam ainda mais o intercâmbio de componentes e a flexibilidade da experiência.

Os dois protótipos são construídos com materiais e *softwares* que possibilitam ao aluno uma experiência com a realidade utilizada na indústria de sistemas de controle e automação. Do ponto de vista do desenvolvimento de *software*, utilizamos o protocolo OPC para modularizar e tornar mais simples a interação entre sistemas de automação e planta.

Utilizando-se desta arquitetura de comunicação, pode-se referir ao OPC como uma “cola” que une diferentes fontes de sistemas de automação e de plantas (vide Figura 67).

Essa arquitetura é transparente ao usuário, que não precisa saber as fontes dos sistemas nem a forma como é feita a comunicação. A transparência pode ser entendida como *ubiquitous computing* (computação ubíqua), isto é, o usuário não tem o conhecimento do sistema como um todo, sabe, contudo, sua funcionalidade.

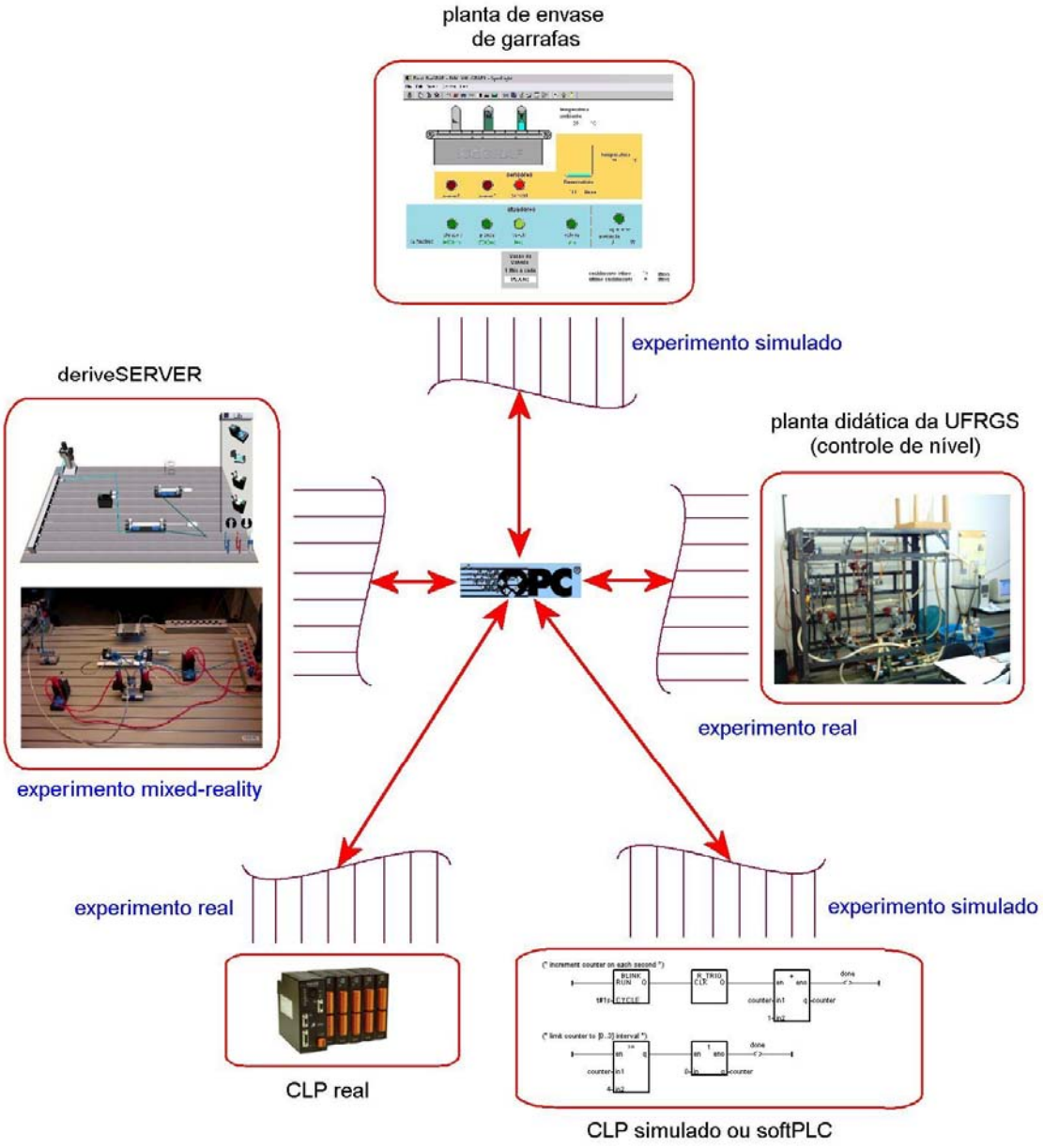

Figura 67. OPC unindo diferentes experimentos.

Na Figura 67, pode-se notar todos os componentes que podem ser utilizados para montar o sistema de realidade mista com componentes intercambiáveis. A interface comum de comunicação é o OPC que “une” diferentes experimentos e componentes.

5.7.1 Protótipo 1

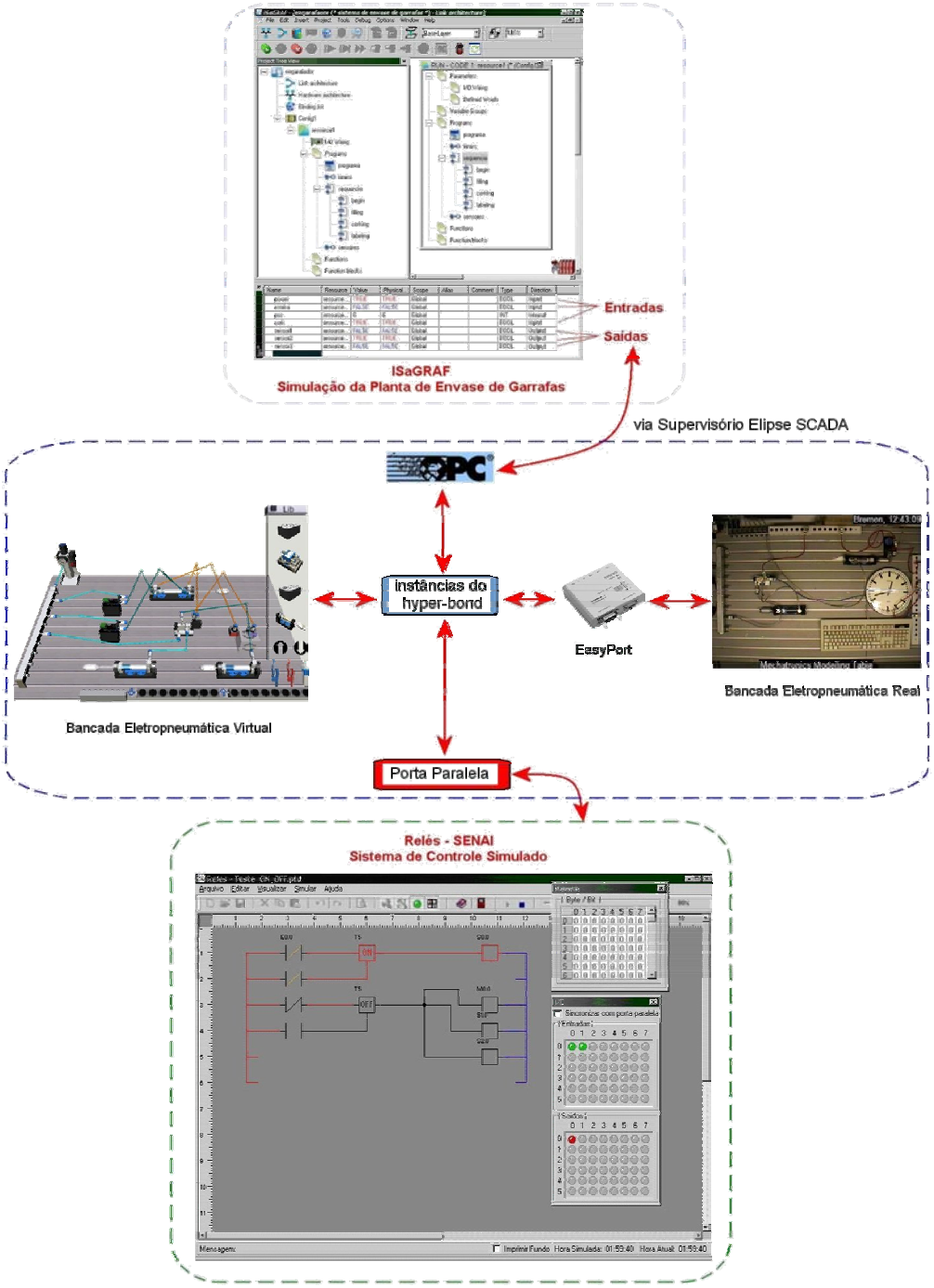

Figura 68. Protótipo 1.

O protótipo 1 faz uso do experimento remoto de mecatrônica (sistema *deriveSERVER)* do projeto de laboratórios remotos do SENAI. Com este protótipo, pode-se combinar todos os tipos possíveis de componentes, propostos na arquitetura de componentes intercambiáveis.

A construção faz uso *Elipse SCADA,* como centralizador da comunicação OPC e do experimento, e possibilita combinar o *ISaGRAF* (modelo da planta de envase), o experimento de remoto de mecatrônica e o *Relés* (com a lógica de relés). A bancada virtual serve de meio para conectarmos diversas variáveis *booleanas* externas através de diversas instâncias de *hyper-bonds* tanto locais quanto remotas.

A Figura 68 mostra a configuração do sistema usado neste protótipo, onde pode-se dizer que as três estações da planta de envase são direcionadas a equipamentos reais no experimento remoto de mecatrônica (cilindros pneumáticos). O simulador do SENAI pode controlar os equipamentos virtuais ou reais através da *I/O* da porta paralela, lida pela interface *hyper-bond*.

5.7.2 Protótipo 2

O protótipo 2 procura unir todas as funcionalidades dos estudos de caso anteriores de forma a produzir o máximo de vantagens. Nele temos a planta de envase como elemento central e conectada a ela diversos sistemas que podem ser utilizados de acordo com a seleção do usuário remoto (através da interface do experimento integrado no MOODLE).

Com este protótipo obtem-se um experimento completamente misto, combinando-se a planta de envase com experimentos reais da planta didática FF, planta térmica e atuadores reais do experimento remoto de mecatrônica. Deste modo, a planta de envase poderia ter um controle de nível de reservatório real (planta didática FF), um controle de temperatura real (planta térmica) e um acionamento de prensa real (cilindro pneumático ligado ao experimento de eletro-pneumática). Ainda ligado a este experimento pode-se ter um sistema de automação simulado remoto conectado ao servidor por meio de uma interface *hyper-bond* remota.

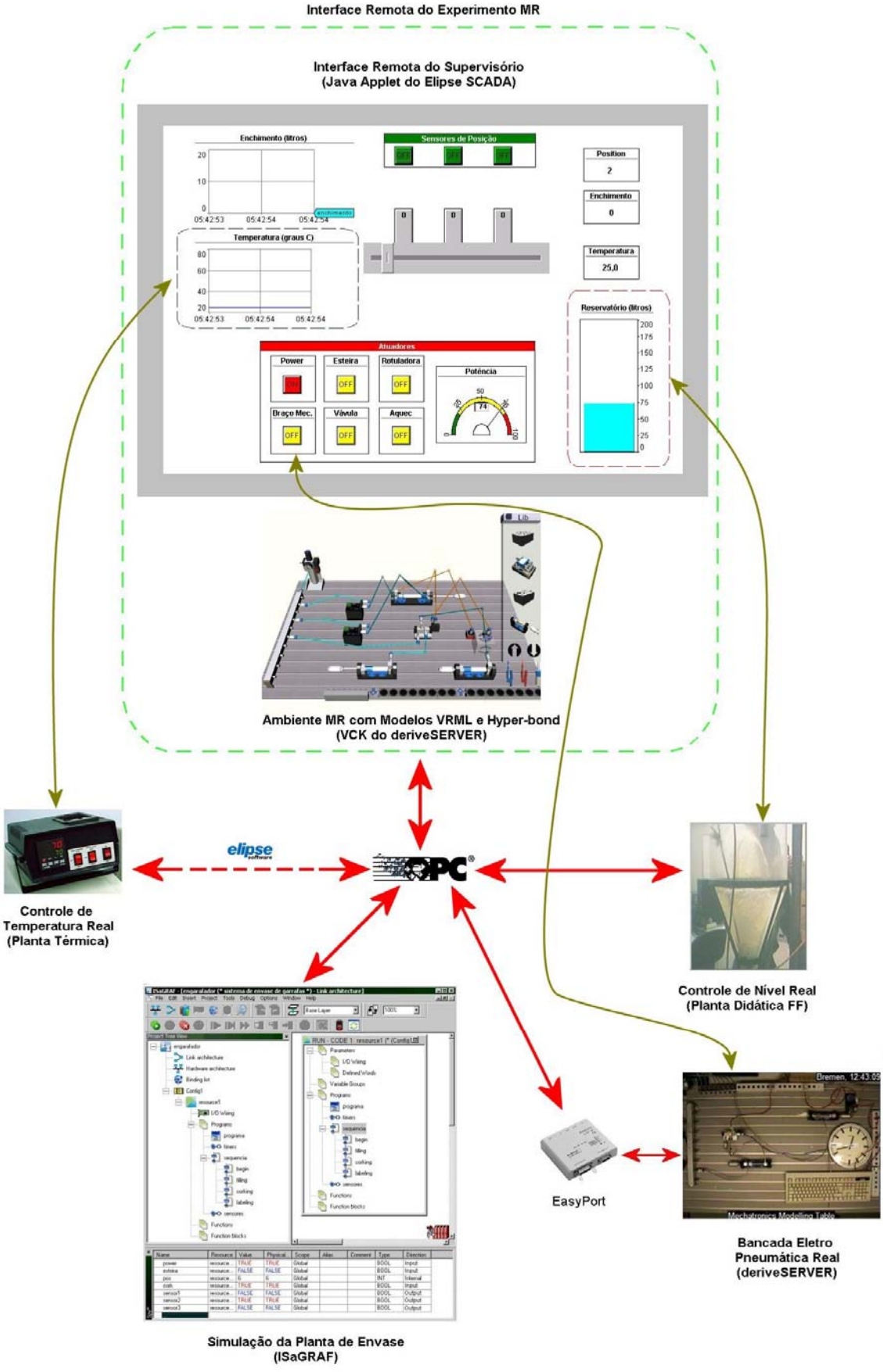

Figura 69. Interações do Protótipo 2.

Na Figura 69 está representada todas as possibilidades de combinações deste sistema. A seleção dos diferentes elementos utilizados é feita através da escrita no banco de dados remoto, o qual é lido no servidor pelo *Elipse SCADA* que seleciona (direciona) os dispositivos que irão ser usados no experimento com base na seleção passada para o banco de dados. A interface remota conta tanto com *Applets Java* gerados pelo *Elipse* para a planta de envase (as figuras foram simplificadas para melhorar o rendimento) quanto com a bancada virtual no caso de interação de *software*/*hardware* remoto. Com a interface *hyper-bond* de cliente OPC, pode-se usar qualquer *software* ou *hardware* que possua um servidor OPC para controlar as variáveis envolvidas na simulação. No caso de usar a interface *hyper-bond* com interação com a porta paralela podem ser usados o *Relés* do SENAI assim como qualquer *hardware* conectado a porta paralela para efetuar tanto controle real/virtual quanto mais elementos reais/virtuais. O ponto positivo desta construção é que pode-se interligar diversos equipamentos reais e virtuais em um mesmo ensaio. Os controladores são distribuídos e podem ser personalizados pelo usuário remoto livremente.

6 CONCLUSÕES

A comunidade científica anseia por uma nova metodologia de ensino de laboratórios remotos. Esforços de instituições de ensino no mundo inteiro foram apresentados no que poderíamos chamar de intercâmbio educacional de experimentos remotos e sistemas de aprendizagem. Comprovadamente pode-se adaptar diversos experimentos existentes e transformá-los em experimentos remoto, atendendo a uma faixa maior de estudantes, e podendo proporcionar maior segurança a equipamentos da instituição utilizados na experiência. Apesar do alto custo de desenvolvimento, os experimentos remotos com plataformas de aprendizagem são amplamente benéficos para a educação.

A arquitetura proposta supre uma grande necessidade causada pelo uso de experimentos remotos, atendendo a falhas educacionais, e podendo ser implantada em cursos de educação à distância (EAD) com o uso de experimentos remotos. As aplicações deste trabalho não se limitam somente às instituições de ensino, pois pode-se utilizá-lo no treinamento de assistência técnica remota em ambientes industriais.

A adaptação de sistemas existentes a novas tecnologias, como o *mixed reality* e os componentes intercambiáveis, proporciona pioneirismos sempre prezados em instituições de ensino a nível internacional. A utilização de plataformas de aprendizagem com propostas coerentes e colaborativas para a educação representa adaptação padrões mundiais de aprendizagem.

Nota-se que este trabalho está inserido em dois projetos internacionais que contam com parceiros reconhecidos mundialmente. Um dos projetos, o consórcio *RExNet*, visa a criação de uma rede mundial de experimentos remotos que serão compartilhados para a utilização em cada instituição, de forma a auxiliar e melhorar a educação na qual o experimento se insere. O outro projeto, da UFRGS com o SENAI e a Universidade Bremen,

visa a criação e adequação do experimento remoto de mecatrônica para ser usado em aulas de eletro-pneumática no SENAI.

