

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

“Implementação de uma plataforma HW/SW para
Automação Industrial, utilizando Hardware
Reconfigurável com Processador NIOS II em
conformidade com o padrão IEEE 1451”

EDSON ANTONIO BATISTA

Orientador: Prof. Dr. Alexandre César Rodrigues da Silva

Co-Orientador: Prof. Dr. Aparecido Augusto de Carvalho

Tese apresentada à Faculdade de
Engenharia - UNESP – Campus de Ilha
Solteira, para obtenção do título de Doutor
em Engenharia Elétrica.
Área de Conhecimento: Automação.

Ilha Solteira – SP

Setembro/2009

FICHA CATALOGRÁFICA

Elaborada pela Seção Técnica de Aquisição e Tratamento da
Informação/Serviço Técnico
de Biblioteca e Documentação da UNESP-Ilha Solteira

- B333i Batista, Edson Antonio.
Implementação de uma plataforma HW/SW para automação industrial, utilizando hardware reconfigurável com processador NIOS II em conformidade com o padrão IEEE 1451 /
Edson Antonio Batista. -- Ilha Solteira : [s.n.], 2009
194 p. : il., fots. color.
- Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de Conhecimento : Automação, 2009
- Orientador: Alexandre César Rodrigues da Silva
Co-Orientador: Aparecido Augusto de Carvalho
Bibliografia: p. 130-137
1. Automação industrial. 2. Sistemas embarcados. 3. Software de aplicação.
4. Software – Reutilização. 5. Hardware – Linguagem descritiva. 6. VHDL(Linguagem descritiva de hardware).

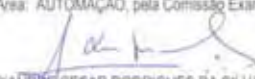
CERTIFICADO DE APROVAÇÃO

TÍTULO: implementação de uma plataforma HWISW para Automação Industrial utilizando Hardware Reconfigurável com Processador NIOS II em conformidade com o padrão IEEE 1451

AUTOR: EDSON ANTONIO BATISTA

ORIENTADOR: Prof. Dr. ALEXANDRE CESAR RODRIGUES DA SILVA

Aprovado como parte das exigências para obtenção do Título de DOUTOR em ENGENHARIA ELÉTRICA, Área: AUTOMAÇÃO, pela Comissão Examinadora:


Prof. Dr. ALEXANDRE CESAR RODRIGUES DA SILVA

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. DIONIZIO PASCHOARELI JUNIOR

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. LUIS CARLOS ORIGA DE OLIVEIRA

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. EDUARDO DO VALLE SIMÕES

Departamento de Sistemas de Computação / Universidade de São Paulo


Prof. Dr. MAURO CONTI PEREIRA

Departamento de Engenharia Mecatrônica / Universidade Católica Dom Bosco

Data da realização: 04 de setembro de 2009

A Deus

A minha esposa Luciana

Ao meu pai Ovídio e a minha mãe Eunice

Aos meus irmãos Cristina, Ana, Silvana, Roberto, Rosiani, Núbia e Junior

Ao meu cunhado e filho Vinicius

DEDICO

AGRADECIMENTOS

A Deus, pela minha vida, pelo aprendizado, por ajudar nas superações e proteger nos momentos difíceis.

A minha esposa pela paciência, pelos auxílios diretos a este trabalho, pelo amparo nos momentos difíceis ajudando a superá-los.

Aos meus Pais por serem um exemplo de superação e uma das fontes motivadoras deste trabalho.

Ao meu orientador, Professor Alexandre César Rodrigues da Silva, pela orientação, pela confiança, pela sinceridade e, principalmente, por contribuir para o meu crescimento, não apenas como profissional, mas também como pessoa.

Ao meu co-orientador, Professor Aparecido Augusto de Carvalho pela orientação, pelas sugestões e contribuições, pela confiança e por ser um exemplo de profissional.

Ao amigo Silvano Renato Rossi pelas contribuições fundamentais nos trabalhos de mestrado e doutorado. Deus o colocou em meu caminho.

Aos amigos Mauro, Wanderlei, Alexsandro, Baroni, Isaias, Naka e Jenner pelo companheirismo e pelas sugestões.

Aos Engenheiros Fábio D'ELia e Carlos Vinicius pelas contribuições.

Aos orientados Raphael Ceni, Luis Henrique, João Carlos, Gabriel Buin, Carlos Nakazone, Jeandro e Thiago da Silva pelas contribuições.

A direção da Universidade Católica Dom Bosco de Campo Grande por permitir a realização deste trabalho.

A minha sogra Ruth e ao meu sogro João por me apoiar inclusive financeiramente, tornando possível este trabalho.

A meu cunhado Vinicius pela convivência e apoio constante.

A minha tia Leonice, meu tio João e minha prima Rita pelo amparo.

Ao meu primo Luis Eduardo e sua esposa Rosana pelas orientações.

A minha cunhada Cristina, aos meus sobrinhos, André Luis, Emmanuel e Luis Paulo e ao meu primo e sobrinho Matheus, pelos bons momentos que ajudam a renovar minha energia.

Aos meus irmãos Cristina, Ana, Silvana, Roberto, Rosiani, Núbia e Junior pelo apoio incondicional ao desenvolvimento deste trabalho.

Resumo

A aplicabilidade da rede de comunicação junto com o avanço tecnológico é constantemente explorada pelos projetistas de automação e controle, pois, estas vertentes podem melhorar o desempenho de um processo industrial. O padrão IEEE 1451, surge em meio a estes desafios, com intuito de homologar conceitos e tecnologias para implementar uma rede de transdutores inteligentes. Neste trabalho desenvolveu-se uma plataforma de hardware/software para ser utilizada na automação industrial, tanto cabeamento como sem fio, de acordo com os padrões IEEE 1451.2 e IEEE 1451.5. Essa plataforma, denominada neste trabalho por plataforma IEEE 1451, é composta por um hardware, o Módulo de Interface para Transdutores (TIM – *Transducer Interface Module*), e por um software Processador de Aplicação para Rede de Comunicação (NCAP – *Network Capable Application Processor*). A lógica de controle e as especificações dos transdutores (TEDS – *Transducer Electronics Data Sheet*) foram inseridas no TIM por meio da programação (linguagem C/C++) do processador NIOS II e o hardware sintetizado em FPGA da família Cyclone II, especificamente na placa de desenvolvimento DE2 da Altera Corporation. A programação do processador NIOS II baseou-se em um *template* definido neste trabalho como IEEE 1451 que possui funções e bibliotecas específicas para atender às funcionalidades das aplicações e das normas IEEE 1451. O NCAP possui características de um software supervisor e foi desenvolvido com tecnologia Java no ambiente NetBeans IDE (*Integrated Development Environment*) versão 6.5. Entre as principais funções deste NCAP está a capacidade de enviar e receber os dados através da porta RS232, geração de relatório incluindo a TEDS, interface gráfica dinâmica e identificação de usuários. A plataforma IEEE 1451 foi testada na automação de diferentes aplicações, nas quais se constataram a flexibilidade e rápida prototipagem para desempenhar as funções de um sistema de controle. Outras vantagens desta plataforma IEEE 1451 é a utilização de linguagem orientada a objeto para desenvolver o software NCAP, o que facilita a reutilização de códigos, e o uso de hardware reconfigurável na implementação do TIM. Os resultados obtidos neste trabalho demonstraram que a tecnologia utilizada na implementação desta plataforma é suficientemente robusta para ser usada na automação industrial.

Palavras-chave: NCAP, TIM, IEEE 1451.5, TEDS, FPGA, processador NIOS II, SOPC Builder, Tecnologia Java e Automação Industrial.

Abstract

Designers usually exploit the fast evolution of technology along with the application of communication networks to improve the performance on industrial processes. The IEEE 1451 standard comes to aid in the development of networks of intelligent transducers, by defining concepts and technologies used in their implementations. This work intends to provide an application consisting of a hardware/software platform to be used in industrial automation, either wireless or not, according to the 1451.2 and 1451.5 IEEE standards. This IEEE 1451 platform is composed by a hardware part, the Transducer Interface Module (TIM), and a software part, the Network Capable Application Processor (NCAP). The control logic and the transducer specifications (TEDS – Transducer Electronics Data Sheet) were inserted in the TIM by programming in C/C++ a NIOS II processor, synthesized in a FPGA of the Cyclone II family, using the DE2 development board from Altera Corporation. The NIOS II programming was based on an IEEE 1451 template, with functions and libraries to implement the functionalities of the IEEE 1451 applications and guidelines. The NCAP software resembles a supervisory system and was developed in Java in the NetBeans integrated development environment, version 6.5. Amongst its main functions are the capabilities of report generation including TEDS, a dynamic graphical interface, user identification and the ability to send and receive data through a RS232 port. This IEEE 1451 platform was tested in the automation of different applications, demonstrating its flexibility and rapid prototyping suited for the development of control systems. Other advantages are the use of an object oriented language in the development of the NCAP software, which facilitates the code reuse, and the use of reconfigurable hardware for the TIM implementation. The results from this work showed that the technology applied in the implementation of this platform is robust enough to be used in industrial automation.

Key-words: NCAP, TIM, IEEE 1451.5, TEDS, FPGA, NIOS II processor, SOPC Builder, Java, Industrial Automation.

Lista de Figuras

Figura 1.1: Arquitetura do sistema IEEE 1451.....	17
Figura 1.2: Configuração de um NCAP, em conformidade com o padrão IEEE 1451...18	
Figura 1.3: Tela do software supervisor do NIST IEEE 1451.....	20
Figura 1.4: Arquitetura da plataforma JDDAC.....	21
Figura 1.5: Informações necessárias em uma medida.....	22
Figura 1.6: Triângulo de características para a tecnologia sem fio.....	26
Figura 1.7: Arquitetura da plataforma HW/SW proposta neste projeto.....	31
Figura 1.8: Blocos de um módulo sensor com o circuito de condicionamento.	39
Figura 2.9: BSSs conectados por um sistema de distribuição.....	46
Figura 2.10: Infra-estrutura de rede AdHoc.....	46
Figura 2.11: Protocolo Bluetooth.....	51
Figura 2.12: Piconet.....	52
Figura 2.13: Scatternet.....	53
Figura 3.14: Configuração dos parâmetros para comunicação serial.....	58
Figura 3.15: Painel do MHI-01.....	59
Figura 3.16: Tela Inicial e tela para monitoramento do SAF0.....	60
Figura 3.17: Tela do SAF 1.0.....	62
Figura 3.18: Pacotes que compõem o NCAP da plataforma IEEE 1451.....	64
Figura 3.19: Descrição dos métodos da classe DriverTIM.....	65
Figura 3.20: Diagrama de classe do software NCAP.....	66
Figura 3.21: Tela inicial do NCAP.....	68
Figura 4.22: Esquema para projeto de Hardware com NIOS II.....	71
Figura 4.23: Configuração do NIOS II na Placa DE2.....	72
Figura 4.24: Configuração da memória SDRAM, do DriverMMI.....	73
Figura 4.25: Tipo de processador NIOS II utilizado no MMI.....	74
Figura 4.26: Definição do nível de debugging JTAG do DriverMMI.....	75
Figura 4.27: Configuração da RS232 do DriverMMI.....	75
Figura 4.28: Interconexão dos componentes que compõem a UART para o MMI.....	76
Figura 4.29: Dispositivo de saída configurado para compor o DriverMMI.	77
Figura 4.30: Configuração dos dispositivos de entrada do DriverMMI.....	77
Figura 4.31: Definição da PLL_MMI.....	78
Figura 4.32: Definição da frequência da PLL_MMI.....	79
Figura 4.33: Definição da fase da PLL_MMI.....	80
Figura 4.34: Características da PLL_MMI.	80
Figura 4.35: Símbolo da PLL_MMI.....	81
Figura 4.36: Parte do programa do módulo embarcado MMI.	82
Figura 4.37: Propriedades da parte lógica do MMI.....	83
Figura 4.38: Operação realizada entre SAF e MMI.....	83
Figura 4.39: Arquitetura básica do DriverTIM.....	85
Figura 4.40: Template IEEE 1451 disponibilizado no NIOS II IDE.....	87
Figura 4.41: Implementação da função para tratamento de interrupção na biblioteca...88	
Figura 4.42: Biblioteca para enviar a descrição da TEDS para o NCAP.....	89
Figura 4.43: Método que recebe a TEDS enviada pelo DriverTIM.	90
Figura 4.44: Primeira etapa para implementar a TEDS com a plataforma IEEE 1451...91	
Figura 4.45: Segunda etapa para implementar a TEDS.	91
Figura 4.46: Implementação da TEDS da plataforma IEEE 1451.....	92
Figura 5.47: Fluxograma da lógica de controle para automação do MHI-01.....	94

Figura 5.48: Parte da lógica de controle das atividades do MHI no NIOS II IDE.....	95
Figura 5.49: Arquitetura do hardware do TIM para automatizar o MHI-01.....	96
Figura 5.50: TIM para automação do MHI-01.	97
Figura 5.51: Interface gráfica do SAF utilizada na automação do MHI-01.....	98
Figura 5.52: Telas para automação do MHI-01 do software NCAP.....	99
Figura 5.53: Monitoramento dos sensores (MHI-01) através da placa DE2.....	99
Figura 5.54: Etapa para geração de relatório.....	100
Figura 5.55: Arquitetura utilizada na automação do MHI-01.....	101
Figura 5.56: Configuração do processo de Filtragem do Biogás.....	102
Figura 5.57: Lógica de controle para automação - Filtragem de Biogás.	103
Figura 5.58: Equipamentos que compõe parte do processo de Filtragem do Biogás....	104
Figura 5.59: DriverTIM utilizado para medir pressão e vazão.....	105
Figura 5.60: Placa de circuito impresso e o esquema elétrico para conversão de corrente para tensão.....	107
Figura 5.61: Placa de circuito impresso para conversão de sinais A/D com o respectivo esquema elétrico.....	108
Figura 5.62: Configuração para automação - Filtragem do Biogás.....	109
Figura 5.63: Comparação entre valores obtidos no transmissor com os apresentados no NCAP. 110	
Figura 5.64: Parte da lógica de controle com base na Equação 5.1.....	111
Figura 5.65: Comparação entre valores lidos no transmissor de pressão e os apresentados na tela do NCAP com fator de correção.....	111
Figura 5.66: Resultado da solicitação da TEDS e a tela de supervisão.....	112
Figura 5.67: Valores coletados sem tempo real de pressão e vazão.....	113
Figura 5.68: Tela para acionamento do compressor e de monitoramento da pressão..	113
Figura 5.69: Tela de monitoramento da vazão.....	114
Figura 5.70: Linha de Filtragem do Biogás com os medidores instalados.....	114
Figura 5.71: Placa e esquema elétrico para tratamento de sinais da rede de transdutores do processo de Filtragem do Biogás.....	116
Figura 5.72: Tela do software NCAP para a automação do processo de Filtragem do Biogás 117	
Figura 5.73: Módulos utilizados para implementar a interface IEEE 1451.5.....	118
Figura 5.74: X-CTU parâmetros para analisar a eficiência da transmissão.....	119

Lista de Abreviaturas

A/D	Analógica/Digital
AoA	<i>Angle of Arrival</i> (Ângulo de Chegada)
AP	<i>Access Point</i>
API	<i>Applications Programming Interface</i> (Interface de Programas Aplicativos)
ASIC	<i>Application Specific Integrated Circuit</i> (Circuito Integrado de Aplicação Específica)
BPSK	<i>Binary Phase Shift Keying</i>
BSA	<i>Basic Service Area</i>
BSS	<i>Basic Service Set</i>
CAD	<i>Computer Aided Design</i>
CAN	<i>Controller Area Network</i>
CANOpen	<i>Controller Area Network Open</i>
CSMA/CD	<i>Carrier Sense Multiple Access with Collision Detection</i> (Múltiplo Acesso com Verificação de Portadora com Detecção de Colisão)
D/A	Digital/Analógica
DFWMAC	<i>Distributed Foundation Wireless Medium Access Control</i>
DH	<i>Detination Address High</i>
DHCP	<i>Dynamic Host Configuration Protocol</i>
DL	<i>Destination Address Low</i>
DLL	<i>Dynamic-link library</i> (Biblioteca de ligação dinâmica)
DMC	<i>Distributed Measurement and Control</i> (Medição e Controle Distribuído)
DPRAM	<i>Dual Port RAM</i>
DSP	<i>Digital Signal Processor</i> (Processador Digital de Sinais)
DSSS	<i>Direct Sequence Spread Spectrum</i>
ESA	<i>Extend Service Area</i>
ESS	<i>Extend Service Set</i>
ESSID	<i>Extended Service Set ID</i>
FFD	<i>Full Functions Device</i>
FHSS	<i>Frequency Hopping Spread Spectrum</i>
FM	<i>Frequency Modulation</i>
FPGA	<i>Field Programmable Gate Array</i> (Matriz de Portas Lógicas Programáveis em Campo)
GUI	<i>Graphical User Interface</i> (Interface Gráfica com o Usuário)
HAL	<i>hardware abstraction layer</i>
I/O	<i>Input/Output</i> (Entradas / Saídas)
IDL	<i>Interface Definition Language</i> (Linguagem de Definição de Interfaces)
IEEE	<i>Institute of Electrical and Electronics Engineers</i> (Instituto de Engenheiros Eletricistas e Eletrônicos)
IP	<i>Internet Protocol</i> (Endereço na Internet)
ISM	<i>Industrial Scientific Medical</i>
ISO	<i>International Standards Organization</i> (Organização Internacional de

	Padronização)
J2EE	<i>Java 2, Enterprise Edition</i>
J2ME	<i>Java 2, Micro Edition</i>
J2SE	<i>Java 2, Standard Edition</i>
JDDAC	<i>Java Distributed Data Acquisiton and Control</i>
JDK	<i>Java Development Kit</i> (Kit de desenvolvimento Java)
JIT	<i>Java Transducer Interface</i>
JMCI	<i>Java Measurement Calculus Interface</i>
JMDI	<i>Java Measurement Dataflow Interface</i>
JRE	<i>Java Run-Time Environment</i> (Ambiente para execução Java)
JTAG	<i>Joint Test Action Group</i>
JVM	<i>Java Virtual Machine</i> (Máquina Virtual Java)
L2CAP	<i>Logical Link Control and Adaption layer Protocol</i>
LAM	<i>Link Adaptation Management</i>
LAN	<i>Local Area Network</i> (Redes Locais)
LCD	<i>Liquid Cristal Display</i>
MAC	<i>Media Access Control</i> (Controle de Acesso ao Meio)
MES	<i>Manufacturing Execution Systems</i>
MMI	<i>Modulo Multi Interface</i>
MMI	<i>Mixed-Mode Interface</i>
MPU	<i>Microprocessor Unit</i> (Unidade Microprocessadora)
MY	<i>Source Address</i>
NCAP	<i>Network Capable Application Processor</i> (Processador de Aplicação com Capacidade de Operar em Rede)
NIST	<i>National Institute of Standards and Technology</i> (Instituto Nacional de Padrões e Tecnologias)
NRSS	<i>Numeric Really Simple Syndication</i>
OFDM	<i>Orthogonal Frequency Division Multiplexing</i>
OO	<i>Object Oriented</i> (Orientado a Objeto)
OQPSK	<i>Offset Quarternary Phase Shift Keying</i>
OSI	<i>Open System Interconnection</i>
PC	<i>Personal Computer</i> (Computador Pessoal)
PLD	<i>Programmable Logic Device</i> (Dispositivo de Lógica Programável)
PLL	<i>Phase-Locked Loop</i>
RFD	<i>Reduced Function Device</i>
RFID	<i>Radio-Frequency IDentification</i>
RISC	<i>Reduced Instruction Set Computer</i>
RSSI	<i>Received Signal Strength Indication</i> (Indicação da Força de um Sinal Recebido)
RTD	<i>Resistance Temperature Detector</i>
RTOS	<i>Real Time Operating System</i>
SAF	Software para Automação com FPGA
SCADA	<i>Supervisory Control and Data Aquisition</i> (Sistema de Supervisão Controle e Aquisição de Dados)
SDRAM	<i>Synchronous Dynamic RAM</i>
SoC	<i>System on chip</i>
SOPC	<i>System on a Programmable Chip</i> (Sistema programável em um único

	Chip)
SPC	<i>Set Parameter Communication</i> (Configuração de Parâmetro de Comunicação)
SPI	<i>Serial Peripheral Interface</i>
SPWM	<i>Sinusoidal Pulse Width Modulation</i>
STIM	<i>Smart Transducer Interface Module</i> (Módulo de Interface para Transdutores Inteligentes)
TBIM	<i>Transducer Bus Interface Module</i> (Módulo de Interface em Barramento para transdutor)
TCP	<i>Transmission Control Protocol</i>
TEA	<i>Transmission Error Approximation</i> (Erro Aproximado da Transmissão)
TEDS	<i>Transducer Electronic Data Sheet</i> (Especificações de Transdutor em Formato Eletrônico)
TII	<i>Transducer Independent Interface</i> (Interface Independente para Transdutores)
TIM	<i>Transducer Interface Module</i> (Módulo de Interface para Transdutor)
TOA	<i>Time of Arrival</i> (Tempo de Chegada)
UDP	<i>Transmission Control Protocol</i>
USB	<i>Universal Serial Bus</i>
VHDL	<i>Very High Speed Integrated Circuit - Hardware Description Language</i> (Linguagem de descrição de Hardware para circuitos integrados de velocidade muito elevada <i>VHSIC</i>)
WEP	<i>Wired-Equivalent Privacy</i>
WTIM	<i>Wireless Transducer Interface Modules</i> (Módulo de Interface para Transdutor Sem Fio)
XML	<i>eXtensible Markup Language</i>

Lista de Tabelas

Tabela 2.1: Especificações dos módulos 802.11b.....	50
Tabela 2.2: Tipos de enlace Bluetooth.....	53
Tabela 2.3: Especificações dos módulos Bluetooth.....	54
Tabela 2.4: Especificações dos módulos Xbee.....	56
Tabela 4.5: Informações sobre o hardware MMI.	81
Tabela 4.6: Recursos básicos para compor o TIM no Cyclone II.....	84
Tabela 4.7: Configuração da TEDS da plataforma IEEE 1451.....	88
Tabela 5.8: Recursos do TIM, para automação do kit de hidráulica no Cyclone II.....	96
Tabela 5.9: Especificações do medidor de Vazão.....	104
Tabela 5.10: Especificações do medidor de Pressão.....	104
Tabela 5.11: Recursos utilizados pelo TIM - processo de Filtragem de Biogás.	105
Tabela 5.12: Conversão de corrente para tensão.....	107
Tabela 5.13 : Análise da transmissão do módulo BTS1009C com linha de visada.....	120
Tabela 5.14: Análise da transmissão do módulo Xbee com linha de visada.	120
Tabela 5.15: Análise da transmissão do módulo INT550 ME com linha de visada.....	120
Tabela 5.16: Análise da transmissão do módulo BTS1009C sem linha de visada.....	121
Tabela 5.17: Análise da transmissão do módulo Xbee sem linha de visada.....	121
Tabela 5.18: Análise da transmissão do módulo INT 550 ME sem linha de visada.....	121

Sumário

Edson Antonio Batista.....	1
Abstract.....	5
Lista de Figuras.....	6
Lista de Abreviaturas.....	8
Lista de Tabelas.....	11
Sumário.....	13
CAPÍTULO 1 - Introdução	15
1.1 Revisão do Estado da Arte – Padrão IEEE 1451.....	15
1.1.1 IEEE 1451.1.....	18
1.1.2 Software NCAP - NIST 1451.....	19
1.1.3 JDDAC	20
1.1.3.1 Configuração da plataforma JDDAC.....	21
1.1.3.2 JDDAC com Tecnologia Java.....	23
1.1.4 IEEE 1451.2.....	23
1.1.5 IEEE 1451.5	25
1.2 Objetivos.....	29
1.2.1 Objetivos Específicos.....	29
1.3 Descrição do Trabalho	30
1.4 Justificativa.....	33
1.5 Motivação.....	34
1.6 Trabalhos Correlatos	35
1.7 Organização do Trabalho.....	43
1.CAPÍTULO 2 - Tecnologias IEEE 1451.5.....	43
2.1 Introdução.....	44
2.2 IEEE 802.11 – Wi-Fi	44
2.2.1 Protocolo MAC.....	47
.....	47
2.2.2 Roaming.....	47
2.2.3 Técnicas de Codificação.....	48
2.2.4 Segurança.....	49
2.2.5 Configuração de dispositivo Wi-Fi utilizado na plataforma IEEE 1451	49
2.3 IEEE 802.15.1 – Bluetooth.....	50
2.3.1 Descrição Técnica.....	51
2.3.2 Configuração da Comunicação de um sistema Bluetooth.....	52
2.4 IEEE 802.15.4 – ZigBee.....	54
2.4.1 Modulação da tecnologia ZigBee.....	55
CAPÍTULO 3 – Softwares NCAP da Plataforma IEEE 1451.....	57
3.1 Introdução.....	57
3.2 Software NCAP da Plataforma IEEE 1451 – Primeira Versão.....	57
3.3 Software NCAP da Plataforma IEEE 1451 – Segunda Versão.....	63
CAPÍTULO 4 – TIM da Plataforma IEEE 1451.....	69
4.1 Introdução.....	69
4.2 MMI e DriverTIM.....	69
4.2.1 Desenvolvimento do Módulo Embarcado da plataforma IEEE 1451.....	72
4.2.2 Desenvolvimento do Firmware do MMI.....	82
4.3 Parte Hardware da Plataforma IEEE 1451	84
4.3.1 Template IEEE 1451 e Firmware do DriverTIM.....	85

4.4 TEDS da Plataforma IEEE 1451.....	88
2.CAPÍTULO 5 - Plataforma IEEE 1451 – Testes e Resultados.....	93
5.1 Introdução.....	93
5.2 Automação do Kit de Hidráulica – MHI-01.....	93
5.2.1 Circuitos de Condicionamento de Sinais – MHI-01.....	96
5.3 Automação do Processo de Filtragem de Biogás.....	101
5.3.1 Circuitos de Condicionamento de Sinais - Filtragem de Biogás.....	105
5.3.2 Automação do protótipo de Filtragem do Biogás	108
5.3.3 Plataforma IEEE 1451 – Processo de Filtragem do Biogás.....	114
5.4 Comunicação do Software Elipse com o DriverTIM.....	117
5.5 Análise de desempenho dos módulos sem fio utilizados para implementar as interfaces IEEE 1451.5.....	118
5.5.1 Testes das interfaces IEEE 1451.5 com linha de visada.....	120
5.5.2 Testes das interfaces IEEE 1451.5 sem linha de visada.....	121
CAPÍTULO 6 - Conclusões e Trabalhos Futuros.....	123
6.1 Trabalhos Futuros.....	127
Referências	129
APÊNDICE A – CONFIGURAÇÕES MÓDULOS SEM FIO.....	138
3.APÊNDICE B – DOCUMENTAÇÃO DO SOFTWARE NCAP.....	146
APÊNDICE C – DESENVOLVIMENTO DE HARDWARE COM NIOS II.....	160
APÊNDICE D – INFORMAÇÕES COMPLEMENTARES.....	172
4.APÊNDICE E – FIRMWARE DO DRIVERTIM.....	187
ANEXO A – COMUNICAÇÃO DO ELIPSE E3 COM DRIVERTIM.....	193

CAPÍTULO 1 - Introdução

A conexão de transdutores em rede de comunicação de maneira flexível e padronizada favorece ao sistema de monitoramento e controle distribuído mais funcionalidades. O desenvolvimento e a implementação de uma rede de transdutores envolve áreas multidisciplinares, como por exemplo, instrumentação eletrônica, engenharia de software, técnicas de controle, barramento de campo e rede de comunicação. Desta forma, o desenvolvimento de um sistema padronizado para conectar transdutores em rede de comunicação é complexo do ponto de vista operacional, ou seja, necessita-se do domínio de conhecimento em conceitos e ambientes de atuação diversificados. Atualmente, a conexão de transdutores em rede de comunicação é realizada de forma proprietária, tornando as aplicações com pouca flexibilidade de expansão e com dependência de um único fabricante na manutenção. O padrão *IEEE 1451* possui normas que viabilizam a conexão de transdutores em rede de comunicação de forma padronizada. Neste trabalho de doutorado o tema central é o desenvolvimento e a implementação de um sistema para conectar transdutores em rede de comunicação em conformidade com o padrão *IEEE 1451*.

1.1 Revisão do Estado da Arte – Padrão IEEE 1451

O padrão *IEEE 1451* surgiu por volta de 1997 para suprir as necessidades na construção de redes de transdutores inteligentes. A proposta de implementação deste padrão foi feita pelo *National Institute of Standard and Technology (NIST)* e o *Institute of Electrical and Electronics Engineers (IEEE)*. Estes dois órgãos formaram comitês específicos para gerar normas que devem servir como base para a construção de rede de transdutores inteligentes. As normas *IEEE 1451* contém os quesitos que uma interface padronizada deve possuir para a construção de uma rede de transdutores inteligentes. Neste aspecto, o comitê *IEEE 1451* define como transdutores inteligentes, os dispositivos (sensores/atuadores) que podem ser conectados em rede de comunicação através de interfaces padronizadas.

O padrão *IEEE 1451*, não define qual tecnologia deve ser utilizada para a construção das interfaces, mas é enfático na utilização de ferramentas padronizadas,

visando a interligação amigável dos diversos tipos de transdutores, nas diferentes redes de comunicação industrial. Atualmente, existem sete comitês *IEEE 1451* responsáveis em desenvolver as seguintes interfaces :

Comitê *IEEE 1451.0*: Este comitê é responsável por elaborar as normas para especificar as características dos transdutores, na rede de comunicação. As informações dos transdutores devem estar em formato eletrônico, regidas pela norma, sendo esta definida como *Transducer Electronic Data Sheet (TEDS)*. Na primeira homologação do padrão *IEEE 1451.2*, a *TEDS* deveria estar acoplada ao *Smart Transducer Interface Module (STIM)*. Por volta de 2004, a *TEDS* passou a ser tratada pelo comitê *IEEE 1451.0*, sendo comum para as demais interfaces. Com a implementação da *TEDS* um transdutor pode ser identificado em uma rede de comunicação, fornecendo informações como fabricante, número de manobras realizadas, faixa de atuação e número do canal no qual está conectado .

Comitê *IEEE 1451.1*: Definida como *Network Capable Application Processor (NCAP)*, esta interface é comum para interligação entre as diferentes redes industriais e as demais interfaces *IEEE 1451*. Para desenvolver o *NCAP*, o projetista deve atentar para duas funcionalidades básicas deste processador: trocar informações com a rede de controle e gerenciar as atividades dos transdutores .

Comitê *IEEE 1451.2*: Descreve a norma para a construção da rede de transdutores pela conexão por cabeamento, utilizando protocolo serial. Este comitê inicialmente não especificava o tipo de protocolo dos *STIM*, porém, desde 2004 o comitê *IEEE 1451.2* sugere a utilização de interfaces como *RS232*, *RS485*, *RS422*, *I2C*, *USB* e *SPI*. Na primeira homologação do padrão *IEEE 1451.2*, o *STIM* deveria ser conectado na Interface para Transdutores Inteligentes, *Transducer Interface Independent (TII)*. A *TII* deve possuir um barramento com dez linhas de dados, sendo que a principal função dessa interface é possibilitar o sistema de *plug and play* entre o *NCAP* e os *STIMs*, ou seja, diferentes *STIMs* poderiam ser conectados e controlados por um mesmo *NCAP*. A norma *IEEE 1451.2* vem passando por atualizações, que devem ocorrer principalmente em relação à necessidade ou não da *TII* .

Comitê *IEEE 1451.3*: Desenvolve as normas para construir uma interface que possibilite a conexão dos módulos de transdutores em barramento, *Transducer Bus Interface Module (TBIM)*. O *TBIM* deve conter um barramento para conectar dispositivos com sinal analógico ou digital, viabilizando a construção de rede de transdutores dotados destes sistemas de operação .

Comitê *IEEE 1451.4*: Responsável pelo desenvolvimento de uma interface de modo mista, cuja principal função é de possibilitar a conexão de transdutores analógicos e digitais em módulos comuns. Esta interface é denominada Modo de Interface Mista, *Mixed-Mode Interface (MMI)* .

Comitê *IEEE 1451.5*: Este comitê é responsável por desenvolver as normas para interligar o *NCAP* à rede de transdutores através da comunicação sem fio. O comitê *IEEE 1451.5* teve a adesão de empresas como *Ericson*, *Siemens*, *Agilent Technologies*, *Sun Microsystems*, entre outros. As tecnologias apresentadas para a construção da interface *IEEE 1451.5* são as tecnologias: *ZigBee* , *Bluetooth* e *Wi-Fi* . Estas três tecnologias têm em comum o fato de serem padronizadas *IEEE 802.15.1*, *IEEE 802.15.4* e *IEEE 8012.11*, respectivamente, além de serem as tecnologias populares no mercado. A interface *IEEE 1451.5* é definida como *Wireless Transducer Interface Module (WTIM)* .

Comitê *IEEE 1451.6*: A rede *Controller Area Network (CAN)* é uma rede de comunicação desenvolvida pela *Bosch* e tem alto nível de utilização na indústria automobilística em sistemas embarcados. O comitê propõe a tecnologia *CanOPEN* *Cia (Control in Automation) DS404* para implementar uma rede de sensores .

Comitê *IEEE 1451.7*: Recentemente vêm-se discutindo a possibilidade de desenvolver uma norma *IEEE 1451* para identificação por rádio frequência *Radio Frequency Identification (RFID)* .

Na Figura 1.1, mostra-se a configuração do padrão *IEEE 1451*, sendo que os transdutores devem ser conectados nas interfaces conforme a arquitetura da rede de transdutores a ser desenvolvida.