Todos os estudos de caso e protótipos estarão disponíveis para serem usados em disciplinas oferecidas na UFRGS de maneira a incrementar os cursos de sistemas de automação e controle existentes. O estudo de caso do projeto de laboratório remotos do SENAI será exaustivamente manuseado por alunos do curso técnico em Caxias do Sul. Esse uso exaustivo servirá como validação e futuro aprimoramento do sistema, assim como da arquitetura.

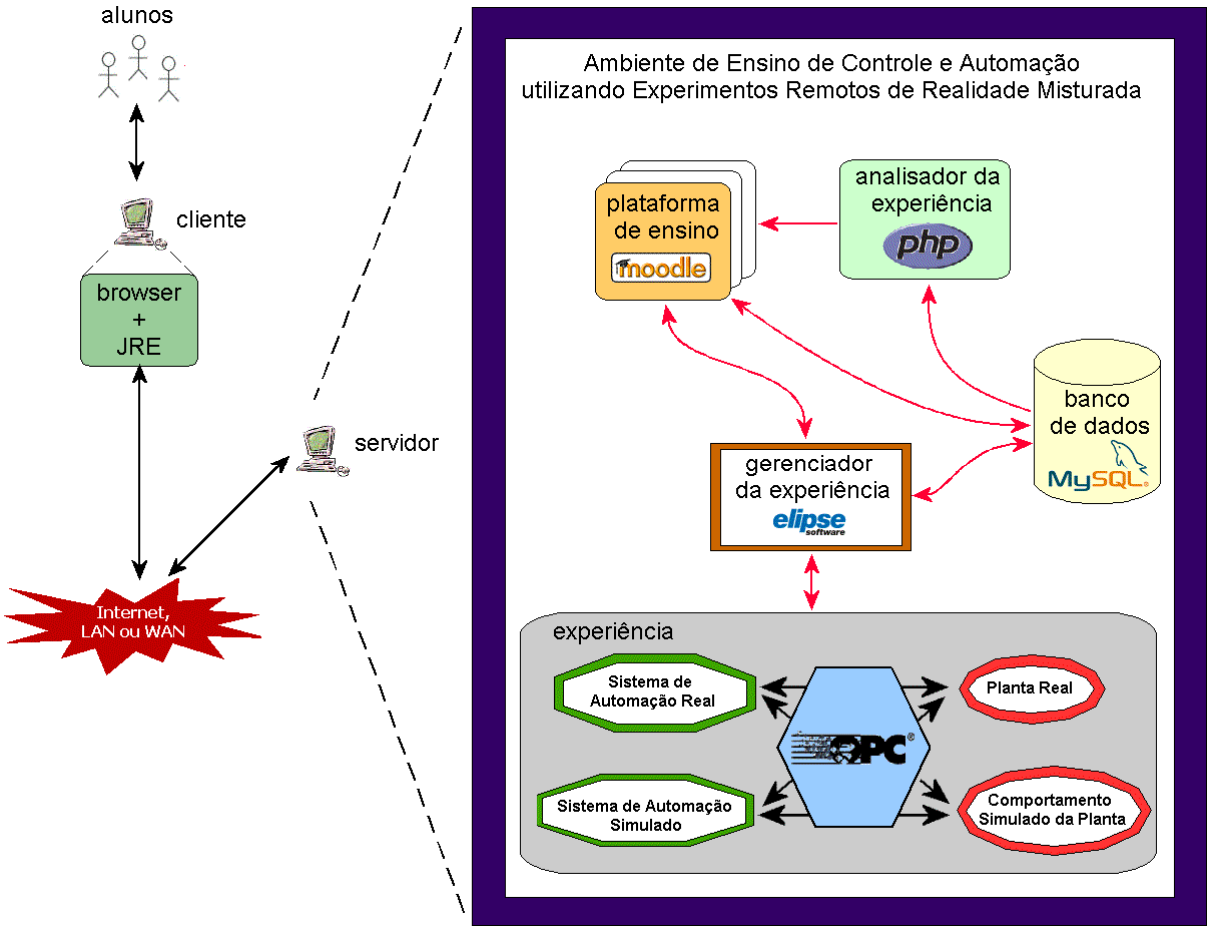

Figura 70. Estudo de caso

A planta FF da UFRGS integrada completamente ao ambiente virtual de aprendizagem MOODLE serve de estudo de caso principal para a validação da arquitetura. O conceito de

realidade mista foi incluído na atual planta FF para seguir o planejamento da arquitetura, o mesmo acontecendo com os componentes intercambiáveis. O sistema tutorial e analisador de experiência, integrado no MOODLE, foi desenvolvido especificamente para analisar sistemas de controle que envolvam projetos de controladores para sistemas contínuos (dinâmicos). A Figura 70 resume todas as funcionalidades e componentes envolvidos nos estudos de caso da arquitetura proposta com as tecnologias idealizadas neste estudo.

Tabela 6. Características dos Estudos de Caso.

Característica	**Planta FF**	**Planta Térmica**	**Experimento Mecatrônica**	**Planta de Envase**
Topologia de Cliente Remoto	*thin-client*	*thin-client*	*fat-client*	*thin-client*
Simulador Testado	ISaGRAF	ISaGRAF	ISaGRAF, Relés	ISaGRAF
Analisador de Experiência	Sim	sim	não	Sim
Sistema Tutorial (*Feedback*)	aconselhamento de material educacional	aconselhamento de material educacional	nenhum	aconselhamento de material educacional
Tipo de sistema	Contínuo	contínuo	discreto	discreto e contínuo
Execução do Controlador	software ou hardware	software ou hardware	software ou hardware	software ou hardware
Controle que pode ser utilizado	malha aberta ou malha fechada com controlador PID	malha aberta ou malha fechada com controlador PID	qualquer (projeto definido pelo cliente)	Manual
Tipo de planta utilizada	real ou simulada	real ou simulada	qualquer (depende dos equipamentos reais disponíveis)	Simulada

A planta de envase simulada, com a arquitetura local, foi pioneira para identificar erros de simulação provocados por defeitos no sistema de automação projetado pelos alunos, mas ela não oferece nenhum tipo de retorno educacional, pois somente aponta o erro ocasionado na simulação.

A Tabela 6 analisa características que apontam vantagens e desvantagens de cada um dos estudos de caso e a Tabela 7 compara os protótipos apresentados.

Tabela 7. Características dos protótipos.

Característica	**Protótipo 1**	**Protótipo 2**
Topologia de Cliente Remoto	*fat-client*	*thin-/fat-client*
Simulador Testado	ISaGRAF, Relés	ISaGRAF e Relés
Analisador de Experiência	componentes separados podem ser analisados	componentes separados podem ser analisados
Sistema Tutorial (*Feedback*)	componentes separados podem gerar feedback	componentes separados podem gerar feedback
Tipo de sistema	componentes discretos e contínuos	componentes discretos e contínuos
Execução do Controlador	*software e/ou hardware*	*software e/ou hardware*
Controle que pode ser utilizado	*variados* *(componentes utilizados podem possuir controlador próprio)*	*variados* *(componentes utilizados podem possuir controlador próprio)*
Tipo de planta utilizada	*real e/ou simulada*	*real e/ou simulada*

A necessidade de uma arquitetura flexível só foi identificada graças a um estudo do estado da arte realizado na disciplina de Trabalho Individual e posto em prática na disciplina de estágio em docência. O estudo culminou no desenvolvimento da planta simulada de envase de garrafas. O estudo de caso da planta de envase requisitou analisar o protocolo de comunicação OPC o que se tornou a base de comunicação dos componentes intercambiáveis. A planta de envase é utilizada em disciplinas de sistemas de automação na graduação e pós-graduação, demonstrando o caráter útil e versátil da simulação. Este estudo de caso possibilitou a criação de um curso adicional no MOODLE chamado de norma IEC61131-3, onde um material didático sobre a norma e sobre a planta de envase foi criado. Os estudantes de sistemas automação podem interagir com a planta de envase através do MOODLE que oferece uma interface remota com a planta de envase simulada ou “links” para instalação do simulador localmente.

A constituição da arquitetura utiliza diversos *softwares* e trabalhos já desenvolvidos anteriormente por outros desenvolvedores e instituições, apesar disso, esta apresenta aspectos inéditos de integração com componentes intercambiáveis e analisadores de experiência. Um guia de aprendizagem mais elaborado deverá ser futuramente desenvolvido para proporcionar melhorias de identificação de falhas de aprendizagem e sugestões aprimoradas embora o sistema vigente funcione corretamente.

O MOODLE é usado largamente na comunidade cientifica acadêmica e representa um avanço nos tradicionais sistemas tutoriais que se utilizam apenas de materiais educacionais dispostos em *HTML*s. A instalação desse *software, no entanto* não propicia geração automática de cursos que foram desenvolvidos, contribuindo para este trabalho. Os cursos contidos no MOODLE foram compilados com o intuito de facilitar o entendimento das experiências remotas das UFRGS. Os cursos apresentam um material elaborado com diversas ilustrações para facilitar a transferência do conhecimento no ambiente colaborativo de

aprendizagem MOODLE. O material educacional contido nos cursos soma comparativamente mais de 50 páginas de livros e apostilas didáticas.

O analisador de experiência que está integrado na arquitetura oferece métricas de sistemas de controle associadas aos controladores projetados de maneira simples, não obstante seu desenvolvimento contenha aproximadamente 150 linhas de código PHP.

O sistema de guia de experiência também foi desenvolvido com o intuito de preencher a arquitetura com detectores de falhas na experiência, da mesma forma que no sistema de envase, isto é, com comparadores de resultados e especificações previamente estabelecidas.

O estudo de caso da planta FF com misturas de sistemas de automação simulados foi inteiramente baseado em uma comunicação preexistente do sistema SCADA com o configurador de rede *FF*, para tornar o sistema interativo com simulações de sistemas de controle, algumas pequenas modificações na programação do sistema SCADA e da interface *Web* foram desenvolvidas para efetuar o chaveamento automatizado.

O protótipo 1 possui o maior potencial de interatividade entre os protótipos e estudos de caso desenvolvidos e, apesar de se basear em um experimento *mixed-reality* “pronto”, foi expandido e agora se torna atrativo do ponto de vista de flexibilidade e uso por usuários remotos. A expansão só foi possível por meio de reprogramação de parte de seu *software* e desenvolvimento de uma nova interface de aquisição de dados (*hardware*) de custo muito inferior à utilizada originalmente. A reprogramação exigiu um estudo profundo do sistema existente e das interações entre seus componentes. Novos *softwares* foram integrados somando mais de 500 linhas com *DLL*s e executáveis para gerenciamento do novo *hardware*. A nova interface OPC do sistema possibilita integração completa na arquitetura proposta.

O protótipo 2 reúne as qualidades de todos os outros estudos de caso apresentados e se mostra extremamente eficiente para demonstrar a capacidade de intercambiar componentes simulados com reais, fazendo com que a experiência se torne distribuída, podendo-se utilizar

uma gama muito grande de equipamentos reais que normalmente não estão conectados numa mesma experiência. O controle também está distribuído e podem se utilizar diversas técnicas de controle para se automatizar os processos envolvidos.

O conceito para avaliação interdisciplinar de ferramentas de aprendizagem (*Concept for Interdisciplinary Evaluation of Learning Tools* – CIELT), utilizado no Projeto Lab@future (LAB@FUTURE, 2006; MWANZA, 2003) é ideal para avaliar a arquitetura proposta e futuramente para aprimorá-la. A Figura 71 exemplifica o conceito CIELT (GRUND, 2002) usado na avaliação de sistemas que envolvem o aprendizado utilizando sistemas automatizados.

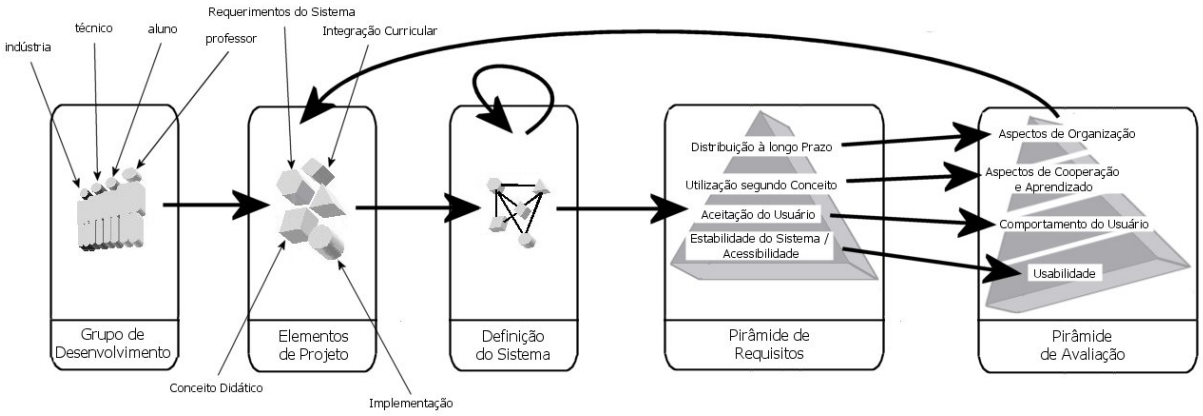

Figura 71. Metodologia de avaliação.

A arquitetura também implementa uma nova metodologia para criação de experimentos remotos com ênfase educacional utilizando equipamentos que normalmente não estão combinados.

A Tabela 8 identifica vantagens da arquitetura proposta sobre a implementação da dissertação de Zeilmann (Zeilmann, 2002), que é o trabalho correlato de maior importância na UFRGS. Nota-se que a arquitetura proposta estende o trabalho de Zeilmann (Zeilmann, 2002) em diversos aspectos.

Tabela 8. Tabela comparativa da arquitetura proposta.

Implementação de Zeilmann	*Arquitetura Proposta*
Experimento remoto real	Experimento de realidade mista remoto
Acesso limitado de 1 usuário por vez	Acesso depende da utilização dos equipamentos reais
Sem mecanismo de aprendizagem associado	Mecanismo de aprendizagem integrado
Experiência não produz nenhuma métrica. Aluno não recebe nenhuma avaliação da experiência.	Analisador da experiência produz métricas associadas ao experimento realizado.
Depois de efetuada a experiência, aluno não recebe nenhum aconselhamento.	Sistema tutor detecta falha(s) na experiência e aconselha material de acordo com a(s) falha(s).
Material educacional não organizado.	Plataforma de aprendizagem integrada propicia organização de material educacional, assim como registros de materiais visualizados, questionários e aprendizagem colaborativa e ativa dos alunos.
Sistema de controle restrito a controladores PID.	Com a utilização de componentes intercambiáveis pode-se utilizar qualquer tipo de controlador existente.
Sistema de *booking* não organizado e estático.	Sistema de *booking* automatizado integrado na plataforma de aprendizagem.
Cadastro de usuários remotos para acesso ao experimento é estático.	Acesso de usuários remotos ao experimento é gerenciado pela plataforma de aprendizagem dinamicamente.
Planta didática é usada para experimentação de alunos de cursos da UFRGS.	Experimentos usados na validação da arquitetura serão utilizados por uma rede de experimentação remota internacional.

As desvantagens da arquitetura, no entanto não estão representadas na Tabela 8, pois se devem à escolha da topologia e das limitações técnicas dos equipamentos dos experimentos reais. Podem-se citar as seguintes desvantagens da arquitetura em relação a outros tipos de arquiteturas:

- Limitação temporal dos equipamentos reais:

  Recursos precisam ser reservados, pois somente um aluno (ou grupo) pode ter acesso ao experimento durante sua execução. Em experiências puramente simuladas não há esta preocupação.

- Limitação de projeto de controladores:

  De forma a utilizarmos a topologia thin-client projetar-se um controlador que não esteja parametrizado (pré-estabelecido) no servidor depende de uma interface mais elaborada onde se possa projetar (interface gráfica de programação) ou enviar o projeto de um controlador (arquivo com a lógica) para o servidor. O projeto e execução do controlador no cliente fazem necessário o uso de softwares adicionais no cliente, diminuindo a interatividade do usuário com o experimento.

É importante notar que ainda há muito a ser feito para completar a validação desse trabalho, mas resultados preliminares com estudantes da UFRGS, utilizando a arquitetura proposta e a planta de envase simulada, indicam um futuro promissor aos estudos realizados devido à aceitação dos alunos.

Ainda devem ser direcionados melhoramentos, em trabalhos futuros, para tornar os experimentos livres de licenças comercias de *softwares* empregados, assim como um sistema de direcionamento de material mais amplo e inteligente, utilizando não somente comparadores para avaliação da experiência. O sistema, assim como se encontra, toma decisões baseadas em somente uma experiência. É óbvio que baseado em uma única experiência, não pode-se avaliar o aprendizado do aluno, mesmo assim, o sistema oferece uma resposta e sugere material de forma a melhorar a transferência de conhecimento.

Frutos deste trabalho já foram colhidos através da publicação de dois artigos científicos no Congresso Internacional de Problemas na Manufatura (INCOM), propiciado

pela Federação Internacional de Controle Automático (IFAC), em 2006. O primeiro artigo (SCHAF, 2006a), apresentado na seção especial de *e-learning*, trata justamente da arquitetura proposta nessa dissertação. Já o segundo (SCHAF, 2006b), apresentado na seção especial de realidade mista, descreve o projeto desenvolvido em conjunto com o SENAI para tornar o experimento remoto de mecatrônica mais versátil, interativo e flexível. A publicação desses artigos demonstra o caráter inovador e pertinente do texto apresentado. No *WIER IV* (Workshop Internacional de Experimentação Remota), que teve a Universidade do Porto como organizador, foram apresentados vídeos que comprovam o funcionamento dos estudos de caso.