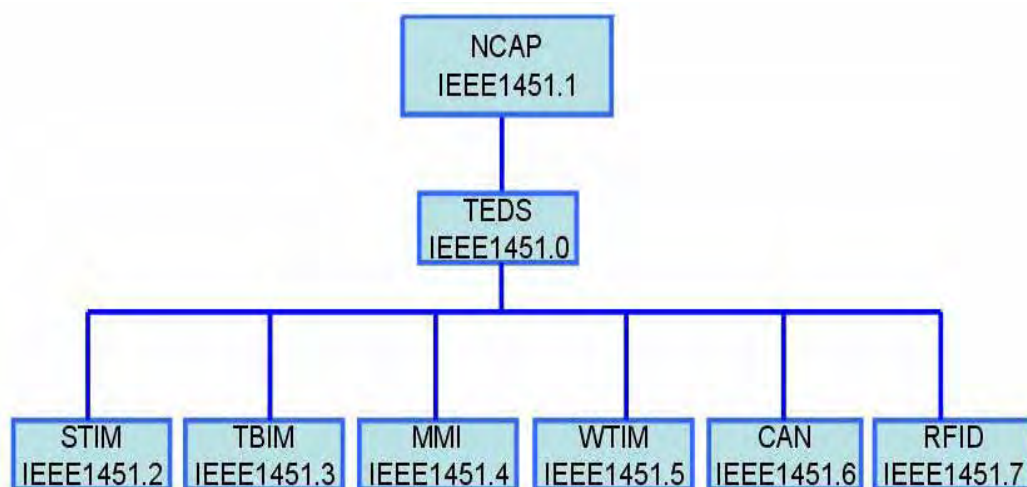


Figura 1.1: Arquitetura do sistema *IEEE 1451*.

Neste trabalho utilizou-se basicamente a norma *1451.1*, *1451.2* e *1451.5*, descrita nos itens 1.5.1, 1.5.4 e 1.5.5 respectivamente.

1.1.1 IEEE 1451.1

O *NCAP* é o elemento central de uma rede de transdutores inteligentes, tendo como principal função gerenciar as atividades dos transdutores e se comunicar com uma rede de controle. Um módulo *NCAP* deve possuir um processador que seja capaz de implementar atividades específicas, como por exemplo, conectar em qualquer tipo de rede e possuir capacidade de controlar as interfaces *IEEE 1451*. O *NCAP* é basicamente um nó de rede composto de partes lógicas e partes físicas .

Na Figura 1.2 apresenta-se a configuração do *NCAP* definida no padrão *IEEE 1451.1*, sendo que as Interfaces de Programação para Aplicações (*API*) são os blocos softwares responsáveis pela funcionalidade da aplicação, atuando diretamente no tratamento de dados da rede de comunicação, na geração da *TEDS* e nas interfaces *IEEE 1451.x* (*IEEE 1451.2*, *IEEE 1451.3*, *IEEE 1451.5*, *IEEE 1451.6* e *IEEE 1451.7*). As interfaces *IEEE 1451.x* desempenham as atividades do *drivers* de comunicação com *TIM* e a *TEDS* deve fornecer as especificações da rede de transdutores ao operador.



Figura 1.2: Configuração de um *NCAP*, em conformidade com o padrão *IEEE 1451*.

O *NCAP* deve possuir as seguintes características:

- a) **Interoperabilidade:** A interoperabilidade é a característica pela qual a interface *NCAP* deve abstrair o tipo de rede na qual será conectada. Em termos práticos, um hardware para desempenhar a interoperabilidade deve possuir um

processador que permita a instalação de softwares que controlam as atividades da rede de comunicação. A interoperabilidade do *NCAP* fornece maior flexibilidade na construção de rede de transdutores.

- b) **Bloco Software Orientado a Objeto:** A parte lógica do processador realiza a comunicação com as interfaces *1451.x* e com a rede de controle. Este bloco deve possuir alto grau de abstração da aplicação, sendo sugerida pelo comitê *1451.1* a utilização de programação orientada a objeto, para desenvolver o software. Outra característica essencial deste bloco é a interface gráfica, que deve permitir ao operador detalhes das atividades dos transdutores como os diversos softwares supervisórios utilizados na indústria de automação, como por exemplo, *Eclipse SCADA da Eclipse Software*, *RSView da Rockwell* e *WinCC da Siemens*.
- c) **Driver do Módulo para Transdutor Inteligente:** Este hardware tem a função de possibilitar ao *NCAP* o controle do módulo para interface de transdutor (*TIM*), ou seja, um *driver* de comunicação. Este *driver* deve estar de acordo com o módulo no qual serão conectados os transdutores, por exemplo, se o módulo é para a construção de uma rede de transdutores sem fio, então o *driver* deve seguir a norma *IEEE 1451.5*.

Nos itens 1.1.2 e 1.1.3 apresentam-se uma breve descrição de *NCAPs* desenvolvidos pelo *NIST* e pela *Agilent Technologies* respectivamente.

1.1.2 Software NCAP - NIST 1451

O *NIST* desenvolveu um software supervisório com base no padrão *IEEE 1451.1*, utilizado no monitoramento das atividades de um sistema de refrigeração. Este software foi desenvolvido com tecnologia Java e especificamente a biblioteca *Applet* para fornecer informações atualizadas dos estados dos transdutores (sensores de temperatura e válvulas) e da direção do fluxo de água para a refrigeração de um ambiente. Na Figura 1.3 mostra-se a tela do sistema de *DMC* com supervisão via *Internet* desenvolvida pelo *NIST* com base no padrão *IEEE 1451.1*.

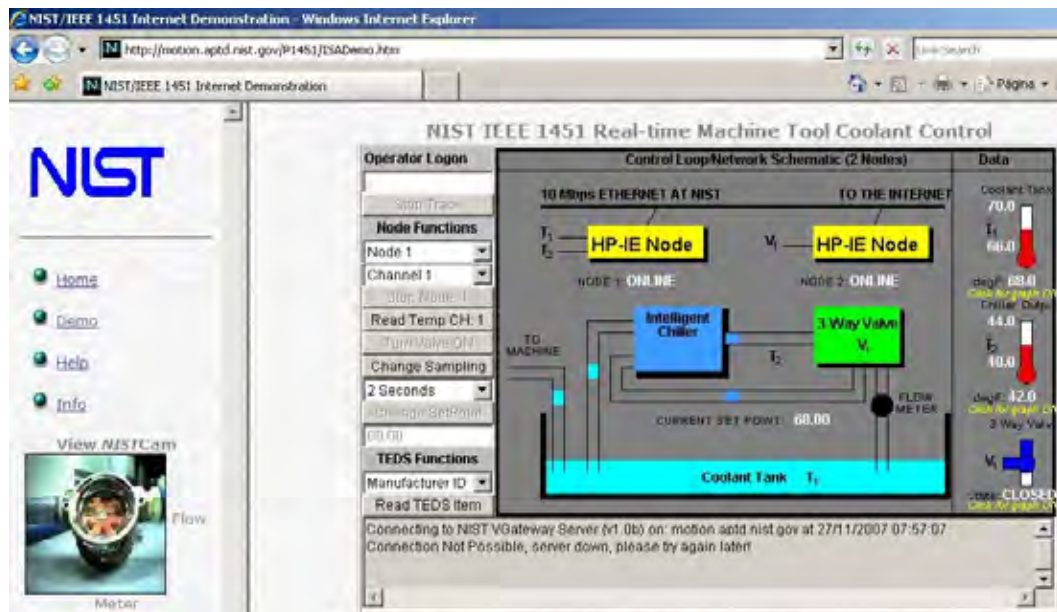


Figura 1.3: Tela do software supervisorio do *NIST IEEE 1451*.

Fonte: [HTTP://motion.aptd.nist.gov/P1451/ISADemo.htm](http://motion.aptd.nist.gov/P1451/ISADemo.htm)

1.1.3 JDDAC

A *Agilent* e a *Sun Microsystems* desenvolveram uma plataforma para software supervisorio com base no padrão *IEEE 1451.1*. Esta plataforma é denominada de *JDDAC* (*Java Distributed Data Acquisition and Control*). O *JDDAC* é uma plataforma de código aberto, orientado a objeto que visa a aquisição de dados de uma rede de sensores.

As principais características do *JDDAC* são:

- Baseado no padrão *IEEE 1451*;
- Possui todas as plataformas Java: *J2SE*, *J2ME* e *J2EE*;
- Funcionalidade com rede sem fio;
- Projeto para um ambiente com ciclo de vida dinâmico;
- Suporta diferentes tipos de aplicações.

A arquitetura do pacote da plataforma *JDDAC* é apresentada na Figura 1.4.



Figura 1.4: Arquitetura da plataforma *JDDAC*.

1.1.3.1 Configuração da plataforma JDDAC

A *Agilent Technologies* definiu diversos parâmetros para compor a plataforma *JDDAC*, como por exemplo, o modelo objeto, modelo de comunicação, as funções das mensagens, modelo dos dados e modelo das medidas. Sendo que, no modelo objeto um nó baseado na plataforma *JDDAC* deve conter múltiplas funcionalidades (*F-Blocks*) para atuar e monitorar diferentes processos. No modelo de comunicação a rede deve ser construída com blocos de comunicação que suportam um ou mais canais de comunicação. Os blocos de comunicação devem possuir as seguintes características:

- O formato do dado deve ser independente do tipo de rede;
- Estar sempre disponível para a comunicação;
- Suportar os modelos de comunicação *cliente-servidor* e *publish-subscribe*.

As mensagens podem ser armazenadas, reenviadas e circular entre diferentes nós e dentro de cada nó, a mensagem passa por um processo de *pipeline*. Como o processo de *pipeline* inclui criptografia e compressão, a comunicação deve ser em *XML*

(*eXtensible Markup Language*) sobre *http* e outros tipos de transportes devem ser disponíveis na comunicação.

No modelo de dados, o *JDDAC* deve operar com os diferentes tipos de dados como inteiros, string, além de incluir tipos de dados primitivos (*int* e *float*). E no modelo de medidas uma medida típica pode conter: valor, *timestamp*, localização, qualidade, unidade, margem de erro e propriedade. Na Figura 1.5, apresentam-se as informações que uma medida deve possuir.

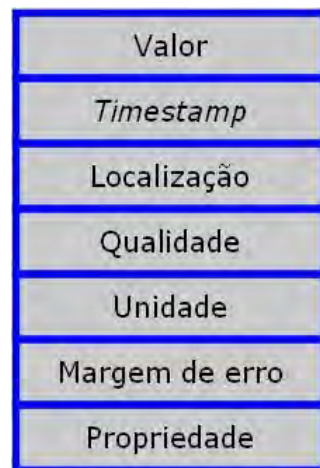


Figura 1.5: Informações necessárias em uma medida.

Sendo que:

- Valor – valor atual do dado.
- *Timestamp* – o momento que a medida foi realizada.
- Localização – a localização onde a medida foi realizada.
- Qualidade – a fonte de medida (medição, simulação, etc).
- Unidade – unidade de medida.
- Margem de erro – a margem de erro da medida.
- Propriedade – as propriedades da medida.

Um software supervisor baseado na plataforma *JDDAC* deve proporcionar auto-identificação dos transdutores, fornecer informações sobre a camada física e como a comunicação é realizada com o transdutor, informar a descrição de cada transdutor e a caracterização das medidas, ou seja, implementar os conceitos do padrão *IEEE 1451.0*.

1.1.3.2 JDDAC com Tecnologia Java

As plataformas Java que compõem o *JDDAC* são:

- *J2EE*: plataforma para desenvolver aplicações em servidor .
- *J2ME*: para desenvolver aplicações para dispositivos móveis e *TV* digital .
- *J2SE*: plataforma para desenvolver aplicações em *desktop* .

A plataforma *JDDAC* é composta pelas seguintes bibliotecas:

- *JTI* - *Java Transducer Interface*: nesta biblioteca são definidas como são as integrações com os transdutores, com base nas interfaces *IEEE 1451*.
- *JMDI* - *Java Measurement Dataflow Interface*: especifica como o dado é representado, transportado e processado, esta biblioteca é baseada nas funcionalidades do padrão *IEEE 1451.1*.
- *JMCI* - *Java Measurement Calculus Interface*: deve suportar operações aritméticas e lógicas para realizar medidas.
- *NRSS* - *Numeric Really Simple Syndication*: essa biblioteca deve fornecer um banco de dados com histórico de acesso, além de divulgar os dados.

A plataforma *JDDAC* é de grande relevância para concretização do padrão *IEEE 1451* na indústria, pois, compõe-se de recursos que podem atender as necessidades atuais das redes de transdutores.

Assim como o sistema *JDDAC*, o software que compõe a plataforma *IEEE 1451*, desenvolvida neste projeto, busca atender os requisitos de um sistema *SCADA* e em conformidade com o padrão *IEEE 1451.1*. O desenvolvimento deste software é descrito no item 3.3.

1.1.4 IEEE 1451.2

A primeira versão desta norma foi homologada por volta do ano 1997, porém, recentemente está sendo revisada, principalmente na interface de conexão entre *NCAP* e

STIM. Recentemente, vem sendo discutido a possibilidade de tornar esta interface (*TII*) opcional, quando o *plug-and-play* pode ser obtido por outras configurações. Uma configuração que pode substituir a *TII* são as interfaces seriais, como por exemplo, *RS232*, *RS485*, *422* e *Universal Serial Bus (USB)* .

Em 2008, pesquisadores do *NIST*, publicaram dois artigos. Uma publicação foi na revista *Intrumentation and Measurement* e a outra no congresso *IMTC (Instrumentation and Measurement Technology Conference)*, descrevendo a implementação de um nó *IEEE 1451* utilizando os recursos da interface *RS232*. No artigo publicado na revista foi apresentado um nó *IEEE 1451.5* no qual a parte física do *NCAP* foi implementada em *PC* e a parte lógica programada com tecnologia Java. O módulo para transdutores foi desenvolvido em um *PC* em conjunto com uma placa de microcontroladores. O *NCAP* comunica-se com o *TIM* através de *Access Point (IEEE 802.11)* e a placa de microcontroladores foi conectada ao *TIM* por cabeamento via interface *RS232*. A placa de microcontrolador compõe-se de um sensor de temperatura, ou seja, neste nó *IEEE 1451*, desenvolveu-se um canal transdutor . O artigo publicado no *IMTC* descreve a arquitetura de um nó *IEEE 1451.2*, sendo que o *NCAP* foi desenvolvido em um *PC* programado com tecnologia Java e uma placa com microprocessador *8052*, memória e um elemento sensor para executar as funções. Esta placa foi conectada ao *PC* por cabeamento via interface *RS232* e a *TEDS* pôde ser programada com linguagem *C/C++* e alocada em uma memória *RAM*. Em ambos os artigos mostram-se a versatilidade de tornar as informações do sensor em rede de comunicação, que é um dos tópicos fundamentais, porém, o módulo para transdutor possui pouca flexibilidade para gerar mais de um canal transdutor .

Um grupo de pesquisadores da *NASA* e da universidade de *Houston* publicou um trabalho que descreve um sistema para rede de sensores sem fio (*IEEE 1451.5*). Neste sistema o *TIM* é composto por uma placa com microprocessador, módulo *ZigBee*, memória flash e conversor *A/D*. Na placa pode-se acoplar um ou mais sensores e as informações destes dispositivos são salvos na memória flash. Outra placa contendo um módulo *ZigBee* é conectada ao *PC* através da interface *RS232* ou *Ethernet*, as atividades e as informações dos sensores podem ser monitoradas por uma aplicação Web .

Em 2009, foi publicado um artigo descrevendo as funcionalidades de uma placa contendo interfaces para *Can*, *Ethernet*, *RS485*, *RS232* e *USB*. Esta placa foi denominada de *Multi-port serial NCAP* e têm como finalidade possibilitar o desenvolvimento de diferentes normas *IEEE 1451*, ou seja, realiza as atividades do

NCAP. A comunicação entre esta placa e o módulo para transdutores é realizada com um protocolo unificado, permitindo a interoperabilidade entre algumas normas *IEEE 1451*. Além do suporte para diferentes *drivers* de comunicação a *Multi-port serial NCAP*, possui memórias *RAMs* e *ROM*, um microcontrolador da família *8051* e um *chip* produzido pelo laboratório *Silicon*, que permite a seleção dos *drivers* de comunicação.

Um software foi desenvolvido para gerenciar as atividades com a rede de comunicação e com os módulos para transdutores, possibilitando que as informações das *TEDS* sejam disponibilizadas na Internet. A comunicação sem fio pode ser desenvolvida com uma placa contendo um módulo *ZigBee* e interface *RS232*, propõe os autores . Este artigo escrito por integrante do comitê *IEEE 1451*, evidencia a inserção de interfaces seriais na revisão da norma *IEEE 1451.2*, com a finalidade de possibilitar o *plug and play* entre o *NCAP* e o *TIM*.

1.1.5 IEEE 1451.5

O *NIST* e o comitê *IEEE 1451.5* organizaram diversos *workshops*, para discutir a formatação da norma *IEEE 1451.5*. Estes eventos foram realizados nos Estados Unidos da América (*USA*) e teve a participação de empresas e centros de pesquisas.

A interface *IEEE 1451.5* foi oficialmente proposta no primeiro *Workshop on Wireless Sensing*, realizado no Centro de Convenções *Rosemont* na cidade de *Chicago, Illinois (IL)* em Junho de 2001 e teve a participação de instituições como *NIST*, *3e Technology International*, *Oak Ridge National*, *Ohio State University* e a *Oceana Sensor*. Neste evento, foi apresentada uma descrição do padrão *IEEE 1451 a importância e as características* que a comunicação sem fio deve possuir para desenvolver a interface *IEEE 1451.5*. Entre as características apresentadas está o triângulo de funcionalidade, no qual deve ser projetado neste tipo de comunicação. O triângulo de funcionalidade tem nos vértices da base a confiabilidade e a integridade dos sinais e no vértice superior a eficiência da tecnologia . Na Figura 1.6 pode-se observar a estrutura deste triângulo .



Figura 1.6: Triângulo de características para a tecnologia sem fio.

O segundo *workshop*, sobre a interface *IEEE 1451.5* foi realizado na cidade de *Philadelphia, Pennsylvania (PA)* em 4 de Outubro de 2001. Neste evento pode-se destacar a participação de empresas como *Motorola* descrevendo o sistema *IEEE 802.15.4 (ZigBee)* e a *Ericsson Mobile Communications* apresentando sobre a tecnologia *Bluetooth*.

Em 20 de Maio de 2002 em *San Jose, California (CA)* foi realizado o terceiro *workshop*. Neste evento foram apresentados os seguintes seminários :

- *Kang Lee* realizou uma descrição do padrão *IEEE 1451* e apresentou uma proposta de comunicação sem fio entre o *NCAP* e o *TIM*. Nesta proposta foram sugeridas as características de hardware e software para desenvolver a automação sem fio em conformidade com o padrão *IEEE 1451.5*. Entre as características está o consumo de energia, formato das mensagens, o tipo de antena e a faixa de frequência utilizada pelos módulos sem fio.
- *Mike Moore* e *Steve Smith* da *Oak Ridge National Laboratory* apresentaram como proposta para desenvolver a camada física da *P1451.5*, o padrão *IEEE 802*, especificamente os sistemas *IEEE 802.11*, *IEEE 802.15*, e *IEEE 802.16*.
- *Prabal Dutta* realizou uma apresentação intitulada de *Security Considerations for Wireless Sensor Networks*, na qual, destacou-se os aspectos da segurança que uma rede sem fio deve possuir. Segundo *Dutta*, neste tipo de rede é fundamental avaliar os seguintes aspectos: integridade na comunicação, confiabilidade na transmissão de dados (proteger os dados não somente com criptografia) e autenticidade das mensagens.
- *David Perrussel*, da *Naval Surface Warfare Center de Dahlgren Division Dahlgren, Virginia (VA)*, destacou as seguintes características que uma rede de

sensores sem fio deve possuir como aplicação, faixa de atuação (frequência), alcance da transmissão, quantidade de nó suportada pela rede, tempo de espera, tipo de alimentação e posição da antena (interna ou externa).

- *Lance Hester e Ken Cornett da Florida Communication Research Lab Motorola* apresentaram uma descrição do padrão *IEEE 802.15.4*.

Em 24 de Setembro de 2002 na cidade de *Boston, Massachusetts (MA)*, foi realizado o quarto *workshop*. Neste evento foram apresentados os seguintes seminários :

- *Kent Cornett da Motorola* apresentou uma proposta para desenvolver a *IEEE 1451.5* utilizando como base os padrões *IEEE 802.11*, além de destacar algumas áreas para aplicação da automação sem fio, como por exemplo: segurança, agricultura de precisão, robótico, entretenimento, domótica e biomédica.
- *Myung Lee da Samsung* apresentou uma proposta para implementar a *IEEE P1451.5*. O padrão *802.15.4 (ZigBee)* é utilizado como protocolo de comunicação e são descritas todas as etapas necessárias para desenvolver a comunicação entre o *NCAP* e o *TIM*, porém o tema central da proposta está na adaptação da camada de aplicação para o *Link Adaptation Management (LAM)*. Nessa proposta apresenta as seguintes justificativas para utilizar o *802.15.4*: baixo consumo de energia, fácil instalação, baixo custo, comunicação segura devido ao controle de acesso, possibilidade de 256 nós e possibilidade de taxa de transmissão acima de 113 bytes.

Em 22 de Setembro de 2003 na cidade de *Anaheim, California (CA)*, foi realizado o quinto *workshop*. Neste evento pode-se destacar as seguintes apresentações :

- Integrante da *3eTI (3e Technologies International)* apresentou uma proposta para a arquitetura do protocolo de comunicação *1451.5*, como por exemplo, as terminologias do protocolo e as especificações de diferentes tecnologias para compor a interface sem fio (*IEEE 802.15.1, IEEE 802.15.3, IEEE 802.15.4 e IEEE 802.11*).
- *Wayne W. Manges da Oak Ridge National Laboratory (U.S. Department of Energy)*, apresentou uma palestra intitulada “*Wireless Industrial*”.

Networking Alliance (WINA) supported 1451.5 Committee”. Pode-se observar nos slides desta apresentação uma evidência no mercado da automação sem fio, as dificuldades para utilizar este sistema na indústria e às regras para atender as necessidades deste tipo de aplicação.

Em Junho de 2004, na cidade de *Detroit, Michigan (MI)*, foi realizado o sexto *workshop*, para desenvolvimento da norma *IEEE1451.5*. Neste evento ocorreram as seguintes apresentações :

- *Kang Lee* apresentou uma descrição da família *IEEE 1451* e a proposta de utilizar as tecnologias *Wi-Fi*, *Bluetooth* e *ZigBee* para desenvolver a interface *1451.5*.
- *Jeff Burch* da *Agilent Technologies*, apresentou uma proposta para implementar uma interface entre *IEEE P1451.5* e *IEEE 1451.0*. Foram apresentadas as especificações de comunicação do tipo cliente/servidor ou *publish/subscribe* e os modelos das informações para identificar os transdutores.
- *Ryom Colemam*, engenheiro sênior do *3eTI* apresentou uma proposta para implementar o protocolo de comunicação *1451.5* utilizando *IEEE 802.11*. Nesta proposta utiliza-se na camada de rede o sistema *IP (Internet Protocol)* na versão 4 e na versão 6 e na camada de transporte o *TCP (Transmission Control Protocol)* ou *UDP (User Datagram Protocol)*.
- *Peter Flittner* apresentou uma proposta para implementar o protocolo de comunicação *P1451.5* utilizando *Bluetooth*. Nesta proposta destacou as características da tecnologia *Bluetooth*, como robustez, custo e consumo.
- *Ryom Colemam* apresentou alguns conceitos para desenvolver o protocolo de comunicação *1451.5*, conforme a tecnologia adotada (*Wi-Fi*, *Bluetooth* e *ZigBee*).
- *Kenneth D. Cornett* da *Motorola* apresentou uma proposta para implementar o protocolo de comunicação *P1451.5* e *IEEE 1451.0* utilizando *ZigBee*. Nos slides apresentados por *Kenneth* mostra-se os

atributos do sistema *ZigBee* na camada de acesso ao meio (*MAC*) e o tamanho (em *bytes*) de cada operação.

No dia 26 de março de 2007 foi aprovado o padrão *IEEE 1451.5*, nomeado de *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*. No texto da norma *IEEE 1451.5* são apresentadas 4 tecnologias para desenvolver a interface sem fio: *Bluetooth*, *ZigBee*, *Wi-Fi* e *6LoWPAN*. A tecnologia *6LoWPAN* (*Low Power Wireless Personal Area Networks*), utiliza na camada física e de enlace de dados o protocolo *IEEE 802.15.4* (*ZigBee*), na camada de rede suporta o Protocolo Internet (IP versão 6). Como a tecnologia *6LoWPAN* possui praticamente a mesma estrutura do *ZigBee*, as principais tecnologias para se desenvolver uma interface *IEEE 1451.5* são *Wi-Fi*, *Bluetooth* e *ZigBee*.

Com a interface *1451.5* o *NCAP* pode-se comunicar com diferentes *TIMs*, pois, as comunicações sem fio permitem que diversos nós troquem informações em uma rede. Desta forma, os *WTIM's* (*Wireless Transducer Interface Modules*) devem possuir um endereço específico, para atuar em rede de comunicação.

1.2 Objetivos

O objetivo deste trabalho é o desenvolvimento de uma plataforma de Hardware/Software para automação de processos industriais em conformidade com o padrão *IEEE 1451*.

1.2.1 Objetivos Específicos

A plataforma proposta neste trabalho é composta por hardware reconfigurável, processador embarcado e software que proporciona a reutilização de código. Almeja-se analisar a robustez destas tecnologias e inserir os conceitos do padrão *IEEE 1451* em ambiente industrial. Neste trabalho destacam-se as seguintes metas:

- Utilizar os recursos da tecnologia Java para desenvolver as funcionalidades da parte lógica de um *NCAP* (*Network Capable Application Processor*) e o

planejamento de um sistema de supervisão. O *NCAP* deve ser composto por pacotes e classes que possam ser utilizados em diferentes aplicações sem sofrer modificações bruscas. O *NCAP* deve requisitar a *TEDS* ao *TIM* e armazenar as informações recebidas em banco de dados para posteriormente ser acessada pelo operador;

- Desenvolver a comunicação entre *NCAP* e *TIM* através de interface *RS232*. Com base nesta interface deve-se desenvolver a comunicação por cabeamento (*IEEE 1451.2*) e sem fio (*IEEE 1451.5*). A comunicação sem fio deve suportar as tecnologias *Wi-Fi*, *Bluetooth* e *ZigBee*;
- O hardware da plataforma *IEEE 1451* deve realizar as funções de um *TIM* (*Transducer Interface Module*), ser dotado de processador *NIOS II* e utilizar os recursos da placa *DE2* da Altera. Este módulo deve favorecer a implementação de diferentes quantidades de canais transdutores limitando-se pela capacidade da placa *DE2*;
- Desenvolver um *template* que possa servir como base para diferentes aplicações. Este *template* deve viabilizar a comunicação com o *NCAP* e com a rede de transdutores, além de possuir suporte para descrever a *TEDS* e inserir a lógica de controle da aplicação. A descrição da *TEDS* deve ser realizada através de uma tabela de códigos;
- Testar a plataforma *HW/SW* na automação de processos por cabeamento ou sem fio, com base nas interfaces *IEEE 1451.2* e *IEEE 1451.5*. Realizar um diagnóstico de desempenho das diferentes tecnologias utilizadas na comunicação entre *NCAP* e *TIM*.

1.3 Descrição do Trabalho

Neste item são descritas as funções do *NCAP* e do *TIM* da plataforma *IEEE 1451* proposta neste trabalho, junto com uma breve metodologia para alcançar os objetivos. A plataforma *IEEE 1451* compõe-se de duas estruturas básicas, uma para realizar as atividades dos blocos funcionais do *NCAP* e outra para desenvolver as funções do *TIM*.

Na Figura 1.7 apresenta-se a descrição da plataforma *IEEE 1451*, sendo que o *NCAP* executa suas atividades em um Computador Pessoal (*PC*) e o *TIM* na placa de

FPGA DE2 da *Altera Corporation*. O *PC* possui blocos funcionais compatíveis com a norma *IEEE 1451.1*, além de ser popular em sistema de automação. Os principais blocos funcionais são: processador, sistema operacional, interface de comunicação para módulos de transdutores (*RS232*) e interface de comunicação com rede controle.

O *NCAP* é composto por pacotes desenvolvidos com tecnologia Java, que viabilizam a comunicação com a rede de controle e com o *TIM*, proporciona a implementação da *TEDS*, a geração de relatórios e o gerenciamento das atividades de aplicações. Cada pacote deve possuir classes que serão comuns nas diferentes aplicações viabilizando a reutilização de códigos. De acordo com a Figura 1.7, a parte software do *NCAP* corresponde aos blocos 1, 2, 3 e 4 e os blocos 5, 6, 7 e 8 compõem o *firmware* do módulo embarcado (*TIM*).

Para desenvolver o *TIM*, utiliza-se o *SOPC Builder* do ambiente *Quartus II* e o *NIOS II IDE* (ambos da *Altera Corporation*), descrito em *VHDL* e programado com linguagem *C/C++*, respectivamente. O módulo embarcado junto com a instrumentação eletrônica forma o *TIM*.



Figura 1.7: Arquitetura da plataforma *HW/SW* proposta neste projeto.

Na Figura 1.7 têm-se as seguintes especificações do software do *NCAP*:

1. Produzir um arquivo da *TEDS* que possa ser disponibilizado em rede de comunicação;

2. Gerar a *TEDS (Transducer Electronic Data Sheet)* no *NCAP* pela decodificação das informações fornecida pelo Módulo de Interface para Transdutores, *TIM (Transducer Module Interface)*. Essas informações devem ser devidamente armazenadas em um banco de dados e devem estar relacionadas com a rede de transdutores que compõe uma aplicação. Desta forma, o *NCAP*, possuirá uma tabela de códigos específica para gerar a *TEDS*;
3. Desenvolver as Interfaces de Programação para Aplicações, (*Application Programming Interface*) *API* do *NCAP*. As funções destas *APIs* são: fornecer ao usuário detalhes do funcionamento da aplicação (interfaces gráficas dinâmicas e geração de relatórios);
4. Possibilitar a realização de comunicação serial, especificamente com a interface *RS 232* do *PC*. Esta atividade é definida como Configuração dos Parâmetros de Comunicação, (*Set Parameter Communication*) *SPC*. Como conexão física utilizada é a interface *RS232*, quando for utilizada a comunicação sem fio, acoplam-se ao *PC* os módulos, conforme a tecnologia desejada (*Wi-Fi, ZigBee e Bluetooth*).

A plataforma *IEEE 1451* possui um *template* composto por um *firmware* que possibilita a descrição da *TEDS*, a comunicação com interface *RS232*, troca de informações com barramento dos transdutores, inserção da lógica de controle e a comunicação com *display LCD (Liquid Cristal Display)*. Este *template* foi desenvolvido com linguagem *C* e *XML* e fornece suporte para executar as funções do *TIM*.

Com base na Figura 1.7 têm-se especificações do *TIM*:

5. O *TIM* deve possuir funções que proporcione a comunicação através da interface *RS232* e quando for realizar a comunicação sem fio acoplam-se os módulos, conforme a tecnologia desejada (*Wi-Fi, Bluetooth e ZigBee*). O *firmware* do *TIM* deve ser desenvolvido em linguagem *C/C++*. Portanto o *FPGA* que compõem o *TIM* deve possuir um processador embarcado;
6. A lógica de controle das aplicações deve ser programada no *TIM*;

7. A *TEDS* no *TIM* deve ser armazenada na memória *SDRAM*, na qual, o projetista descreve as especificações de cada transdutor que compõe a aplicação por meio de uma tabela de códigos proposta neste trabalho. Essas especificações são enviadas ao *NCAP* para serem devidamente decodificadas;
8. O *TIM* deve possuir funções que proporcionem a captação de sinais oriundos da rede de transdutores.

1.4 Justificativa

O padrão *IEEE 1451* possibilita maior flexibilidade na construção da rede de transdutores e favorece a instrumentação distribuída que execute suas atividades na rede de comunicação de forma padronizada. Ou seja, os conceitos definidos no padrão *IEEE 1451*, são aplicáveis na estrutura da automação industrial.

Atualmente, a automação de processos industriais é predominantemente baseada em softwares supervisórios que se comunicam com Controladores Lógicos Programáveis (*CLP*) por meio de *drivers* específicos. Para atender a norma *IEEE 1451* o sistema de Supervisão, Controle e Aquisição de Dados (*SCADA*) deverá sofrer alterações nos protocolos de comunicações, permitindo que hardware e software de diferentes fabricantes possam se reconhecer de modo único.

Para implementar as funcionalidades de um Módulo de Interface para Transdutor (*TIM*) em um *CLP*, deve-se atentar que o número de canais são definidos pelo fabricante do dispositivo e a descrição da *TEDS* deverá ser armazenada na memória deste hardware.

Uma escolha de hardware que vem se tornando popular na construção de interfaces *IEEE 1451* são os microcontroladores , , . Estes dispositivos podem ser empregados na construção de módulos embarcados e para realizar funções específicas.

Existem fabricantes que possuem ambiente de desenvolvimento específico para módulos embarcados com microcontroladores, sendo que este ambiente é baseado em interface gráfica para facilitar na prototipagem . Os microcontroladores possuem arquitetura definida pelos fabricantes, tornando sua utilização direta para fins específicos, restringindo sua versatilidade em relação a modificações futuras. Na implementação de um nó *IEEE 1451* utilizando microcontroladores, deve-se gerar protocolo de comunicação com softwares que realizará as funções do (*NCAP*).

Com intuito de oferecer uma solução para implementar automação de processos, baseado nos conceitos do padrão *IEEE 1451*, neste trabalho desenvolveu-se uma plataforma de hardware e software que viabiliza a construção de rede de transdutores inteligentes. No desenvolvimento desta plataforma utilizou-se basicamente a tecnologia Java e placa de *FPGA* com processador embarcado. Estas tecnologias foram utilizadas por satisfazer os requisitos na norma *IEEE 1451*, são pouco exploradas na automação industrial e no desenvolvimento de nó *IEEE 1451*, possuem flexibilidade para gerar hardware e favorecem uma rápida prototipagem. A utilização de *FPGA* com processador embarcado possibilita ao *TIM* realizar o controle de atividades complexas e de suportar diferentes protocolos de comunicações.

A plataforma possui suporte para desenvolver as interfaces *IEEE 1451.2* e *IEEE 1451.5*, além de favorecer a descrição da *TEDS* no módulo para transdutores (*TIM*) e armazenar estas informações em banco de dados no *NCAP*. O *NCAP* foi desenvolvido de forma modular, ou seja, possui pacotes que podem servir como base para diferentes aplicações. Assim, os resultados apresentados neste trabalho servem como parâmetro para inserção de computação reconfigurável e software desenvolvido com ferramenta gratuita na automação industrial, além de fornecer uma arquitetura flexível para implementar a *TEDS*.

1.5 Motivação

O controle das atividades de dispositivos via Internet e aparelhos móveis de maneira simples e segura para o usuário engloba áreas como, domótica, telemedicina, agroindústria, automação industrial, etc. O padrão *IEEE 1451* surge em meio a esta necessidade de integração entre a instrumentação eletrônica, aquisição de dados e rede de comunicação. O *IEEE* e o *NIST* são os precursores e certificadores deste padrão, porém diversos grupos de pesquisa e empresas vêm inserindo em seus planos estratégicos os conceitos da norma *IEEE 1451*, como por exemplo, *NASA*, *IBM*, *Ericsson*, *Siemens*, *Agilent Technologies* e *National Instruments*. O crescimento constante do número de empresas e institutos de pesquisa, participando do desenvolvimento do padrão *IEEE 1451*, representa a busca por um mercado promissor. As empresas buscam soluções que atendam à expansão e aos gargalos da inovação tecnológica. Os institutos de pesquisa almejam fornecer soluções tecnológicas e tornar-se um referencial neste segmento.

Outro fator motivacional para o desenvolvimento deste trabalho está na aplicabilidade de *FPGA* com processador embarcado na automação de processos. Estes dispositivos reconfiguráveis suportam recursos para processamento de imagem, técnicas de controle, algoritmos de inteligência artificial, *FFT (Fast Fourier Transform)*, entre outros. Para implementar hardware composto de tais recursos, deve-se selecionar um ambiente *EDA (Electronic Design Automation)* que possibilite a redução no tempo de projeto .

Apesar de serem considerados de custo elevado quando comparado com microcontroladores e *CLPs* os dispositivos reconfiguráveis possuem características fundamentais que acompanham a evolução da automação industrial. Assim, os resultados de pesquisas podem tornar estes dispositivos uma solução real para o setor da automação e controle, favorecendo a produção em maiores escalas e conseqüentemente redução de custo.

1.6 Trabalhos Correlatos

Centros de pesquisas e empresas de vários países vêm desenvolvendo um sistema de automação e controle utilizando a norma *IEEE 1451*. A busca por um sistema automatizado baseado no padrão *IEEE 1451* favorece à construção de rede de transdutores mais versáteis, com grande aplicabilidade nos diferentes cenários da engenharia, como por exemplo, biomédica, agroindústria, aeronáutica, naval, sistema de tubulação de gás, petrolíferas, linhas de produção, laboratórios remotos, metrologia, etc. Diversos trabalhos foram publicados em importantes congressos e revistas de engenharia.

Em abril de 2001, *Miroslav Sveda* do Departamento de Ciência da Computação e Engenharia e *Radimir Vrba* do Departamento de Microeletrônica, ambos da *Brno University of Technology da República Tcheca*, publicaram um artigo no qual, se descreveu as tecnologias emergentes na implementação de rede de sensores para aplicação em sistemas industriais . Segundo os autores do artigo, a indústria busca a utilização de sistemas padronizados e abertos, como por exemplo, Java, *Ethernet*, *TCP/IP* e o padrão *IEEE 1451*. No artigo, descreveu-se as diferentes arquiteturas dos barramentos de campo, ponto-a-ponto e hierárquico, os equipamentos para conectar transdutores com diferentes redes de comunicação (*roteador* e pontes) e os conceitos para implementar o padrão *IEEE 1451*.

O estudo de caso analisado no artigo refere-se a um conjunto de sensores de pressão, conectado na rede *Ethernet*, através de microcontroladores (*ARM* ou *Rabbit*), sendo que na saída dos sensores utilizou-se conversores *A/D* (*ADuC18*), desta forma, cada sensor foi conectado a um microcomputador. Para desenvolver as funcionalidades do bloco lógico do *NCAP* para Internet, utilizou-se o pacote Java Applet e um *gateway* para conectar na rede *Ethernet*.

Este *gateway* possibilita aos sensores serem conectados em diferentes redes de transdutores, como por exemplo, *DeviceNet*, *LonWorks* e *CANopen* e *Fieldbus Foundation*. Apesar de utilizar o padrão *IEEE 1451.2* no projeto, os autores não descrevem o protocolo de comunicação entre o *NCAP* e o *STIM*, sendo essencial neste aspecto o *plug and play* exigido neste padrão. Um aspecto interessante é a possibilidade de comunicação do *NCAP* com diferentes barramentos de campo.

Em julho de 2002, pesquisadores da *Universidade Politécnica de Madrid* apresentaram um projeto para construir um *System-on-Chip (SoC)* baseado no padrão *IEEE 1451*, para ser aplicado no interfaceamento de sistemas *DMC (Distributed Measurement Control)*. No projeto, empregou-se a linguagem de descrição de hardware *Very High Speed Integrated Circuit (VHDL)* na construção de um módulo que implementasse ambas as funções: de *STIM* e de *NCAP*. Este módulo foi sintetizado em *Field Programmable Gate Array (FPGA)* e conectado na porta serial do *Personal Computer (PC)*.

Em novembro de 2002, *Wobschall* publicou um artigo descrevendo a construção de um *NCAP* utilizando duas placas *EX01* e *EM04*. A placa *EX01* contém um microcontrolador (*PIC16F877*) que permite a comunicação *TCP/IP*. Essa placa foi conectada em um computador pessoal (*PC*) pela porta paralela (*DB-25*). A placa *EM04* controla a *EX01*, realizando a comunicação através de um *MUX* que seleciona serialmente uma interface *WebSensor*. A interface *WebSensor* foi projetada com base no padrão *IEEE1451*. Com esta interface foi possível a comunicação na Internet por meio da *Ethernet*.

Em maio de 2003, pesquisadores do *Instituto de Telecomunicações e da Escola Superior de Tecnologia*, ambos de Portugal, utilizaram um *PC* e o *LabView 6.1* para implementar o *NCAP*. A parte física do *NCAP* foi construída em um *PC* e devidamente conectada na rede *Ethernet* e a parte lógica do *NCAP* foi desenvolvida em *LabView 6.1*. Uma biblioteca do tipo *dynamic-link library (dll)*, desenvolvida através da linguagem de programação *C++* executa as funções necessárias para controlar o dispositivo

conectado na porta paralela do *PC*. Este artigo é um grande referencial, pois emprega ferramentas de grande utilização nos laboratórios de engenharia. Porém, há uma tendência do comitê *IEEE 1451.2* em homologar a interface *RS232* para realizar o *plug and play* entre *NCAP* e *STIM*.

Em 2004, *Darold Wobschall* e *Wai Sing Poh*, do Departamento de Engenharia Elétrica da *Universidade de Búfalo em Nova York*, publicaram um artigo focando as atividades do *NCAP* e do *STIM*. A comunicação entre o *STIM* e o *NCAP* foi realizada por meio da interface *RS232* e tanto o *STIM* como o *NCAP* são módulos embarcados. O *NCAP* foi implementado em uma placa *EM04a* e o *STIM* em um microcontrolador *PIC 16F877* da *Microchip*. A aplicabilidade deste sistema foi o monitoramento da temperatura via Internet. Desta forma, um sensor do tipo *RTD (Resistance Temperature Detector)* foi devidamente conectado no *STIM* e a placa *EM04* foi conectada ao *PC* pela rede *Ethernet*. A placa *EM04a* converte dados seriais em formatos *TCP/IP*. Neste trabalho não foi implementado a interface para transdutor inteligente *TII*, porém, a utilização de protocolo *RS232*, favorece a capacidade de *plug and play* na comunicação *NCAP* e *STIM*. A arquitetura descrita neste artigo para implementar um sistema para transdutores inteligentes com base no padrão *IEEE 1451* apresenta as seguintes limitações: o *NCAP* monitora um único sensor e fica restrito apenas ao monitoramento.

Em 2005 *Darold Wobschall* da *Esensors, Inc.* e *Deepak Lakshmanan* do Departamento de Engenharia Elétrica da *Universidade de Búfalo no Estado de Nova York*, publicaram um artigo que descreve o desenvolvimento de um sensor capacitivo para medir a umidade do solo cuja transmissão de dados é realizada através da comunicação sem fio. Segundo os autores, o formato dos dados está em conformidade com o padrão *IEEE 1451*. O sensor de umidade pode ser utilizado na monitoração da estabilidade do solo, atividade de construção civil, agricultura de precisão, etc. Para realizar a comunicação do sensor com a unidade de monitoramento (*PC*) os autores utilizaram basicamente um transmissor de 433 *MHz* da *Microchip (RF-PIC)*, com taxa de transmissão de 9600 *bit/seg*, um receptor composto de um transceptor (*CC1000* da *Chipcom Corporation*) e um microcontrolador (*PIC16F873*).

O microcontrolador do receptor exerce a função de *gateway*, convertendo os dados em formato *RS232* para serem disponibilizados em um display. A transmissão de dados tem o seguinte formato: “*PPPPSSIIAADDQ*”, sendo *P* o preâmbulo, *S* a sincronização, *I* a identificação, *A* o endereço, *D* o dado e *Q* o tamanho do dado. O alcance da transmissão de dados foi de 100 metros. Segundo os autores, a transmissão

de dados está em conformidade com a norma *IEEE 1451.5* e a decodificação destes dados está em conformidade com a proposta *IEEE 1451.4*. A aplicabilidade do sistema de comunicação descrito neste artigo restringe ao monitoramento de processos, porém a interface *IEEE 1451.5*, deve permitir também o acionamento de atuadores.

Uma alternativa para melhor atender a norma *IEEE 1451.5* seria desenvolver este sensor utilizando a tecnologia *IEEE 802.15.4*, como por exemplo, os módulos *Xbee* da *MaxStream*. Esses módulos possuem capacidade de transmissão/recepção e operaram com dados no formato *RS232*, ou seja, evitar-se-ia de utilizar um transceptor e o *gateway*, acoplando o módulo *Xbee* diretamente no *PC*. No artigo não está descrito em quais condições foram testadas a capacidade de transmissão do sistema, sendo estas, fundamentais para avaliar uma interface *IEEE 1451.5*.

Em 2006 *Darold Wobschall* da *Esensors Inc.* publicou um trabalho nomeado de “*A wireless Gas Monitor with IEEE 1451 Protocol*”. Este trabalho foi publicado no *Sensor Applications Symposium (SAS-2006)* em *Houston*, no estado do *Texas, EUA*. Destacam-se neste trabalho as aplicações e justificativas para monitoramento de gás no ambiente e os tipos de sensores utilizados para tais propósitos. *Darold Wobschall* (membro do comitê *IEEE 1451.2*) descreve como o sensor de gás é identificado e calibrado automaticamente, com base nas informações da *TEDS*. Neste sistema a *TEDS* e o sensor compõem um único módulo, sendo que o elemento sensor tem o sinal amplificado antes de passar por um conversor *A/D* (10 bit) e devidamente acoplado ao microcontrolador. Este microcontrolador controla o ganho do amplificador, a excitação do sensor, além de receber as informações da *TEDS* e fornecer a quantidade de gás de forma digital. A excitação do sensor pode ser realizada de duas formas: através de uma tensão 1 a 5 Volts, acima de 200 mA e/ou com uma tensão de referência de 0.1 até 2.5 Volts. O ajuste da tensão ocorre por meio de um microcontrolador conectado na porta serial de um *PC*. O microcontrolador aciona um potenciômetro digital, conforme solicitado pelo elemento sensor. Na Figura 1.8 pode-se observar o diagrama de bloco do módulo sensor e o circuito de condicionamento de sinal proposto por *Darold*.

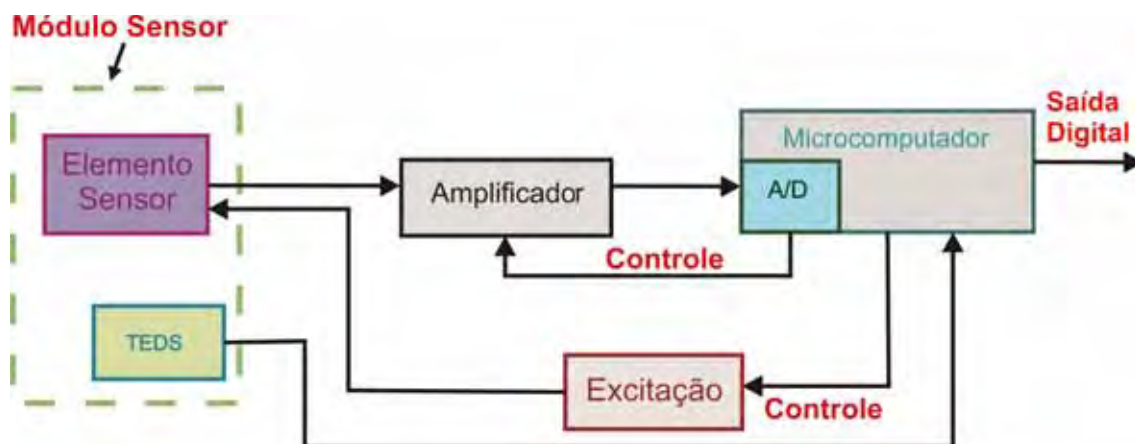


Figura 1.8: Blocos de um módulo sensor com o circuito de condicionamento.

O formato da *TEDS* tem as seguintes especificações:

- *TEDS* informações básicas (8 bytes)
 - Identificação da Fabricação do dispositivo (4 bits).
 - Modelo do dispositivo (15 bits).
 - Versão do dispositivo (6 bits).
 - Serial do dispositivo (6 bits).
- Construção da *TEDS*
 - Tipo de Sensor e calibração (16 bytes).

Neste artigo são apresentadas duas diferentes interfaces sem fio: a simples utiliza a configuração ponto-ponto e é implementada por um transceptor *CC1000* que pode operar a até 433 MHz. Neste tipo de comunicação, a faixa de atuação é de 30 – 100 metros; em malha, utiliza o transceptor *ZigBee*, com capacidade de operar em 2.4 GHz. Assim, o equipamento completo (*Monitor*) desenvolvido por *Darold* pode enviar informações de um *Monitor* para outro. Um *Monitor* contendo um *gateway* para Internet é conectado em um *PC*, através da *Ethernet* e, desta forma, os dados (leitura dos gases, conforme o sensor) são disponibilizados pela Internet.

Um dos resultados apresentado no artigo é a implementação do *plug and play* do sensor de gás desenvolvido. O autor afirma que a decodificação das informações da *TEDS* está baseada no *IEEE 1451.4*. Ou seja, o *plug and play* está fundamentalmente associado à capacidade de um sensor ser acoplado em diferentes redes de comunicação sem mudanças físicas e lógicas e ser reconhecido automaticamente nesta rede. O

reconhecimento é baseado nos dados eletrônicos contidos na *TEDS* e estes dados eletrônicos devem ser acessados pelo *NCAP*. O autor não descreve como se implementa o *NCAP* para acessar os dados e não apresenta uma proposta para o protocolo de comunicação sem fio. O módulo embarcado implementado neste artigo atende apenas à parte de monitoramento do padrão *IEEE 1451.5*, tornando o sistema com pouca aplicabilidade na automação de processos industriais.

Em 2006, foi publicado na *International Conference on Mechatronics e Automation* realizado na China, um artigo sobre um controlador digital implementado em *FPGA* com o processador *NIOS II*, com a finalidade de controlar um sistema de servo mecanismo (braço robótico). O esquema proposto realiza não só a aquisição de dados, mas também o controle de posicionamento das junções, controle de torque e controle da trajetória, além de controlar através de *FPGA* as atividades dos motores. O sistema de servo mecanismo é um braço robótico denominado de *HIT/DLR*, em homenagem às instituições que pertencem os autores do projeto, *Harbin Institute of Technology* e *Institute of Robotics and Mechatronics*. Segundo os autores, as vantagens de utilizar *FPGA* em um sistema de controle para servo mecanismo são a alta capacidade de processamento para controlar o motor e a facilidade em implementar algoritmos de controle.

O braço foi projetado de forma modular e possui um circuito eletrônico embarcado para cada junção, vários tipos de sensores (sensor de torque, sensor de posicionamento das juntas, sensor de posicionamento dos motores, sensor *hall* digital e sensor de temperatura), motores e engrenagens. O *FPGA* com *NIOS II* foi responsável pela aquisição de dados, por implementar a lógica dos comandos e por fornecer interface para a comunicação serial e cartão de controle. O *FPGA* controla os parâmetros (torque, temperatura e posição) através da interface *SPI (Serial Peripheral Interface)* e de um conversor *A/D*. O sistema de modulação utilizado foi *SPWM (Sinusoidal Pulse Width Modulation)* e foi implementado em *VHDL*, assim como o controlador do tipo *PI (Proporcional Integrativo)*, ambos foram sintetizados em *FPGA*. No artigo pode-se observar a potencialidade dos recursos da utilização do processador *NIOS II*, em implementar hardware com diversas funcionalidades, porém a comunicação deste braço robótico com *PC* foi realizada através de um *Digital Signal Processor (DSP)*, dotado de um cartão *Dual Port RAM (DPRAM)*. Uma alternativa para redução de custo deste projeto e até mesmo aumentar a velocidade da comunicação

entre o servo mecanismo e o *PC* é utilizar interface *USB*, suportada pelos *FPGA* da família *Cyclone*.

Em 2006 pesquisadores do *Departamento de Engenharia Elétrica e da Computação da Universidade de Rhode na Irlanda*, publicaram um artigo que descreve a análise da capacidade de transmissão dos dispositivos *ZigBee*. O método utilizado foi o Erro Aproximado da Transmissão, (*Transmission Error Approximation - TEA*), no qual, analisam-se os resultados estatísticos da taxa de desempenho na transmissão de dados. A estimação da distância de dispositivos sem fio pode ser baseada nos seguintes métodos: Tempo de Chegada, (*Time of Arrival ToA*), Ângulo de Chegada, (*Angle of Arrival - AoA*) e Indicação da Força de um Sinal Recebido, *Received Signal Strength Indication (RSSI)*.

No método *ToA*, a distância é calculada pelo tempo de propagação das ondas entre o emissor e o receptor. Esta categoria fornece alta precisão e necessita de um equipamento com alta capacidade de processamento para realizar as medidas. No *ToA*, as medidas podem ser combinadas com as medidas acústicas para obter maior precisão, porém, os sinais acústicos dependem da temperatura, requer a linha de visada além da inserção de dispositivos especiais. No método *AoA* deve-se utilizar o posicionamento da antena para medir o ângulo de chegada de sinal, e desta forma, encontrar a distância entre os dois dispositivos.

Segundo os autores, a técnica *AoA* é de alto custo, devido aos hardwares utilizados para implementar este método, desta forma, esta aplicação só é utilizada quando o custo é irrelevante. No método *RSSI* é medido a potência que o sinal chega ao rádio. A potência dos sinais de rádio cai exponencialmente em relação à distância. A categoria utilizada para analisar *TEA* no artigo foi a *RSSI* por apresentar menor custo para análise. Nesta categoria a distância é calculada com base na (1.1).

$$P_r = \frac{P_t}{d^\alpha} \quad (1.1)$$

sendo P_t potência do transmissor, d a distância entre o transmissor e receptor, α constante entre 2 e 4, dependendo do canal utilizado e P_r a potência do receptor. Quando o transmissor emite um sinal inicial com potência fixa, o receptor obtém uma atenuação definida como P_r/P_t . Embora o método *RSSI* seja simples para implementar, não tem precisão por três motivos básicos:

Primeiro pelo fato de P_r variar rapidamente, devido aos canais que a comunicação sem fio deve realizar.

Segundo pelo fato de P_r ser afetado por ruídos e interferências ou quando a potência do receptor não corresponder a distância entre os dispositivos.

Terceiro motivo consiste em um pequeno erro na P_r com possibilidade de resultar em um grande erro no cálculo da distância, especialmente quando essa distância for longa .

Os autores utilizaram a placa *MCF5208* da *FreeScale* dotada de um transceptor *ZigBee* modelo *MC13192*. Os resultados são baseados em ambientes com linha de visada, porém é significativo saber a capacidade da transmissão sem fio, em ambientes sem linha de visada e em ambientes com alto grau de ruído, além de comprovar a eficiência do método para *TEA* em diferentes aplicações.

Em 2007, na Conferência Internacional em Sistemas, *International Conference on Systems (ICON'07)*, foi publicado um artigo de *Miroslav Sveda e Roman Trchalik* ambos da Faculdade de Tecnologia de Informação da República Tcheca, intitulado de *ZigBee-to-Internet Interconnection Architectures*. Este artigo tem foco principal em uma proposta de arquitetura de software para realizar a comunicação entre Internet e sensores dotados de tecnologia *ZigBee*, com base no padrão *IEEE 1451*. Os autores descrevem dois componentes fundamentais para atuar na interface entre a Internet e o sistema *ZigBee*, o *ZigBee gateway* e *ZigBee bridge* ou *router*.

O *gateway ZigBee* é capaz de converter do protocolo *wireless* para diferentes protocolos de cabeamento, como por exemplo, *LonWorks* e *ModBus* para redes industriais e *HTML* e *XML* para aplicação na Internet. O *gateway* pode fornecer uma interface entre o protocolo *ZigBee* e o *IP* da máquina, traduzindo os endereços e comandos do *ZigBee* e do *IP*. A ponte ou *bridge ZigBee* é mais simples que o *gateway* e deve ligar a rede *ZigBee* sobre a rede baseada em *IP*.

A ponte *ZigBee* pode trabalhar com protocolo *Ethernet* e *WiFi*, realizando a conectividade *IP* transparente para os dispositivos *ZigBee*. Este artigo descreve o padrão *IEEE 1451*, especificamente sobre o padrão *IEEE 1451.1* e *IEEE 1451.5*. Foram apresentadas soluções para implementar uma rede de transdutores utilizando a tecnologia *ZigBee* . No artigo não se descreve como os sensores/atuadores poderiam ser conectados em um módulo *ZigBee*, nem como o software gerencia a comunicação com

os componentes (*gateway e ponte*) e qual a eficiência (taxa de transmissão, alcance, imunidade a ruído) destes componentes. Ou seja, neste artigo apresenta-se a descrição de alguns componentes que podem ser utilizados para desenvolver um sistema de comunicação sem fio baseado no padrão *IEEE 1451.5*.

1.7 Organização do Trabalho

Os próximos capítulos deste trabalho estão organizados com os seguintes assuntos: no Capítulo 2 descreve-se as principais tecnologias para desenvolver interface com base no padrão *IEEE 1451.5*, conforme a homologação do comitê responsável por esta norma, sendo que as configurações dos módulos sem fio são descritas no Apêndice A. No Capítulo 3, apresentam-se as etapas de desenvolvimento, as características dos pacotes e classes, relação de dependência e o funcionamento do software *NCAP*.

No Capítulo 4 mostra-se o desenvolvimento do hardware básico para exercer as funcionalidades do *TIM* proposto neste trabalho, do *template* e da *TEDS*. No Capítulo 5 são apresentados os testes realizados com a plataforma *IEEE 1451* em diferentes aplicações, além de um comparativo entre o desempenho das interfaces *IEEE 1451.2* e *IEEE 1451.5*. No Apêndice B apresentam-se etapas da documentação do software *NCAP*. No Apêndice C, encontra-se as etapas para desenvolver um hardware com processador NIOS II embarcado. No Apêndice D pode-se observar as configurações utilizadas para desenvolver e testar a plataforma *IEEE 1451*. No Apêndice E demonstra-se parte da *firmware* do processador *NIOS II*. No Anexo A, apresenta-se uma breve descrição da automação de um kit de Pneumática utilizando o software *Eclipse E3* com o módulo embarcado desenvolvido neste trabalho.

CAPÍTULO 2 - Tecnologias IEEE 1451.5

2.1 Introdução

O comitê *IEEE 1451.5* apresentou, desde o início das discussões as tecnologias que poderiam ser utilizadas para implementar a interface *1451.5*. Essas tecnologias deveriam atender os seguintes quesitos: padronizada, aberta e de domínio público. O padrão *IEEE 1451.5* foi homologado em março de 2007, no qual foi estabelecido para desenvolver a interface sem fio, o sistema de comunicação *Wi-Fi*, *Bluetooth* e *ZigBee*. Desta forma, neste capítulo descreve-se estas tecnologias.

2.2 IEEE 802.11 – Wi-Fi

Por volta de 1990 o *Institute of Electrical and Electronics Engineers (IEEE)* definiu um grupo que ficou responsável por alavancar o desenvolvimento de rede de computadores utilizando a tecnologia sem fio. Este projeto foi denominado de *IEEE 802.11*. Porém, devido à baixa taxa de transferência alcançada que era em torno de dezenas de *kbps* o projeto ficou inerte por aproximadamente sete anos. Com o desenvolvimento da tecnologia, a rede sem fio começou a ser vista como uma solução promissora e a receber mais investimento para a construção de equipamentos.

O *IEEE 802.11* refere-se a uma família de especificações sobre tecnologias sem fio para uma rede local. A família *802.11* é subdividida em :

- *IEEE 802.11*: aplicada às redes locais sem fio *Local Area Networks (WLANs)*, com taxas de 1 ou 2 *Mbps*, frequência de 2,4 *GHz*, usando o método de transmissão *Frequency Hopping Spread Spectrum (FHSS)* ou *Direct Sequence Spread Spectrum (DSSS)*;
- *IEEE 802.11a*: aplicada às redes locais sem fio que prevê até 54 *Mbps* em 5 *GHz*. Utiliza o esquema *Orthogonal Frequency Division Multiplexing (OFDM)*. É incompatível com o *802.11b* e deve ser substituído pelo *802.11g*;
- *IEEE 802.11b*: aplicada às redes locais sem fio com taxas de transmissão de até 11 *Mbps*, operando a 2,4 *GHz* e utiliza somente o esquema de codificação *DSSS*;

- *IEEE 802.11 g*: aplicada às redes locais sem fio que prevê taxas de 20 *Mbps*, operando na frequência de 2,5 *GHz*.

O padrão *IEEE 802.11* define basicamente uma arquitetura para as *WLANs* que abrange os níveis físicos e de enlace. No nível físico são tratadas apenas as transmissões com frequência de rádio (*RF*) e infravermelho (*IR*), embora outras formas, como o *laser* possam ser usadas.

Atualmente, a maioria das redes sem fio é baseada no padrão *802.11* para comunicação entre dispositivos e uma *LAN* (rede local). O *IEEE 802.11* define uma arquitetura para as redes sem fio, baseada na divisão da área coberta pela rede em células. Essas células são denominadas de *Basic Service Area (BSA)*, cujo tamanho depende das características do ambiente e da potência dos transmissores/receptores usados nas estações. Outros elementos fazem parte do conceito da arquitetura de rede sem fio:

- *Basic Service Set (BSS)*: representa um grupo de estações que se comunica por rádio-fusão ou infravermelho em uma *BSA*;
- *Access Point (AP)*: são estações especiais responsáveis pela captura das transmissões realizadas pelas estações de sua *BSA*, destinada a estações localizadas em outras *BSAs*, retransmitindo-as, usando um sistema de distribuição;
- *Sistema de Distribuição*: representa uma infra-estrutura de comunicação que interliga múltiplas *BSAs* para permitir a construção de redes cobrindo áreas maiores que uma célula;
- *Extend Service Area (ESA)*: representa a interligação de vários *BSAs* pelo sistema de distribuição através dos *APs*;
- *Extend Service Set (ESS)*: representa um conjunto de estações formado pela união de vários *BSSs* conectados por um sistema de distribuição.

Na Figura 2.9, apresenta-se os principais elementos de uma configuração para *BSSs*.

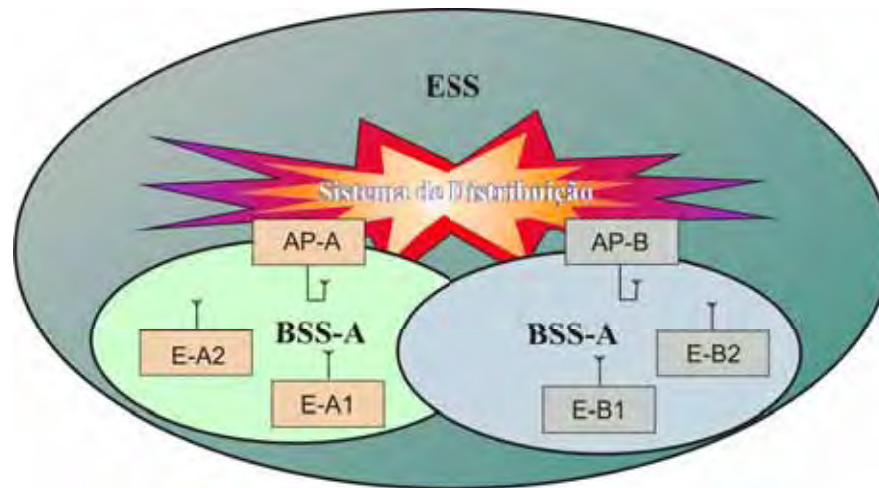


Figura 2.9: BSSs conectados por um sistema de distribuição.

A construção de uma rede abrangendo áreas maiores do que um ambiente local, o *IEEE 802.11* limita-se ao padrão das redes locais, podendo atuar nos modelos de comunicação Cliente/Servidor ou *AdHoc*. No modelo Cliente/Servidor, várias células fazem parte da arquitetura e as estações se comunicam com outras células através de pontos de acesso, usando um sistema de distribuição. No sistema *AdHoc* as comunicações são estabelecidas entre várias estações de uma mesma área (célula) sem o uso de um servidor. Na Figura 2.10, mostra-se um exemplo de uma rede com infraestrutura e uma rede local sem fio *AdHoc*.

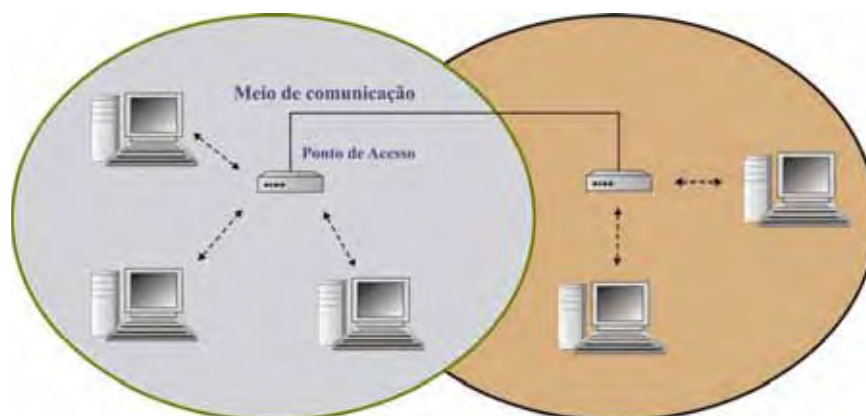


Figura 2.10: Infra-estrutura de rede *AdHoc*.

Um elemento fundamental na arquitetura de rede local sem fio com infraestrutura é o ponto de acesso que desempenha as seguintes funções:

- Autenticação e associação: permite que uma rede de estação móvel mesmo saindo de sua célula de origem continue conectada a infraestrutura e não perca a comunicação;
- *Handoff*: a função que permite manter a continuidade da comunicação quando um usuário passa de uma célula para outra;
- Gerenciamento de potência: permite que as estações operem economizando energia através de um modo chamado de *power save*;
- Sincronização: garante que as estações associadas a um ponto de acesso estejam sincronizadas por um relógio comum.

2.2.1 Protocolo MAC

O *IEEE* definiu um protocolo de acesso ao meio denominado de *Distributed Foundation Wireless Medium Access Control (DFWMAC)* que suporta dois métodos de acesso: um sistema distribuído, que é obrigatório e um centralizado, que é opcional, podendo esses dois coexistir. Além de tratar de problemas relacionados com estações que se deslocam para outras células (*roaming*) e com estações perdidas (*hidden node*).

Diferente do método tradicional em rede *Ethernet*, que não garante a entrega correta dos dados, o *DFWMAC* acrescenta ao *CSMA/CA* um mecanismo opcional que envolve a troca de quadro de controle antes da transmissão dos dados, melhorando efetivamente a transmissão .

2.2.2 Roaming

O *roaming* é uma importante característica de comunicação sem fio, pois permite que estações mudem de célula e continuem enviando e recebendo informações.

A transferência (*handoff*) entre pontos é totalmente transparente para o usuário. Quando um usuário caminha com um laptop, dispositivo inteligente ou *PDA*, todos têm que se mover de uma célula para outra, executando a função de *roaming* que funciona da seguinte forma :

- Uma estação móvel, ao entrar em nova célula e não estando em conversação registra-se automaticamente pelo *AP*, que controla a célula destino;
- Na célula visitada, o *AP* desta irá verificar se a estação móvel visitante não havia se registrado anteriormente. Caso esse procedimento não tenha se efetuado, o referido *AP* irá informar ao *AP* da célula origem sobre a nova posição;
- Com isso, o *AP* da célula origem fica sabendo da nova posição da estação móvel e envia uma informação como se esta estação estivesse em sua própria célula.

Um dos grandes problemas da rede sem fio ocorre quando uma estação fica incomunicável por um período de tempo com o *AP*. Seu desligamento, a saída da área de atuação ou locais com grande degradação de sinal são alguns dos motivos que ocasionam estes problemas e .

Ao tentar comunicar-se com a estação móvel, inúmeras vezes sem obter resposta, o protocolo *MAC* do *AP* envia um sinal de *request* para todas as outras dentro da área de cobertura. Cada uma destas envia um *request communication*.