Outro artigo científico publicado no CBA 2006 em Salvador descreve as modificações efetuadas na arquitetura da planta FF e apresenta um novo ensaio de controle, chamado de Bang-bang (ECKHARD, 2006). Este tipo de controle ainda não foi utilizado na arquitetura proposta, mas também deve ser incorporado futuramente.

A continuação desse trabalho se torna viável através de um possível doutorado com base na proposta encaminhada junto ao PPGEE da UFRGS.

REFERÊNCIAS

ALBU, M. M. et al. Embedding Remote Experimentation in Power Engineering Education. **IEEE Transactions on Power Systems**, New York, v. 19, p. 139 – 143, Feb. 2004.

ALLEGRO – a lifelong education and training environment for microelectronic. **Allegro Final Report**, European Commission, Directorate. General XXII - Education, Training and Youth, CONTRACT N°: P / 99 / 1/ 075270 / PI / III.3.a / FPC, 1999. Disponível em: <http://www.fe.up.pt/~allegro/finalreport.pdf>. Acesso em: 10 de Dez. 2005.

ALVARÉS, A. J.; FERREIRA, J. C. E. Metodologia de Implantação de Laboratórios Remotos via Internet na Área de Automação e Manufatura. In: CONGRESSO BRASILEIRO DE ENGENHARIA DE FABRICAÇÃO, 2., 2003, Uberlândia, Brazil. **Anais...** [S. l.: s. n.], 2003.

ALVES, G. R. et al. Remote Experimentation Network Yielding an Inter-University Peer-to-Peer e-Service. In: IEEE INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA), 10., 2005, Catania, Italy. **Proceedings...** New York: IEEE, 2005. v. 2, p. 1023 – 1030.

ATKAN, B.; BOHUS, C. A.; CROWL, L. A.; SHOR, M. H. Distance Learning Applied to Control Engineering Laboratories. **IEEE Transactions on Education**, New York, v. 39, p. 320 – 326, Aug. 1996.

AUER, M.; PESTER, A.; URSUTIU, D.; SAMOILA, C. Distributed Virtual and Remote Labs in Engineering. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL TECHNOLOGY (ICIT), 2003, Maribor, Slovenia. **Proceedings...** New York: IEEE, 2003. v. 2, p. 1208 – 1213.

BAUDRILLARD, J. **Simulacra and Simulation**. Michigan, USA: University of Michigan Press, 1994. ISBN: 0472065211.

BAZANELLA, A. S.; SILVA, J. M. G. **Sistemas de Controle:** princípios e métodos de projeto. Porto Alegre, Ed. da UFRGS, 2005. ISBN: 85-7025-849-6 v.1

BILLINGHURST, M.; KATO, H. Mixed-reality: merging real and virtual worlds. In: INTERNATIONAL SYMPOSIUM ON MIXED-REALITY (ISMR), 1999, Yokohama, Japan. **Proceedings...** [S.l.]: Ohmsha-Springer Verlag, 1999. p. 261-284.

BOCKHOLT, U. et al. Augmented Reality for Enhancement of Endoscopic Interventions. In: IEEE ON VIRTUAL REALITY, 2003, Los Angeles, USA. **Proceedings...** New York: IEEE, 2003. p. 97 – 101.

BOHL, O.; SCHEUHASE, J.; SENGLER, R.; WINAND, U. The Sharable Content Object Reference Model (SCORM): a critical review. In: INTERNATIONAL CONFERENCE ON COMPUTERS IN EDUCATION (ICCE), 2002, Auckland, New Zealand. **Proceedings...** [S. L.: s. n.], 2002. v. 2, p. 950 – 951.

BORTZ, J.; DÖRING N. **Forschungsmethoden und Evaluation für Human Sozialwissenschaftler**. Berlin: Springer Verlag, 2002. 897 p. ISBN: 3540333053.

BRUNS, F. W.; ERBE, H. H. Mixed-reality with Hyper-Bonds – a means for remote labs. In: IFAC SYMPOSIUM ON INFORMATION CONTROL PROBLEMS IN MANUFACTURING (INCOM), 11., 2004, Salvador, Brazil. **Proceedings...** [S. l.]: IFAC, 2004. p. 55 – 68.

BÖHNE, A.; FALTIN, N.; WAGNER, B. Self-directed Learning and Tutorial Assistance in a Remote Laboratory. In: INTERACTIVE COMPUTER AIDED LEARNING CONFERENCE (ICL), 2002, Villach, Austria. **Proceedings...** New York: ACM Press, 2002.

CADOLINI, P.; DAVOLI, F.; MARESCOTTI, G.; MARYNI, P. Developing a Distance Learning System using Java Applets. In: GLOBAL TELECOMMUNICATIONS CONFERENCE (GLOBECOM), 1996, London, England. **Proceedings...** New York: IEEE, 1996, p. 63 – 66.

CASINI M.; PRATTICHIZZO D.; VICINO A. The Automatic Control Telelab: a web-based technology for distance learning. **IEEE Control Systems Magazine**, New York, v. 24, p. 36 – 44, Jun. 2004.

CASINI, M.; PRATTICHIZZO, D.; VICINO, A. The Automatic Control Telelab – a user-friendly interface for distance learning. **IEEE Transactions on Education**, New York, v. 46, n. 2, p. 3242 – 3247, May 2003.

CHICULITA, C.; FRANGU, L. A Web Based Remote Control Laboratory. In: WORLD MULTICONFERENCE ON SYSTEMIC, CYBERNETICS, INFORMATICS, 6., 2002, Orlando, USA. **Proceedings...** [S. l.]: Elsevier, 2002.

COOPER, M. The Challenge of Practical Work in a eUniversity - real, virtual and remote experiments. In: INFORMATION SOCIETY TECHNOLOGIES (IST), 2000, Nice, France. **Proceedings...** [S. l.: s. n.], 2000.

COOPER, M.; DONNELY, A.; FERREIRA, J. M. Remote Controlled Experiments for Teaching over the Internet: comparison of approaches developed in PEARL project. In: AUSTRALASIAN SOCIETY FOR COMPUTERS IN LEARNING IN TERTIARY EDUCATION (ASCILITE), 19., 2002, Auckland: Australia. **Proceedings...** Auckland: UNITEC Institute of Technology, 2002.

DAVOLI, F.; MARYNI, P.; PERRANDO, M.; ZAPPATORE, S. A General Framework for Networked Multimedia Applications Enabling Access to Laboratory Equipment: the LABNET project experience. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: CODING AND COMPUTING, 2001, Las Vegas, USA. **Proceedings...** New York: IEEE, 2001, p. 389 – 365.

DUAN, B.; LING, K. V.; HOSSEINI, H. Developing a Framework for Online Laboratory Learning Objects. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 10., 2003, Sentosa, Singapura. **Proceedings...** New York: IEEE, 2003, v. 3, p. 467 – 157.

ECKHARD, D.; SCHAF, F. M; SILVA, J.M. G.; PEREIRA, C. E. Uma Plataforma de Experimentação Remota para Ensino de Controle. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA (CBA), 16., 2006, Salvador, Brazil. **Anais...** [S. l.]: IFAC, 2006. p. 2305 – 2310, 2006.

FALTIN, N.; BÖHNE, A.; TUTTAS, J.; WAGNER, B. Distributed Team-Learning in an Internet-Assisted Laboratory. In: INTERNATIONAL CONFERENCE ON ENGINEERING EDUCATION, 2002, Manchester, UK. **Proceedings...** Manchester: University of Manchester Press, 2002.

FAUST M.; YOO, Y.-H. Haptic Feedback in Pervasive Games. In: INTERNATIONAL WORKSHOP ON PERVASIVE GAMING APPLICATIONS, 3., 2006, Dublin, Ireland, May 2006, **Proceedings...** New York: IEEE Computer Society, 2006.

FERREIRA, J. M. M.; COSTA, R. J.; ALVES G.; COOPER, M. The PEARL Digital Electronics Lab: full access to the workbench via the web. In: ANNUAL CONFERENCE ON INNOVATIONS IN EDUCATION FOR ELECTRICAL AND INFORMATION ENGINEERING (EAEEIA), 13., York, England, Apr. 2002, **Proceedings...** [S. l.: s. n.], 2002.

FERREIRA, J. M.; MÜLLER, D. The MARVEL EU Project: a social constructivist approach to remote experimentation. In: REMOTE ENGINEERING AND VIRTUAL INSTRUMENTATION INTERNATIONAL SYMPOSIUM (REV), 1., 2004, Villach, Austria. **Proceedings...** [S. l.: s. n.], 2004.

FOULK, P.; DESMULLIEZ, M.; MACKINNON, L.; FERREIRA, J. M. The ASTEP Educational Multimedia Framework. In: EUROPEAN WORKSHOP ON MICROELECTRONICS EDUCATION (EWME), 2., 1998, Enschede, Holand. **Proceedings...** [S. l.]: Kluwer Academic Publishers, 1998.

GONZÁLES, V. M.; MATEOS, F.; NG, A. H. C. MLAV: the object-oriented methodology of the virtual automation lab. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2004, New Orleans, USA. **Proceedings...** New York: IEEE, 2004. v. 5, p. 5153 – 5158.

GRUND, S.; WINDLINGER, L.; GROTE, G. Concept for Interdisciplinary Evaluation of Learning Tools (CIELT). In: INTERNATIONAL CONFERENCE ON NEW EDUCATIONAL ENVIRONMENTS, 4., 2002, Lugano: Switzerland. **Proceedings...** [S. l.]: Elsevier, 2002. p. 2.4, 11 – 2.4, 14.

GUIMARÃES, E. et al. REAL: A Virtual Laboratory for Mobile Robot Experiments. **IEEE Transactions on Education**, New York, v. 46, n. 1, p. 37 – 42, Feb. 2003.

HARTMANN, S. The World as a Process: simulations in the natural and social sciences. In: MODELLING and Simulation in the Social Sciences from the Philosophy of Science Point of View. Dordrecht, Germany: Kluwer Verlag 1996. p. 77-100.

HINE, N. et al. **Modelling the Behaviour of Elderly People as a means of monitoring well being**. Lecture Notes in Computer Science. Berlin: Springer Verlag, 2005. v. 3538, p. 241 – 250. ISBN: 3-540-27885-0

Java Distributed Data Acquisition and Control **(**JDDAC), **User's Guide**, Version 2006-03-14. Disponível em: <https://jddac.dev.java.net/files/documents/1315/31001/JDDAC-20060314.pdf >. Acesso em: 20 de Maio 2006.

KARNOPP, D.; MARGOLIS, D. L.; ROSENBERG, R. C. **System Dynamics**: a unified approach. New York: John Wiley, 1990. ISBN: 0471621714.

KHARNIS, A.; RIVERO, D. M.; RODRÍGUEZ, F.; SALICHS, M. Pattern-based Architecture for Building Mobile Robotics Remote Laboratories. In: IEEE INTERNACIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2003, Taipei, Taiwan. **Proceedings...** New York: IEEE, 2003. v. 3, p. 3284 – 3289.

KIKUCHI, T.; KENJO, T. Distance Learning applied to a Small Motor Laboratory – insight into the stepping motor. In: IEEE CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS (SMC), 1999, Tokyo, Japan. **Proceedings...** New York: IEEE, 1999. v. 2, p. 259 – 264.

KO, C. C. et al. A Web-Based Virtual Laboratory on a Frequency Modulation Experiment. **IEEE Transactions on System, Man and Cybernetics**, New York, v. 31, n. 3, p. 295 – 303, Aug. 2001.

KO, C. C. et al. Development of a Web-based Control Experiment for a Coupled Tank Apparatus. **IEEE Transactions on Education**, New York, v. 44, n. 1, Feb. 2001.

KOJIJANCIC, S. Online experiments in Physics and Technology Teaching. **IEEE Transactions on Education,** New York, v. 45, n. 1, p. 26 – 32, Feb. 2002.

KRETSCHMER, U. et al. Virtual Reality, Archeology, and Cultural Heritage. In: INTERNATIONAL WORKSHOP ON PERVASIVE GAMING APPLICATIONS, 2001, Glyfada, Greece. **Proceedings...** [S. l.: s. n.], 2001.

LAURILLARD, D. **Rethinking University Teaching**. London: Routledge Falmer Press, 2001. ISBN: 978-0415256797.

LEIDNER, D.; S. JARVENPAA, The Use of Information Technology to Enhance Management School Education: a theoretical view. **MIS Quarterly**, v. 19, p. 265 – 292, Sep. 1995.

LI, L.; WANG, F.-Y.; LAI, G.; WU, F. Online Autonomous Guidance System for Remote Experiments in Control Engineering. IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS (SMC), 2003, Washington, USA. **Proceedings...** New York: IEEE, 2003. v. 3, p. 2444 – 2449.

LIPOVSZKI, G.; ARADI, P. Simulating Complex Systems and Processes in LabVIEW. **Journal of Mathematical Sciences**, [S. l.], v. 132, n. 5, p. 629-636, Feb. 2006.

MICHAELIDES, I.; ELEFTHREIOU, P.; MÜLLER, D. A Remotely Accessible Solar Energy Laboratory – a distributed learning experience. In: REMOTE ENGINEERING AND VIRTUAL INSTRUMENTATION INTERNATIONAL SYMPOSIUM (REV), 1., 2004, Villach, Austria. **Proceedings...** [S. l.: s. n.], 2004.

MILGRAM, P. et al. Merging Real and Virtual Worlds. In: IMAGINA, 1995, Monte Carlo. **Proceedings...** [S. l.: s. n.], 1995.

MILGRAM, P.; KISHINO, F. A Taxonomy of Mixed-reality Visual Displays. **IEICE Transactions Information Systems**, New York, v. E77-D, n. 12, p. 1321-1329, 1994.

MITSUI, H.; SUGIHARA, H.; KOIZUMI, H. A Scheduling Method of Distance Learning Classes Including Remote Experiments. In: INTERNATIONAL CONFERENCE ON

ADVANCED INFORMATION NETWORKING AND APPLICATION (AINA), 2004, Fukuoka, Japan. **Proceedings...** New York: IEEE, 2004. v. 1, p. 439 – 444.

MWANZA, D.; ENGESTRÖM, Y. Pedagogical Adeptness in the Design of E-learning Environments: experiences from the Lab@Future project. In: INTERNATIONAL CONFERENCE ON E-LEARNING IN CORPORATE, GOVERNMENT, HEALTHCARE, & HIGHER EDUCATION, 2003, Phoenix, USA. **Proceedings...** [S. l.: s. n.], 2003. v. 2, p. 1344 – 1347.

MÜLLER, D. Creating Hybrid Learning Spaces for Mechatronics Education. In: INTERACTIVE COMPUTER AIDED LEARNING CONFERENCE (ICL), 2006, Villach, Austria. **Proceedings...** New York: IEEE, 2006.

NIEH, J.; YANG, S. J.; NOVIK, N. **A Comparison of Thin-Client Computing Architectures.** TECHNICAL REPORT CUCS-022-00, Columbia University, Nov. 2000. Disponível em <http://www.ncl.cs.columbia.edu/publications/cucs-022-00.pdf >. Acesso em: 10 de Jan. 2006.

OVERSTREET, J. W.; TZES, A. An Internet Based Real-Time Control Engineering Laboratory. **IEEE Control Systems Magazine**, New York, v. 19, p. 19 – 34, 1999.

POINDEXTER, S. E.; HECK, B. S. Using the Web in your Courses: what can we do? what should you do? **Control on the Web Magazine**, 1999. Disponível em: <http://users.ece.gatech.edu/~bonnie/web-use/>. Acesso em: 5 Nov. 2005.

SALAZAR-SILVA, G.H.; MARTINEZ-GARCIA, J.C.; GARRIDO, R. Enhancing Basic Robotics Education on the Web. In: AMERICAN CONTROL CONFERENCE, 1999, San Diego, USA. **Proceedings...** NEW YORK: IEEE, 1999. v. 2., p. 1470 – 1471.

SCHAF, F.M.; PEREIRA, C. E. PID Controller Tuning Remote Experiment with Learning Environment Integration. In: IFAC SYMPOSIUM ON INFORMATION CONTROL PROBLEMS IN MANUFACTURING (INCOM), 12., 2006, Saint Etienne, France. **Proceedings...** [S. l.]: Elsevier, 2006a. v. 2, p. 753 – 758.

SCHAF, F.M.; PEREIRA, C. E. Mixed Reality Experiment for Distributed Learning Environment in Pneumatics Systems. In: IFAC SYMPOSIUM ON INFORMATION CONTROL PROBLEMS IN MANUFACTURING (INCOM), 12., 2006, Saint Etienne, France. **Proceedings...** [S. l.]: Elsevier, 2006b. v. 2, p. 261 – 266.

SCHMID, C.; ALI, A. A Web-based System for Control Engineering Education. In: AMERICAN CONTROL CONFERENCE, 2000, Chicago: USA. **Proceedings...** New York: IEEE, 2000. v. 5, p. 3463 – 3467.

SCORM DOCUMENTATION, **SCORM Documentation Public Draft 3rd Edition**. ADL Secretary of Defense – USA, 2004. Disponível em: <http://www.adlnet.gov/news/articles/375.cfm>. Acesso em: 25 de Mar. 2005.

SHEN, H. et al. Conducting Laboratory Experiments over the Internet. **IEEE Transactions on Education**, New York, v. 42, p. 180 – 185, Aug. 1999.

SILVA, J. M. G.; BAZANELLA, A. S. Um Sistema Didático para o Ensino de Ajuste de Controladores PID. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA (CBA), 14., 2002, Natal, Brazil. **Anais...** [S. l.]: IFAC, 2002.