A estação perdida recebe este sinal e envia um *response request* para todos avisando que está ativa. As que receberam a *response request* da estação perdida enviam um *bridge request* diretamente para o *AP*, podendo assim encontrar o melhor caminho, seja via “ponte” ou diretamente para estação perdida. Assim, quando a estação receber a comunicação, não há mais a necessidade da ponte. As pontes são equipamentos utilizados para auxiliar a conexão entre dispositivos com o mesmo protocolo de comunicação.

Se o *AP* perder a comunicação com a ponte ou vice-versa, o *AP* escolhe outra entre as estações que responderam inicialmente. Com este método, o *AP* tem a chance de recuperar uma estação que por algum motivo tornou-se incomunicável com a rede.

2.2.3 Técnicas de Codificação

O padrão *IEEE 802.11* implementa duas técnicas de codificação: a *DSSS (Direct Sequence Spread Spectrum)* e *FHSS (Frequency Hopping Spread Spectrum)*. A escolha dependerá de vários fatores relacionados com a aplicação dos usuários e o ambiente onde a rede operará. Ambos transmitem quadro de dados enviando-os por vários canais disponíveis dentro de uma frequência e, não somente para um único canal, possibilitando, desta forma, a transmissão simultânea de vários quadros. A técnica *DSSS* distribui o sinal em cima de uma gama extensiva de frequência e reorganiza os pacotes no receptor. Já a *FHSS* envia segmentos curtos de dados que são transmitidos pelas frequências específicas, controlando o fluxo com o receptor que negocia velocidades menores comparadas às oferecidas pela *DSSS*, mas menos suscetíveis a interferências.

O transmissor *802.11* escolhe uma frequência onde o nível de interferência não seja crítico e se mantém nela durante o período de operação, a menos que o ruído atinja um ponto proibido. Neste caso, procurará outra disponível entre 11 canais. Os transmissores podem utilizar qualquer uma das faixas em busca da banda mais limpa, garantindo flexibilidade na comunicação, apesar de que na banda de trabalho das redes *802.11*, pelo menos quatro sistemas importantes dividem a utilização do espectro: *Bluetooth*, alguns modelos de telefone sem fio, forno de microondas e outros pontos de acesso *802.11* que estão próximos.

2.2.4 Segurança

A segurança na rede de comunicação sem fio é um fator altamente discutido pelos fabricantes e usuários desta tecnologia. Para garantir a segurança, existem várias ferramentas, sendo as mais populares o *ESSID (Extended Service Set ID)* um código que identifica os computadores e *APs* que fazem parte da rede, algumas alterações em parâmetros padrão do ponto de acesso e o *WEP (Wired-Equivalent Privacy)*. Uma ferramenta de criptografia equivalente à das redes cabeadas, que se encarrega de decodificar os dados transmitidos por meio da rede .

2.2.5 Configuração de dispositivo Wi-Fi utilizado na plataforma IEEE 1451

Neste trabalho utilizou-se os módulos *INT550WI* da *Digi International* para testar a funcionalidade da comunicação *Wi-Fi*, como interface *IEEE 1451.5*. Estes módulos permitem conectar dispositivos seriais na rede *802.11b*. As especificações dos dispositivos *INT550WI* são apresentadas na Tabela 2.1 . No Anexo A apresenta-se as etapas de configuração do módulo *INT 550*.

Tabela 2.1: Especificações dos módulos 802.11b

Potência de Transmissão	Sensibilidade de Recepção	Frequência
16 dBm	- 80 dBm	2,4 GHz

2.3 IEEE 802.15.1 – Bluetooth

A tecnologia Bluetooth recebe este nome em homenagem ao rei medieval dinamarquês *Harald Blaatand II*, que foi cognominado de *Bluetooth* por possuir um dente azul. Esse rei tornou-se notável por sua luta para unificar Dinamarca e Suécia, da mesma forma que os engenheiros modernos passaram a lutar pela padronização internacional do *Bluetooth* .

O *Bluetooth* é uma tecnologia destinada a servir como padrão universal de conexão entre equipamentos (microcontrolador, celular e impressora) ou entre *PC* e seus periféricos por meio de uma faixa limitada de frequência. Os periféricos interconectados podem ser desde terminais móveis como impressora, *mouse*, *webcam*, bem como qualquer tipo de equipamento eletrônico como máquina de café, relógio eletrônico, brinquedo de criança, entre outros .

Nos meados dos anos 90 a *Sony Ericsson AB*, localizada em *Lund*, Suécia, iniciou um estudo para viabilizar módulos de baixo custo que permitissem comunicação sem fio entre telefones móveis, computadores pessoais, periféricos, etc. Este projeto teve posteriormente a adesão de outras grandes empresas, como por exemplo, a *IBM* e a *Intel*, resultando na padronização internacional do produto desenvolvido. A proposta de produzir transmissores de baixo custo, suficientemente pequenos para serem incluídos em praticamente qualquer tipo de equipamento e o fato de ser padronizado pelo *IEEE (IEEE 802.15.1)*, despertou o interesse de muitos fabricantes e atualmente o consórcio possui cerca de 2164 empresas e entidades.

Os transmissores *Bluetooth* consomem pouca energia, ocupa uma área menor que 1 cm² e a taxa de operação do *Bluetooth* pode ser de 720 Kbps até 1 Mbps .

2.3.1 Descrição Técnica

O sistema de comunicação *Bluetooth* é composto por um rádio, um controlador digital, um gerenciador de enlace, uma interface de controle e uma biblioteca de programas de aplicação. O rádio opera na faixa *Industrial, Scientific, Medical (ISM)* de 2,4 GHz com uma potência que varia de 1 W a 100 W e a transmissão de dados é feita através de pacotes em espalhamentos espectral de 79 canais. Os radiotransceptores saltam de um canal para o outro de um modo pseudo-randômico, determinado por um terminal mestre. Na camada de banda base, o controlador digital é responsável por construir e decodificar os pacotes de dados, além de tratar a correção de erro e cuidar da criptografia .

O gerenciador de enlace é responsável por criar os *links* entre as conexões e verificar a eficiência das comunicações entre os dispositivos *Bluetooth*, através da codificação e decodificação dos dados. No protocolo de comunicação *Bluetooth* o *Logical Link Control and Adaption layer Protocol (L2CAP)* proporciona a comunicação entre as camadas de enlace de dados e às camadas de rede e controla o fluxo de dados entre os canais. Comparando com o modelo *Open System Interconnection (OSI)*, o gerenciador de enlace e o *L2CAP* estão na segunda camada . Na Figura 2.11, pode-se observar a arquitetura do protocolo *Bluetooth*.

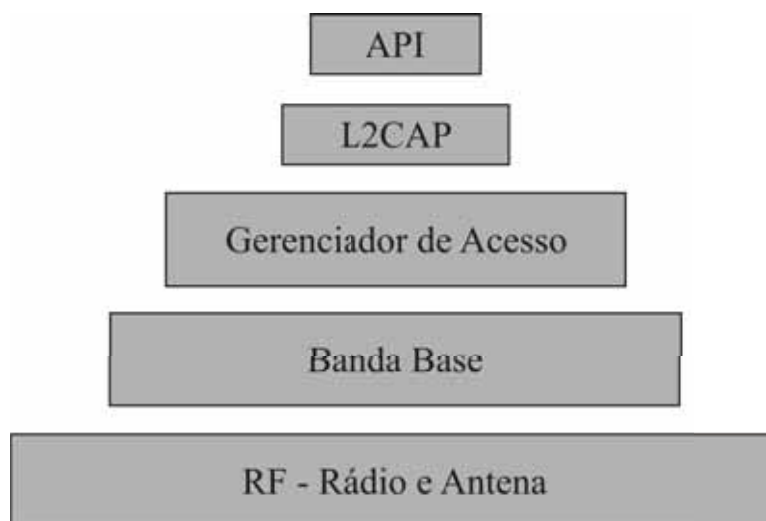


Figura 2.11: Protocolo *Bluetooth*.

Os dispositivos *Bluetooth* têm capacidade de localizar dispositivos próximos, formando as chamadas *piconets*. Uma vez estabelecida a rede, eles determinam um padrão de transmissão usando os canais possíveis. Isto significa que os pacotes de dados da rede serão enviados em canais diferentes numa ordem que apenas os dispositivos da rede conhecem. No item 2.3.2, apresenta-se uma breve descrição do funcionamento do sistema *Bluetooth*.

2.3.2 Configuração da Comunicação de um sistema *Bluetooth*

A tecnologia *Bluetooth* utiliza uma modulação *Frequency Modulation (FM)* binária, aplicada para minimizar a complexidade de transceptor e os canais são formados pela alocação por tempo (*slotted*), com extensões de abertura nominal de 625μ s. Nos canais são trocadas informações por pacotes, sendo que cada pacote é transmitido em uma variação de frequência diferente e um pacote ocupa normalmente uma única abertura, mas pode ser estendido para cobrir até cinco aberturas .

O conjunto de unidades *Bluetooth*, comunicando em uma determinada região é definido como *piconet*. Em uma *piconet* são suportados até oito dispositivos (um mestre e sete escravos). Duas ou mais *piconets* numa mesma área podem estabelecer conexão entre si denominada “*scatternets*”. Todas as unidades *Bluetooth* de uma *piconet* são sincronizadas através de *slots* de 625μ s e numerados de 0 a 277, conforme o relógio de contagem síncrona do terminal mestre. Na Figura 2.12 apresenta-se um exemplo de *piconet*, sendo que a letra *M* corresponde a um dispositivo mestre e a letra *E* são os dispositivos escravos.

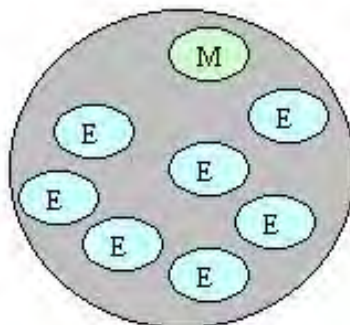


Figura 2.12: *Piconet*.

Na Figura 2.13: mostra-se uma configuração de uma *scatternet*.

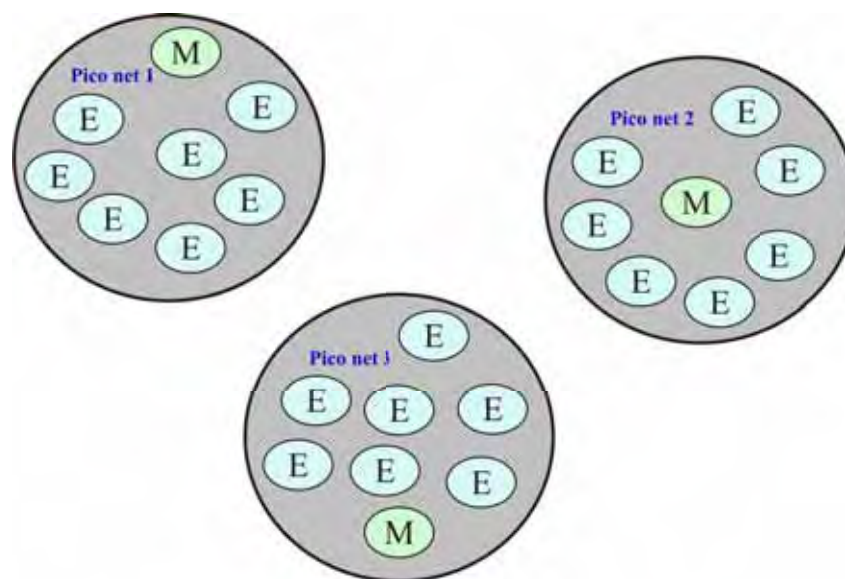


Figura 2.13: *Scatternet*.

Ao estabelecer uma comunicação, todos os dispositivos iniciam em modo de espera (*Standby*), procurando por mensagem periodicamente (*Inquiry*). Uma conexão é iniciada por qualquer um dos dispositivos (*Page*) que vai tornar-se então “mestre”. A sequência de conexão ocorre da seguinte forma: *Standby* → *Inquiry* (endereço desconhecido) → *Page* (endereço conhecido) → conexão.

Na Tabela 2.2 mostram-se as características de enlace da tecnologia *Bluetooth*.

Tabela 2.2: Tipos de enlace *Bluetooth*.

Orientada a conexão síncrona (<i>SCO</i>).	Equivalente a conexão de circuitos.	Comunicação ponto-a-ponto <i>full duplex</i> .	Aplicação de comunicação de voz.	Intervalos de tempo fixo alocados entre dispositivos.
Enlace não conectados-síncrono.	Equivalente a comutação por pacotes, de forma síncrona ou assíncrona.	Conexão temporária com a transmissão de um quadro.	Aplicações de comunicação de dados.	Acesso <i>polling</i> .

Neste trabalho utilizaram-se os módulos BTS1009C da *Sunix* para testar a funcionalidade da comunicação Bluetooth, como interface *IEEE 1451.5*. As especificações dos dispositivos *BTS1009C* são apresentadas na Tabela 2.3 .

Tabela 2.3: Especificações dos módulos Bluetooth.

Potência de Transmissão	Sensibilidade de Recepção	Frequência
16 dBm	- 80 dBm	2,4 GHz

2.4 IEEE 802.15.4 – ZigBee

A origem da tecnologia *ZigBee* ocorreu quando algumas empresas resolveram se reunir para propor uma tecnologia de rede sem fio. Este grupo, liderado pela Philips, estava descontente com o desenvolvimento do Bluetooth e começou a desenvolver uma tecnologia para as redes sem fio, cuja principal característica é o baixo custo e o baixo consumo de energia. Esta tecnologia foi denominada de *ZigBee*, é também conhecida por *PULNet*, *Firefly*, *RF Lite*, *HomeRF* ou *RFEasyLink*. A origem da denominação *ZigBee* está relacionado aos movimentos executados pelas abelhas quando estão trabalhando .

As empresas que participam do fórum *ZigBee Alliance* são a *Mitsubishi*, a *Honeywell*, a *Invensys*, a *Motorola* e a *Philips*. Este fórum está responsável por definir e desenvolver aplicativos, programas de certificação, logotipos, estratégia de marketing e os protocolos acima da camada de Controle de Acesso ao Meio, *Media Access Control (MAC)*. Os meios físicos utilizados pelo sistema *ZigBee* são especificados no padrão *IEEE 802.15.4*. A tecnologia *ZigBee* possui um período que pode ficar inativo, conseqüentemente, os módulos *ZigBee* têm uma duração estimada de funcionamento (contínuo) entre seis meses a dois anos sendo que a tensão média de alimentação é de 9 Volts.

Esta tecnologia possibilita a interligação de até 250 nós por rede. Segundo os pesquisadores da *Honeywell*, a *ZigBee* utiliza pouco processamento, o que permite a sua implementação em um microcontrolador de 8 *bits* com 10 *Kbytes* de memória para o código.

O principal modo de atuação da *ZigBee* é o modelo mestre-escravo, mas pode funcionar no modo *peer-to-peer* ou em modo misto (*mesh mode*). No modelo mestre

escravo sempre há um dispositivo que opera como coordenador, sendo responsável em acionar outros dispositivos em *sleep mode* (modo de espera) os dispositivos podem estar em modo de espera momentos antes de enviar os pacotes. Originalmente o *ZigBee* só operava na banda *Industrial Scientific Medical (ISM)* de 2,4 GHz com tecnologia de espalhamento espectral. Atualmente, também opera em 868/915 MHz .

2.4.1 Modulação da tecnologia ZigBee

A tecnologia *IEEE 802.15.4* foi desenvolvida para suportar apenas comunicação digital e operação *half-duplex*¹, com intuito de desenvolver um modo de espera eficiente na economia de energia. A modulação pode ser *BinaryPhase Shift Keying (BPSK)* para faixa de 868/915 MHz e *Offset Quarterternary Phase Shift Keying (OQPSK)* para 2,4 GHz. O termo *Phase Shift Keying (PSK)*, ou seja, chaveamento em fase refere-se ao sistema de modulação digital, na qual, a fase da onda portadora varia de acordo com o nível lógico do pulso a ser transmitido .

No chaveamento em fase, 2n estados de fase, os mais utilizados são n = 1 para modulação *BPSK* de duas fases e o n = 2 para modulação *QPSK* de quatro fases. O sistema *QPSK* ou quadrifásico suporta o dobro de bits de dados por mudanças de fase da portadora em relação à modulação *BPSK*, realizando as fases de 45°, 135°, 225° e 315°. Neste sistema para os quatro níveis determinam o dobro de estados binários, sendo que cada mudança de fase transmitida chama-se de *dibit* ou duplo dígito binário.

Os dispositivos *IEEE 802.15.4* podem ser classificados de duas formas em relação à execução de serviços, que são os dispositivos de função completa e os dispositivos de função reduzida. Os dispositivos de função completa, *Full Functions Device (FFD)* possuem as configurações completas da camada *MAC*, possibilitando que este atue como coordenador de rede ou como componente da rede. Os dispositivos de função reduzida, *Reduced Function Device (RFD)* possuem funções reduzidas na camada *MAC*, podendo atuar somente como componente de uma rede *IEEE 802.15.4*.

A banda de 868 MHz é disponível para Europa, a banda 915 MHz para América do Norte e 2,4 GHz em praticamente todo mundo. Na banda 869/915 utiliza-se 10 canais e na banda de 2.4 GHz, utiliza-se 16 canais. O padrão *IEEE 802.15.4* possui um controle de erro, realizado pelo protocolo *full handshake* .

¹ *Half-duplex*: a comunicação ocorre nos dois sentidos, porém não pode ser simultaneamente.

Os dispositivos *ZigBee* utilizados neste trabalho são os módulos *XBee* da *MaxTream*. No Anexo A apresenta-se configuração destes módulos *XBee* com base no software da Digi. As especificações dos dispositivos *Xbee* são apresentadas na Tabela 2.4 .

Tabela 2.4: Especificações dos módulos *Xbee*.

Potência de Transmissão	Sensibilidade de Recepção	Frequência
0 dBm	- 92 dBm	2,4 GHz

No próximo capítulo são descritos os procedimentos que implementam a parte lógica do *NCAP*.

CAPÍTULO 3 – Softwares NCAP da Plataforma IEEE 1451

3.1 Introdução

Neste capítulo, descreve-se o desenvolvimento do software, responsável por executar as funções lógicas do *NCAP*. A parte lógica do *NCAP* pode ser comparada com um software supervisorio utilizado nas indústrias, pois neste tipo de aplicação o operador tem detalhes das atividades da rede de transdutores, podendo monitorar e acionar os dispositivos conforme estabelecido no sistema de automação e controle.

O sistema *SCADA* (*Supervisory Control and Data Acquisition*) é fundamental na automação de processos industriais, pois fornece ao operador detalhes das atividades dos transdutores, além de possibilitar a intervenção remota no funcionamento do processo. Para desenvolver um software supervisorio é essencial que o projetista escolha uma ferramenta que possibilite a construção de interfaces gráficas e a comunicação com periféricos. Estes sistemas vão ao encontro das funcionalidades da parte lógica do *NCAP*.

3.2 Software NCAP da Plataforma IEEE 1451 – Primeira Versão

Neste projeto foi desenvolvido e implementado um software supervisorio com capacidade de enviar e receber dados através da porta serial denominado de software para Automação com *FPGA* ou *SAF0*, sendo que zero (0) é uma referência à primeira versão deste software. Assim como os softwares *NCAP* do *NIST* e da *Agilent*, o *SAF0* foi desenvolvido com a tecnologia Java. O desenvolvimento de software supervisorio através da tecnologia Java é atrativo pelos seguintes aspectos:

- É uma plataforma de desenvolvimento gratuita: pode reduzir o custo do produto final (software de automação).
- A tecnologia Java possui flexibilidade para ser executada em diferentes sistemas operacionais: facilita a inserção de dispositivos móveis e sistemas embarcados na automação de processos, além da portabilidade.

- A linguagem de programação é orientada a objeto: facilita a reutilização de código e de polimorfismo, programação em alto nível, além de atender uma das exigências da norma *IEEE 1451.1*.
- Facilidade de interação com acesso remoto: favorece a comunicação com Internet.

O *SAF0* possui interação com o usuário por meio de telas de informações e o ambiente *MS-DOS* da *Microsoft*, proporcionando a comunicação serial, ou seja, o envio e recebimento de dados pela interface *RS232*. As configurações dos parâmetros de comunicação, das atribuições de eventos e da definição do endereço da porta utilizada, são apresentadas na Figura 3.14. A identificação do endereço da porta, utilizou-se principalmente os métodos *getPortType* e *getName* da classe *CommPortIdentifier*, sendo que no *SAF0* definiu-se para a comunicação com o *TIM* a porta *COM1*.

Os parâmetros para transmissão e recepção de dados são definidos no método *setSerialPortParams* da classe *serialPort* sendo que no *SAF0*, utilizou-se 8 bits como tamanho de dados, *StopBit* igual a um e sem bit de paridade, esses parâmetros são definidos com base no protocolo *UART*. Os parâmetros de eventos são configurados pelo método *SerialPortEvent* na recepção de dados e quando utiliza-se a porta serial.

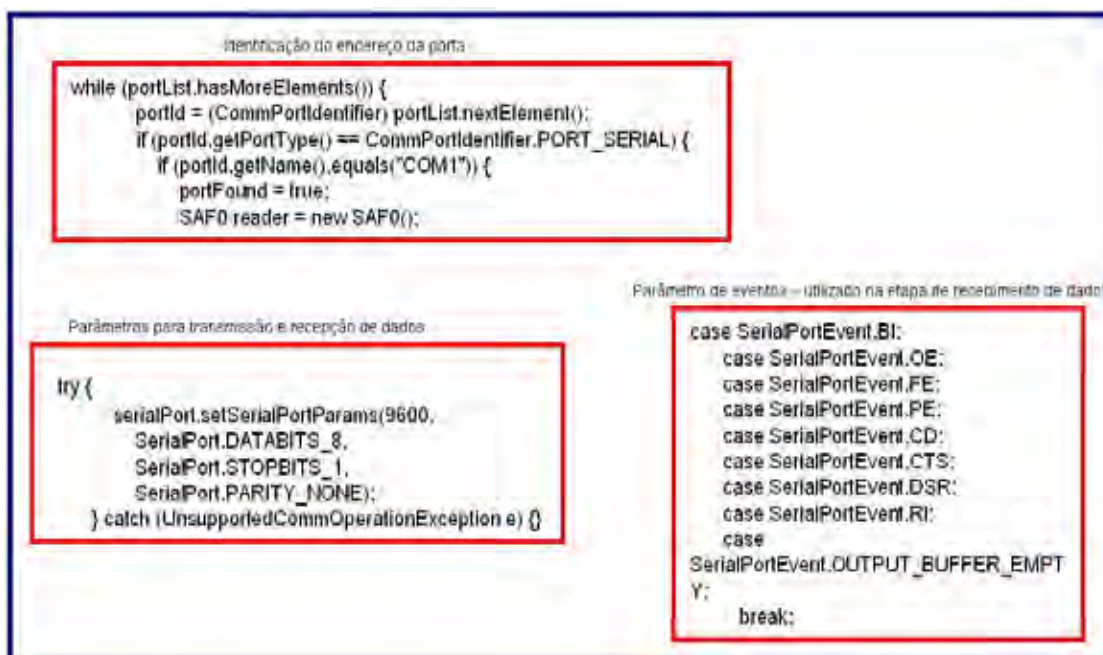


Figura 3.14: Configuração dos parâmetros para comunicação serial.

Nas aplicações em Java para comunicação serial, é essencial utilizar as interfaces *Runnable* e *SerialPortEventListener*, para habilitar os métodos *run* e *serialEvent*. Estes métodos favorecem a etapa de recebimento de dados.

No *SAF0*, a lógica de controle foi desenvolvida para o monitoramento e acionamento de acordo com os transdutores do Módulo de Hidráulica Industrial (*MHI-01*) da *Hidroeleetro*. O kit *MHI-01* possui os seguintes transdutores: 2 sensores indutivos, 2 chaves de fim de curso, 5 solenóides, relés, motor e válvulas. Os sensores indutivos são denominados de *FC3* e *FC4*, as chaves de fim de curso *FC1* e *FC2* e os solenóides *S1*, *S2*, *S3*, *S4* e *S5*. Na Figura 3.15 mostra-se o painel do *MHI-01*, com os respectivos transdutores.



Figura 3.15: Painel do *MHI-01*.

A interface com o usuário gerada pelo *SAF0* de acordo com os transdutores do *MHI-01* é mostrada na Figura 3.16. No *SAF0*, o operador pode monitorar ou acionar individualmente cada transdutor do *MHI-01*. No *menu* de *Opções* (Figura 3.16) tem-se:

- 0 – para finalizar o programa.
- 16 – comando para acionar os solenóides.
- 2 – comando para monitorar os sensores.

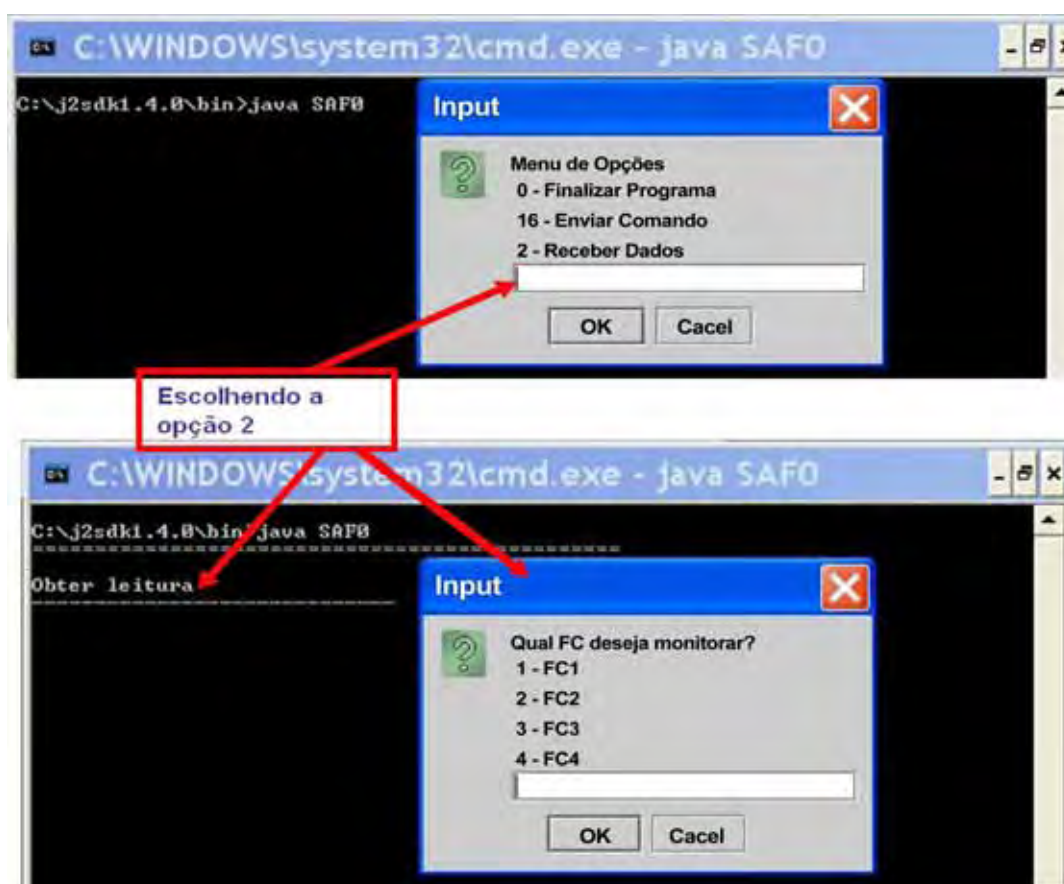


Figura 3.16: Tela Inicial e tela para monitoramento do *SAF0*.

No *SAF0* concentrou-se em viabilizar a comunicação entre o *PC* e a placa *DE2* através da interface *RS 232*, sem preocupar-se com a interface gráfica, armazenamento de informações e hierarquia entre as classes. Na versão *SAF 1.0* utilizou-se o ambiente de programação *NetBeans 6.1*, por ser uma ferramenta com diversos recursos para construir interfaces gráficas e facilitar no desenvolvimento de software , . Entre os recursos do *NetBeans*, está a detecção imediata de erros, sugestões de métodos, geração de documentação, etc.

Na segunda versão do *SAF* inseriu-se interfaces gráficas com mais detalhes da aplicação, maiores informações sobre o sistema automatizado, além da construção de métodos e classe para facilitar a reutilização de código, sendo estes métodos gerados a partir do *SAF0*. No *SAF 1.0* o operador tem possibilidade de selecionar as seguintes aplicações:

Acionamento Manual: possibilita ao usuário acionar e monitorar individualmente todos os transdutores.

Retificadora Paralela: possibilita a ação do *MHI-01* como uma retificadora.

Usinagem: simula a ação do *MHI-01* em um processo de usinagem.

Estas aplicações são simulações de processos industriais e podem ser interrompidas em qualquer instante. No funcionamento do *MHI-01*, como retifica paralela, o algoritmo de execução tem as seguintes etapas:

1. Aciona-se a válvula direcional de duas posições *S1* e em seguida *S3*.
2. Quando *FC3* for igual a 1, *S4* fica 10 milisegundos ligado, em seguida aciona-se *S2*.
3. Quando *FC4* for igual a 1, aciona-se *S3* e o processo se repete como em 2.
4. Quando *FC2* for igual a 1, aciona-se *S5* e o primeiro ciclo foi realizado.

O operador define no *SAF1.0* quantos ciclos o processo deve executar. No funcionamento do *MHI-01*, como usinagem, tem-se o seguinte algoritmo de execução:

1. Aciona-se *S1* em seguida aciona-se *S3*.
2. Quando *FC3* for igual a 1, aciona-se *S4*.
3. Quando *FC2* for igual a 1, aciona-se *S2*.
4. Se o ciclo de operação for maior que 1, *S3* é acionado e repete-se as etapas 2), 3) e 4).
5. Quando o número de ciclo é finalizado *FC1* e *FC4*, devem estar em 1.

Na Figura 3.17 mostra-se a tela do *SAF 1.0*, sendo que no centro desta, mostra-se a foto do *MHI-01*. Pode-se observar na Figura 3.17, quando os sensores estão acionados são representados pela cor verde, caso contrário ficam com a cor vermelha. No *Menu Principal* tem-se as seguintes funcionalidades:

- Equipe: apresenta uma breve descrição de cada integrante da equipe responsável pelo desenvolvimento do projeto.
- *IEEE 1451*: apresenta informações sobre o padrão *IEEE 1451*.
- Usinagem por Furação: atividade que simula um processo de usinagem no *MHI-01*.
- Retificadora Paralela: atividade que simula o funcionamento de uma retificadora no *MHI-01*.
- Sensores: apresenta informações, sobre os sensores que compõe o *MHI-01*, simulando a implementação da *TEDS*, ou seja, informa o tipo, fabricante e faixa de atuação de cada sensor.
- Solenóides: apresenta informações sobre os solenóides que compõe o *MHI-01*.
- Vídeo: neste *Menu* apresenta-se alguns vídeos de demonstração das atividades do *MHI-01* executadas com auxílio do *SAF 1.0*.
- Descrição: mostra-se as informações a plataforma *IEEE 1451*.

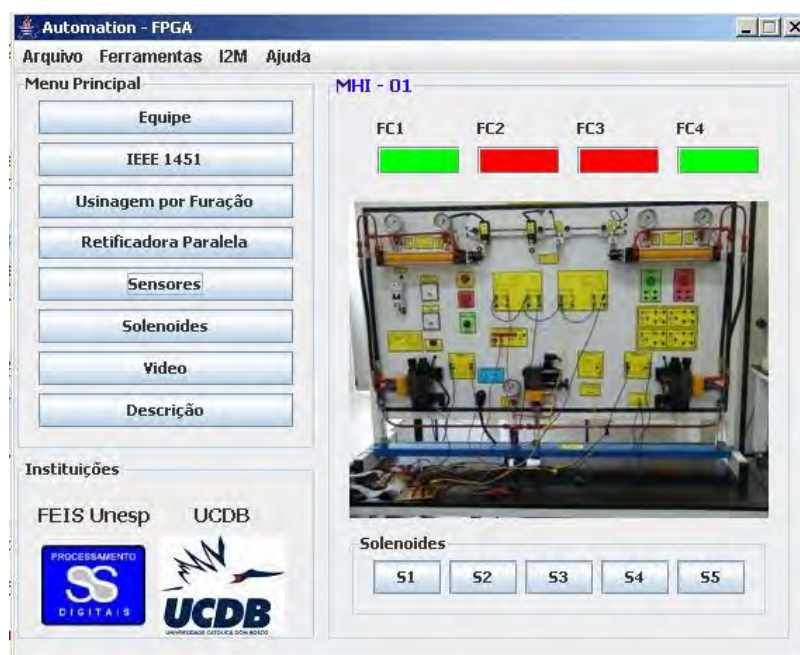


Figura 3.17: Tela do *SAF 1.0*.

No *SAF 1.0* existem os *Menus Arquivo*, *Ferramentas*, *I2M* e *Ajuda* que auxiliam na compreensão da proposta de implementação da plataforma *IEEE 1451* sendo que, no *Menu Arquivo*, encontra-se o texto relacionado ao padrão *IEEE 1451*. No *Menu Ferramentas*, encontram-se informações sobre as principais ferramentas utilizadas para

desenvolver as interfaces *IEEE 1451.1*, *IEEE 1451.2* e *IEEE 1451.5* por diferentes grupos de pesquisa.

No *Menu I2M*, encontra-se uma breve descrição do funcionamento do módulo Multi-Interface (*Interface Mult Module*) definido nesta *plataforma IEEE 1451*. No *Menu Ajuda* encontra-se informações sobre a utilização do *SAF 1.0*. O *SAF 1.0* foi composto das classes *InterfaceGráfica* e *Serial* nas quais desenvolveu-se métodos que proporcionam a inserção de telas e a comunicação serial respectivamente e independente uma das outras. Esta divisão de funções favorece a implementação de um software baseado em blocos funcionais como deve ser a parte lógica do *NCAP*.