SILVA, J. M. G.; SILVEIRA, M. A. L.; PEREIRA, C. E. A Web Based Remote Multivariable Control Experiment. In: IFAC SYMPOSIUM ON INFORMATION CONTROL PROBLEMS IN MANUFACTURING (INCOM), 11., 2004, Salvador, Brazil. **Anais...** [S. l.]: IFAC, 2004.

SMITH, K.A. The Craft of Teaching Cooperative Learning: an active learning strategy. In: CONFERENCE FRONTIERS IN EDUCATION, 1999, Pittsburg: USA. **Proceedings...** New York: IEEE, 1998. p. 188 – 193.

SOBH, T. et al. Case Studies in Web-Controlled Devices and Remote Manipulation, Laboratory Based Distance Learning. In: WORLD AUTOMATION CONGRESS (WAC), 2002, Orlando, USA. **Proceedings...** New York: IEEE, 2002. v. 14, p. 435 – 440.

SRINAVASAGUPTA, D.; JOSEPH, B. An Internet-Mediated Process Control Laboratory. **IEEE Control Systems Magazine**, New York, v. 23, p. 11 – 18, Feb. 2003.

STERGIOULAS, L.K.; AHMED, H.; XYDEAS, C.S. Adaptive Web-based Engine for the Evaluation of eLearning Resources, **IEEE Educational Technology and Society**, New York, v. 5, p. 138 – 149, 2002.

SWAMY, N.; KULJACA, O.; LEWIS, F. L. Internet-Based Educational Control Systems Lab Using NetMeeting. **IEEE Transactions on Education**, New York, v. 45, n. 2, p. 145 – 151, May 2002.

TUTTAS, J.; WAGNER, B. Distributed Online Laboratories. In: INTERNATIONAL CONFERENCE ON ENGINEERING EDUCATION, 2001, Oslo, Norway. **Proceedings...** Oslo: Stipes Publishing, 2001.

VALERA, A.; VALLÉS, M.; DÍEZ, J. L. A Global Approach for Remote Simulation and Process Control. In: IFAC WORLD CONGRESS ON AUTOMATION AND CONTROL, 15., 2002, Barcelona, Spain. **Proceedings...** [S. l.]: IFAC, 2002. p. 76 – 81.

WATSON, K. Utilization of Active and Cooperative Learning in EE courses: three classes and the results. In: CONFERENCE FRONTIERS IN EDUCATION, 1995, Atlanta, USA. **Proceedings...** New York: IEEE, 1995. v. 2, p. 3c2.1 – 3c2.6.

WU, D. et al. Streaming Video over the Internet: approaches and directions. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v. 11, n. 3, p. 282 – 300, Mar. 2001.

ZAGLER, W.L.; EDELMAYER, G.; MAYER, P. Studies on Vision Enhancement with Optoelectronical Devices. In: IEEE SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS, 10., 1997, Maribor, Slovenia. **Proceedings...** New York: IEEE, 1997. p. 28-33.

ZEILMANN, R. P. **Uma estratégia para controle e supervisão de processos industriais via Internet**, 2002. 141 p. Tese (Mestrado em engenharia) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

ZEILMANN, R. P.; SILVA, J. M. G.; BAZANELLA, A. S.; PEREIRA, C. E. Web-based Control Experiment on a Foundation Fieldbus Pilot Plant. In: IFAC INTERNATIONAL CONFERENCE ON FIELDBUS SYSTEMS AND THEIR APPLICATIONS, 2003, Aveiro, Portugal. **Proceedings...** [S. l.]: IFAC, 2003. p. 325 – 330.

ZEILMANN, R. P.; SILVA, J. M. G.; PEREIRA, C. E.; BAZANELLA, A. S. Uma Estratégia para Controle e Supervisão de Plantas Industriais através da Internet. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE (SBAI), 5., 2001, Canela, Brazil. **Anais...** [S. l.]: IFAC, 2001.

ZHANG, S. et al. NETLAB – an internet based laboratory for electrical engineering education. **Journal of Zheijang University**, 2004. Disponível em: <http://www.zju.edu.cn/jzus/2005/A0505/A050507.pdf >. Acessado em: 20 de Maio 2006.

ZHENG, L.; NAKAGAWA, H. OPC (OLE for process control) specification and its developments. In: SOCIETY OF INSTRUMENT AND CONTROL ENGINEERS ANNUAL CONFERENCE (SICE), 41., 2002, Osaka, Japan. **Proceedings...** [S. l.]: Springer Verlag, 2002. v. 2, p. 971 – 920.

APÊNDICE – SIMULADORES

Nesta seção um estudo sobre as funcionalidades de simuladores comerciais é apresentado de forma a tornar mais claras citações do trabalho.

MatLab – MathWorks

Tecnicamente o MatLab não é considerado um simulador, mas sim um ambiente numérico e uma linguagem de programação. Com o auxílio deste programa computacional (*software*) pode-se gerar e extrair modelos matemáticos que representam sistemas reais. Utilizando sua funcionalidade de programação, pode-se criar um simulador de qualquer experimento físico.

O *MatLab* é um produto comercial da *MathWorks* (MATHWORKS, 2006). Seu nome provém de **Mat**rix **Lab**oratory e foi criado por Cleve Moler, da Universidade do Novo México – EUA, nos anos 70 com o intuito de disponibilizar aos estudantes pacotes como o LINPACK e o EISPACK sem a necessidade de aprenderem a linguagem de programação *Fortran*. Logo que foi criado encontrou diversos usuários entre matemáticos. Mais tarde, encontrou nos projetistas de engenharia de controle os seus mais fortes usuários. O *software* hoje em dia está escrito em linguagem *C* com a qual tem fortes ligações.

Características Interessantes

Para programação o ambiente utiliza uma linguagem de programação própria, o *m*. Sua linguagem é bastante simples não necessitando de declarações de variáveis. As variáveis utilizadas na programação não possuem tipos, somente os valores em tempo de execução (*runtime*) armazenados nestas variáveis que possuem tipos. Assim, a linguagem se aproxima do PHP ou JavaScript na forma como trata as variáveis.

O ambiente é composto de uma máquina principal (*engine*) e diversos pacotes de ferramentas (*toolboxes*). Cada pacote tem uma funcionalidade diferente e aplicações específicas. Podem-se citar as seguintes categorias principais de pacotes: matemática básica e

otimização; estatística e análise de dados; projeto e análise de sistemas de controle; processamento de sinais e comunicação; processamento de imagens; medição e testes; biologia computacional; análise e modelagem financeira; distribuição de aplicações; alvos de distribuição de aplicações; relatórios e conectividade de base de dados.

Além dos pacotes ligados à máquina principal do MatLab existem alguns pacotes que estão diretamente relacionados ao Simulink.

Simulink

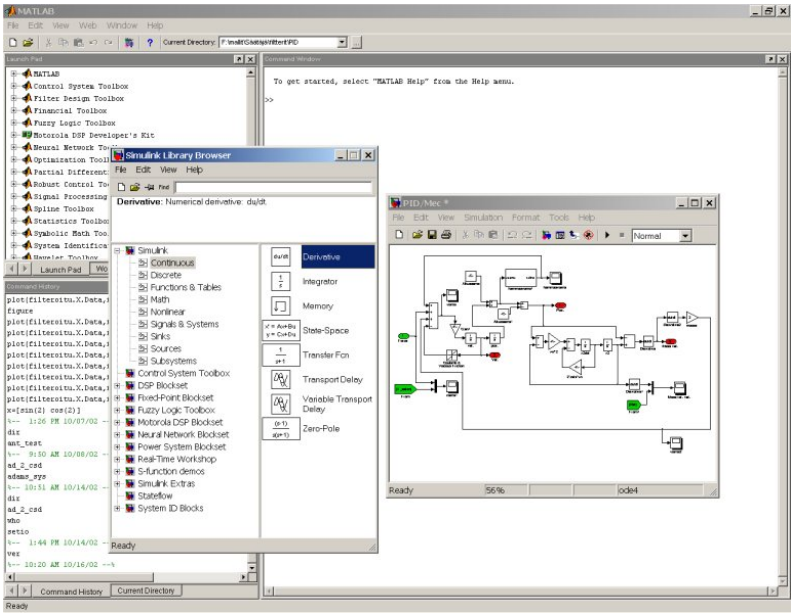

Figura 72. MatLab com Simulink.

Simulink é um módulo do MatLab que é uma plataforma multidomínio e projetado em modelos baseados em sistemas dinâmicos. Este módulo proporciona um ambiente gráfico interativo com uma configuração personalizada de bibliotecas de blocos funcionais que pode ser estendida para aplicações especializadas. Como esta ferramenta é mais ligada à simulação as categorias de pacotes são ligadas a esta funcionalidade como: modelagem de ponto fixo; modelagem baseada em eventos; modelagem física; gráficos de simulação; análise e projeto de sistemas de controle; processamento de sinais e comunicações; geração de códigos;

prototipação de controle rápido baseado em *PCs*; alvos embarcados; verificação, validação e testes. A Figura 72 ilustra o ambiente MatLab ao fundo com módulo Simulink na frente

O MatLab possui recursos adicionais que são muito desejáveis para processamento mais pesado, como a ampliação da máquina para computação distribuída. Um pacote especial trata da comunicação entre as máquinas que estão distribuídas.

Manipulação de Sistemas de Controle

Pode-se encontrar os seguintes pacotes, amplamente usados na área de sistemas de controle na categoria de projeto e análise de sistemas de controle: sistemas de controle; identificação de sistemas; lógica *fuzzy*; controle robusto; e controle preditivo de modelos.

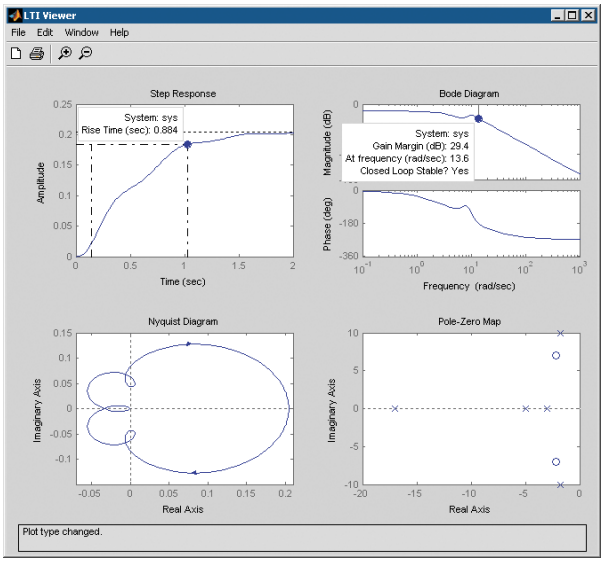

Figura 73. LTI Viewer.

Na Figura 73 está demonstrado o *LTI Viewer* com gráficos de resposta em freqüência, diagramas de Bode e de Nyquist e o mapa de pólos e zeros.

Alunos das disciplinas de sistemas de controle geralmente são apresentados ao MatLab para utilizarem o pacote de sistemas de controle. Neste pacote existem ferramentas especializadas no projeto e análise de controladores em malha fechada para sistemas dinâmicos. Podem ser utilizadas técnicas de controle clássicas e modernas, incluindo lugar

das raízes (*root locus*), alocação de pólos, e projeto de reguladores LSQ. Podem-se modelar sistemas invariantes no tempo (*Linear Time Invariant* – LTI) informando função de transferência; zeros, pólos e ganho; ou espaço de estados em qualquer dos domínios de tempo, seja contínuo ou discreto. Funções possibilitam além de conversões entre representações diferentes de modelos e domínios de tempo, gráficos de respostas temporais, respostas em freqüência e lugar das raízes.

Aquisição de Dados

Um pacote especial para aquisição de dados esta localizado na categoria de Medição e Testes. Este pacote fornece funcionalidade para interfaces de *hardware* de entrada e saídas (I/O) tanto digitais como analógicas de placas de aquisição de dados (*Data Acquisition Card* – DAC) que são compatíveis com *PC*s, assim como placas de som e portas paralelas. Uma ferramenta configura o dispositivo de *hardware* externo para leituras e escritas provindas da máquina do *MatLab*.

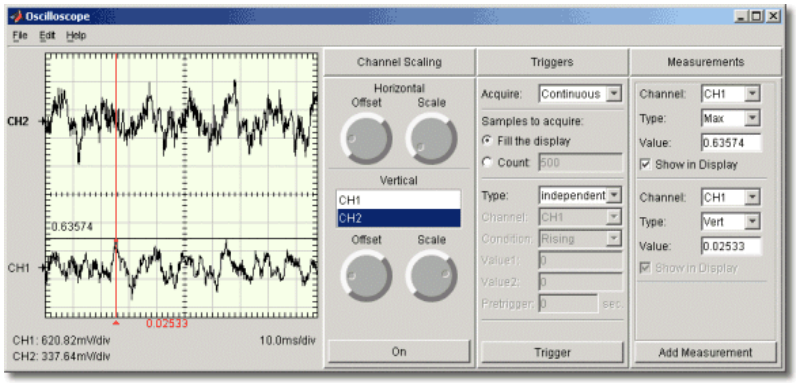

Figura 74. SoftScope.

Na Figura 74 está ilustrada a visualização do *soft-Scope* dentro do pacote de aquisição de dados do MatLab.

Uma função chamada de *soft-Scope* apresenta dados adquiridos com uma interface simulada de um osciloscópio. Esta simulação de osciloscópio apresenta todas as

funcionalidades de um verdadeiro osciloscópio, como *trigger* (função de apresentação da onda), número de amostras adquiridas, *start* e *stop* (iniciar e parar aquisição de dados), número de amostras apresentadas, erros e funções simples de regulação da amplitude, freqüência, *offset* (nível de corrente contínua) e amostragem de canais.

Controle de Instrumentos

Também na categoria de Medição e Testes um pacote de controle de instrumentos possibilita comunicação com instrumentos de medição como osciloscópios, geradores de funções, e instrumentos analíticos direto do MatLab. A comunicação se dá através de *drivers* IVI e VXIplug&play e protocolos de comunicação comumente utilizados como o GPIB (HPIB, IEEE-488), VISA, TCP/IP, UDP, USB e RS-232 (porta serial).

Os comandos utilizados para comunicação seguem os comandos padrões para instrumentos programáveis, os *SCPI*s (Standard Commands for Programmable Instruments). Uma interface gráfica, chamada de *TMtool*, guia o usuário no processo de conciguração do instrumento e da comunicação com o MatLab. Blocos para o Simulink também podem ser construídos para serem utilizados na biblioteca do módulo.

Interação com OPC

Ainda na categoria de Medição e Testes é oferecido um pacote OPC que estende o MatLab para interagir com servidores OPC (vide Figura 75). Nele pode-se ler, escrever, e manter registros (*logs*) de dados de dispositivos que obedeçam ao padrão de acesso de dados da fundação OPC, o OPC-DA (OPC - Data Access), como: sistemas de controle distribuídos (do inglês “Distributed Control Systems” – DCS), supervisórios de controle e aquisição de dados (do inglês “Supervisory Control and Data Acquisition” – SCADA) e controladores lógicos programáveis – *CLP*s (em inglês PLC).

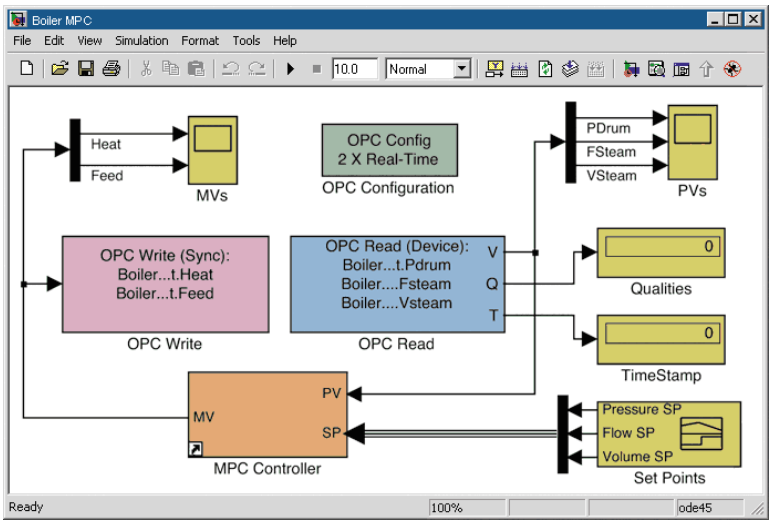

Figura 75. Simulink com elementos de OPC.

O pacote ainda possibilita ações guiadas por mudanças de variáveis e iniciação e parada de servidores. Em outras palavras a funcionalidade se restringe a clientes OPC o que é desejável já que muitas funcionalidades podem ser obtidas com este serviço.

MatLab Web Server – MWS

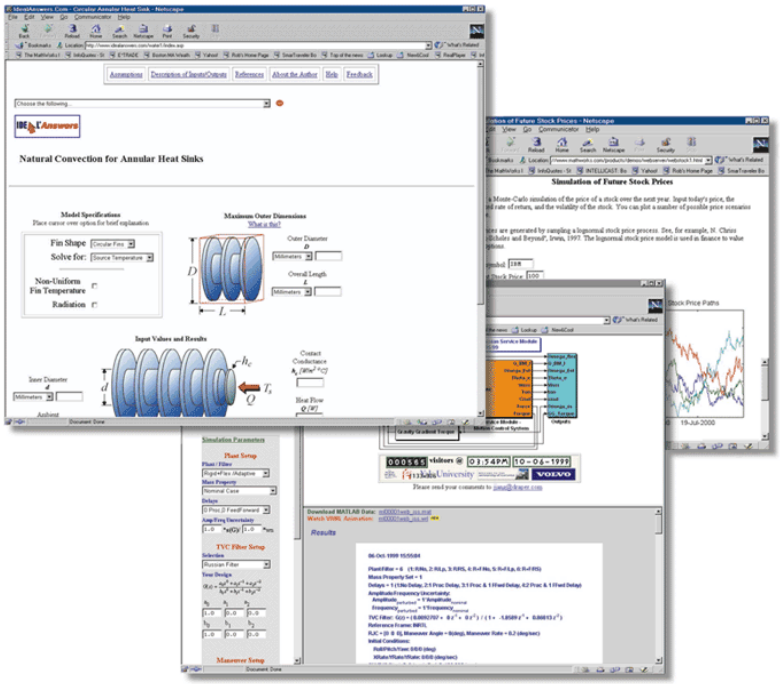

Figura 76. Interfaces do MatLab Web Server.

Na categoria de Distribuição de Aplicação temos uma funcionalidade muito utilizada por instituições de ensino que desejam disponibilizar interfaces de simulação do MatLab através da *Web*, o *MatLab Web Server* (vide Figura 76). O usuário remoto (cliente) acessa a simulação através de *HTML*s, *FORM*s (elementos de entrada de dados) e gráficos.

Todas as funcionalidades do *MatLab* podem ser acessadas pelo cliente sem necessidade de instalação do *software*. O desenvolvedor da interface para *Web* configura e restringe opções desejadas. Simulações mais elaboradas como as do Simulink também podem ser incorporadas no *Web Server*.

MatLab Real Time Workshop – RTW

Na categoria de geração de códigos temos um pacote de tempo real chamado de RTW. O RTW gera e executa códigos em *C* para testes e desenvolvimento de algoritmos modelados no *Simulink* (vide Figura 77).

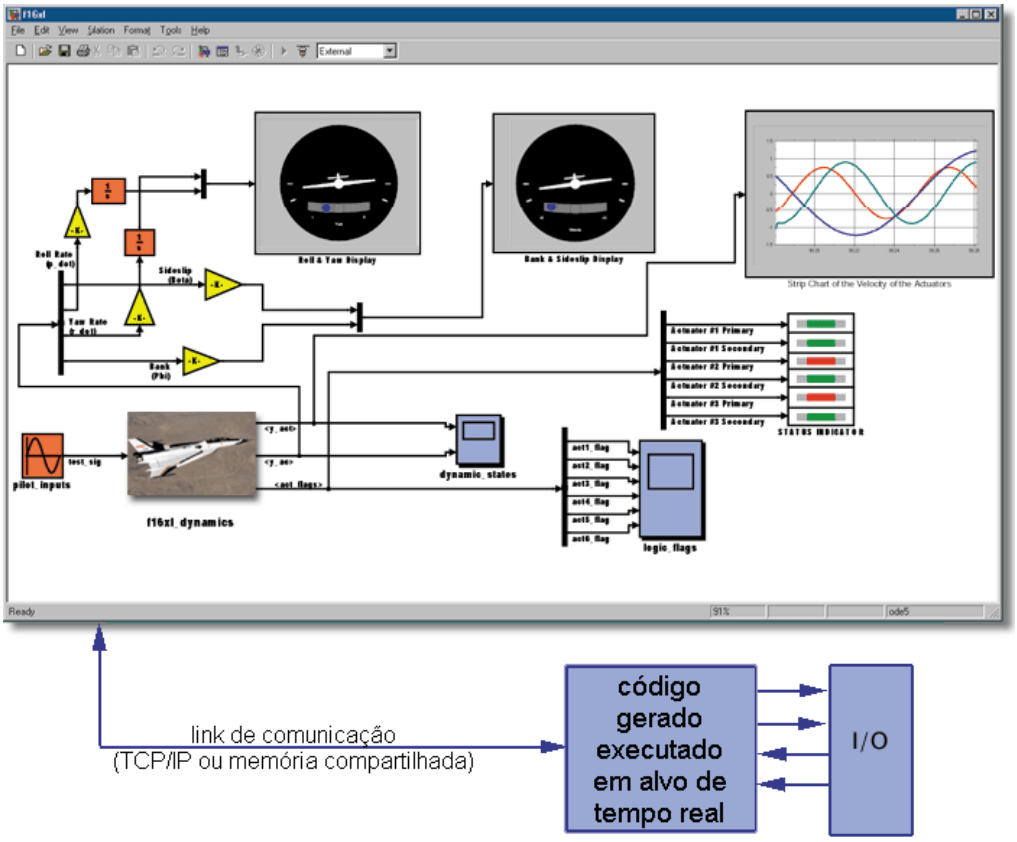

Figura 77. MatLab RTW.

O código gerado é compatível com ANSI/ISO C possibilitando execução em qualquer microprocessador ou sistema operacional tempo real (RTOS). Os alvos de execução podem ser de cinco tipos diferentes:

i. Código Genérico de Tempo Real:

   O código é utilizado para ajustar interativamente parâmetros de modelos, para mostrar e registrar resultados da simulação, e para alocação estática de dados, facilitando, assim, a execução eficiente em tempo real.

ii. Alocação de Memória em Código Genérico de Tempo Real

   A alocação possibilita criação de múltiplas instâncias de um modelo ou vários modelos em um programa executável.

iii. Alvo de Função *S*

   Ele converte modelos de simulação do *Simulink* em *DDL*s de funções *S*, possibilitando compartilhar modelos sem comprometer a propriedade intelectual da simulação.

iv. Alvo de Simulação Rápida (*RSim*)

   Ele proporciona geração rápida de códigos que executam em modo *Batch* ou usando simuladores Monte Carlo com solucionadores de passo fixo ou variável, possibilitando fácil variação dos parâmetros e dados de entrada para cada execução, registrando dados de saída em um único arquivo.

v. Alvo Tornado

   Ele gera código para execução no *VxWorks*.

ISaGRAF – ICS Triplex

Este *software* simulador é baseado nas linguagens de programação da norma IEC 61131-3. Lógicas de programação são simuladas como se estivessem sendo executadas em CLPs (Controlador Lógico Programável), este recurso é chamado de *softPLC*, ou seja um

CLP em *software*. O *ISaGRAF* (ISAGRAF, 2006) é um produto comercial da ICS Triplex que pode ser usado como um *software* supervisório (SCADA) em sistemas de controle.

Características

Por utilizar as seis linguagens de programação, cinco abrangidas na IEC 61131 mais o *Flow Chart* (diagrama de fluxo), este *software* é bastante utilizado na simulação e programação de CLPs para controles locais ou distribuídos. Esta capacidade torna mais fácil a adaptação do sistema de controle para o padrão de segurança internacional IEC 61508, que é uma norma de segurança de sistemas de controle elétricos, eletrônicos e programáveis, assim como de aplicações de nível crítico de segurança SIL 3.

O *software* é disponibilizado em diversas plataformas, como Windows com tecnologia NT (NT, 2000 e XP), Windows CE 3.0 e QNX. Dois elementos distintos fazem parte do *software*: a "bancada de aplicações" (do inglês *Application Workbench*) e o alvo de execução (do inglês *Runtime Target*) ou máquina virtual.

Os alvos de execução são máquinas exeqüíveis, que cumprem a lógica desenvolvida na bancada de aplicações, descritas em linguagem "C" ou código independente de sistema (*Target Independent Code* – TIC). Alvos de execução podem ser os seguintes sistemas: Embedded NT; Linux; MS-DOS; NT com Intime; NT com extensão de tempo real (Real Time eXtention - RTX); OS-0/9000; Phar Lap; pSOSytem; QNX; VRTX; VxWorks; Windows CE; Windows NT e Windows XP. Como pode-se ver alguns dos alvos são sistemas tempo real (*Real Time* – RT) ou que utilizam alguma técnica de relógio tempo real.

Linguagens de Programação

Como vimos anteriormente estas linguagens são as cinco da norma IEC 61131 mais o diagrama de fluxo. As cinco linguagens abrangidas pela norma IEC 61131 estão representadas na Tabela 6, onde também estão comentados os tipos de aplicações.

A linguagem de programação mais utilizada por técnicos da área de eletrotécnica no Brasil é o *Ladder* (LD), ou linguagem de relés. Sua lógica é muito parecida com o próprio circuito elétrico que reproduz a lógica. Chaves contatoras e relés são seus principais componentes. Entradas lógicas são representadas por contatos e saídas (resultados) através de bobinas. Funcionalidades, como temporizadores (*timers*) e comparadores, também podem ser utilizadas na lógica, dando mais possibilidades ao programador.

A Figura 78 mostra quatro linguagens de programação diferentes utilizadas no ISaGRAF que simulam exatamente as mesmas funções (comportamentos).

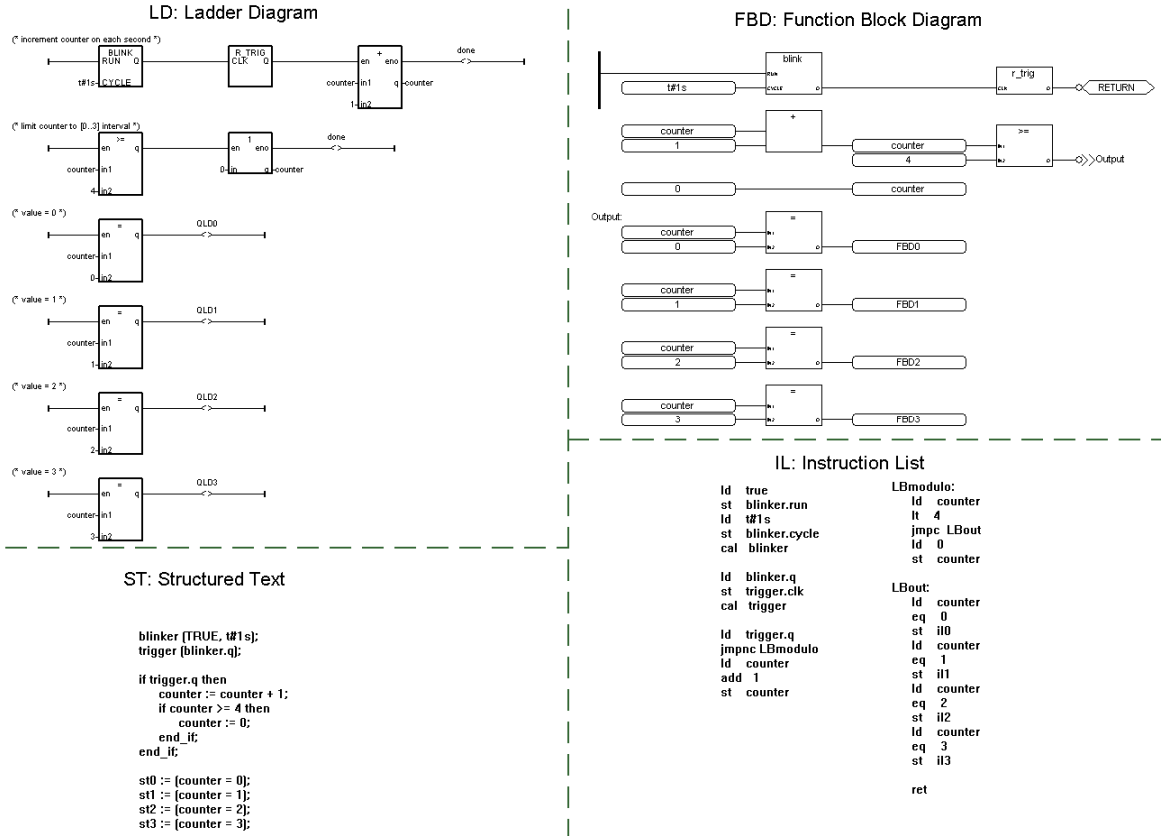

Figura 78. Demonstração de 4 linguagens da IEC 61131.

O Texto Estruturado (ST) lembra a linguagem de programação *PASCAL* do *PC*. Operações complexas com instruções aninhadas são suportadas dando mais flexibilidade à programação.

A lista de instruções (IL) é a linguagem de mais baixo nível, entre as abrangidas pela IEC 61131. A linguagem *booleana* de baixo nível representa a linguagem de CLP simples a nível de registradores, lembrando o *assembler*.

O diagrama de funções seqüenciais (SFC) é uma ótima linguagem para representar processos seqüenciais graficamente em que há processos concorrentes, pois divide o processo em passos definidos separados por transições. Outras linguagens descrevem as ações realizadas em cada passo.

O diagrama de blocos funcionais (FBD) é uma linguagem gráfica, que lembra a lógica booleana de circuitos digitais, em que temos blocos de funções simples como AND, OR, NOT, NOR e NAND em conjunto com funções de operadores matemáticos mais complexos e blocos funcionais utilizados para automatizar processos.

O diagrama de fluxo (FC), que não faz parte da IEC 61131, é talvez a mais simples das linguagens. Diagramas de tomadas de decisão e de laços são facilmente montados, até por desenvolvedores que não conhecem programação, para representar sistemas seqüenciais

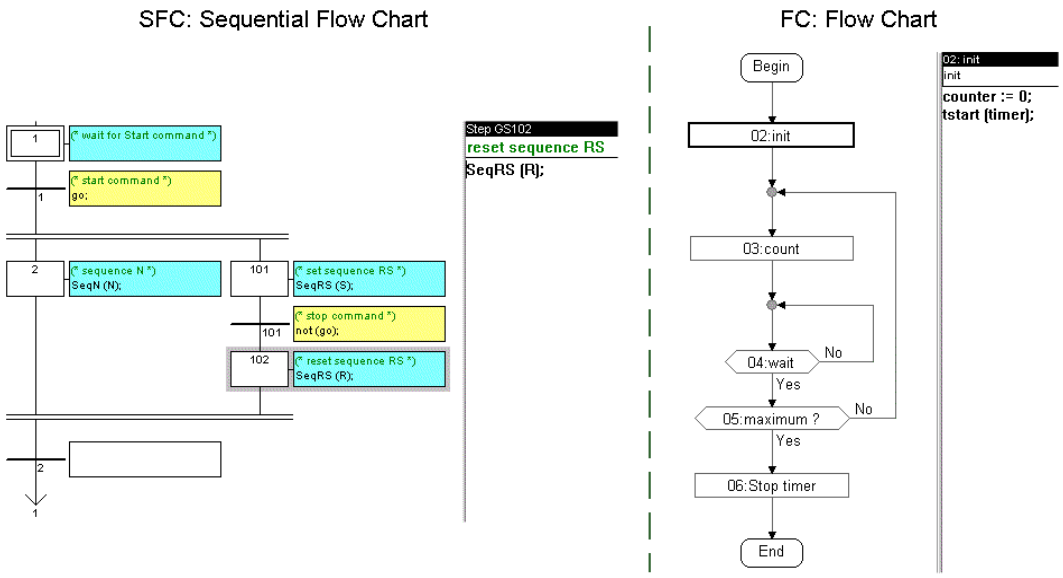

Figura 79. Diagramas seqüenciais SFC e FC.

A Figura 79 ilustra dois diagramas seqüenciais, o SFC da IEC 61131 e o FC. Pode-se notar que o SFC possui dois ramos que são executados em paralelo.

A idéia básica de utilizar um simulador que utilize estas seis linguagens de programação é combiná-las para produzir uma lógica que possibilite tirar vantagens de cada linguagem.

Funcionalidades do Simulador

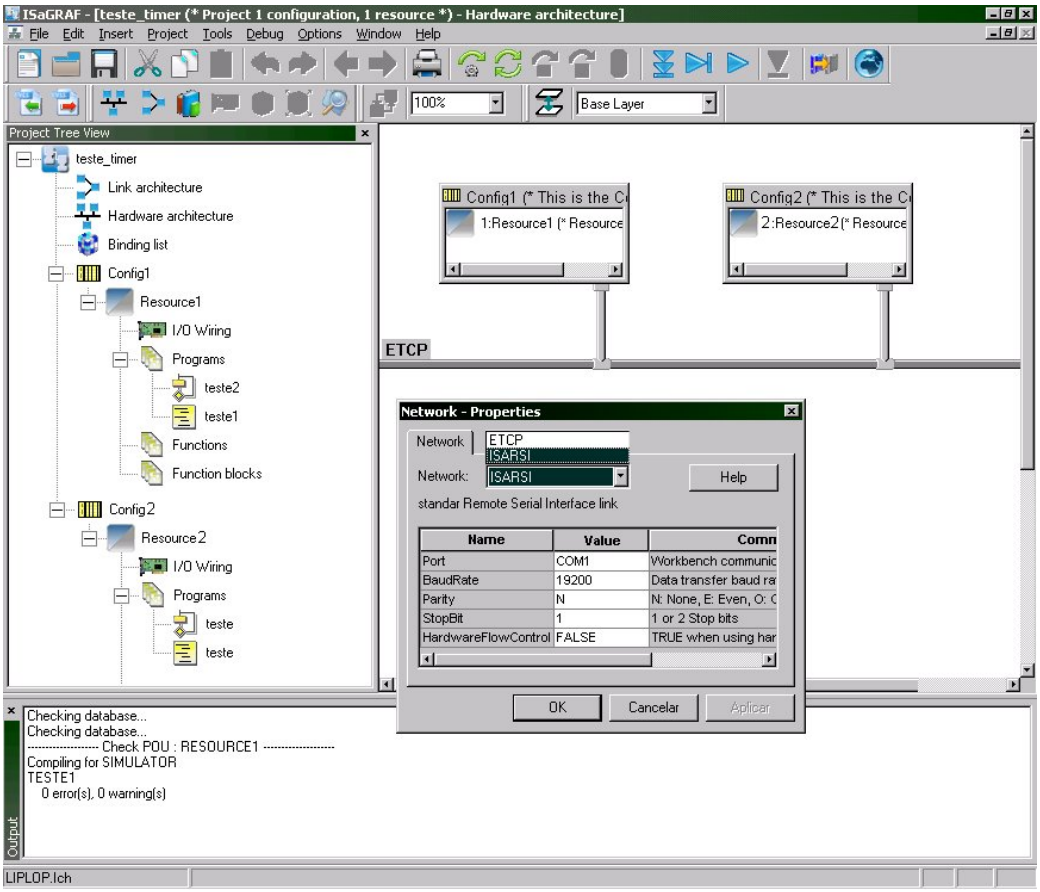

Figura 80. Configuração de rede no ISaGRAF.