No aperfeiçoamento do *SAF 1.0* inseriu-se funções como geração de relatórios contendo a *TEDS*, identificação de operadores, configuração de módulos para comunicação sem fio com base nas tecnologias *Wi-Fi*, *Bluetooth* e *ZigBee*, além da modificação do layout deste software. Nesta nova versão o software da plataforma *IEEE 1451* foi denominado de *NCAP*, por possuir um nome já conceituado por outros grupos de pesquisa. O software *NCAP* possui três pacotes contendo classes e objetos que podem ser manipulados individualmente, dependendo da aplicação.

Estes pacotes alinhados com a reutilização de códigos executam funções básicas como: comunicação com *driver TIM* e *driver* de rede (pacote *Communication*), armazenamento de informações em banco de dados (pacote *DataBase*) e gerenciamento das atividades conforme a aplicação (pacote *Interface*).

3.3 Software *NCAP* da Plataforma *IEEE 1451* – Segunda Versão

Neste item descreve a configuração do software *NCAP* proposto na plataforma *IEEE 1451*.

Os pacotes *Communication*, *DataBase* e *Interface* proporcionam interação nos níveis da automação, desde relatórios para composição de *MES (Manufacturing Execution Systems)*, passando por sistema de supervisão, até a comunicação com módulos de controle. Estes três pacotes servem como base para qualquer automação com a plataforma *IEEE 1451*, sendo que as inserções de classes devem ocorrer de modo a interagir com o pacote *Interface*, não alterando a estrutura principal deste software. O principal objetivo de deixar pacotes, classes e métodos comuns nas diferentes aplicações é auxiliar nas implementações das funcionalidades da plataforma *IEEE 1451*. Na Figura 3.18, mostra-se a configuração geral do software *NCAP*.



Figura 3.18: Pacotes que compõem o NCAP da plataforma IEEE 1451.

No pacote *Communication* encontra-se a classe *DriverTIM* responsável por viabilizar a comunicação serial, ou seja, gerencia o envio e recebimento de dados por meio da interface *RS232*. A leitura (recebimento de dados) disponível por *default* nesta classe é implementada com um vetor de 8 bits. Na plataforma *IEEE 1451*, o projetista pode modificar os parâmetros da comunicação (*baude rate*, *stop bit* e *paridade*) e o tamanho do vetor para leitura da interface *RS232*. Para viabilizar o dinamismo dos eventos, implementou-se métodos da classe *SerialPortEventListener* e os recursos das *Threads*. Na Figura 3.19 apresenta-se a documentação geral dos métodos pertencentes a classe *DriverTIM()*. Os métodos foram desenvolvidos com base nas definições do padrão *IEEE 1451.5* e a descrição das funções de cada método pode ser observada na Figura 3.19.

Constructor Summary	
<code>DriverTIM()</code>	
Method Summary	
<code>void EventData()</code>	Treates the data receavement from serial port.
<code>void EventInterrupt()</code>	Controls the data flow from serial port.
<code>int GetData()</code>	Converts data from Byte to Int.
<code>void IdPort (java.lang.String port)</code>	Defines the port for connection.
<code>static void init()</code>	Creates the connection with the port.
<code>static void ListPort()</code>	List the serial ports present.
<code>void read()</code>	Receives data from serial port.
<code>void serialEvent (javax.comm.SerialPortEvent ev)</code>	Controls the lecture event from serial port.
<code>static void SetPayloadSize (int baud, int data, int stop, int par)</code>	Defines the connection parameters.
<code>static void write (int OutetArray)</code>	Send the data throught the serial port.
Methods inherited from class java.lang.Object	
<code>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</code>	

Figura 3.19: Descrição dos métodos da classe *DriverTIM*.

No pacote *DataBase*, encontra-se a classe *TEDS()* responsável por alocar as informações da rede de transdutores, enviadas pelo *TIM*. Os principais métodos desta classe são *init*, *insertTEDS* e *UserVerify*. No método *init* cria-se a conexão com o banco e no método *insertData* inserem-se as informações neste banco de dados. Quando o operador realiza a aquisição da *TEDS* ao *TIM*, as informações são armazenadas no banco de dados, através do método *insertTEDS*. No método, realiza-se a busca das informações decodificadas.

O *NCAP* tem uma tela principal que auxilia o operador na utilização deste software. Esta tela principal é desenvolvida na classe *MainInterface.java* e suporta outras telas de forma hierárquicas com base nas solicitações do operador. A classe *MainInterface.java* possui métodos que se relacionam com as classes *DriverTIM.java* e *TEDS.java*, além de possibilitar a inserção de métodos para atender as necessidades das aplicações. Por exemplo, na automação do kit de Hidráulica desenvolveu-se uma classe denominada *Hydraulic.java*, que pertence ao pacote *Interface* e relaciona unicamente com a classe *MainInterface.java*. Todas as atividades desta aplicação são controladas pela classe *MainInterface.java*.

Na Figura 3.20 mostra-se o diagrama de classes do *NCAP* desenvolvido no *NetBeans IDE 6.5*, no qual pode-se observar a relação entre as classes e os pacotes. A implementação da *TEDS* é gerenciada pelas classes *Report.java* e *TEDS.java*, sendo apresentada na tela principal (classe *MainInterface*). As classes *Hydraulic.java* e *BioGasFull.java* são aplicações que não possuem dependência e portanto podem ser manipuladas individualmente, sendo essas controladas pela classe *MainInterface.java*. Nesta Figura 3.20, pode-se observar que a classe *MainInterface.java* implementam atributos das classes *DriverTIM.java* e *TEDS.java*. No Apêndice B, mostra-se parte da documentação do software *NCAP*.

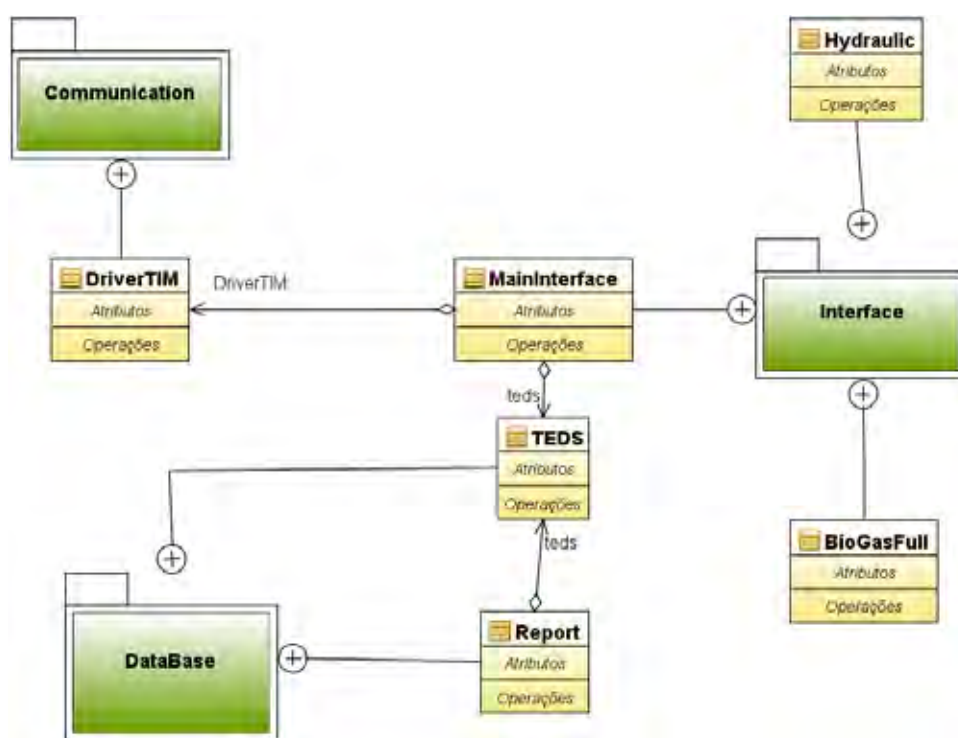


Figura 3.20: Diagrama de classe do software *NCAP*.

Ao executar a plataforma *IEEE 1451*, o projetista pode selecionar qual interface será utilizada (*IEEE 1451.2* ou *IEEE 1451.5*) e posteriormente define-se a aplicação. Caso seja selecionada a interface *IEEE 1451.5*, o projetista deve realizar a configuração da tecnologia sem fio (*Bluetooth*, *ZigBee* e *WiFi*) selecionada. O operador pode solicitar as informações da *TEDS* e terá como resultado as descrições da rede de transdutores fornecida pelo *TIM* implementada em relatório.

A parte software da plataforma *IEEE 1451*, proporciona também um cadastro de usuário e a geração de relatório.

O *NCAP* da plataforma *IEEE 1451*, segue como base as funcionalidades do *SAF*, porém, inseriram-se métodos e classes para melhor atender as atividades do padrão *IEEE 1451.1*, além de outras funções, como por exemplo, banco de dados, hierarquia de navegação de telas e *TEDS*. Esta documentação é gerada com o auxílio do ambiente *NetBeans* e deve otimizar o tempo de desenvolvimento de projetos. As funcionalidades do *NCAP* propostas na plataforma *IEEE 1451* coincidem com as de um software supervisor, desempenhando atividades como comunicação com a rede local, interface gráfica dinâmica e *driver* de comunicação com módulos de controle.

Os métodos e as classes que compõem a plataforma desempenham as funções dos blocos dos softwares em conformidade com o padrão *IEEE 1451.1*. Desta forma, a parte lógica do *NCAP* (nesta versão) possui três pacotes contendo classes e métodos que podem ser manipuladas individualmente dependendo da aplicação. Estes pacotes alinhados com a reutilização de códigos executam funções básicas como: comunicação com *driver TIM* e *driver* de rede (pacote *Communication*), armazenamento de informações em banco de dados (pacote *DataBase*) e viabiliza o desenvolvimento de interfaces gráfica de acordo com a aplicação (pacote *Interface*).

Os pacotes desta plataforma *IEEE 1451*, proporcionam a interação com os níveis da automação, desde relatórios para composição de *MES (Manufacturing Execution Systems)*, passando por sistema de supervisão até a comunicação com módulos de controle.

No software *NCAP* o projetista pode selecionar qual o tipo de interface *IEEE 1451 (IEEE 1451.2 ou IEEE 1451.5)* que deseja realizar a automação. Além disso, disponibilizam-se diferentes aplicações com intuito de tornar o sistema mais flexível.

Na Figura 3.21 mostra-se a tela inicial do *NCAP*, sendo que primeiramente o operador deve inserir a identificação (*login* e *senha*) para poder utilizar o software da classe *MainInterface* e os menus. Maiores informações sobre o software *NCAP* são apresentadas no Capítulo 5 e no Apêndice B.

Figura 3.21: Tela inicial do *NCAP*.

No Capítulo 4 descrevem-se o desenvolvimento do *TIM* da plataforma *IEEE 1451*.

CAPÍTULO 4 – TIM da Plataforma IEEE 1451

4.1 Introdução

Neste capítulo, apresentam-se as etapas de desenvolvimento do módulo embarcado, responsável por implementar as funções de um *TIM*.

Um dos desafios dos projetistas de hardware está em desenvolver sistemas embarcados cada vez mais complexos em menor tempo possível. Neste aspecto uma metodologia atrativa é o desenvolvimento baseado em plataforma que consiste em um conjunto de componentes pré-caracterizados e podem ser configurados e programados pelos projetistas. Cabe então aos projetistas conhecer em detalhes o ambiente de desenvolvimento e o hardware que deseja gerar para sanar os gargalos da produtividade.

A Altera possui bibliotecas de componentes que podem ser configurados para desenvolver hardwares específicos. Os componentes são disponibilizados no *Symbol*, na *MegaFunctions*, na *MegaWizard Plug in Manager* e no *SOPC Builder*. Estas bibliotecas de componentes serviram como base para o desenvolvimento do módulo embarcado que compõe a plataforma *IEEE 1451*.

4.2 MMI e DriverTIM

Neste trabalho implementou-se um módulo embarcado com funcionalidades semelhantes de um *CLP* (possibilita o controle das atividades de transdutores remotamente). Este módulo foi denominado de Modulo Multi-Interface (*MMI*) e, posteriormente, de *DriverTIM* que em conjunto com os circuitos de condicionamento de sinais formam o *Transducer Interface Module (TIM)*. O módulo foi renomeado por um fator estratégico, pois a sigla *MMI* representa a abreviação de *Mixed-Mode Interface* na norma *IEEE 1451.4*.

Entre as principais características deste módulo está na capacidade de *plug and play* com o *PC*, por cabeamento ou sem fio, sem a necessidade de mudanças estruturais no hardware e no *firmware* e o fato de possui processador embarcado. Para criar um hardware contendo processador *NIOS II* utiliza-se um sistema *CAD (Computer Aided Design)*, especificamente o software *Quartus II* e o ambiente *System-On-a-*

Programmable-Chip (SOPC) Builder. Desta forma as etapas do projeto para implementar um hardware dotado de processador *NIOS II* são: geração do hardware no *Quartus II* com auxílio do *(SOPC) Builder*, desenvolvimento da lógica de funcionamento do processador no ambiente *NIOS II IDE* e a integração do hardware e da lógica em *FPGA*. No *SOPC Builder*, o projetista define se o hardware será em *VHDL (Very High Speed Integrated Circuits Hardware Description Language)* ou *Verilog*, além de definir quais componentes são necessários para implementar o hardware. Estes componentes são a Unidade Central de Processamento, *Central Processing Unit (CPU)* que será embarcada no *FPGA* e os periféricos (memórias, interfaces de comunicação, pinos de entrada/saída, etc).

Os periféricos são conectados à *CPU* por meio de um barramento denominado de *Avalon Switch Fabric* e a interface *Blaster* permite o *download* do hardware para o *FPGA*. Com os dispositivos devidamente configurados e conectados ao barramento *Avalon*, gera-se o hardware criando um arquivo *.ptf (Project File)*. Este arquivo *Project File* representa a *CPU* do hardware e é solicitado no ambiente *NIOS II IDE*. No *NIOS II IDE* são disponibilizadas algumas funções e bibliotecas em linguagem *C* que podem auxiliar no desenvolvimento do *firmware*, como por exemplo, a *system.h*. Nesta biblioteca encontra-se o nome e o endereço de cada componente que compõe o hardware gerado no *SOPC Builder*, auxiliando diretamente na programação das atividades do módulo embarcado. Na Figura 4.22 pode-se observar as etapas para projetar um módulo embarcado dotado de processador *NIOS II*.

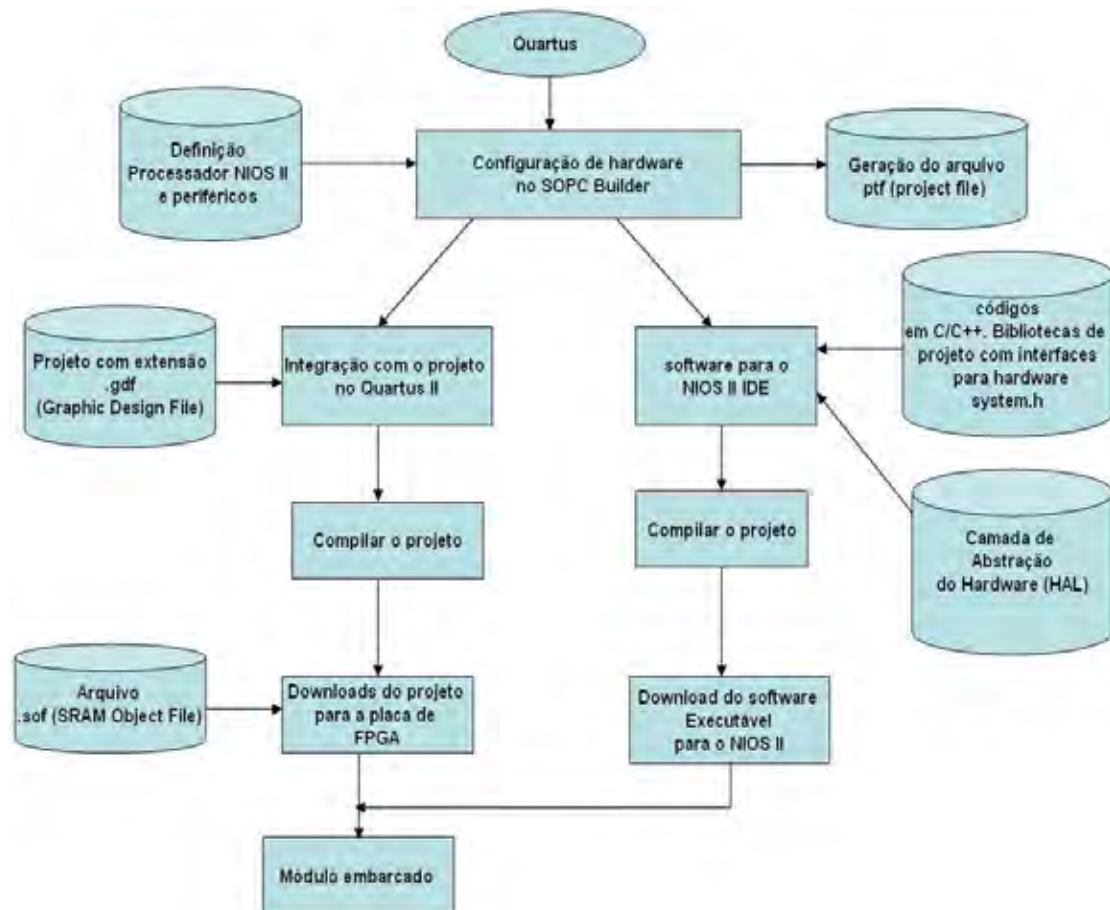


Figura 4.22: Esquema para projeto de Hardware com *NIOS II*.

Adaptado de: <http://www.altera.com/education/univ/materials/manual/unv-lab-manual.html>.

O hardware gerado no *SOPC Builder* aloca no *FPGA* a *CPU* com a memória *on-chip* e o barramento *Avalon*, os demais componentes são os periféricos. Na Figura 4.23 mostra-se uma arquitetura de conexão de um módulo embarcado gerado no *SOPC Builder*. Todos os periféricos que irão compor o módulo embarcado devem ser configurados no *SOPC Builder*, inclusive o *JTAG UART*, utilizado para realizar a comunicação entre os ambientes *NIOS IDE* e *Quartus II*.

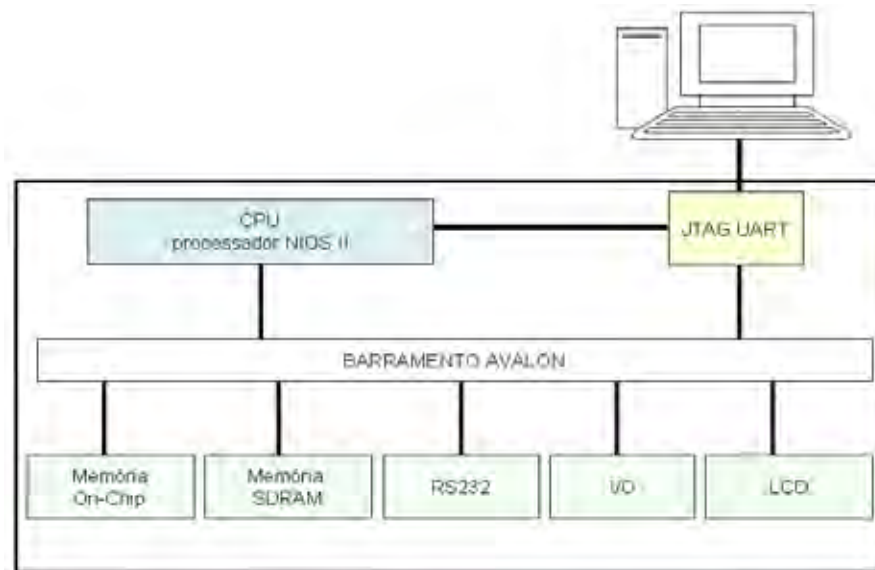


Figura 4.23: Configuração do *NIOS II* na Placa *DE2*.

4.2.1 Desenvolvimento do Módulo Embarcado da plataforma *IEEE 1451*

Desenvolveu-se neste trabalho um módulo embarcado que recebe os comandos enviados de um *PC* por meio da interface *RS232*, por cabeamento ou sem fio e transmite estes dados aos dispositivos de uma rede de transdutores. Na primeira versão, denominou-se este módulo embarcado de *MMI* (*Módulo Multi-Interfaces*), sendo que o termo *Multi-Interface* refere-se à viabilidade de implementar o padrão *IEEE 1451.2* e *IEEE 1451.5*. No primeiro teste realizado com o módulo *MMI*, os transdutores foram representados pelo conjunto de *led's LED[7..0]* da placa *DE2* da *Altera*. Este teste foi realizado para avaliar a capacidade de transmissão por cabeamento e sem fio entre o *PC* e o *MMI*. O software utilizado no teste foi o *SAF0*, que possibilita ao usuário enviar um dado no formato inteiro (valor máximo 255) para a porta serial (*COM1*) e este dado é devidamente implementado na matriz de *leds* da placa *DE2*.

O desenvolvimento de um sistema embarcado como o *MMI*, consiste de duas partes fundamentais, a parte hardware e a parte software. A parte hardware do *MMI* é composta de um *DriverMMI* da *PLL_MMI* e das pinagens necessárias para endereçar as *I/O* deste hardware. O *DriverMMI* foi gerado pelo *SOPC Builder* e é responsável pelas funcionalidades do *MMI*, ou seja, define a arquitetura do sistema embarcado. A *PLL_MMI* foi implementada utilizando-se os recursos da *MegaWizard* (ferramenta de desenvolvimento de funções do *Quartus II*). A parte software foi desenvolvida com linguagem *C* no ambiente *NIOS II IDE*.

Com os recursos do *SOPC Builder* desenvolveu-se o *DriverMMI*, tendo como principais componentes um processador *NIOS II/s*, uma memória *SDRAM* e os periféricos de comunicação *JTAG UART (Universal Asynchronous Receiver Transmitter)* e *UART (RS232)*. A memória *SDRAM* é utilizada, pois a aplicação no *NIOS II IDE* excede o tamanho das memórias *on-chip*. A interface *JTAG UART Blaster* possibilita descarregar a descrição do hardware para placa *DE2*, além de proporcionar o funcionamento da aplicação *NIOS II*. Se o cabo *JTAG UART* for desconectado da placa *DE2*, as atividades do processador *NIOS II* funcionarão durante um período de uma hora, fato que ocorre porque a licença é livre. O dispositivo *UART (RS232)* favorece a comunicação com a porta *RS232* da placa *DE2* com equipamentos externos.

A memória *SDRAM* que compõe o *DriverMMI* possui capacidade de armazenamento de 8 *MBytes*, e foi configurada de modo a possuir 12 linhas por 8 colunas de endereçamento, 16 bits de dados, 4 bancos *chipselect*. O projetista pode seleccionar outros tipos de memória, porém neste trabalho utilizou-se a versão da placa *DE2*. Na Figura 4.24, mostra-se a configuração da memória *SDRAM* utilizada para compor o *DriverMMI*.



Figura 4.24: Configuração da memória *SDRAM*, do *DriverMMI*.

Para desenvolver as atividades do *DriverMMI*, utilizou-se o NIOS II/s, por permitir alocação das instruções memória *cache*, utilização de sistema de *pepipeline* e execução de previsão de desvio (*Branch Prediction*), características ausentes na versão *NIOS/e*. Como as atividades do módulo *MMI* exigem várias execuções de instrução de desvio, as características do *NIOS II/s* tornam este processador mais eficiente. A versão *NIOS II/f* apesar de possuir mais recursos, não foi selecionada nesta aplicação por consumir maior quantidade de elementos lógicos, quando comparado com a versão *NIOS II/s*, porém, para aplicações que exigem maior capacidade de armazenamento e execução de algoritmos complexos, recomenda-se a versão mais completa.

Na Figura 4.25 observa-se as características de cada versão de processador *NIOS II* disponíveis e os parâmetros definidos para a *CPU* do *DriverMMI*.

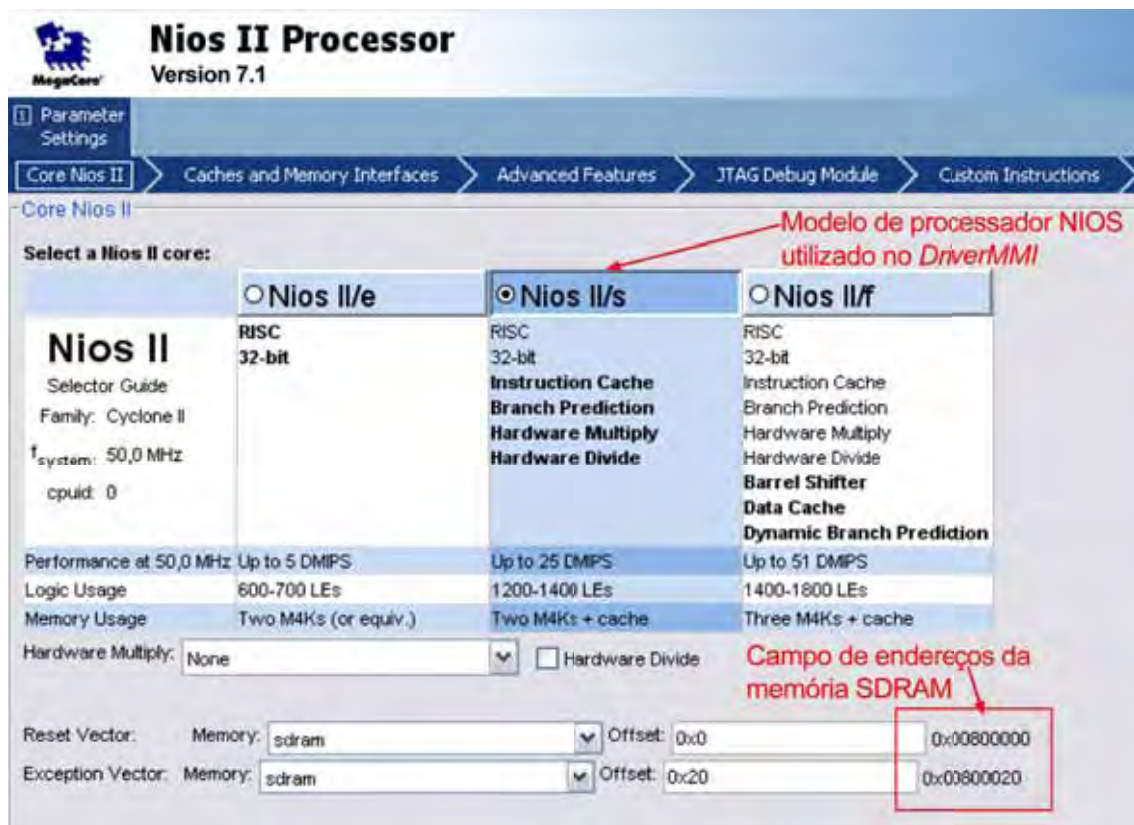


Figura 4.25: Tipo de processador *NIOS II* utilizado no *MMI*.

Na Figura 4.26 ilustra-se as configurações do emulador *NIOS II* utilizado no *DriverMMI*. A seleção do emulador se deve ao fato de necessitar apenas das interrupções via software, além de ocupar o menor número de células lógicas que os demais tipos de emuladores.

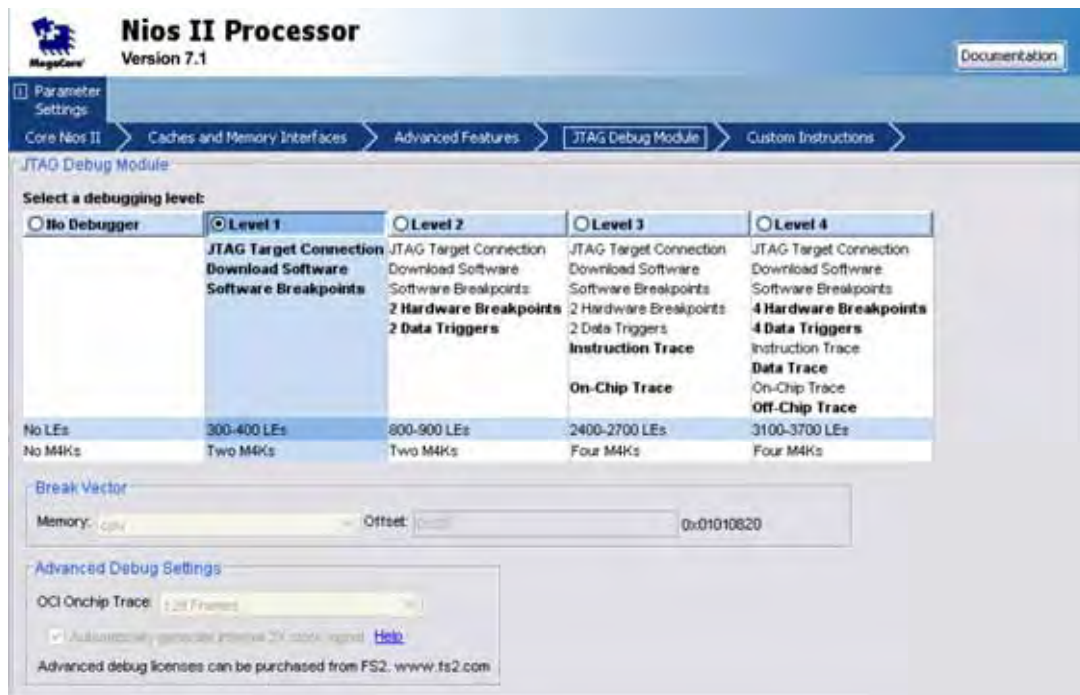


Figura 4.26: Definição do nível de *debugging JTAG* do *DriverMMI*.

Na configuração da interface *RS232* definiram-se os parâmetros da comunicação que serão utilizados entre o módulo embarcado (*MMI*) e o *NCAP*, como se verifica na Figura 4.27.

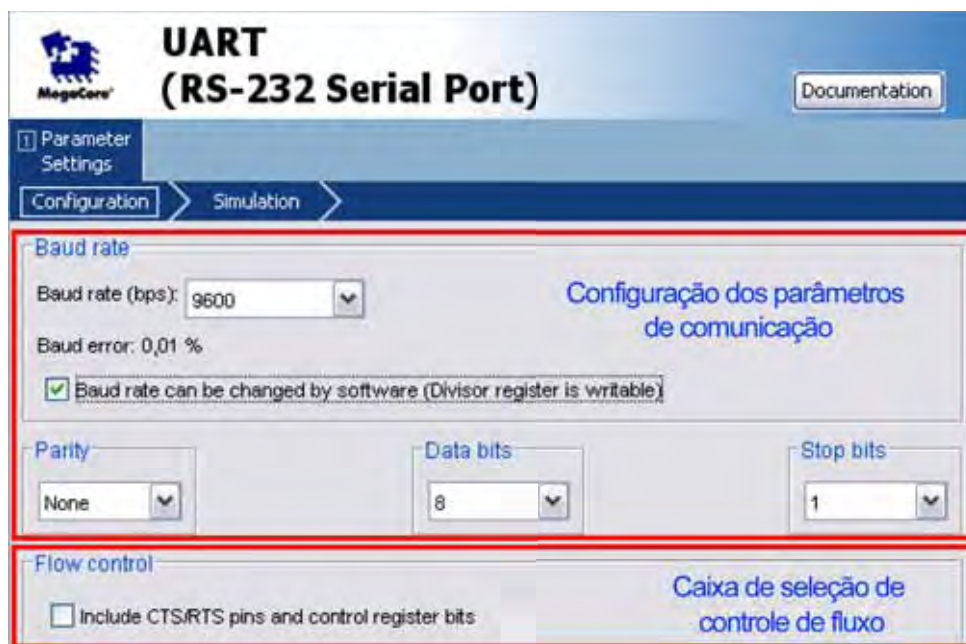


Figura 4.27: Configuração da *RS232* do *DriverMMI*.

Um dos destaques que compõe o *DriverMMI* é a interface *UART (RS232)*. Esta interface possui basicamente três blocos funcionais, como se mostra na Figura 4.28, sendo que: em (1) encontra-se o bloco responsável pelo recebimento de dados com os respectivos sinais de controle, em (2) está o bloco responsável pela transmissão de dados, e em (3) implementa-se os sinais de controle para receber e enviar dados conforme a configuração realizada no *DriverMMI*.

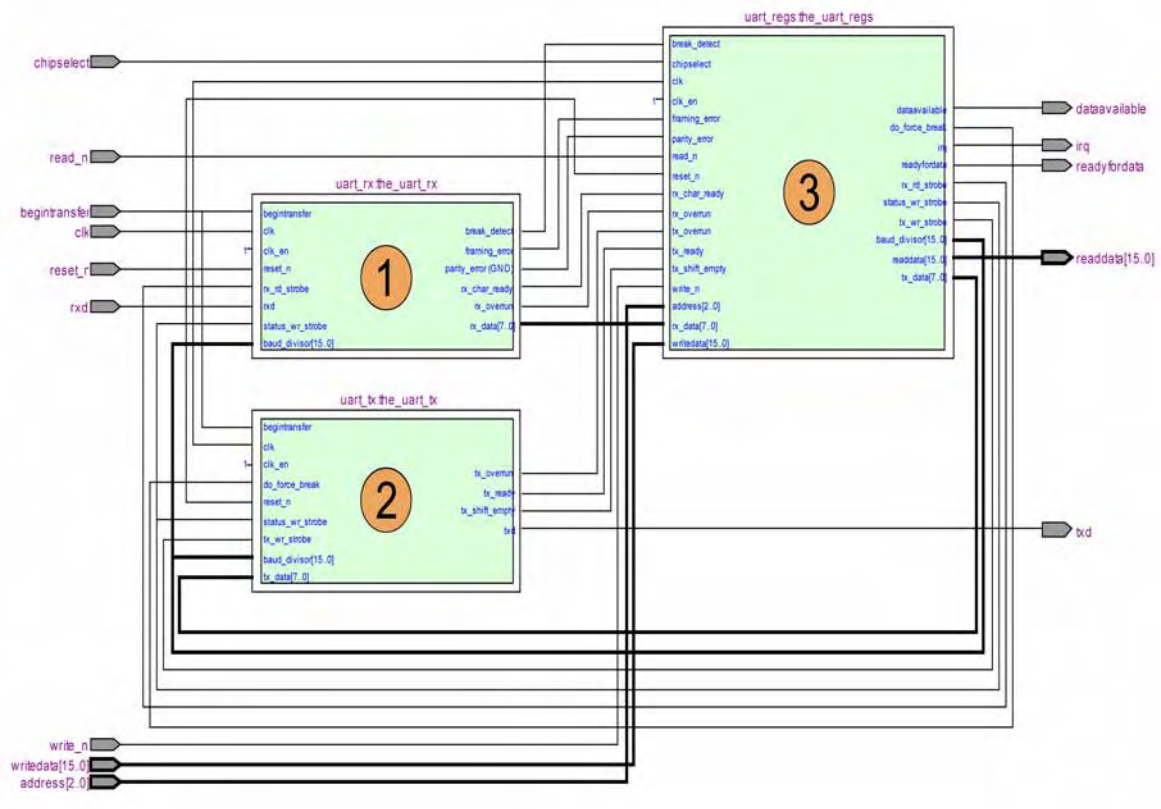


Figura 4.28: Interconexão dos componentes que compõem a *UART* para o *MMI*.