A Figura 80 ilustra a tela do *ISaGRAF* de configuração da rede entre dois dispositivos (configurações), demonstrando sua capacidade programação e simulação distribuída.

O simulador também suporta programas externos escritos em *C* para dar mais flexibilidade aos desenvolvedores. Além das linguagens suportadas uma biblioteca de 60 blocos funcionais está incluída para utilização direta no simulador.

O *software* possui também um pacote (*toolkit*) de desenvolvimento que potencializa o uso de diversas máquinas virtuais para aplicações com um alto volume de equipamentos. Este pacote possibilita flexibilidade na configuração de interfaces de: sistemas, comunicações, entrada e saída (I/O), blocos funcionais, IXL (tipo de lista XML documentada) e de personalização da bancada de desenvolvimento (*workbench*).

O *software* também possui *drives* e *gateways* de comunicação que possibilitam integração com sistemas de:

- estatística, diagnósticos e suporte de debugação pra recuperação de erros;
- pacote de *drivers* para personalização;
- pacote de funções seriais para conexão utilizando protocolos proprietários;
- pacote ODBC para acessar bases de dados ODBC;
- servidor OPC-DA para conexões com clientes OPC e interfaces homem-máquina (*IHM*);
- mestre-escravo (*master-slave*);
- rotulação de tempo (*time stamping*) em milisegundos;
- múltiplas entradas e saídas (*I/O*) em um nódulo;
- *gateway*s de comunicação redundante.

Funcionalidades, como o servidor OPC, a integração com bancos de dados e comunicações com dispositivos externos, dão ao ISaGRAF qualificações para ser usado como supervisórios *SCADA* ou *softDCS* (*Software Distributed Control Systems*) e não como meros *softPLC*s. O servidor OPC-DA suporta projetos múltiplos, isto é, diversos instâncias do servidor podem ser iniciadas com diferentes projetos.

HiBeam

Com um *software* adicional da ICS Triplex, chamado de *HiBeam*, é possível a construção de telas de supervisório gráficas com as variáveis descritas no ISaGRAF. O

HiBeam disponibiliza estas telas em formato que pode ser lido em navegadores de Internet (*browsers*) fazendo com que o sistema desenvolvido no ISaGRAF se torne acessível remotamente.

Este *software* é constituído de três elementos:

1. *Screen Builder* (construtor de telas);

   É uma interface gráfica (GUI) controlada por aplicativos Java (*Applets*) para construção de telas de supervisório.

2. *Embedded Web and Data Server* (servidor embarcado da *Web* e de dados)

   São aplicativos Java responsáveis em coletar e distribuir dados dos controladores que são disponibilizados através de um servidor de dados que funciona via Internet.

3. *Development Kits* (pacotes de desenvolvimento)

   Possibilitam a visualização dos códigos-fonte de aplicativos Java gerados pelo HiBeam assim como portabilidade para o *C++*.

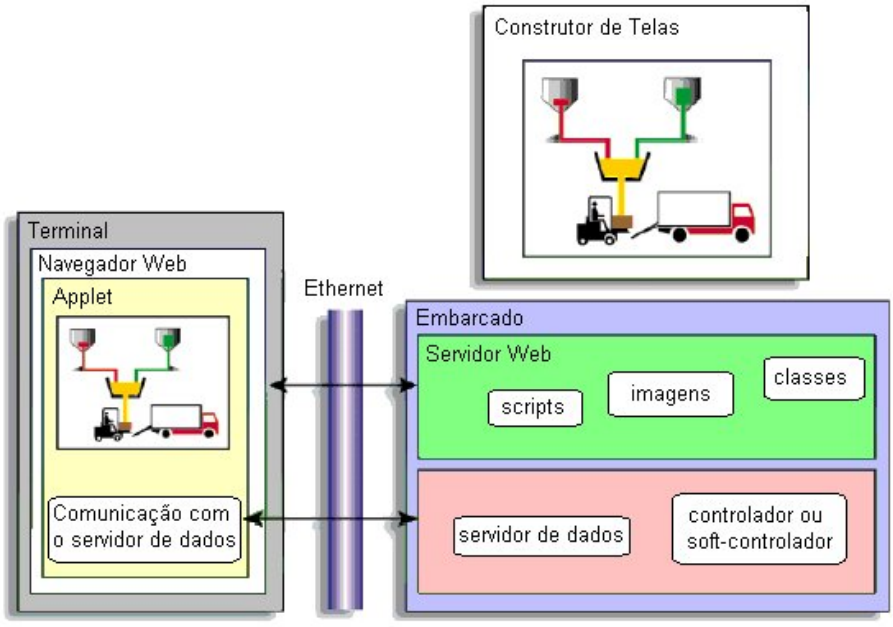

Figura 81. Arquitetura e comunicação entre elementos do HiBeam

O HiBeam é alternativa para sistemas operacionais que não utilizam OPC por não suportar DCOM, como por exemplo o Linux. Os servidores do HiBeam foram construídos para serem altamente portáveis, podendo ser incorporados em controladores e em dispositivos pequenos como: *DCS*s, dispositivos inteligentes (ex. *Foundation Fieldbus*), *CNC*s, etc. O servidor de dados ocupa 100 kbytes, enquanto que o servidor da *Web* 50 kbytes. Adicionais 100 bytes de memória RAM para cada variável são necessários no dispositivo.

A arquitetura assim como a comunicação entre os elementos existentes na configuração e programação da tela do supervisório no *HiBeam* estão ilustrados na Figura 81.

LabVIEW – National Instruments

O *LabView* é uma abreviação de **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench, que significa bancada de laboratório virtual para instrumentação de engenharia. É um *software* simulador comercial da *National Instruments* (NI, 2006) que utiliza uma 1linguagem própria de programação por meio de ícones gráficos e de fluxo de dados. Esta linguagem é chamada de *G* projetada para ser muito simples e fácil, pois conta com inúmeras funções prontas que o usuário arrasta e solta (*drag & drop*) como se fossem ícones.

A Figura 82 ilustra as janelas representando o diagrama de blocos (código) e o painel do instrumento virtual de cálculo de variância e desvio-padrão de medidas.

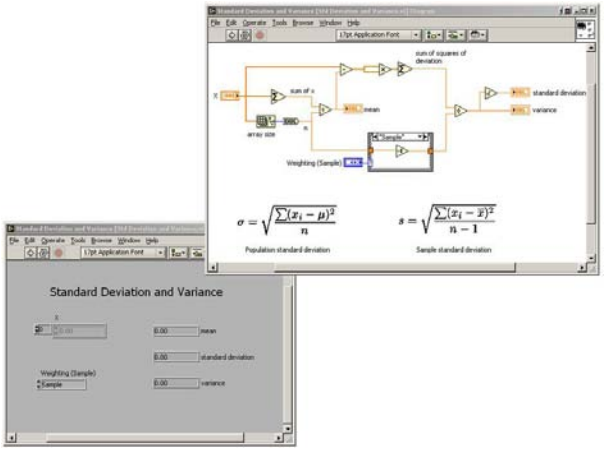

Figura 82. Janelas do LabVIEW.

O *LabView* é largamente usado em diversas áreas de instrumentação, controles de instrumentos e automação industrial por facilitar aquisições de dados e tratar estes dados em instrumentos virtuais (*virtual instruments* – VI's).

Características

A linguagem de programação *G* em oposição a outras linguagens não é determinada por execução seqüencial de instruções. Ao invés disso a execução inicia quando todas as entradas estão disponíveis em um nódulo (função). Isto torna a programação intrinsecamente paralela já que muitos nódulos podem ser usados ao mesmo tempo. Um escalonamento próprio (*built-in*) multi-thread é automaticamente utilizado.

O LabVIEW também é uma ambiente de desenvolvimento da linguagem *G* com compilação e *debugging* (execução passo a passo) de programas da linguagem *G* chamados somente de VI's. Isto o torna uma ferramenta muito flexível podendo ser usada para qualquer aplicação que o programador desenvolver.

A grande vantagem da utilização de fluxo de dados para programação é sua representação através de metáforas gráficas. Representações gráficas em bidimensionais (2-D) utilizam melhor a capacidade visual de programadores.

Os *VI*s possuem dois componentes: o diagrama de blocos, que representa o código, e o painel frontal, que representa o mostrador (*display*) do instrumento virtual com suas funções de entrada (vide Figura 83). O painel frontal também pode ser chamado de IHM. Entradas de dados são postas no painel frontal e tratadas no diagrama de blocos. A programação de painéis frontais se dá da mesma maneira que os diagramas de blocos, pois equipamentos usuais de laboratórios também são utilizados como ícones.

O simulador está no mercado há 20 anos, disponível para diversos sistemas operacionais, entre eles Windows, MacOS e Linux.

Funcionalidades

Diversas funcionalidades podem ser anexadas à programação existente como por exemplo:

- programação direcionada a eventos (*event-driven*);

  Combina programação de fluxo de dados com programação direcionada a eventos.

- desenvolvimento de diagramas de estado

  Adiciona a capacidade de criação de diagramas de estado dentro da programação de fluxo de dados.

- Sistemas dinâmicos de tempo contínuo

  Execução de algoritmos matemáticos, como os do MatLab, podem ser usados em conjunto com a simulação.

- Algoritmos baseados em textos matemáticos

  Execução de algoritmos matemáticos, como os do *MatLab* podem ser usados em conjunto com a simulação.

- Integração de códigos herdados (*legacy code*)

  Viabiliza a integração de códigos existentes em *C* na simulação.

- Programação em tempo real

  Extensões de tempo real podem ser carregadas como módulos para proporcionarem maior determinismo a simulações de fluxo de dados.

Aquisição de Dados

O *software* LabVIEW está dotado de diversos recursos para interfaceamento de equipamentos de aquisição de dados (vide Figura 83). A National Instruments é fabricante de uma grande gama de placas de aquisição de dados para PC das seguintes tecnologias: PCI, PCI Express, PXI, PCMCIA, USB, CompactFlash, Ethernet, FireWire compatíveis com os sistemas operacionais Windows, Linux, MacOS, Pocket PC/Windows CE com RTX. Adquirir

dados de instrumentos de controle é facilmente configurado através de um assistente de *I/O* que possui *drivers* para: GPIB, portas seriais, Ethernet, PXI, USB e VXI.

Controle e monitoramento industrial também são possíveis utilizando ferramentas para monitoramento de sistemas distribuídos para uma grande variedade de dispositivos de *I/O* incluindo *CLP*s e CompactFieldPoint's (*hardware* da National Instruments). O CompactFieldPoint proporciona performance industrial e integração com LabVIEW Real-Time.

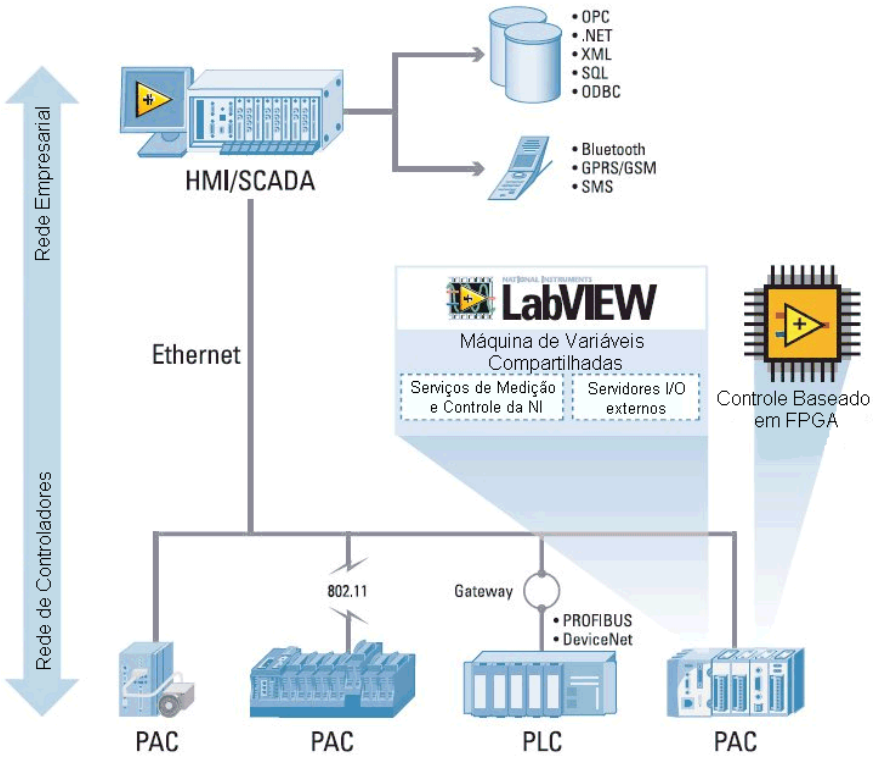

Figura 83. Diagrama de utilizações do LabVIEW.

Módulos do LabVIEW

Na Figura 83 pode-se notar utilização do LabVIEW que engloba desde redes de controladores até redes empresariais. Existem sete módulos do LabVIEW com diferentes aplicações e funções. Os módulos existentes são:

1. Sistemas de desenvolvimento (LabVIEW Development Systems)

   Esta versão de desenvolvimento do *software* possui quatro subcategorias:

   - Básico: produtividade gráfica instantânea de desenvolvimento para aquisição de dados e controle de instrumentos;
   - Completo: adiciona conjuntos de análise e rotinas para medição, processamento de sinais e automação;
   - Profissional: versão para desenvolvedores avançados;
   - Conjunto de desenvolvedores: envolve funcionalidades da subcategoria profissional, todos os pacotes (*toolkits*) e atualizações.

2. Módulo PDA (LabVIEW PDA Module);

   Com esse módulo podem ser gerados alvos de execução, como touchscreens e *PDA*s.

3. Real Time (LabVIEW Real-Time)

   É o módulo responsável pela geração de aplicações determinísticas em alvos de execução tempo real.

4. Módulo FPGA (LabVIEW FPGA Module)

   Estende a versão gráfica do LabVIEW para desenvolvimento em *FPGA*s reconfiguráveis, como o *hardware* da National Instruments RIO (*Reconfigurable I/O*). Dessa forma a linguagem do LabVIEW pode ser usada para criar um *hardware* personalizado sem a necessidade de linguagens baixo nível, como o *VHDL*.

5. Módulo para Aplicações Embarcadas (LabVIEW for Embedded Applications)

   Através deste módulo, sistemas embarcados, como processadores de 32 bits, *DSP*s e *FPGA*s, podem ser programados a partir da linguagem gráfica utilizada no LabVIEW.

6. Módulo de Registro de Dados e Controle Supervisório (LabVIEW Datalogging and Supervisory Control Module)

   Possibilita o desenvolvimento interativo de interface para supervisórios a partir da visualização de dados, configuração de alarmes e eventos. Configurações de segurança também são possíveis, assim como interfaces OPC de servidor e cliente, SQL e ODBC.

7. Visão de Máquina (LabVIEW for Machine Vision)

   Viabiliza a programação de algoritmos de visão de máquina, que possui diversas bibliotecas e assistentes para ajudar no desenvolvimento da aplicação.

Alvos de Execução - runtime targets

A aplicação desenvolvida no LabVIEW, o código, pode ser configurado para uma variedade grande de alvos de execução (*runtime targets*). A Figura 84 ilustra os possíveis alvos de execução que podem ser criados utilizando-se o LabVIEW.

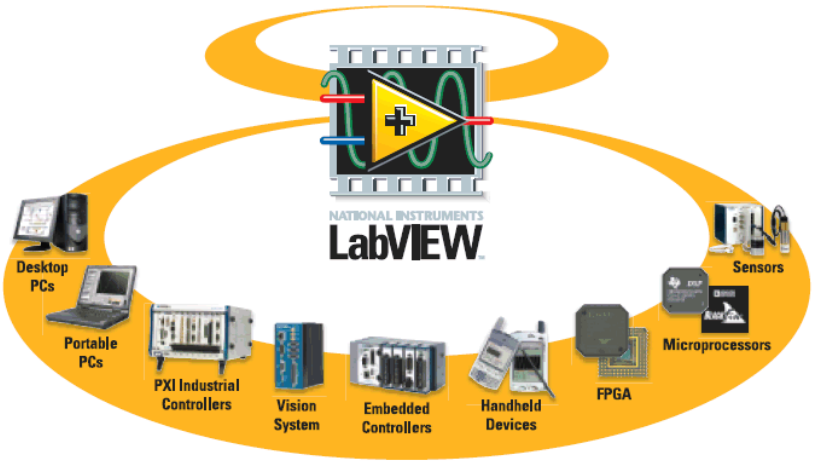

Figura 84. Alvos da aplicação do LabVIEW.