Os dispositivos de entrada e saída são configurados de acordo com a aplicação. Na Figura 4.29 mostra-se um dispositivo de saída com tamanho de um bit utilizado para compor o *DriverMMI*.



Figura 4.29: Dispositivo de saída configurado para compor o *DriverMMI*.

Na configuração dos dispositivos de entrada definiu-se o sistema de interrupção com tamanho de *quatro bits*, sendo que cada bit implementa-se as funções de um sensor digital. Na Figura 4.30 representa-se a configuração dos dispositivos de entrada que compõe o *DriverMMI*.



Figura 4.30: Configuração dos dispositivos de entrada do *DriverMMI*.

Ao solicitar a geração do hardware no *SOPC Builder*, o projetista obtém um relatório informando os principais tópicos deste desenvolvimento, como por exemplo: a versão do *SOPC Builder*, erro na formação do hardware, a data e hora que o hardware foi gerado, etc.

Com a geração do *DriverMMI* definiu-se a arquitetura básica do módulo embarcado que deve servir como base para o hardware de controle utilizado na automação dos processos.

Para completar as funcionalidades do *MMI* foi necessário desenvolver uma *Phase-Locked Loop (PLL)* para realizar o sincronismo entre o *clock* do sistema e o da memória *SDRAM* e a definição das pinagens. Para desenvolver a *PLL* utilizou-se os recursos da *MegaWizard Plug-in Manager* pertencentes a biblioteca de símbolos do *Quartus II*.

Acionando a *MegaWizard Plug-in Manager*, surge um conjunto de componentes denominados de *MegaFunction*. Para desenvolver a *PLL* deste projeto foi utilizada a *MegaFunction I/O* e o componente *ALTPLL*. Esse dispositivo foi denominado de *PLL_MMI*, como pode ser verificado na Figura 4.31.

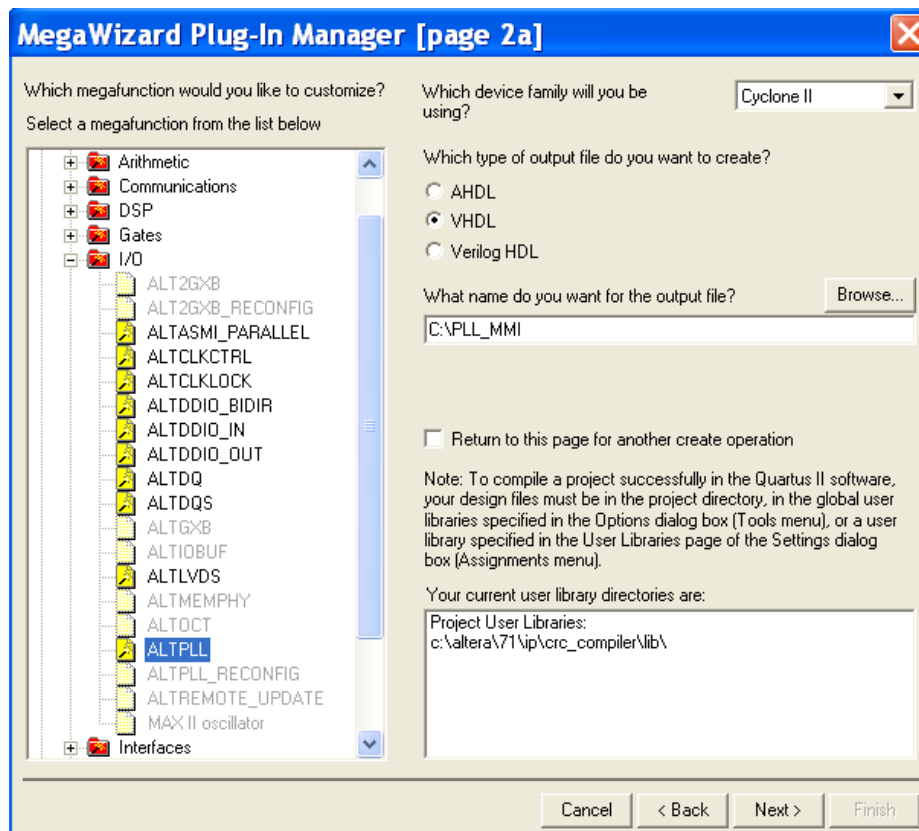


Figura 4.31: Definição da *PLL_MMI*.

A implementação da *PLL* segue oito passos pré-definidos, cabendo ao projetista montar o dispositivo de acordo com cada projeto. Na Figura 4.32 mostra-se a tela inicial para a configuração da *PLL_MMI*.

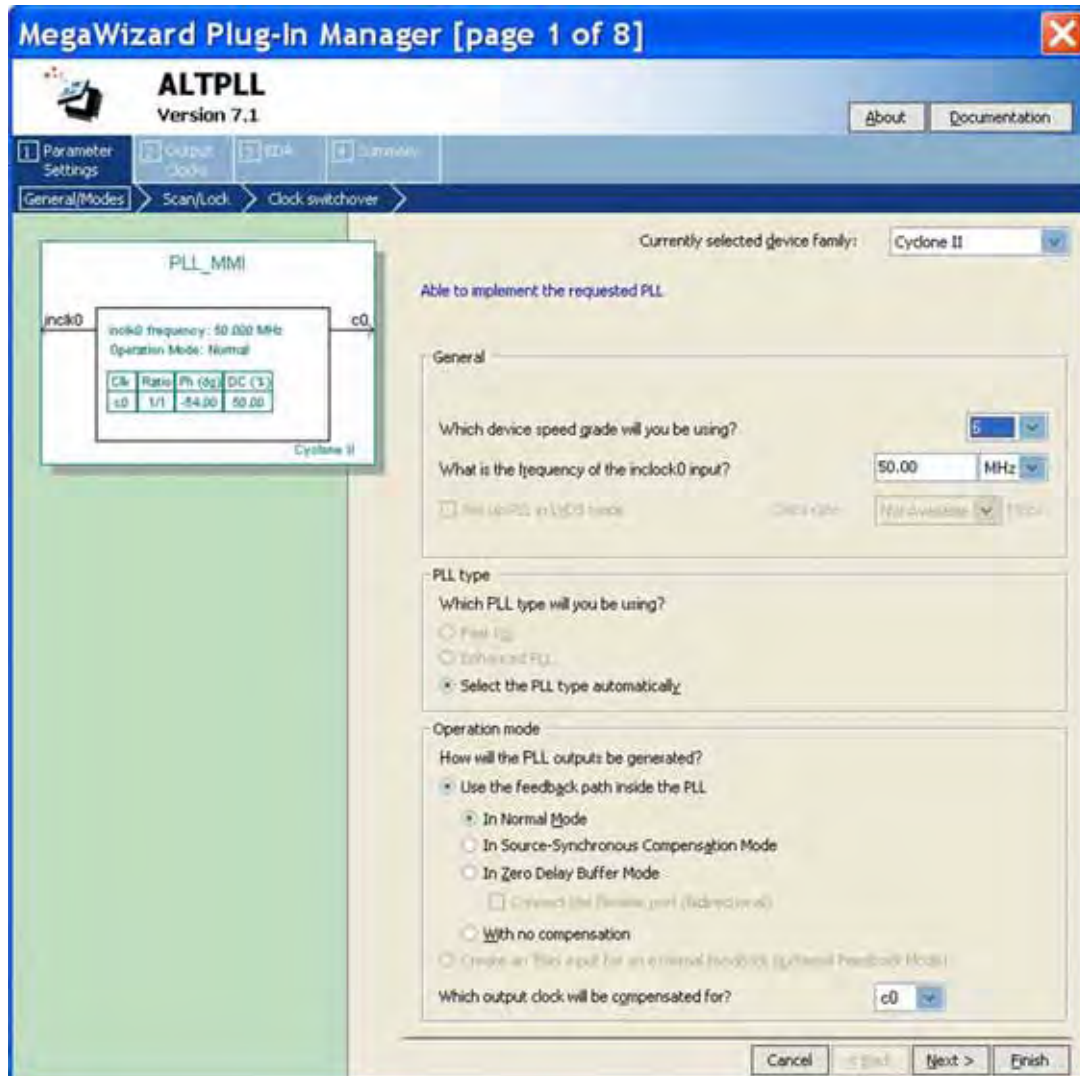


Figura 4.32: Definição da frequência da *PLL_MMI*.

Na *PLL_MMI* foi utilizado um *clock phase shift* de -3 nanosegundos, como pode ser observado através das setas apresentadas na Figura 4.33.

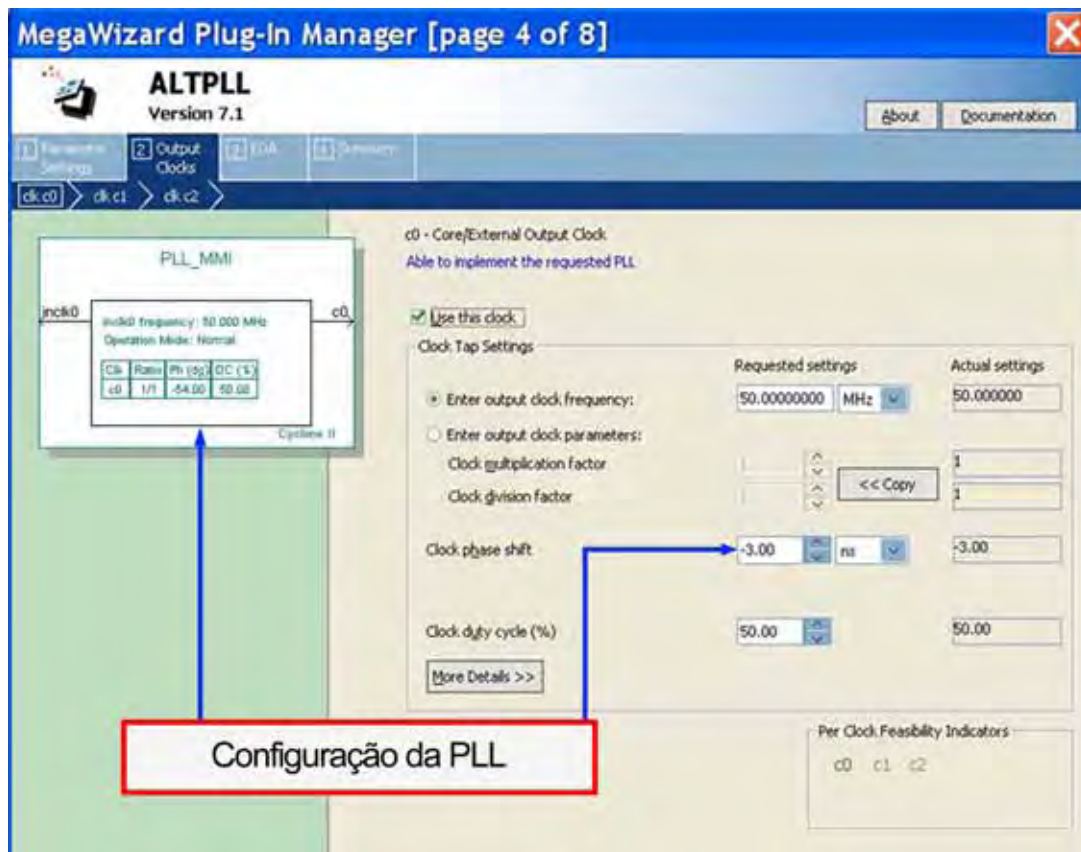


Figura 4.33: Definição da fase da *PLL_MMI*.

Na Figura 4.34 apresentam-se as descrições da *PLL_MMI*.

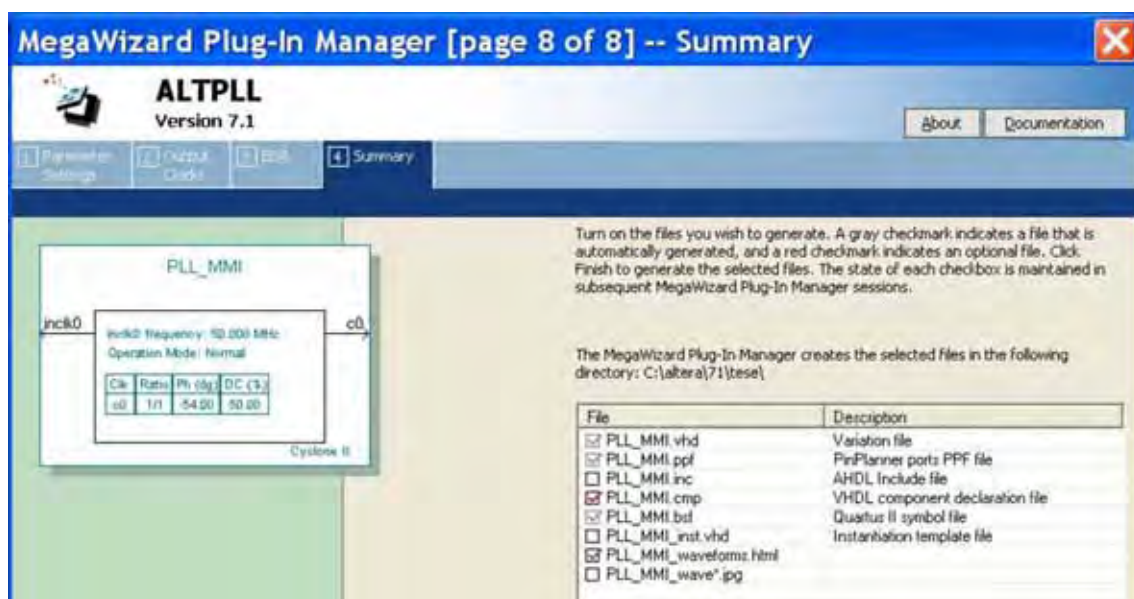


Figura 4.34: Características da *PLL_MMI*.

Construído o *DriverMMI* e a *PLL_MMI*, ambos pertencem à biblioteca de componentes do projeto *MMI*, para ser instanciados conforme a necessidade. Na Figura 4.35 é apresentado a simbologia da *PLL_MMI*, projetada para compor o *MMI*.

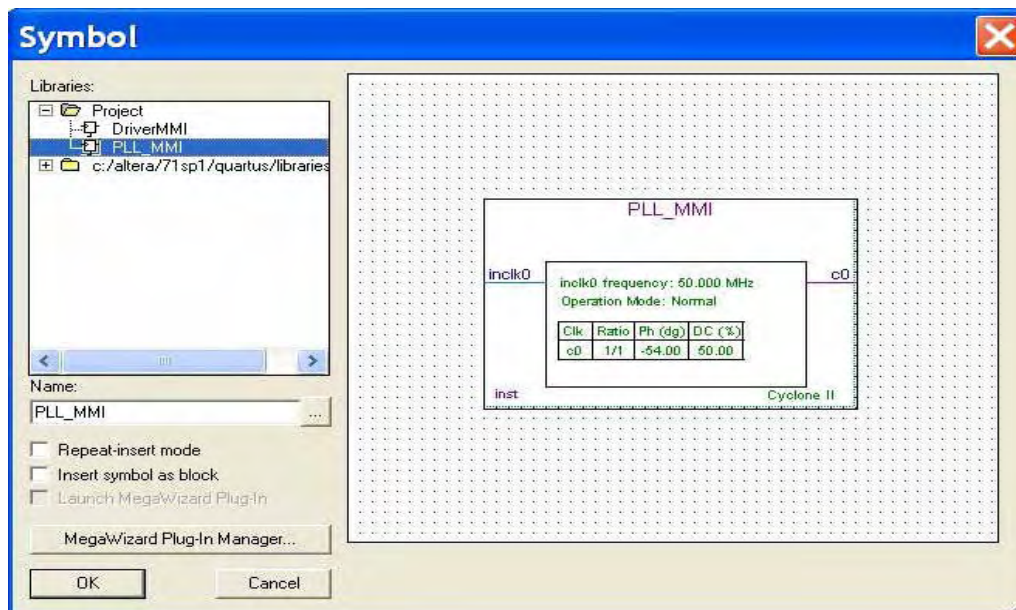


Figura 4.35: Símbolo da *PLL_MMI*.

Realizando a conexão dos blocos lógicos *PLL_MMI* e *DriverMMI* com os respectivos componentes de entrada e saída, pinos de *reset* e *clock* implementa-se o hardware do *MMI*. Com a pinagem devidamente endereçada, foi realizada a compilação do projeto *MMI*, na qual se obteve os resultados apresentados na Tabela 4.5.

Tabela 4.5: Informações sobre o hardware *MMI*.

Família	Cyclone II
Dispositivo	EP2C35F672C6
Total de Elementos Lógicos	3503 (11%)
Total de PLL	1 (25%)
Total de Pinos	131 (28%)
Total de Memória	210.048 (43%)
Total de Registradores	2044

O *DriverMMI* é composto por diversos blocos funcionais, sendo todos descritos em VHDL e devidamente interconectados. Os blocos funcionais são implementados com lógica de circuitos digitais específicos. Após a implementação do hardware *MMI*

desenvolveu-se o *firmware* deste módulo embarcado, no qual, definiu-se a lógica da operação.

4.2.2 Desenvolvimento do Firmware do MMI

No ambiente *NIOS II IDE* desenvolveu-se um *firmware* para gerenciar as funcionalidades do *MMI*. Este software proporciona a implementação do protocolo de comunicação entre *NCAP* e as interfaces *IEEE 1451.2* e *IEEE 1451.5*, além de viabilizar a lógica de controle dos processos a ser automatizados. Desta forma, desenvolveu-se uma lógica de controle que recebe os dados via interface *RS232* e implementa este valor na matriz de *led* da placa *DE2*. Como o tamanho do dado deve ser de oito bits, pois assim foi definido no *SOPC Builder*, então se utilizou oito *leds*, ou seja, o *DriverMMI* nesta aplicação possui oito canais de saída.

Na Figura 4.36 mostra-se a etapa do programa que estabelece a passagem do valor recebido da porta serial para a matriz de *leds*. Este programa utilizou os recursos do *template hello word*.

```
fp = fopen ("/dev/uart", "r+"); //abre a COM para iniciar comunicação

if (fp)
{
    prompt = getc(fp);

    //imprime mensagem no ambiente NIO II IDE - caso a comunicação tenha sucesso
    printf("O MMI recebeu com sucesso o valor %d \n", prompt);

    //Envia o dado para os leds (PIO)
    utport(PIO BASE, prompt);
}
```

Figura 4.36: Parte do programa do módulo embarcado *MMI*.

Na Figura 4.37 apresentam-se algumas propriedades do software utilizado para desenvolver a lógica de operação do *MMI*.

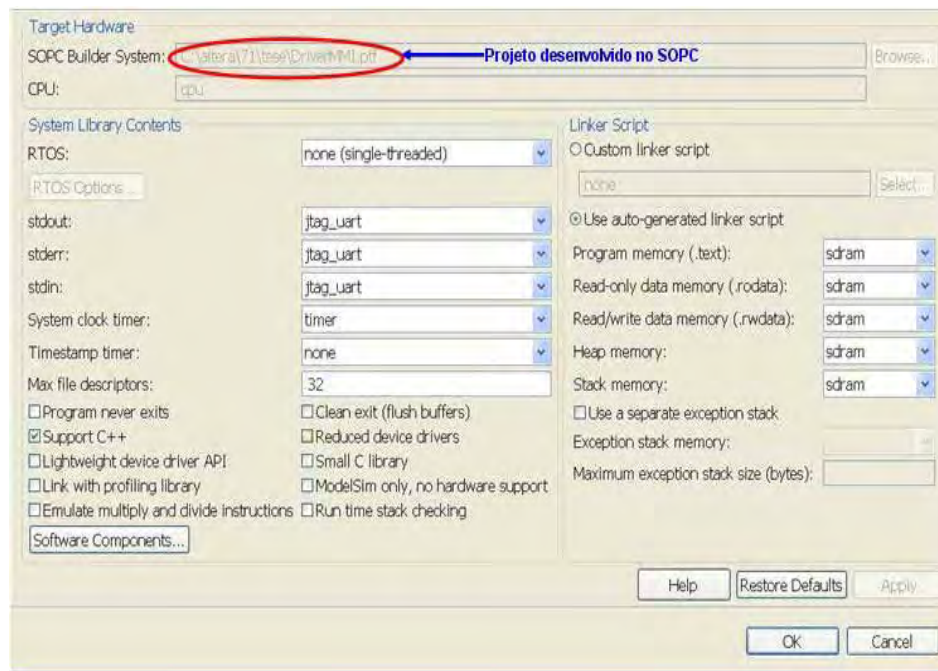


Figura 4.37: Propriedades da parte lógica do *MMI*.

Nesta etapa, o módulo embarcado foi utilizado para receber um dado enviado pelo SAF e disponibilizar esta informação no *prompt* do *NIOS II IDE*, como pode-se observar na Figura 4.38.

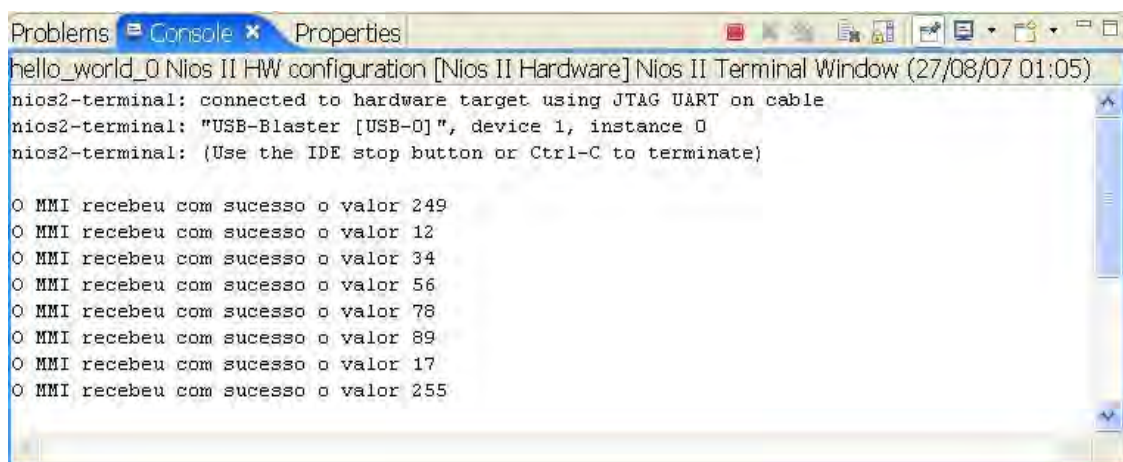


Figura 4.38: Operação realizada entre *SAF* e *MMI*.

No item 4.3 descreve-se a inserção dos conceitos do padrão *IEEE 1451* para desenvolver a plataforma proposta neste trabalho.

4.3 Parte Hardware da Plataforma IEEE 1451

Na composição da plataforma *IEEE 1451*, o *SAF* foi denominado de software *NCAP* e o *MMI* de *TIM*, pois, estas nomenclaturas são universais neste contexto. O *TIM* é composto de um módulo embarcado (*DriverTIM*) e de um *template IEEE 1451*.

Os dispositivos básicos para realizar as atividades de um *DriverTIM*, conforme a plataforma *IEEE 1451* são *CPU*, memória *on-chip*, memória *SDRAM*, *JTAG UART* interface *RS232*, *LCD*, *PLL*, todos com suas respectivas configurações. Esses dispositivos formam o *DriverTIM*, que em conjunto com os canais transdutores formam o módulo embarcado.

Os recursos para compor o hardware básico para o *DriverTIM* são apresentados na Tabela 4.6, sendo que a porcentagem refere-se à quantidade utilizada no *FPGA Cyclone II*. Nesta Tabela 4.6, pode-se observar que a quantidade de recursos exigidos para compor o *TIM* favorece a expansão deste hardware.

Tabela 4.6: Recursos básicos para compor o *TIM* no *Cyclone II*.

<i>Logic Cell</i>	<i>Dedicated Logic Register</i>	<i>Memory Bit</i>	Total de Pinos
3360, 10%	1931, 6%	46720, 10%	58, 12%

Na Figura 4.39 mostra-se a arquitetura básica do *DriverTIM*.

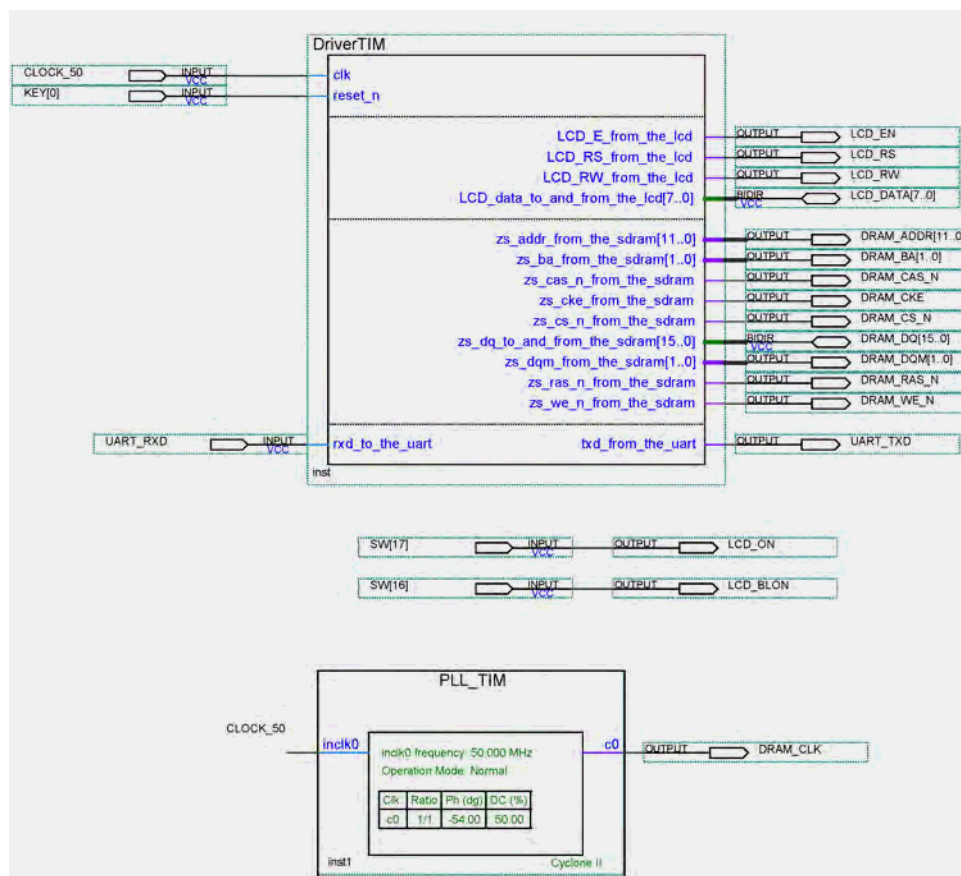


Figura 4.39: Arquitetura básica do *DriverTIM*.

A lógica de controle da aplicação, as informações da rede de transdutores (*TEDS*) e o programa de interfaceamento de comunicação são alocados na memória *SDRAM*. A utilização de alocação dinâmica de memória no *TIM* favoreceu a criação de bibliotecas com funções parametrizáveis que podem ser inseridas de forma padronizada.

Essas bibliotecas foram desenvolvidas no ambiente *NIOS II IDE*, dando origem ao *template IEEE 1451* e conseqüentemente ao *firmware* do *DriverTIM*, no qual viabilizou a programação das funcionalidades do *TIM*.

4.3.1 Template IEEE 1451 e Firmware do DriverTIM

O *template IEEE 1451* foi desenvolvido com linguagem *C* e *XML*, possui bibliotecas e funções para comunicação *RS232*, construção da *TEDS*, descrição do protocolo de comunicação da *TEDS*, além de disponibilizar *RTOS microC/OS-II* e funções para tratamento de interrupção. Os programas que compõem a pasta do *template IEEE 1451* são: *ieee1451.h*, *TEDS.h*, *ProtocolTEDS.h*, *IEEE1451.c*, *readme* e

o *ieee1451.xml*, sendo que os quatro primeiros, correspondem ao *firmware* do *DriverTIM*.

No programa *IEEE1451.c* pode-se desenvolver a lógica de controle, ele possui funções para realizar o tratamento da comunicação serial (interface *RS232*), possibilita a escrita de informações no *LCD*, troca informações com o barramento de expansão e envia para o *NCAP* as informações da *TEDS*. A realização da comunicação com o barramento de expansão ocorre através da implementação de funções pertencentes à biblioteca *IEEE 1451.h* e para enviar as informações da *TEDS*, utiliza-se funções da *TEDS.h*. Portanto no *IEEE1451.c* encontra-se a função *main()* e deve-se incluir necessariamente as bibliotecas *IEEE1451.h* e *TEDS.h*. As informações oriundas do *NCAP* para o *DriverTIM* passam necessariamente pelo *IEEE1451.c*, ou seja, este programa gerencia o protocolo de comunicação.

Na biblioteca *IEEE1451.h* desenvolve-se as operações com o barramento de expansão, de acordo com a aplicação e baseado nas especificações fornecida na *system.h*. Portanto, para cada transdutor implementa-se uma função que esta diretamente relacionada com o endereço no barramento de expansão e com o comportamento deste dispositivo. Quando o *NCAP* necessita enviar ou receber dados da rede de transdutores o gerenciamento desta comunicação é realizado pela biblioteca *IEEE1451.h*.

A biblioteca *TEDS.h* possui duas funções básicas, a *Info_TEDS* e a *Send_Info_TEDS*. Na *Info_TEDS* o projetista pode inserir as informações da rede de transdutores com base em uma tabela de códigos e *Send_Info_TEDS*. Estas informações são enviadas ao *NCAP*. Portanto a função *Send_Info_TEDS* deve ser implementada no *IEEE1451.c*. As especificações disponíveis na tabela de código referem-se aos fabricantes, tipos de transdutor e faixa de atuação de cada transdutor. O funcionamento da *TEDS* da plataforma *IEEE 1451* é descrita no item 4.4.

Na biblioteca *ProtocolTEDS* encontra-se a tabela de código definida neste trabalho. Esta tabela facilita a descrição da *TEDS*, pois o formato das informações pode ser modificado conforme a estrutura da programação.

No arquivo *readme* foram inseridas as informações sobre as atividades do *TIM*, uma breve descrição do template *IEEE 1451* e como a *TEDS* funciona.

No *ieee1451.xml* encontra-se informações visualizadas quando o projetista seleciona o template *IEEE 1451*, como pode ser observado na Figura 4.40, sendo que em:

- 1) Seleção do *template IEEE 1451*;
- 2) Uma breve descrição do template deve ser disponibilizada para os usuários;
- 3) As principais funções utilizadas no template são devidamente demonstradas em um campo específico;
- 4) A seleção da *CPU* para o *DriverTIM*, além das descrições deste template.

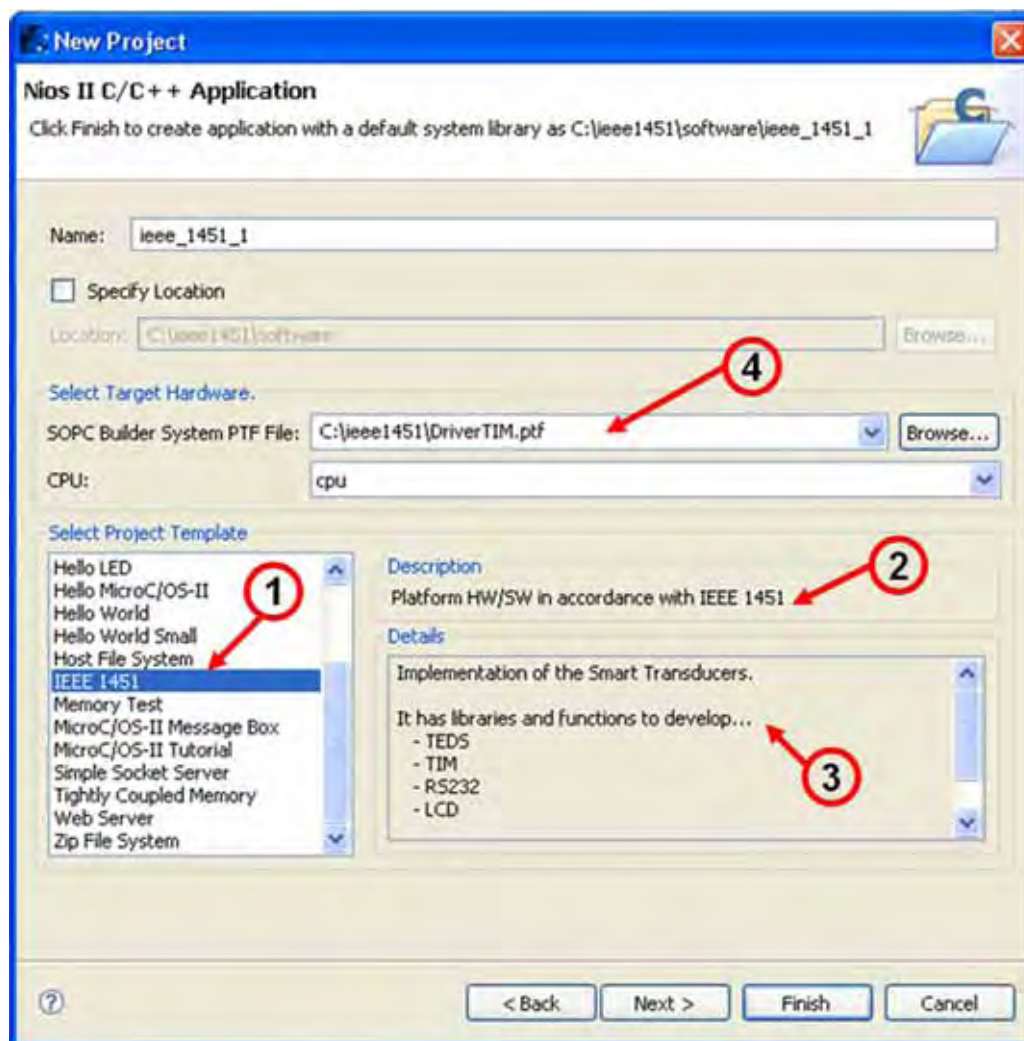


Figura 4.40: Template *IEEE 1451* disponibilizado no *NIOS II IDE*.