Pacotes (*Toolkits*)

Adicionais aos diversos módulos ainda temos pacotes para fins mais específicos, como pacotes de:

- Projeto de filtros digitais;
- Rastreamento de execução;
- Desenvolvimento rápido de VI's;
- Analisador de VI's;
- Diagrama de estados;
- Projeto de controle;
- Identificação de sistemas;
- Módulos de simulação;
- Interface matemática;
- Geração de relatórios;
- Testes de Integração DSP
- Painéis Remotos;
- LabVIEW Internet;
- Controle PID;
- Processamento avançado de sinais;
- Controle de movimento;
- Driver IVI;
- Vibração e sons;
- Servidor de automação industrial;
- Conectividade com base de dados;
- Conectividade de empresas;
- Análise de pedidos.

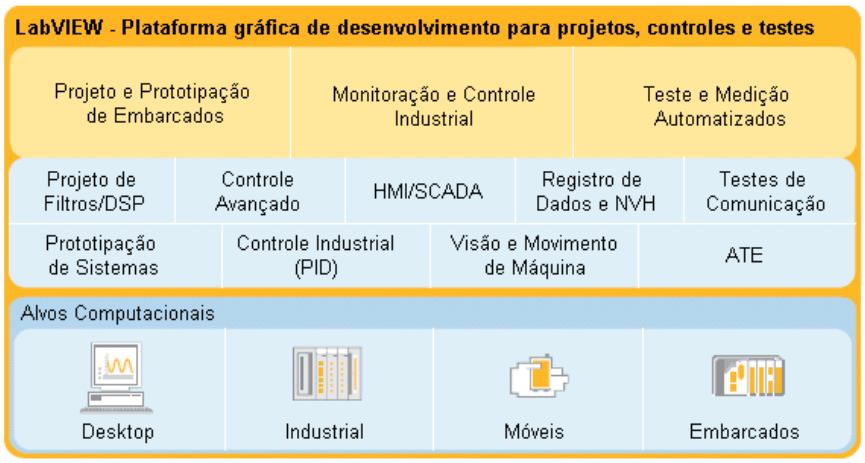

Figura 85. Abrangência do LabVIEW.

O LabVIEW se torna altamente flexível e abrangente (vide Figura 85) com todas essas opções de pacotes e módulos. O pacote de painéis remotos possibilita acesso remoto de usuários a sistemas reais ou simulados desenvolvidos no LabVIEW. Pacotes de automação

possibilitam criação de interface homem-máquina (HMI), assim como telas de supervisório (SCADA). A partir da Figura 85, pode-se afirmar que o LabVIEW pode interagir com qualquer rede, empresarial ou de controle, funcionando tanto como dispositivo medidor, quanto como controlador e supervisório.

FluidSim

O *FluidSim* é um simulador desenvolvido para o ensino de dinâmica dos fluidos no sistema operacional Windows. É um *software* comercial vendido pela FESTO, produzido em parceria *FESTO Didactic,* Universidade de Paderborn e empresa Art Systems Software. Este simulador de circuitos eletropneumáticos pode ser usado em combinação com equipamentos de treinamento produzidos pela FESTO.

Características

A característica mais marcante do FluidSim é sua conexão com funcionalidades de CAD e simulação. Os esquemas dos circuitos eletropneumáticos utilizados são desenhados de acordo com a norma DIN ISO 1219 e podem realizar simulações realísticas dos modelos baseados no funcionamento físico dos componentes. A Figura 86 mostra esquemas eletro pneumáticos à esquerda e à direita a simulação no FluidSim.

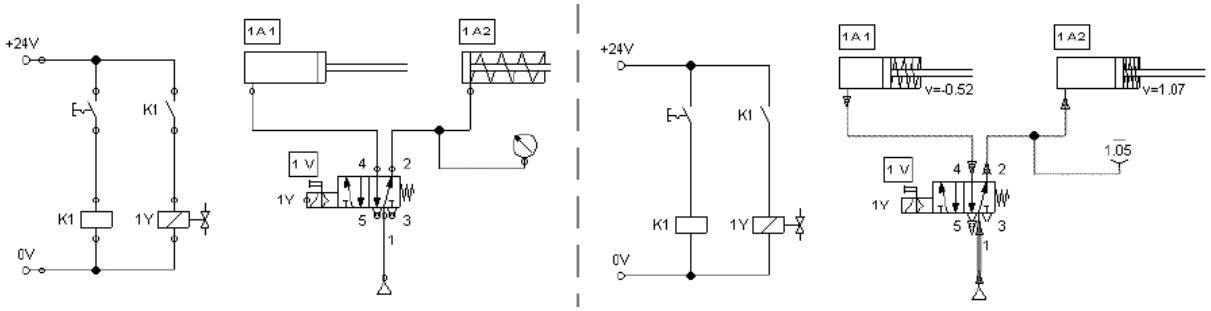

Figura 86. CAD e Simulação no FluidSim.

Como o *software* foi desenvolvido seguindo princípios didáticos, ele pode ser usado com fins educacionais, pois possui uma completa descrição textual de cada componente,

assim como vídeos e animações que demonstram o funcionamento de cada modelo (equipamento) utilizado. Ênfase na interface intuitiva, fácil de aprender, foi dada durante o desenvolvimento.

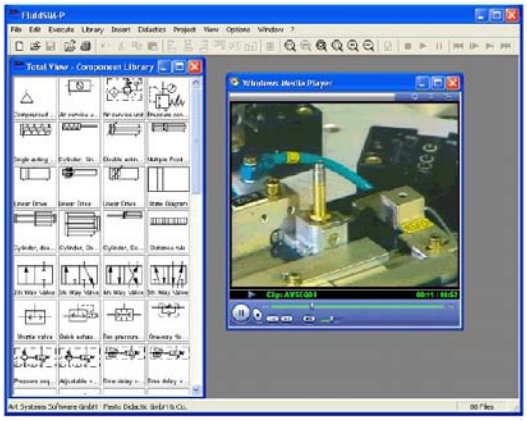

Figura 87. Ajuda no FluidSim.

Todos os componentes da simulação são inspirados em equipamentos reais produzidos pela FESTO, até sons produzidos por equipamentos reais são reproduzidos na simulação. Assim, os experimentos simulados podem ser construídos em bancadas didáticas da FESTO da mesma maneira como são simulados.

Na Figura 87 temos a exemplificação de utilização de um equipamento real da FESTO que é simulado no FluidSim com um vídeo didático demonstrando o funcionamento

Interfaces de Comunicação

Para se tornar um *software* mais flexível, algumas funcionalidades de comunicação foram incorporadas no simulador. Modelos de conectores são inseridos na simulação para representarem as interfaces de comunicação com equipamentos/*softwares* externos.

Algumas das interfaces são:

- Interface DDE

  Um serviço de troca de dados direta (DDE) é iniciado, onde nomes de variáveis são assinalados para serem utilizadas na simulação.

- Interface OPC

  Pode-se trocar valores entre simuladores diferentes ou até com servidores de CLPs através de um cliente OPC-DA.

20-Sim

O *20-Sim* (20-SIM, 2006) é um simulador de comportamento de sistemas dinâmicos como sistemas: elétricos, mecânicos e hidráulicos ou qualquer combinação entre eles. A modelagem é feita de forma gráfica capacitando o projeto e análise dos sistemas dinâmicos modelados como se fossem esquemas de engenharia. É um *software* comercial de produzido pela empresa de mesmo nome.

Na Figura 88 são representados modelos do 20-Sim, sendo à esquerda um esquema gráfico de um cilindro oscilante, e, à direita, o modelo gráfico de Bond.

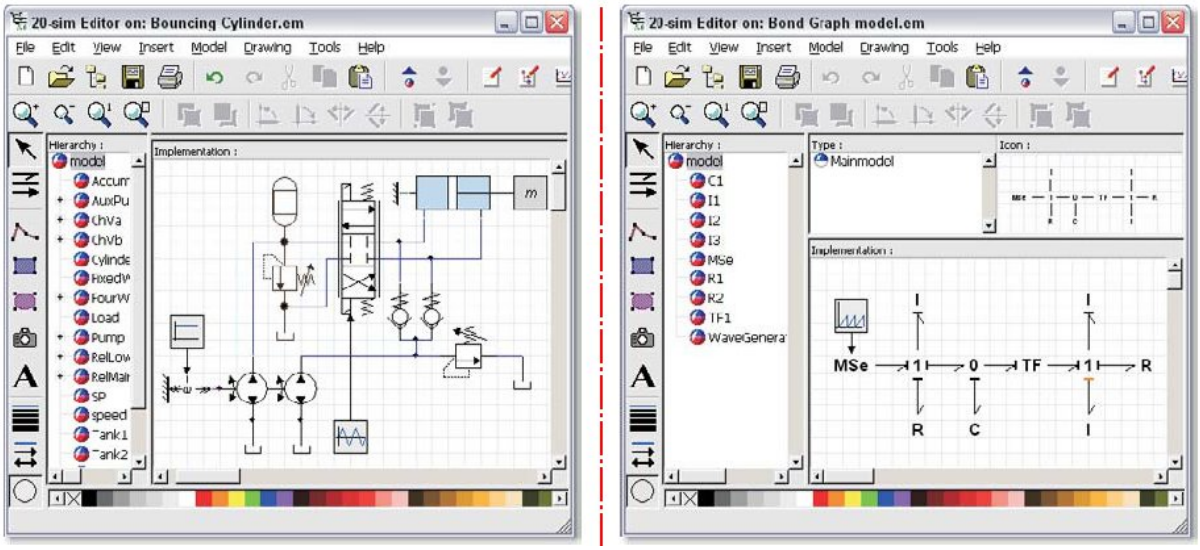

Figura 88. Modelos do 20-Sim.

A biblioteca de modelos possui diversos componentes separados por seis categorias: elétricos, hidráulicos, mecânicos, térmicos, diagramas de blocos e gráficos de Bond. Todos os modelos utilizados na biblioteca são abertos, isto é, podem ser modificados. As equações que representam o comportamento dos modelos são escritas utilizando uma notação matemática

padronizada, o *SIDOPS+*. Esta notação matemática que também pode ser chamada de linguagem é utilizada para representar gráficos de Bond. Com a utilização destes modelos o 20-Sim pode simular uma grande gama de sistemas dinâmicos, incluindo sistemas: lineares, não lineares, de tempo discreto, de tempo contínuo e híbridos.

Características

Os algoritmos de simulação utilizados possibilitam resolver equações diferenciais ordinárias (ODE) e equações algébricas diferenciais (DAE). Diversos métodos de integração numérica podem ser usados como: fixado, passo variável, passo único, múltiplos passo e múltiplas ordens.

Modelos do 20-Sim podem ser exportados para o MatLab ou Simulink como um arquivo *m* ou uma função *S*. A escolha entre exportação como função de transferência ou espaço de estados também pode ser feita.

Este *software* se encontra disponível apenas para *PC*s e sistemas operacionais Windows.

Pacotes (toolboxes)

Assim como outros simuladores o 20-Sim também possui oito pacotes para expansão.

1. pacote de domínio de tempo (*time-domain toolbox*)

   Possibilita troca de parâmetros, otimização e ajuste a curvas para melhorar performance do modelo e análise de sensibilidade, análise de monte carlo e análise de variação para verificação e validação de limites de protótipos virtuais.

2. pacote de domínio de freqüência (*frequency-domain toolbox*)

   Consiste de um editor de linear de sistemas, análises de transformada de Fourier rápida (*FFT*) e linearização de modelos.

3. pacote de controle (*control toolbox*)

   Consiste em editores de projeto de controladores, de filtros e de redes neurais.

4. pacote de tempo real (*real-time toolbox*)

   Oferece geração de códigos C para simulações de *hardware* em laço (*hardware-in-the-loop* - HIL) e prototipação rápida de controle (*rapid control prototyping-RCP*).

5. pacote de tempo real (*real-time toolbox*)

   Oferece geração de códigos C para simulações de *hardware* em laço (*hardware-in-the-loop* - HIL) e prototipação rápida de controle (*rapid control prototyping*).

6. pacote de mecatrônica (*mechatronics toolbox*)

   Contém guias de perfil de movimento (*motion profile wizard*), de câmeras (*cam wizard*) e editores de servo motor (*servo motor editor*).

7. pacote de mecânica 3-D (*3-D mechanics toolbox*)

   Contém um editor de modelos tridimensionais. Desta forma um modelo 3-D pode ser associado a um modelo do 20-Sim e a ele associado, movimentos (Figura 89).

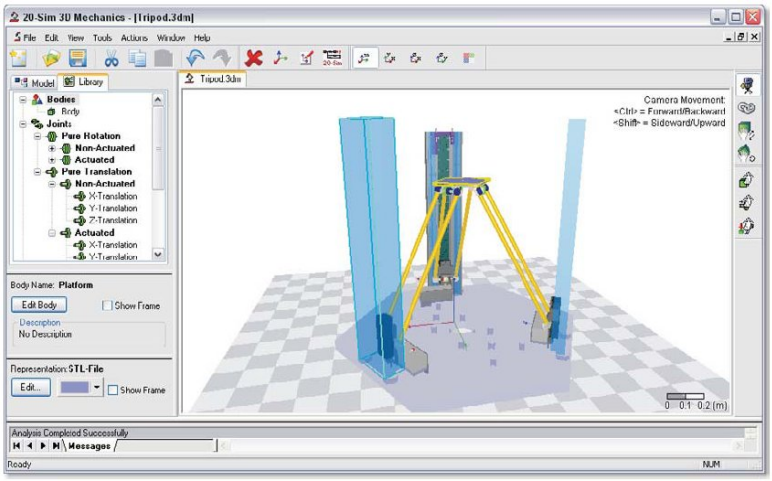

Figura 89. Pacote de mecânica 3-D do 20-Sim.

8. pacote de animação (*animation toolbox*)

   Podem ser criados vídeos através de simulações, assim como importações de modelos CAD.

COSIMIR

O *COSIMIR* é um *software* de simulação de sistemas robotizados criado pela Universidade de Dortmund e mais tarde desenvolvido pelas empresas *EF-Robotertechnik* e *EFR-Systems* (COSIMIR, 2006).

A simulação utiliza modelos tridimensionais para simulação e programação de movimentos de equipamentos industriais como braços de robôs, linhas de montagem automatizadas, *CNC*s e outros.

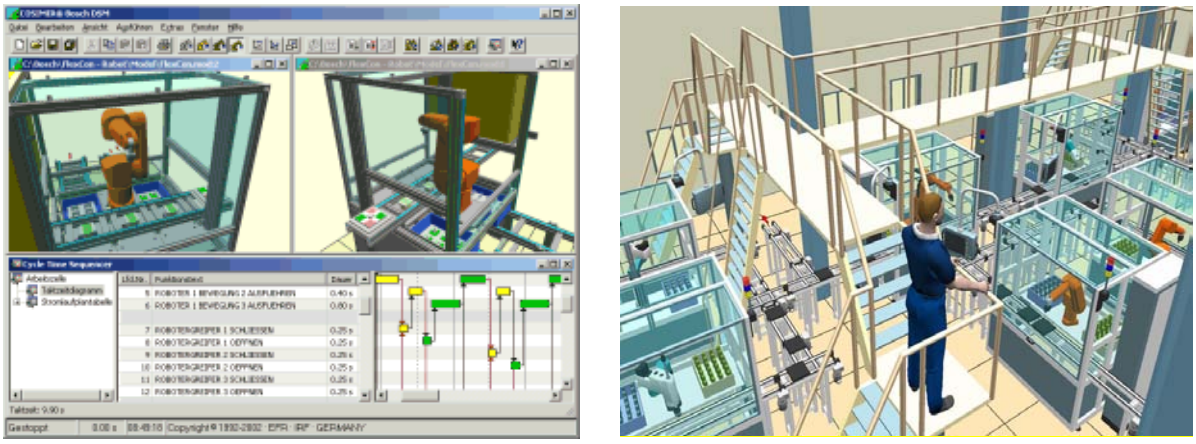

Figura 90. Interface e simulação do COSIMIR.

Na Figura 90 está ilustrado à esquerda uma visão da programação seqüencial de um braço de robô e à direita visualização do funcionamento da linha de montagem em um ambiente virtual em que se pode navegar com a câmera.

Versões

Existem duas versões diferentes do produto:

- COSIMIR Manufacturing (Manufatura)

  Simulador para linhas de montagem que utilizam robôs, esteiras, *CLP*s e sensores.

  Os seguintes fabricantes de robôs podem ser utilizados na simulação: ABB, Adept, FANUK, KUKA, Manutec, Mitsubishi, Reis, Stäubli, Niko, VW e outros. O

simulador suporta as seguintes linguagens de programação para robôs: IRL (Industrial Robot Language), V+, KRL (Kuka Robot Language), RAPID, MRL (Movemaster Command), MELFA Basic III + IV, BAPS (Bosch Automatisierungsprogrammiersprache) e SRPL (Simple Robot Programming Language). A simulação de CLP (em alemão SPS) utiliza a linguagem de programação da Siemens, o S5/S7 (Step 5).

Na Figura 91 pode-se ter uma comparação da simulação em VR com a realidade.

Figura 91. Simulação no COSIMIR vs. realidade.

- COSIMIR Virtual Reality (VR)

  Esta versão do simulador possibilita modelagem e visualização de processos em realidade virtual com a utilização de capacetes de realidade virtual (*head mounted display*), panoramas de retro projeção e *CAVE*s (*Cave Automatic Virtual Environment*).

  Simulações diversas podem ser criadas para treinamento e manuseio de equipamentos. Aplicações na medicina também podem ser simuladas para treinamento.