Na Figura 4.41 podem-se observar a composição do *firmware* da *plataforma IEEE 1451*.

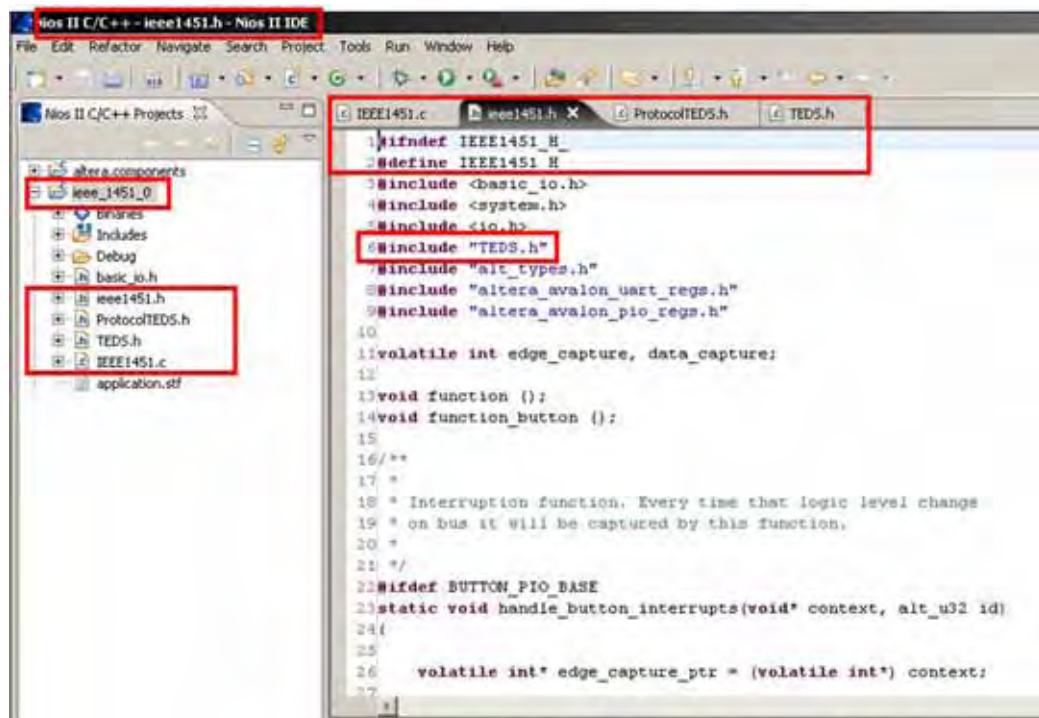


Figura 4.41: Implementação da função para tratamento de interrupção na biblioteca.

4.4 TEDS da Plataforma IEEE 1451

Neste trabalho desenvolveu-se uma arquitetura para implementar a *TEDS*, baseado nas funcionalidades do software *NCAP* e do *DriverTIM*. As informações dos transdutores são alocadas em três campos que representam o *MetaTEDS*, conforme a norma *IEEE 1451.0*. Estes campos são definidos através de uma faixa de código que representam as informações básicas da rede de transdutores. A Faixa de código foi definida neste trabalho, porém pode ser modificado em caso de padronização, assim como, a inserção de outras informações no campo *MetaTEDS*. Na Tabela 4.7, apresenta-se a configuração da *TEDS* da plataforma *IEEE 1451*.

Tabela 4.7: Configuração da *TEDS* da plataforma *IEEE 1451*.

<i>MetaTEDS</i> da plataforma <i>IEEE 1451</i>	Faixa de código
Fabricante do Transdutor	7 a 22
Modelo do Transdutor	0 a 4
Faixa de atuação dos transdutores	24 a 37

Ao descrever a *TEDS* o projetista deve inserir os códigos referentes aos três campos para cada transdutor, além de informar quantos transdutores serão descritos. A

quantidade de transdutor deve ser informada, para realizar a composição de um vetor de dados. Assim, quando o *NCAP* requisitar a *TEDS*, o *DriverTIM* envia as informações de cada um dos componentes, respeitando a formação deste vetor, ou seja, a posição de cada transdutor. Na Figura 4.42 mostra-se as funções da biblioteca *TEDS.h* que definem os campos e as codificações de uma rede transdutores.

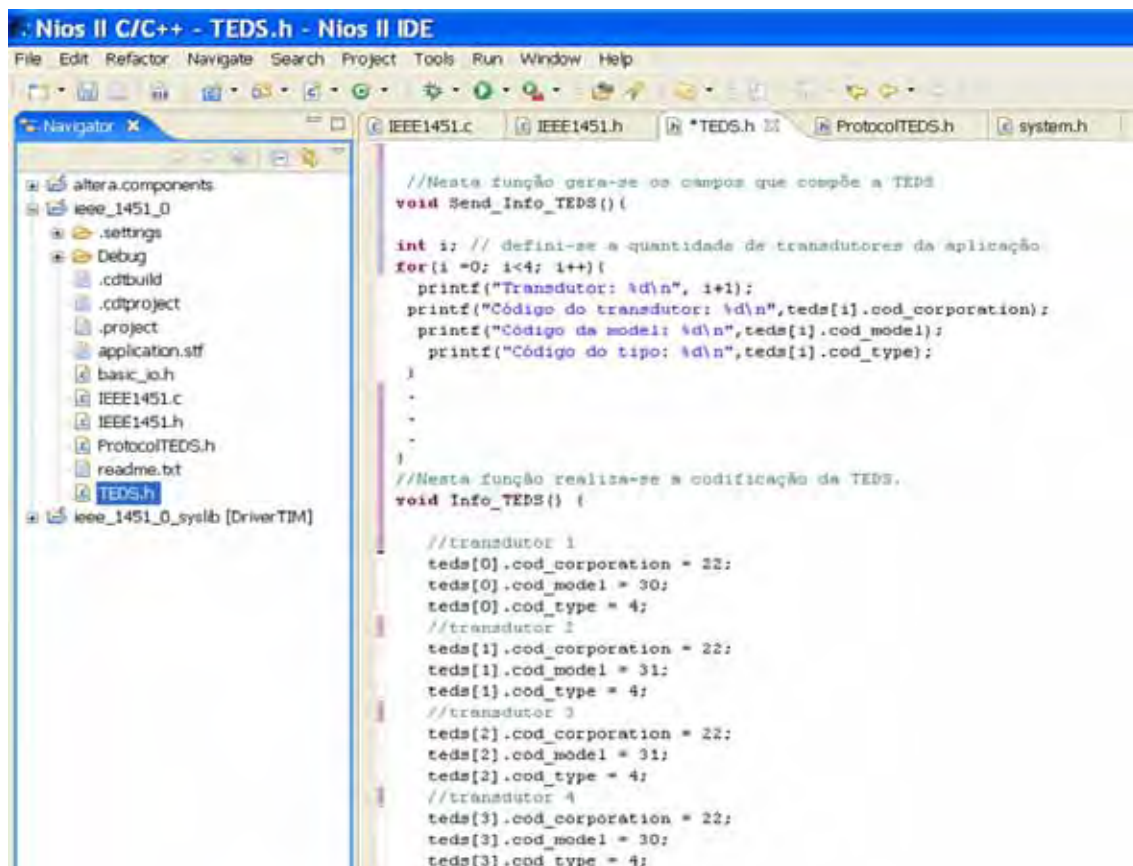


Figura 4.42: Biblioteca para enviar a descrição da *TEDS* para o *NCAP*.

No *NCAP* existe um método para identificar os transdutores conforme a sequência que foram descritos na biblioteca *TEDS.h*. Por exemplo, uma aplicação possui dois transdutores, um sensor e um atuador. Na programação da *TEDS.h* a primeira especificação é do sensor, portanto no *NCAP* esta ordem deve ser mantida. As informações (*TEDS*) enviadas do *DriverTIM* ao *NCAP* são tratadas no método *InfoSend()* da classe *MainInterface.java*, como pode-se observar na Figura 4.43.

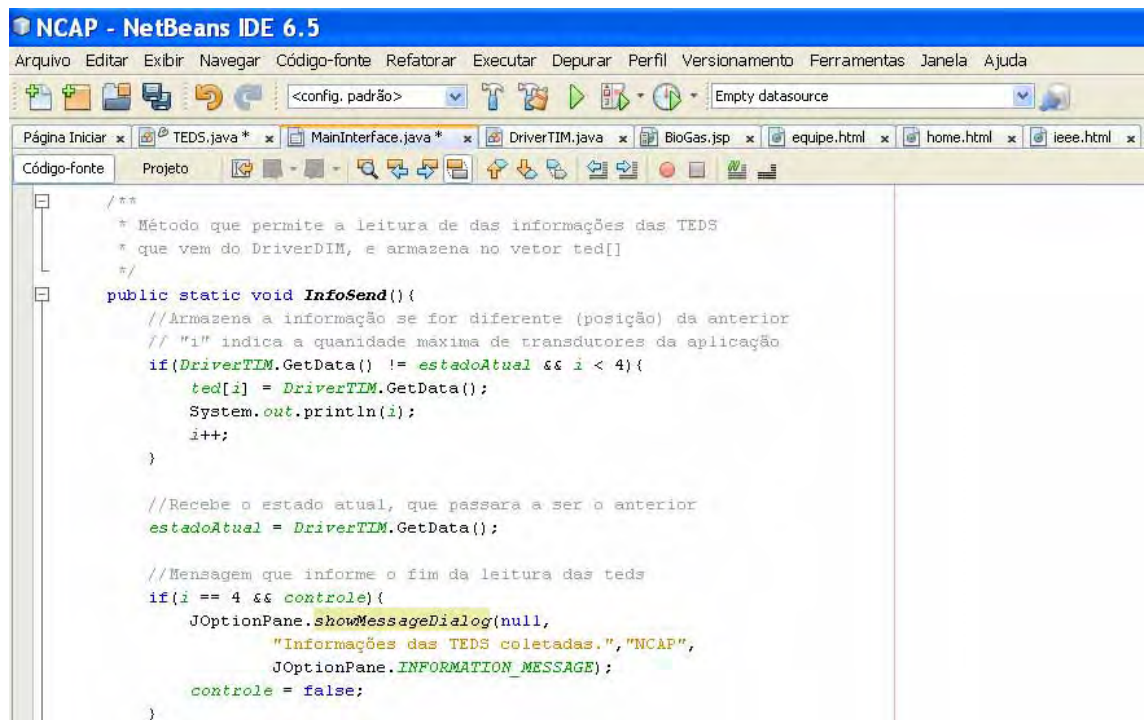


Figura 4.43: Método que recebe a *TEDS* enviada pelo *DriverTIM*.

Após a codificação da rede de transdutores na *TEDS.h*, realiza-se as seguintes etapas para implementar a *TEDS* no *NCAP*:

- 1) Através da opção *Info_TEDS* no menu do software *NCAP* o operador requisita as informações da rede de transdutores;
- 2) O código de requisição da *TEDS* chega ao *IEEE1451.c*. No *IEEE1451.c* existe uma função responsável por buscar as especificações na biblioteca *TEDS.h*;
- 3) As informações são enviadas ao *NCAP* através do *IEEE1451.c*;
- 4) Quando as informações chegam ao *NCAP* são decodificadas e armazenadas no banco de dados;
- 5) A *TEDS* é apresentada no *NCAP* especificamente para cada transdutor, ou seja, o operador deve indicar de qual componente deseja visualizar as informações.

Na Figura 4.44 observa-se a primeira etapa para solicitação da *TEDS* no *NCAP*. A tela que encontra o menu *Report* pertence à classe *MainInterface*. Ao selecionar o

menu *Report* e clicar em *InfoTEDS*, o *DriverTIM* envia as informações que são decodificadas e armazenadas no banco de dados.

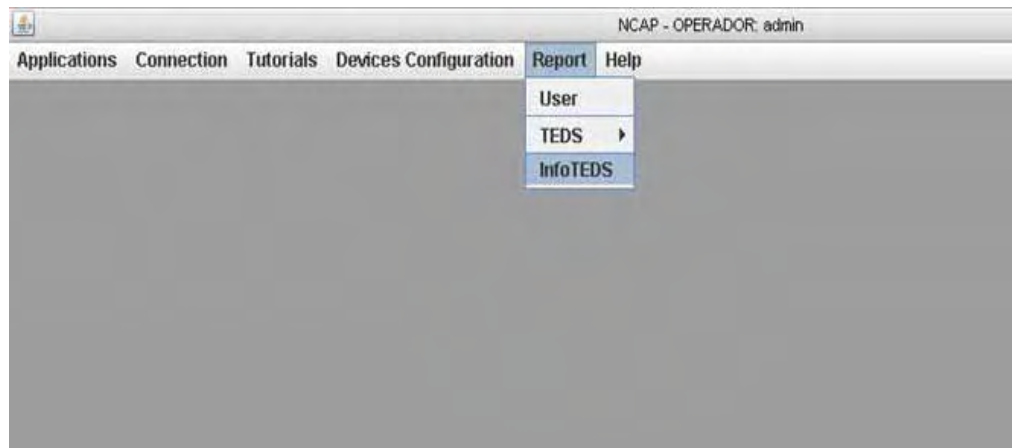


Figura 4.44: Primeira etapa para implementar a *TEDS* com a plataforma *IEEE 1451*.

Na Figura 4.45 as nomenclaturas dos transdutores devem ser colocadas conforme a ordem da descrição da *TEDS*, realizada na biblioteca *TEDS.h*. O operador pode selecionar as informações de qualquer um dos transdutores, sendo que nesta etapa a solicitação é realizada na classe *MainInterface*, porém as especificações são coletadas na classe *TEDS*.

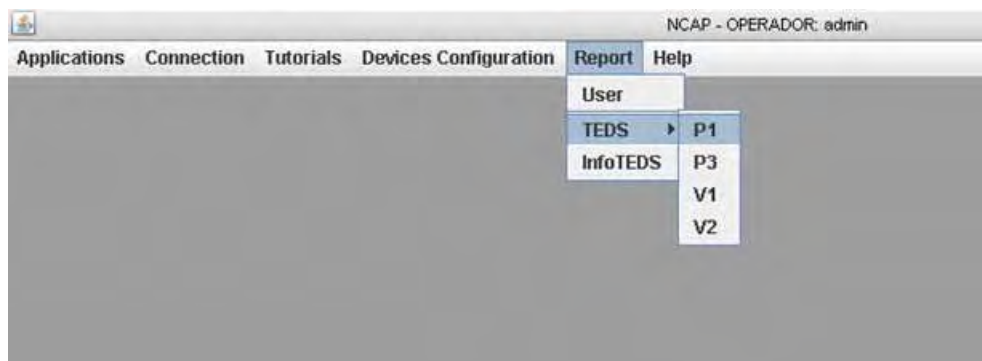


Figura 4.45: Segunda etapa para implementar a *TEDS*.

Na Figura 4.46 mostra a implementação da *TEDS*, contendo os três campos que compõem a *TEDS* da plataforma *IEEE 1451*, sendo que este arquivo é gerado pela classe *Report.java*.

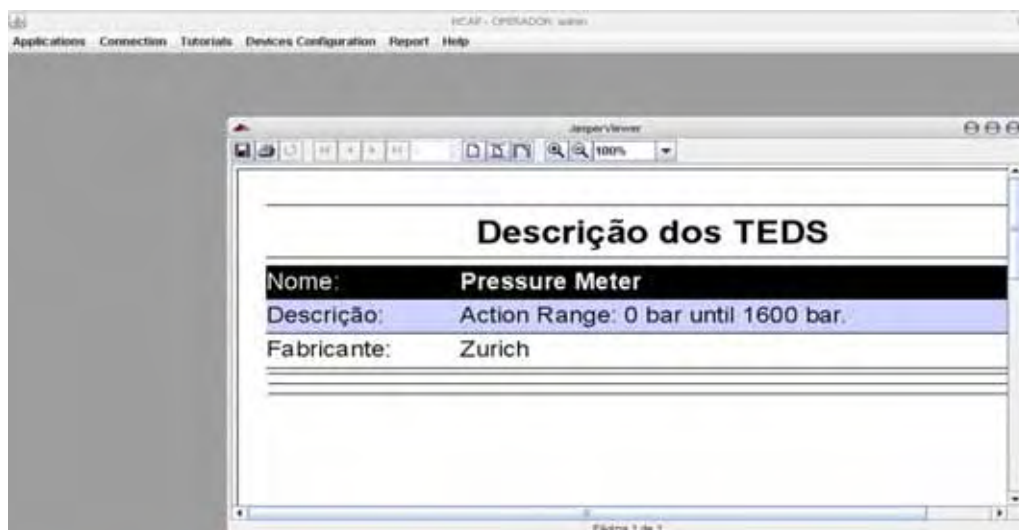


Figura 4.46: Implementação da *TEDS* da plataforma *IEEE 1451*.

No Capítulo 5, são descritos os resultados da automação de processos utilizando a plataforma *IEEE 1451* e os respectivos circuitos de condicionamento de sinais para compor o *TIM*, além da análise de desempenho das interfaces *IEEE 1451.5*.

CAPÍTULO 5 - Plataforma IEEE 1451 – Testes e Resultados

5.1 Introdução

A plataforma *IEEE 1451* foi utilizada na automação de diferentes processos, neste capítulo, são descritos os testes realizados na automação do Kit de Hidráulica *MHI-01* e no controle de processo de filtração de Biogás. Na automação do *MHI-01* testaram-se as funcionalidades do *SAF* e posteriormente do software *NCAP*.

O teste da plataforma *IEEE 1451* na automação do processo de filtração de Biogás foi realizado em duas etapas, uma em nível de protótipo e a segunda em escala real. No Apêndice D apresentam-se outros testes realizados com a plataforma *IEEE 1451*.

5.2 Automação do Kit de Hidráulica – MHI-01

O Kit de hidráulica *MHI-01* da empresa *Hidro Eletro Industrial* é utilizado nas disciplinas de Automação Industrial e Instrumentação Eletrônica do curso de Engenharia Mecatrônica da Universidade Católica Dom Bosco. Na automação do *MHI-01*, desenvolveram-se as mesmas atividades realizadas nas experiências das disciplinas. Porém, nas disciplinas a movimentação dos pistões é realizada com base nas conexões entre solenóides, relés, sensores e do acionamento da botoeira.

Cada ciclo da operação a botoeira deve ser acionada e para que a movimentação seja modificada realiza-se nova configuração de conexão. As lógicas de controle são baseadas no manual do fabricante e podem ser observado no fluxograma da Figura 5.47.

O kit *MHI-01* possui parâmetros e componentes que operam em escala real, ou seja, apesar de ser um kit didático, a aplicabilidade do teste pode ser extrapolado para indústria. Outro ponto atrativo na automação deste kit é a possibilidade de utilizá-lo em disciplinas como projeto digital, programação e controle.

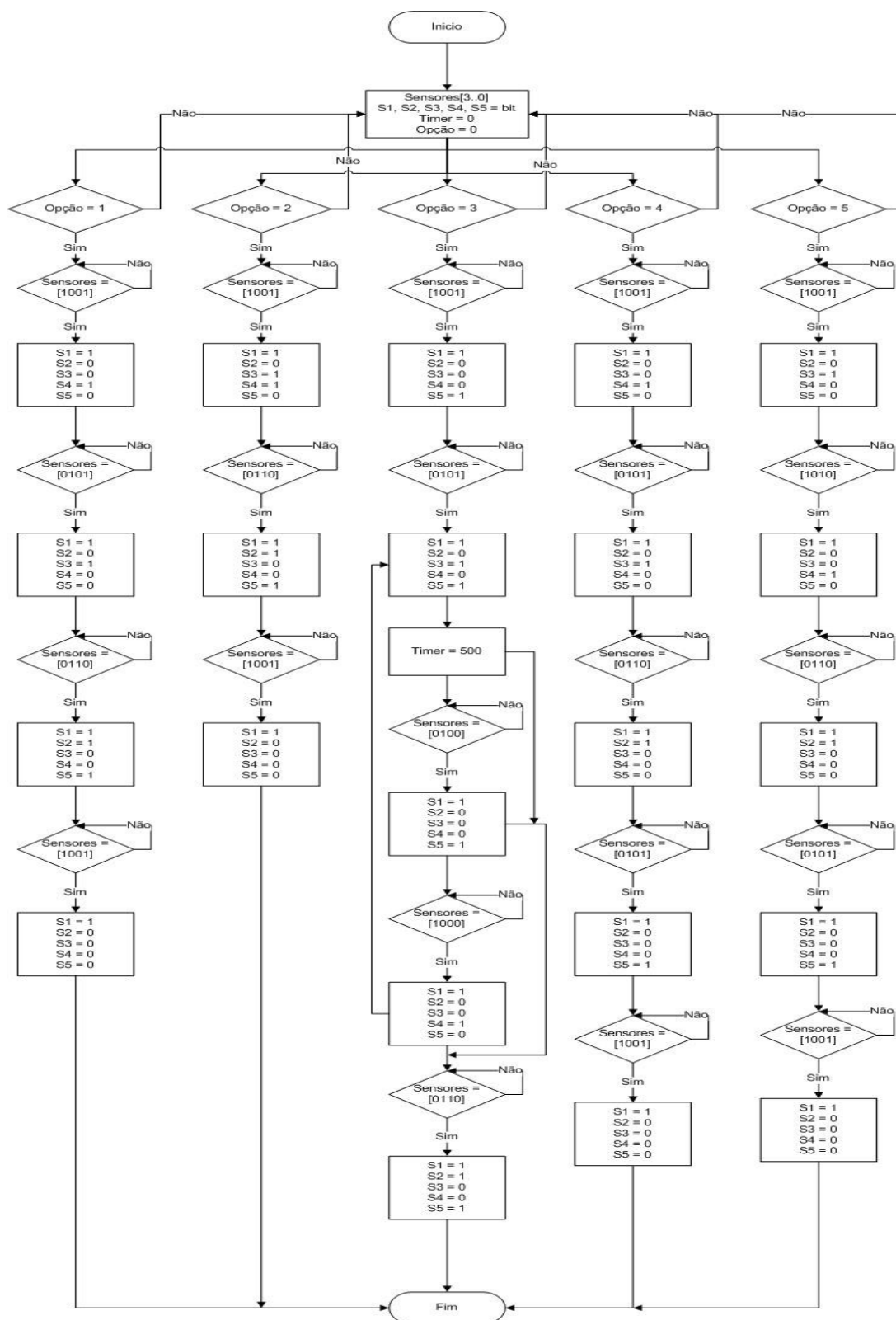


Figura 5.47: Fluxograma da lógica de controle para automação do *MHI-01*.

Estas lógicas de controle foram inseridas no *firmware*, especificamente no *IEEE1451.c* e para cada experiência atribui-se um código. Na Figura 5.48 mostra-se a parte da lógica de controle das atividades desenvolvidas no *MHI-01*, implementada na programação do processador *NIOS II*.

```

1  if(data_capture==5){
2      parametro=1;
3      experimento=5;
4  }
5
6  if(data_capture == 50){
7      outport(S1_BASE,0);
8      outport(S2_BASE,0);
9      outport(S3_BASE,0);
10     outport(S4_BASE,0);
11     outport(S5_BASE,0);
12
13     fprintf(fed, "%d(20",sec);
14     printf(fed, "EXPERIENCIA.....\n TODOS OS SOLENOIDES FORAM DESLIGADOS...");
15     break;
16 }
17 case 1:
18     //com o experimento=1 ymeto o parametro start =1 liberando
19     //a 1ª experiencia para idncipiar.//
20     if(experimento==1){
21         ciclo = data_capture;
22         parametro=0;
23         start=1;
24     }
25     if(experimento==2){
26         ciclo = data_capture;
27         parametro = 0;
28         start = 2;
29     }
30     if(experimento==3){
31         ciclo = data_capture;
32         parametro = 0;
33         start = 3;
34     }
35     if(experimento==4){
36         ciclo = data_capture;
37         parametro = 0;

```

Figura 5.48: Parte da lógica de controle das atividades do *MHI* no *NIOS II IDE*.

O *DriverTIM* utilizado na automação do kit de hidráulica é composto de 5 saídas e 4 entradas digitais. Na Figura 5.49, mostra-se a arquitetura do *DriverTIM*, sendo que em (1) representa-se 4 canais de entrada em um vetor [3..0], em (2) mostra-se 5 canais de saída e em (3) encontra-se a configuração da *PLL* que compõem o *TIM*.

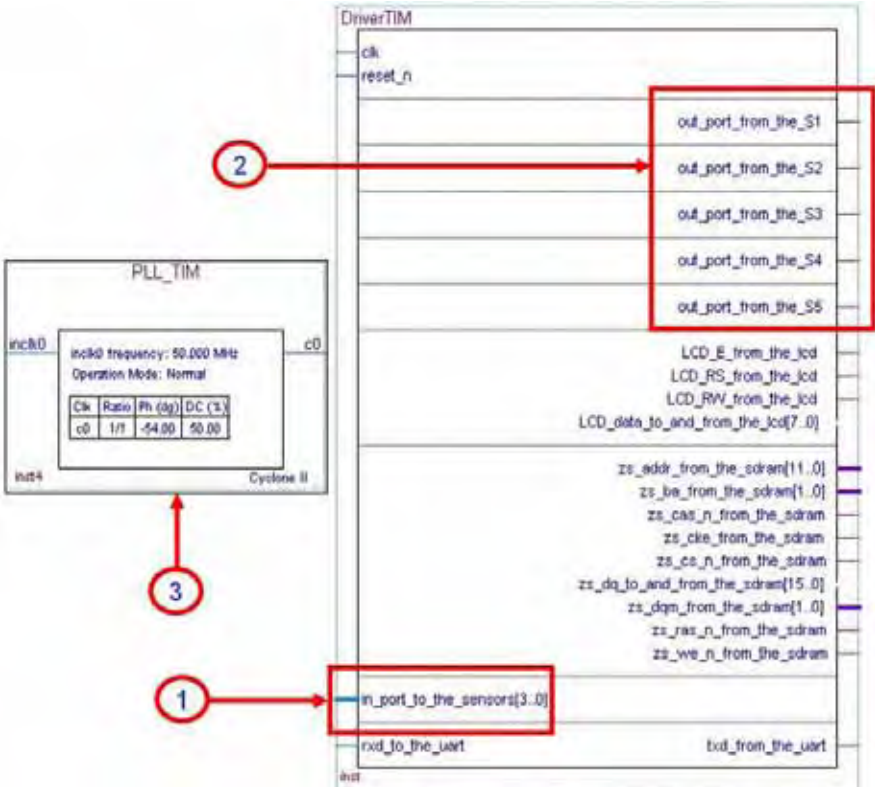


Figura 5.49: Arquitetura do hardware do *TIM* para automatizar o *MHI-01*.

Os recursos utilizados neste hardware são apresentados na Tabela 5.8.

Tabela 5.8: Recursos do *TIM*, para automação do kit de hidráulica no *Cyclone II*.

<i>Logic Cell</i>	<i>Dedicated Logic Register</i>	<i>Memory Bit</i>	Total de Pinos
4313, 13%	2468, 7%	228736, 47%	81, 17%

5.2.1 Circuitos de Condicionamento de Sinais – MHI-01

O circuito de acionamento apresentado na Figura 5.50 é composto de relés, transistores, resistores, *optoacopladores* e *schmitt trigger (CI 40106b)*. A tensão que sai da placa *DE2* através do barramento de expansão é de 3,3V, este sinal passa pelo *optoacoplador HC11* e em seguida alimenta a base do transistor para posteriormente atracar ou não o relé. Assim, a função do *optoacoplador* é de isolar os sinais de baixa tensão (3,3V) e alta tensão (24V), o transistor é polarizado para atuar como chave e os relés atracam com 12V, fornecendo na saída 24V. Desta forma, os transistores são alimentados com 12V (fonte de tensão) e os relés com 24V (kit MHI-01).

Na implementação das placas de circuito impresso disponibilizou-se bornes para facilitar a conexão dos dispositivos ao *TIM*. Na Figura 5.50, apresenta-se o *TIM* utilizado para automatizar o *MHI-01*, sendo que em (1) identifica-se os circuitos de condicionamento de sinais e (2) o módulo embarcado.

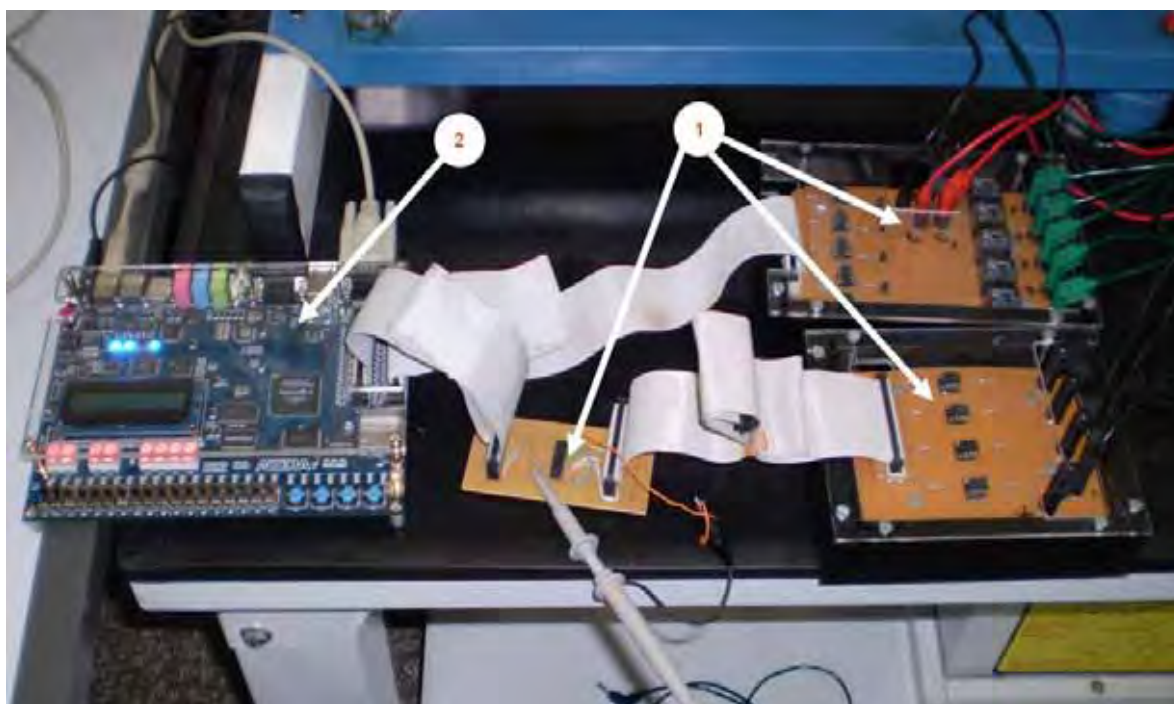


Figura 5.50: *TIM* para automação do *MHI-01*.

A automação do kit de hidráulica *MHI-01* pode ser analisada através do *SAF* e do software *NCAP*. Na primeira versão o operador seleciona no *SAF* qual atividade e o número de ciclos o *MHI-01* deve executar, além de observar em tempo real o monitoramento dos sensores, sendo que a cor verde significa que o sensor está ativado e a cor vermelha que o sensor está desativado. Desta forma, o operador conhecedor das atividades do *MHI-01*, pode ter uma noção do posicionamento dos pistões. A Figura 5.51 apresenta a interface gráfica do *SAF*, durante a execução de uma atividade, na qual pode-se observar as opções de desligar o sistema em caso de emergência, barra de rolagem informando a quantidade de ciclo executado, as instituições envolvidas no desenvolvimento deste trabalho, além das atividades disponíveis.



Figura 5.51: Interface gráfica do *SAF* utilizada na automação do *MHI-01*.

Para o software *NCAP*, acrescentaram-se as seguintes funcionalidades: geração de relatório e gráfico indicativo da quantidade que uma experiência foi executada. Na geração de relatório, o operador pode salvar e/ou imprimir o arquivo.

Na Figura 5.52 mostra-se o monitoramento das atividades dos sensores do kit *MHI-01*, além de apresentar o gráfico da quantidade de solicitação de cada atividade.



Figura 5.52: Telas para automação do *MHI-01* do software *NCAP*.

Outra maneira de monitorar as atividades dos sensores é realizada no display (*LCD*) da placa *DE2*, como pode-se observar na Figura 5.53. Este recurso (monitoramento por meio do *LCD* da *DE2*) é interessante quando ocorrer problemas no *PC* e como a lógica de funcionamento está no processador, o processo continua sendo executado.