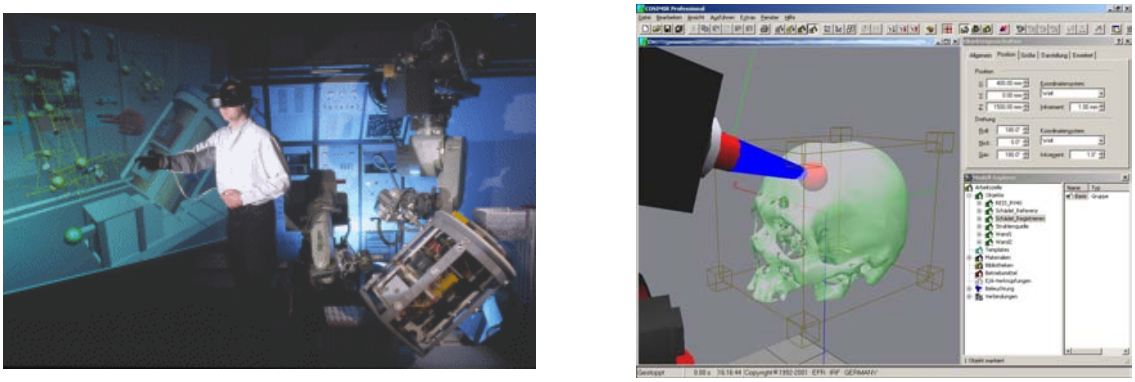

Figura 92. Aplicações do COSIMIR.

A Figura 92 mostra uma foto (à esquerda) que demonstra a interação com modelo em realidade virtual e uma simulação de utilização de equipamento médico em crânio humano (à direita).

Características

A FESTO tem o direito de venda de ambas as versões do simulador que estão disponíveis com 4 diferentes licenças: *Educational*, *Professional*, *Industrial* e *Test Version*. A versão profissional é a mais completa dando possibilidade de:

- Modelagem de robôs através de bibliotecas de diferentes sistemas, garras, sensores, magazines, esteiras e máquinas *CNC*, assim como importação de formatos de projetos *CAD*;
- Simulação para visualização e detecção de colisões e tempos de ciclo em representações tridimensionais. A câmera (vista) da simulação pode ser rotacionada, movida e ampliada (*zoom*). A funcionalidade da programação do CLP também pode ser testada mudando-se parâmetros e visualizando estados dos sensores e de processos.
- Programação de robôs que podem ser testados por um interpretador de comandos que aponta possíveis erros em todas as linguagens de programação suportadas (citadas anteriormente). A possibilidade de enviar e receber

arquivos de configuração (programação) de robôs reais da Kuka, Mitsubishi e ABB também é possível.

Na versão industrial algumas funcionalidades foram retiradas em relação à versão profissional. Os fabricantes suportados são restringidos à Mitsubishi, assim como foram retirados a simulação com diversos robôs, suporte a sensores e *CLP*s. A expansão da versão para suportar VR também não pode ser usada.

Na versão educacional foram retirados os direitos de criação de novas células de trabalho assim como enviar e receber arquivos de programação de robôs reais. Algumas tarefas de punho educacional foram incluídas nesta versão para que estudantes aprendam o funcionamento de certas funcionalidades básicas de linhas de montagem. A criação de animações, que podem ser usadas em apresentações na *Web*, é uma funcionalidade presente nessa licença.

Uma licença de teste também está disponível com funcionalidades da versão completa (profissional), mas com restrições de tempo de simulação de *3 min*, utilização máxima durante *30 dias*, e modificações em modelos não podem ser realizadas assim como a criação de novos arquivos e simulações.

Este simulador está acessível somente para sistemas operacionais Windows. Para a *renderização* tridimensional *drivers OpenGL* (biblioteca gráfica aberta) são utilizados o que necessita de uma placa que suporte este recurso.

Interfaces de Comunicação

Como o *software FluidSim*, também vendido pela FESTO, o COSIMIR possui uma interface de comunicação OPC que pode ser usada tanto para ligar o simulador a um simulador externo quanto a equipamentos reais. Para a aquisição de dados neste caso é usado o *EasyPort* da FESTO. O EasyPort funciona como uma placa de aquisição de dados externa, se comunicando com o computador usando a RS-232 (porta serial). Uma aplicação também

da FESTO é responsável por criar um servidor OPC que irá transmitir os dados medidos no EasyPort para o cliente OPC, isto é, o COSIMIR.

A Figura 93 mostra a utilização da interface OPC do COSIMIR para *softPLC* e CLP real utilizando o EasyPort para adquirir dados e transmiti-los para o PC via RS-232.

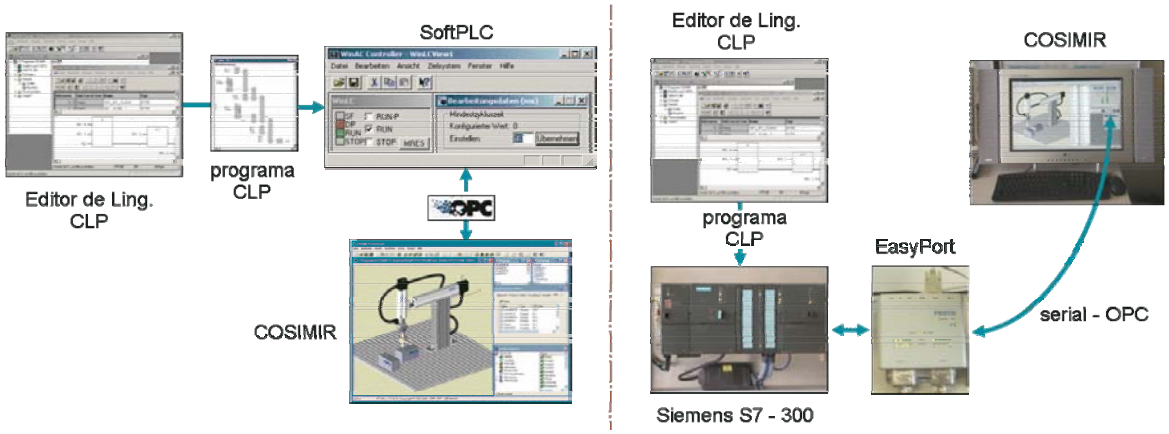

Figura 93. Interface OPC do COSIMIR.

Elipse SCADA

O *Elipse* (ELIPSE, 2006) não é tecnicamente um *software* simulador, mas pode ser programado para simular alguns processos, assim como interagir com simuladores externos. Projetado para suprir todas os requisitos de um *software* supervisório SCADA, o Elipse combina uma ferramenta gráfica de edição com uma linguagem de programação bastante simples baseada em scripts. Desenvolvido pela empresa brasileira *Elipse*, com sede em Porto Alegre, e com 10 anos de experiência em sistemas SCADA, esse *software* tem participação internacional no mercado de *softwares* supervisórios.

Características

O *software* pode ser distribuído em três módulos que são responsáveis por duas funções diferentes do *software*:

1. *Configurador*

   O módulo permite o desenvolvimento da aplicação, restringindo, no entanto, o tempo de execução da aplicação em *2 horas*.

2. *Runtime*

   Exclusivo modo para execução da aplicação desenvolvida no configurador, ou seja, máquina de execução (*Runtime*). Nenhuma alteração na aplicação é permitida neste módulo, que não possui limite de tempo de execução.

3. *Master*

   Une as funcionalidades do configurador com a máquina de execução.

A linguagem de programação utilizada é orientada a eventos (*event driven*), chamada de *Elipse Basic* (baseada no Basic). Entradas de dados como “cliques de mouse” e mudanças de variáveis ativam scripts que são programados nos objetos da ação.

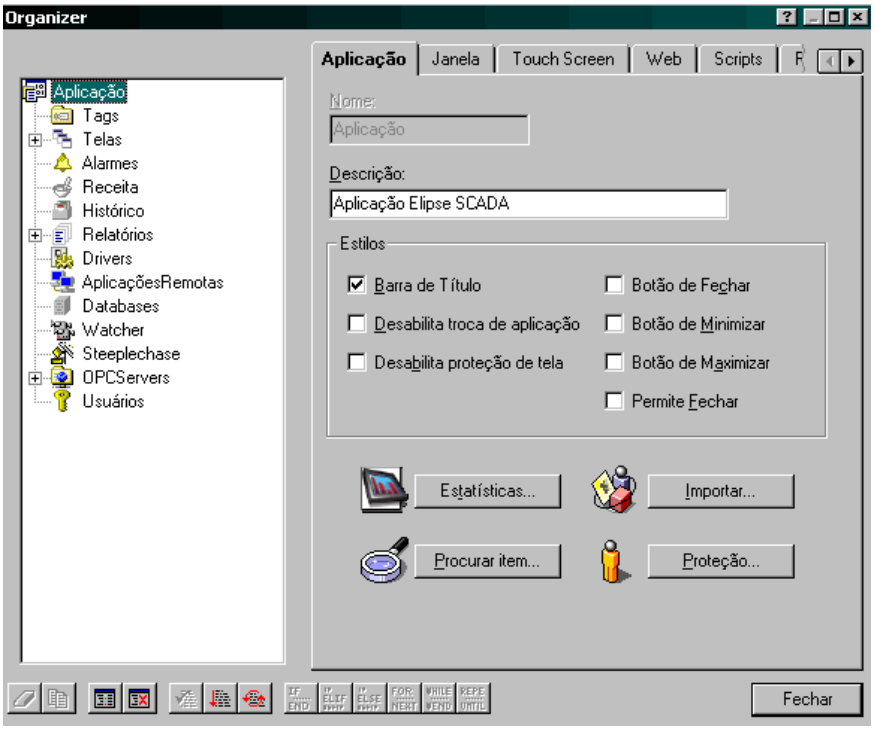

Figura 94. Elipse SCADA “Organizer”.

O *software* utiliza uma tecnologia que possibilita o controle e supervisão de estações à distância chamadas de aplicações remotas. Diversas soluções de comunicação em vários meios físicos são possíveis, sejam em redes fechadas utilizando protocolos *TCP/IP* ou *IPX*, como na *Internet* utilizando o pacote *Elipse Web* ou comunicação serial. Dados da aplicação podem ser acessados por um cliente remoto através do servidor que pode realizar leituras e escritas de qualquer parâmetro da aplicação.

As variáveis utilizadas pelo *software* para caracterizar processos e dispositivos são chamadas de *tags*. Uma janela de organização e configuração da aplicação é chamada de *Organizer* (vide Figura 94) onde são configurados *tags*, equipamentos e serviços. Associações de parâmetros e *I/O*s com *tags* também são configurados no *Organizer* para poderem ser utilizadas em scripts e em objetos como mostradores, históricos, botões, etc.

Plug-ins e pacotes

Para suportar maiores funcionalidades diferentes *plug-ins* e pacotes são oferecidos com objetivos mais específicos. Os *plug-ins* são:

- *Elipse Watcher*

  Permite a monitoração de sistemas através de recursos de captura, registro e transmissão digital de imagens. Suporta diversos padrões de vídeo, possibilitando a visualização em janelas com tamanho e qualidade programáveis pelo usuário. Também possibilita a criação de um banco de imagens com busca por período ou evento e transmissão de imagens para estações remotas via *TCP/IP*.

- *Elipse Web*

  Permite a supervisão de processos através da *Internet*. É possível conectar-se a uma estação de supervisão remota, utilizando qualquer navegador com suporte JRE.

Pacotes que ampliam necessidades específicas da aplicação são:

- *Elipse View - Visualização*

  Permite a visualização de variáveis, inclusive com a utilização de animações, programação de *setpoints*, controle de acesso e funções especiais para *touch screen*. Esta versão não inclui ferramentas para o registro de dados históricos, alarmes ou relatórios, além de outras funcionalidades que venham a surgir em versões mais avançadas. Possui entre outros um cliente/servidor de DDE.

- *Elipse MMI – Interação homem máquina*

  Possui banco de dados proprietário, relatórios formatados, históricos, receitas, alarmes e Controle Estatístico de Processos (CEP), facilmente implementáveis. Pode ainda, ser um servidor de dados para outras estações Elipse assim como registrar alarmes em disco. O *Elipse MMI* é indicado para sistemas de qualquer porte, onde não sejam necessárias conexões com bancos de dados externos (ODBC e DAO) ou aplicações de rede, e quando o usuário precisa enxergar outras estações de supervisão.

- *Elipse Pro – Profissional*

  É o pacote mais completo que oferece soluções para troca de dados entre estações, realização de comandos e programação de *setpoint* via *TCP/IP*. Inclui recursos de conexão com banco de dados ODBC e DAO assim como comunicação OPC (cliente e servidor).

- *Elipse Power – Sistemas de Potência*

  Possui recursos avançados como a conexão com *IED*s e *RTU*s através de protocolos de comunicação como o IEC 870-5 e DNP3.0. O Elipse Power utiliza base de tempo local, permitindo o seqüenciamento de eventos (SOE) com precisão de *1 ms*, oscilografia e telesupervisão. É a ferramenta ideal para sistemas de telemedição, possibilitando a integração com bases de dados relacionais e

comunicação via satélite. Os *drivers* podem capturar eventos de qualidade de energia (*power quality*) como distúrbios de: *sags*, *swells* e interrupções, além de realizar transmissão de arquivos de oscilografia, trabalhando tanto em linha privada quanto em *TCP/IP*. É possível ainda, sincronizar o relógio do computador que controla o processo com os equipamentos remotos via *GPS*.

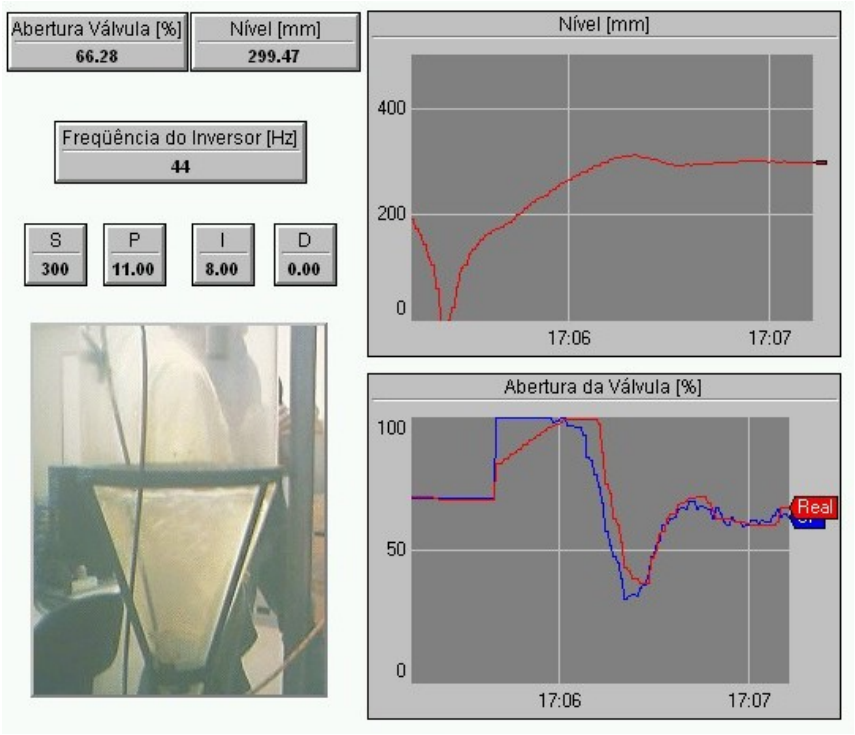

Figura 95. Visualização de vídeo e dados.

A Figura 95 demonstra uma aplicação do Elipse SCADA com os *plug-ins* de visualização de vídeo e visualização via *Web*, utilizados na planta didática da UFRGS.

Portabilidade

O *software* está disponível para sistemas operacionais Windows 98/NT/2000/XP, Windows CE para processadores HPC200, HPC200 ARM, HPC200 MIPS, HPC200 x86 e PocketPC ARM, assim como para Linux (em fase de testes).

Ferramentas para o desenvolvimento de *drivers* (DDK) também estão disponíveis para que usuários possam personalizar interfaces de comunicação de rede e bancos de dados.

Relés – SENAI

O *software Relés* foi desenvolvido pelo Núcleo de Educação à Distância (NEAD) do SENAI-RS. Este simulador bastante simples busca demonstrar aos alunos a linguagem de relé (*Ladder*) para programação em *CLP*s (vide Figura 96). Os componentes são simples contatos, bobinas, memórias e temporizadores. A preocupação principal está em demonstrar a funcionalidade da linguagem relé.

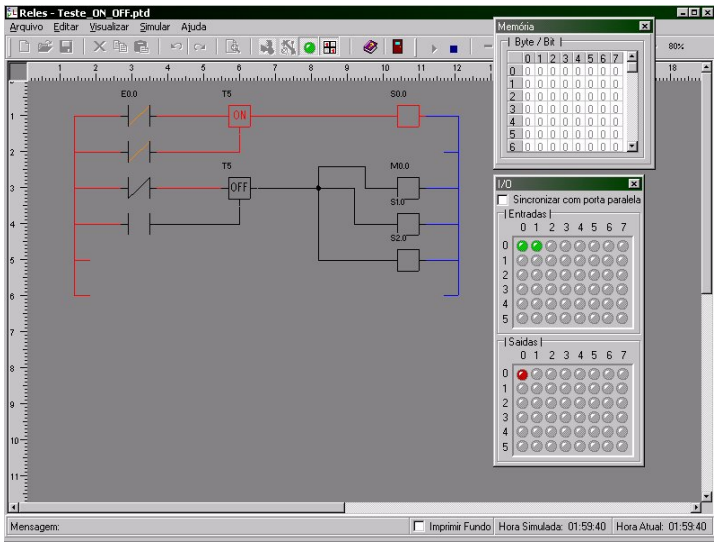

Figura 96. Janela do simulador “Relés”.

Uma pequena interface de *hardware* foi proposta para se adaptar equipamentos reais à porta paralela utilizando os 8 bits de dados.

Atualmente o *software* é utilizado somente em sistemas operacionais Windows.