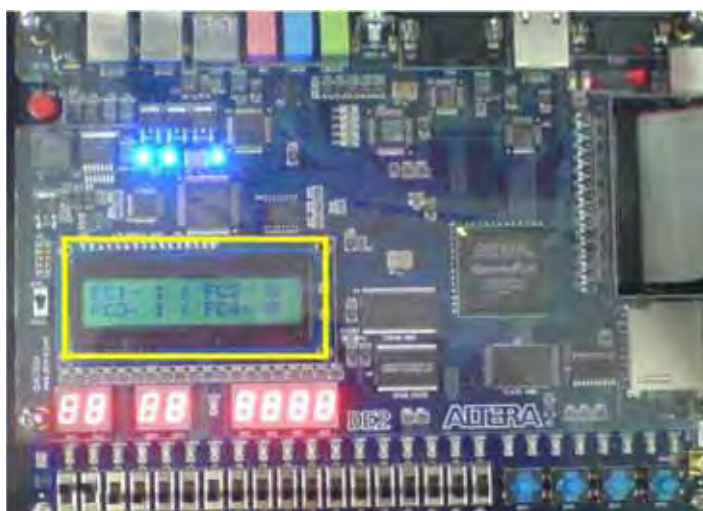


Figura 5.53: Monitoramento dos sensores (*MHI-01*) através da placa *DE2*.

Se o operador necessitar obter as informações de um determinado transdutor que compõem a aplicação, basta clicar na identificação do dispositivo correspondente, por exemplo, na automação do *MHI-01*, se o operador clicar em *FC1* aparecem às informações deste sensor (*TEDS*), como pode-se observar na Figura 5.54. Com a implementação da *TEDS* apresentada na Figura 5.54, verifica-se que o *NCAP*, além de decodificar as informações enviadas pelo *TIM*, possibilita que o operador possa salvar e imprimir este arquivo.

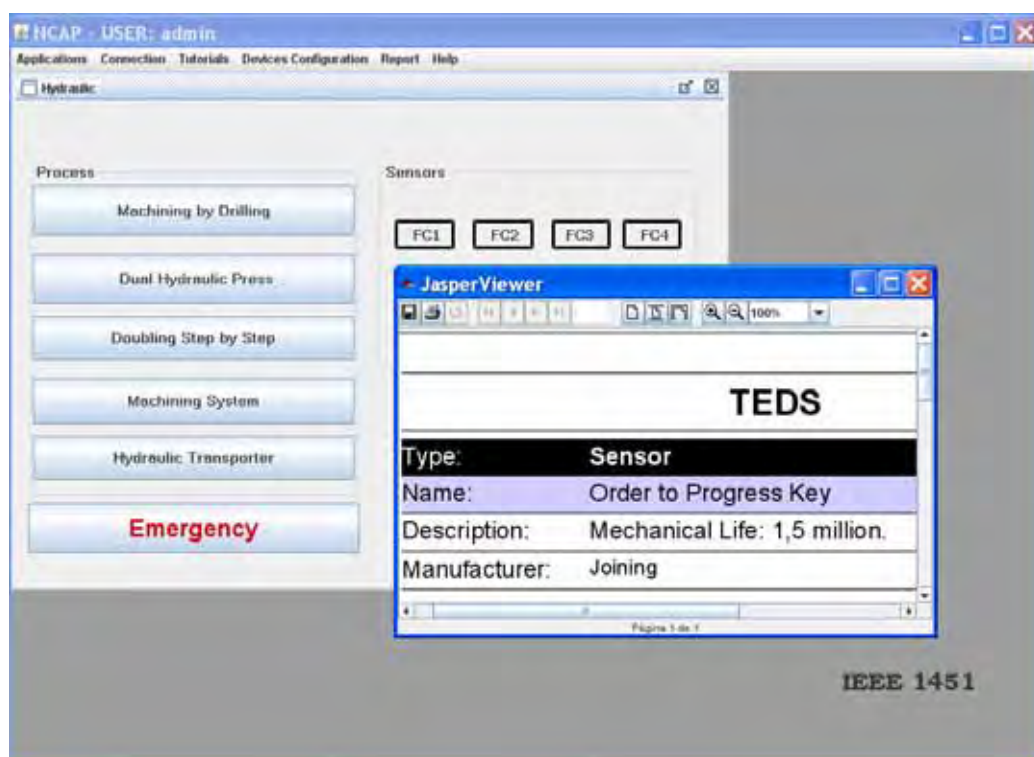


Figura 5.54: Etapa para geração de relatório.

A arquitetura física utilizada para testar e aprimorar o software *NCAP* e o *TIM* é apresentada na Figura 5.55, na qual se destacam os seguintes pontos:

1. O *PC* utilizado para executar as atividades do software *NCAP*. O *NCAP* utilizado nesta etapa possui interface gráfica dinâmica que informa o operador sobre o posicionamento dos pistões.
2. A placa *DE2* utilizada para viabilizar as funcionalidades do módulo embarcado.
3. Os circuitos eletrônicos utilizados para possibilitar o acionamento e monitoramento dos transdutores que compõem o *MHI-01*.
4. O osciloscópio foi utilizado para avaliar o sinal dos sensores.

5. A fonte de tensão foi utilizada para alimentar os relés que compõem o circuito eletrônico para o acionamento.

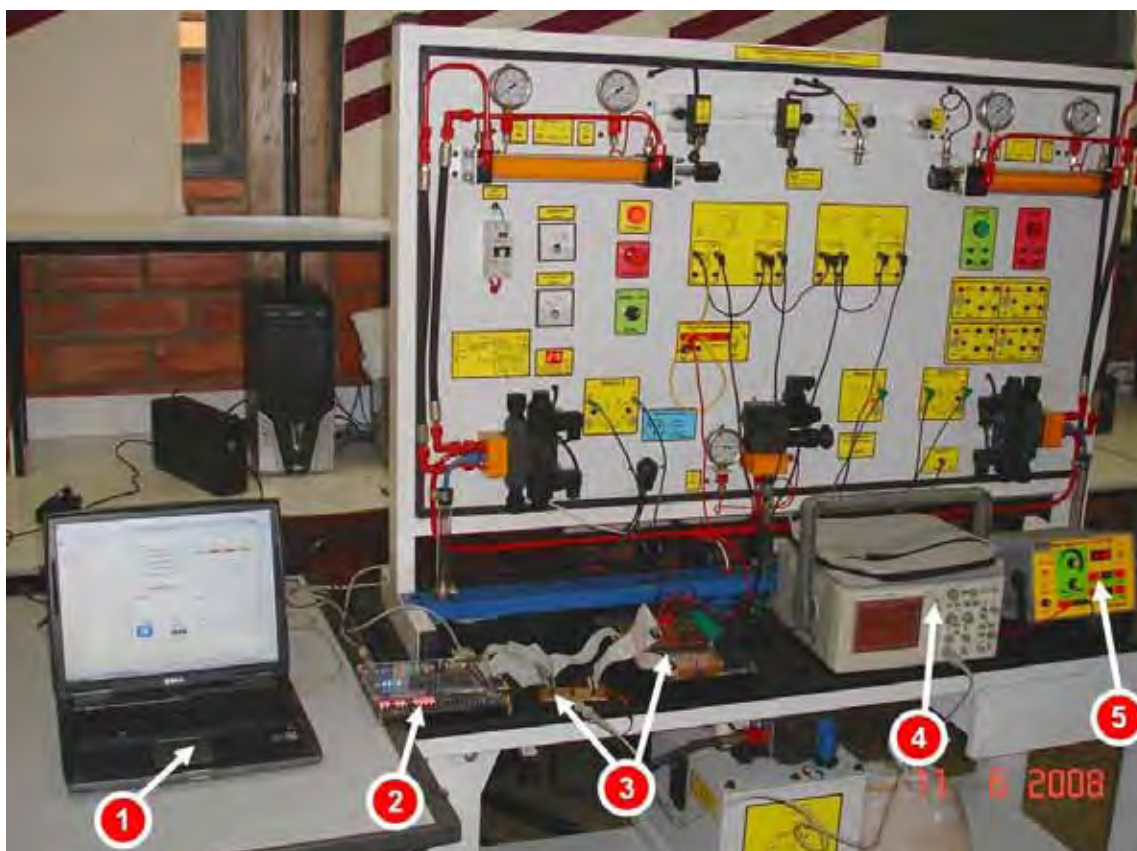


Figura 5.55: Arquitetura utilizada na automação do *MHI-01*.

5.3 Automação do Processo de Filtragem de Biogás

Este processo de filtragem do Biogás tem como principal objetivo tornar este gás com a composição mais próxima possível do gás natural. O biogás utilizado para filtragem é oriundo de um biodigestor da granja de suínos da fazenda Bedin, localizada na cidade de São Gabriel do Oeste – Mato Grosso do Sul. O gás sai do biodigestor passa por um conjunto de filtros e posteriormente é armazenado. Durante o processo de filtragem monitora-se a pressão e a vazão em diferentes pontos da linha, além de controlar as atividades de um compressor. O processo de filtragem deve ser automatizado para tornar o sistema mais preciso e desta forma atender normas técnicas. Na Figura 5.56 mostra-se uma arquitetura do processo de filtragem de Biogás.

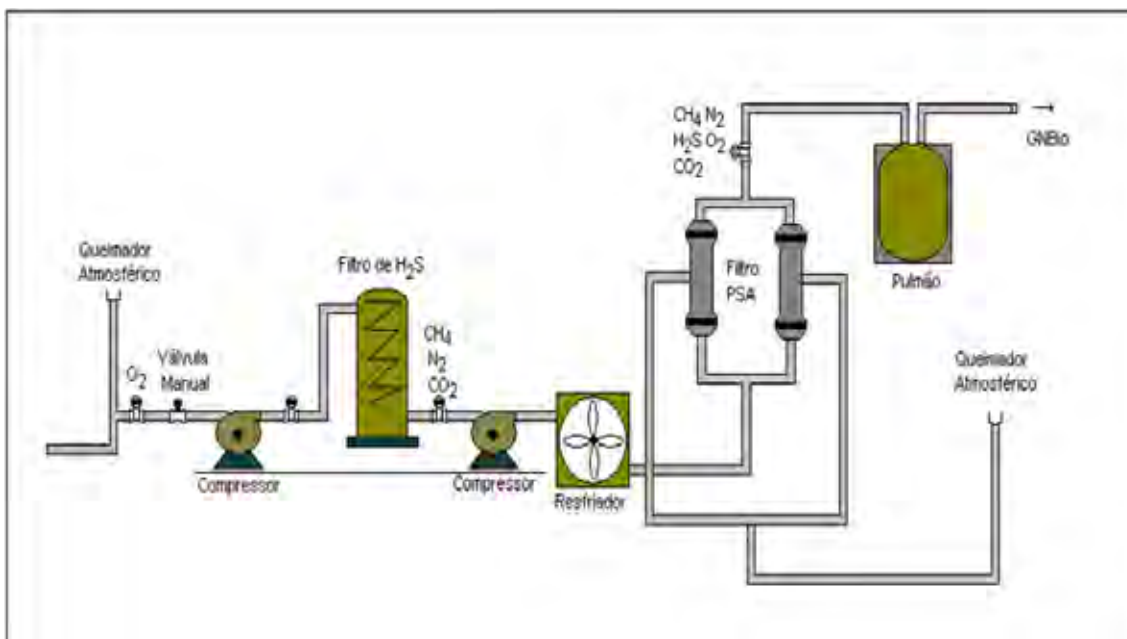


Figura 5.56: Configuração do processo de Filtragem do Biogás.

Para testar a plataforma *IEEE 1451* na automação do processo de filtragem do Biogás, inicialmente, realizou-se o desenvolvimento de um protótipo. Neste protótipo monitorou-se a pressão e a vazão, além de ligar e desligar o compressor. Os equipamentos utilizados no protótipo são os mesmos que formam a linha de filtragem e os parâmetros de monitoramento são compatíveis com os exigidos no processo. Os testes realizados no protótipo serviram para ajustar o funcionamento (lógica de controle e circuitos eletrônicos) da plataforma *IEEE 1451*. Em seguida a plataforma foi inserida na automação do processo em escala real.

Na automação do protótipo o operador aciona o compressor dando início ao monitoramento da pressão e vazão. Com a variação do fluxo de ar absorvido pelo compressor varia-se a pressão e a vazão. Quando a pressão atingir $300\text{mmH}_2\text{O}$ o compressor é desligado. Na Figura 5.57 apresenta-se a lógica de controle utilizada na automação do protótipo, sendo que nesta etapa a vazão não influenciou no funcionamento do sistema.

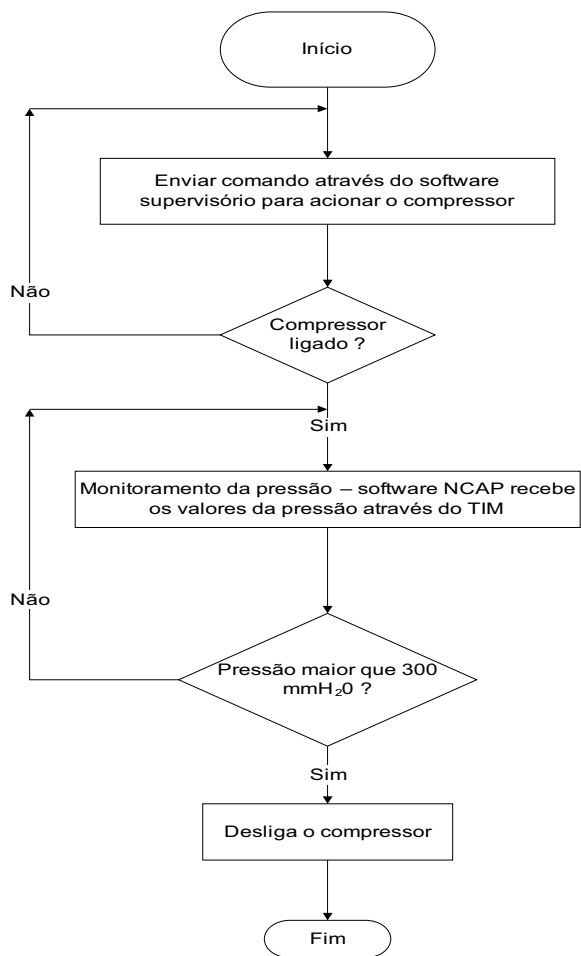


Figura 5.57: Lógica de controle para automação - Filtragem de Biogás.

Foram utilizados os seguintes equipamentos para desenvolver as atividades deste protótipo: um manômetro com saída de 4 a 20 mA que pode medir em uma faixa de 0 a 300 mmH_2O , um medidor de vazão com escala de cm^3/h e um compressor da *Weg S/A*, que possui uma potência de 1 CV e deve ser alimentado com 220 V trifásico. Na Figura 5.58 apresentam-se os principais equipamentos utilizados neste protótipo.

Figura 5.58: Equipamentos que compõe parte do processo de Filtragem do Biogás.

As especificações do medidor de vazão são apresentadas na Tabela 5.9, com base no manual do fabricante.

Tabela 5.9: Especificações do medidor de Vazão.

Características	Desempenho
Faixa de Temperatura	-40°C à + 60°C
Vazão máxima de trabalho	100 m ³ /h
Máxima pressão de operação	12 bar
Máxima velocidade de operação	2350 rpm

O medidor de pressão utilizado neste protótipo realiza uma faixa de 0 a 500 *mmH₂O*, sendo que a descrição deste equipamento fornecida pelo fabricante pode ser analisada na Tabela 5.10.

Tabela 5.10: Especificações do medidor de Pressão.

Características	Desempenho
Faixa desde vácuo até 1600 bar	Precisão ± 0.1 do <i>span</i>
Sensor Piezoresistivo	Temperatura dos invólucros 0-69°C
Saída de 4 – 20 mA ajustável via frontal em 98% da faixa	Saída

Para realizar o monitoramento da vazão e da pressão utilizou-se 16 pinos de entrada, sendo que 8 bits para o transmissor de pressão e 8 bits do medidor de vazão,

além de um canal de saída utilizado para ligar/desligar o compressor. Neste monitoramento foi necessário implementar blocos lógicos em *VHDL* para viabilizar a contagem e tratamento dos pulsos fornecidos pelos sensores. Na Figura 5.59 apresenta-se os hardwares descritos em *VHDL* que compõem o *DriverTIM*, sendo que os blocos lógicos, *calc*, *med* e *temp* proporcionam o tratamento de sinal fornecido pelo medidor de vazão.

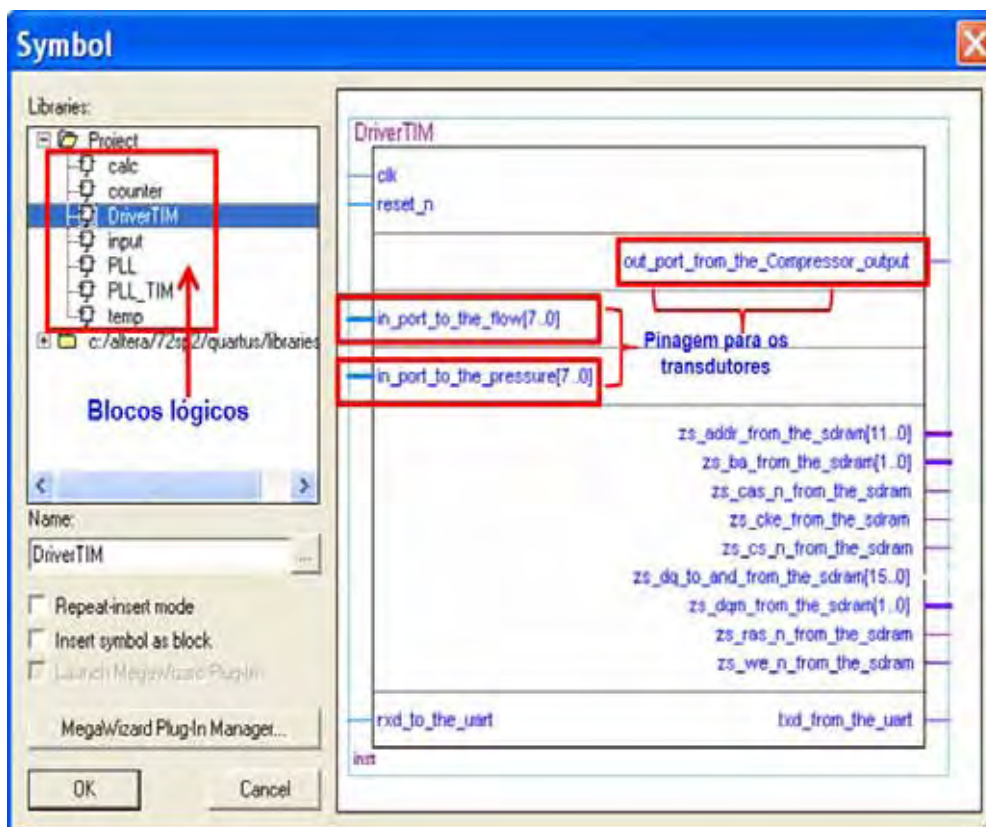


Figura 5.59: *DriverTIM* utilizado para medir pressão e vazão.

Na verificação-se os recursos utilizados pelo *TIM* composto de 2 canais de entrada e 1 canal de saída.

Tabela 5.11: Recursos utilizados pelo *TIM* - processo de Filtragem de Biogás.

<i>Logic Cell</i>	<i>Dedicated Logic Register</i>	<i>Memory Bit</i>	Total de Pinos
3467, 10%	1946, 6%	46720, 10%	88, 19%

5.3.1 Circuitos de Condicionamento de Sinais - Filtragem de Biogás.

Para viabilizar o tratamento de sinais entre os transdutores e o módulo embarcado (placa *DE2*) desenvolveram-se placas de circuitos impressos, conforme a necessidade de cada equipamento.

O sinal de saída do medidor de vazão é diretamente proporcional à tensão de alimentação, ou seja, conforme a tensão que o medidor é alimentado, esta será a amplitude do sinal de saída. Para cada $0,1\text{m}^3$ de vazão que é detectado pelo medidor gera-se um pulso negativo que decai a tensão do valor atual para zero. Assim, ao alimentar o medidor com 24V e após ter detectado que passou $0,1\text{m}^3$ de volume, o mesmo faz com que a tensão desça momentaneamente à zero e logo volte aos 24V. Para eliminar possíveis ruídos, colocou-se na saída do medidor de vazão um *buffer schmitt trigger* (CI 40106b). Como o Medidor de Vazão possui uma limitação de corrente de 10mA foi necessário colocar um resistor na entrada da alimentação para que esta corrente não ultrapasse 10mA . No caso do projeto, foi colocado um resistor de 2400Ω , pois a alimentação é 24V.

O sinal entra no *schmitt trigger* com amplitude máxima de 5V, sendo este valor limitado pelo chaveamento do transistor BC548, ou seja, quando o transistor recebe na base 24V oriundo do medidor de vazão, a tensão na entrada do *schmitt trigger* é de 5V e desta forma o módulo embarcado recebe o sinal de 0 a 5V. Como a saída do transmissor de pressão é de 4 a 20mA , desenvolveu-se circuitos de condicionamentos de sinais, de modo que em 0 bar o equipamento fornece 4mA e em 15bar , o equipamento fornece 20mA , e os valores correspondentes nesta faixa são proporcionais. Estes circuitos eletrônicos proporcionam a transformação de corrente em tensão e posteriormente convertem o sinal analógico para digital. O circuito para conversão de corrente para tensão foi desenvolvido basicamente com amplificador operacional, especificamente com o CI 741. Neste circuito a corrente entra na porta inversora de um amplificador operacional e através da relação de resistências obtém um valor de tensão (negativo) no terminal de saída. Este sinal de saída é inserido em outro amplificador operacional, que tem como principal função inverter para positivo o valor da tensão, ou seja, o circuito é a conexão em cascata de duas configurações inversoras (amplificador operacional). Na Figura 5.60 mostra-se a placa de circuito impresso para a conversão de corrente para tensão.

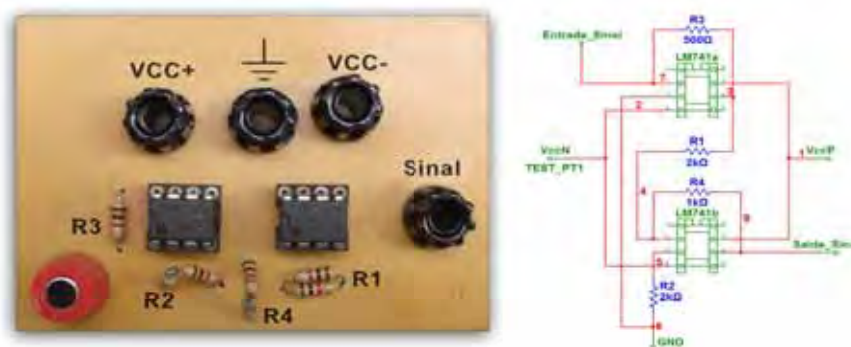


Figura 5.60: Placa de circuito impresso e o esquema elétrico para conversão de corrente para tensão.

Na Tabela 5.12 verificam-se os valores correspondentes de corrente (entrada) e tensão (saída) medidos com base na Figura 5.60.

Tabela 5.12: Conversão de corrente para tensão.

Corrente (mA)	Tensão (V)
4	1,001
5	1,251
6	1,501
7	1,751
8	2,001
9	2,251
10	2,501
11	2,751
12	3,001
13	3,251
14	3,501
15	3,751
16	4,001
17	4,251
18	4,501
19	4,751
20	5,001

Após a conversão de corrente ($4\text{-}20\text{mA}$) para tensão ($1\text{-}5\text{V}$), este sinal foi convertido de analógico para digital. Na Figura 5.61 mostra-se a placa de circuito impresso que viabiliza a conversão de sinal analógico para digital utilizada nesta aplicação.

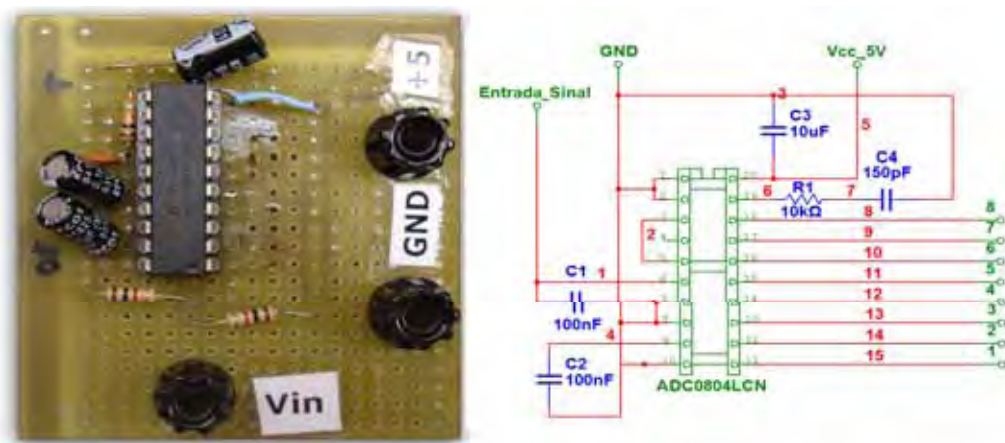


Figura 5.61: Placa de circuito impresso para conversão de sinais *A/D* com o respectivo esquema elétrico.

A conversão de sinal analógico para digital foi implementada com um *CI ADC0804*, no qual a tensão máxima de entrada é de 5V.

Como a bobina da contatora aciona com 127 V, utilizou-se um relé de 28-127 V e por medida de segurança isolou-se os sinais da placa *DE2* do circuito eletrônico através do *optoacoplador (CI H11D2)*.

5.3.2 Automação do protótipo de Filtragem do Biogás

Para testar a automação do protótipo do processo de filtragem do Biogás montou-se uma configuração como pode-se observar na Figura 5.62.

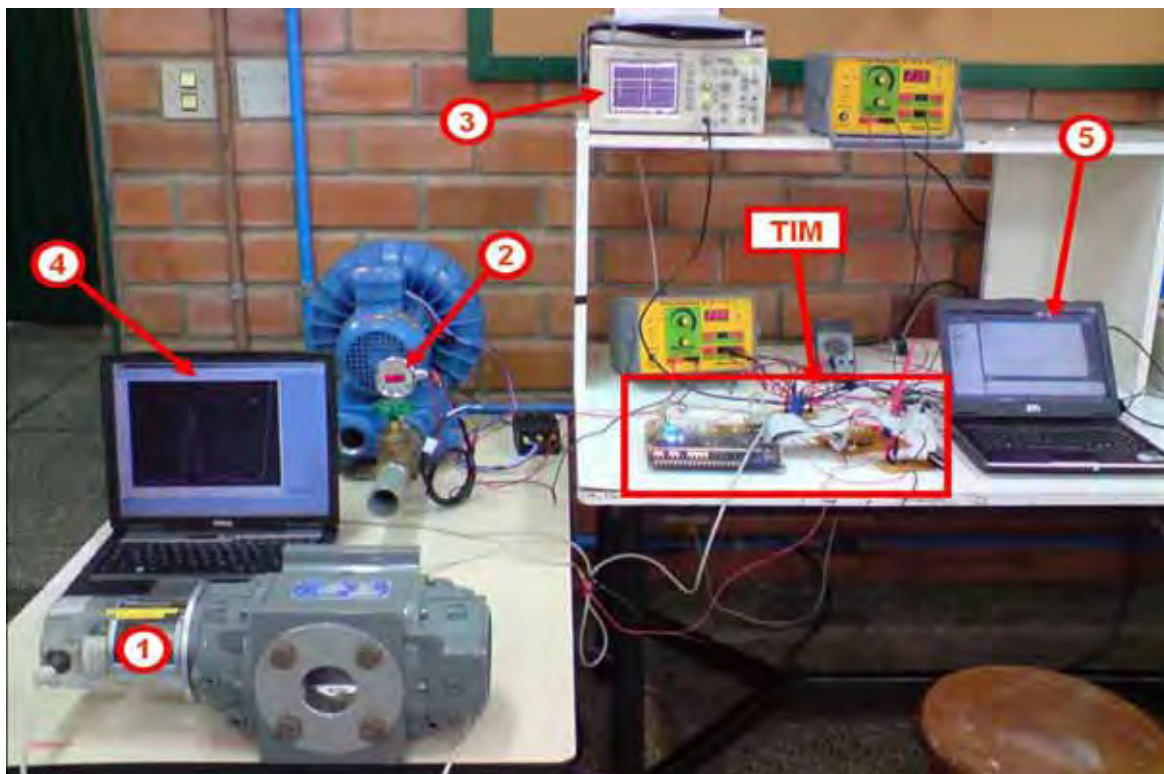


Figura 5.62: Configuração para automação - Filtragem do Biogás.

Sendo que em (Figura 5.62) encontra-se:

- 1) o medidor de vazão.
- 2) o transmissor de pressão.
- 3) e os sinais do medidor de vazão.
- 4) o gráfico de monitoramento da pressão solicitado por um operador (cliente).
- 5) o *PC* (servidor) responsável por implementar as funcionalidades do *NCAP*.

Observação: As fontes de alimentação que aparecem na Figura 5.62 são para alimentar os *CI*s da placa de circuito de condicionamento e conversão de sinais, a placa *DE2*, o transmissor de pressão e o medidor de vazão.

Com a configuração apresentada na Figura 5.62 foi possível comparar o valor medido no transmissor de pressão com o valor mostrado pelo *NCAP* (tela do supervisor). Esses valores foram comparados como mostra-se na Figura 5.63.

Figura 5.63: Comparação entre valores obtidos no transmissor com os apresentados no *NCAP*.

Para a correção da medida, observou-se que um gráfico representando a leitura do supervisor em função da leitura real, apresentava um comportamento linear. Assim sendo, a reta foi discretizada em cada ponto de medição e a média dos coeficientes da equação foi feita. Na equação 5.1 realiza-se a correção entre os valores medidos e o apresentado no software *NCAP*.

$$P_{\text{corrigida}} = \frac{V_{\text{recebido}} - 47,0714}{0,6428} \quad (5.2)$$

Sendo, $P_{\text{corrigida}}$ = pressão corrigida e V_{recebido} = valor recebido

Esta equação 5.1 foi devidamente inserida na lógica de controle da aplicação, ou seja, no *TIM* como pode-se observar na Figura 5.64. Optou-se por corrigir este erro via software por ser mais flexível na implementação quando comparado com a correção via hardware.

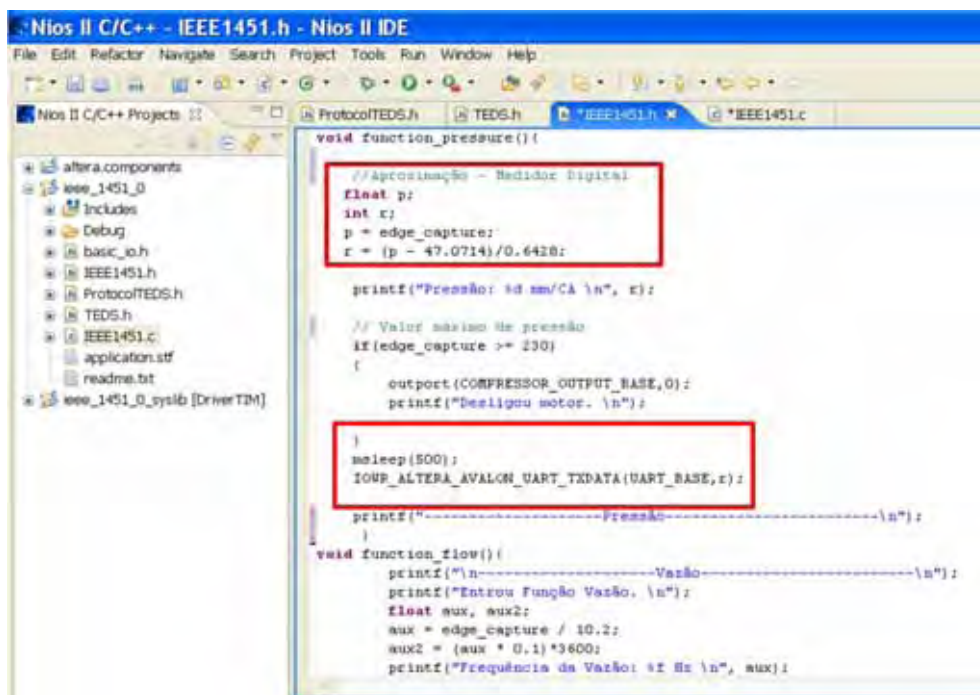


Figura 5.64: Parte da lógica de controle com base na Equação 5.1.

Após a inserção da equação 5.1 na lógica de controle a comparação entre os valores medidos no transmissor de pressão e no *NCAP* foram bem próximos, como observa-se na Figura 5.65. Com a inserção da equação 5.1, o erro máximo foi de 0,66%, sendo esta margem de erro tolerável comparado com os valores de *set point* estabelecido neste processo.

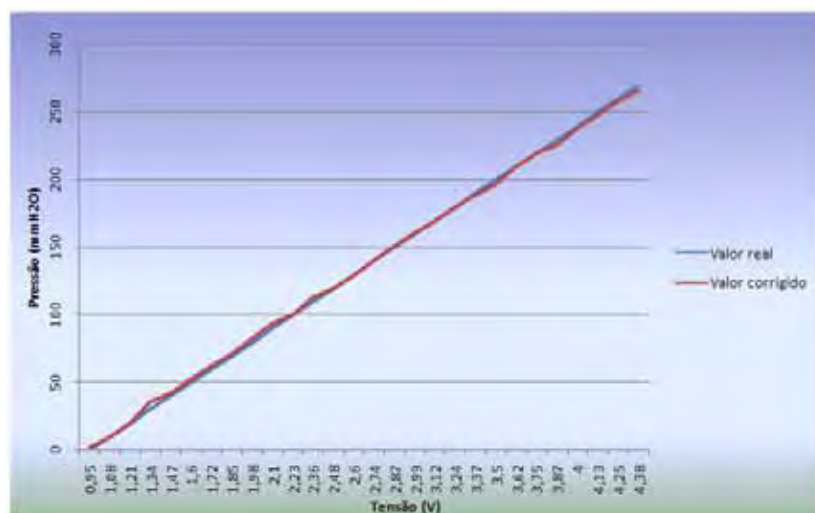


Figura 5.65: Comparação entre valores lidos no transmissor de pressão e os apresentados na tela do *NCAP* com fator de correção.

A solicitação da *TEDS* no *NCAP* pôde gerar a especificação do medidor de vazão utilizado nesta aplicação, como pode ser observado na Figura 5.66.

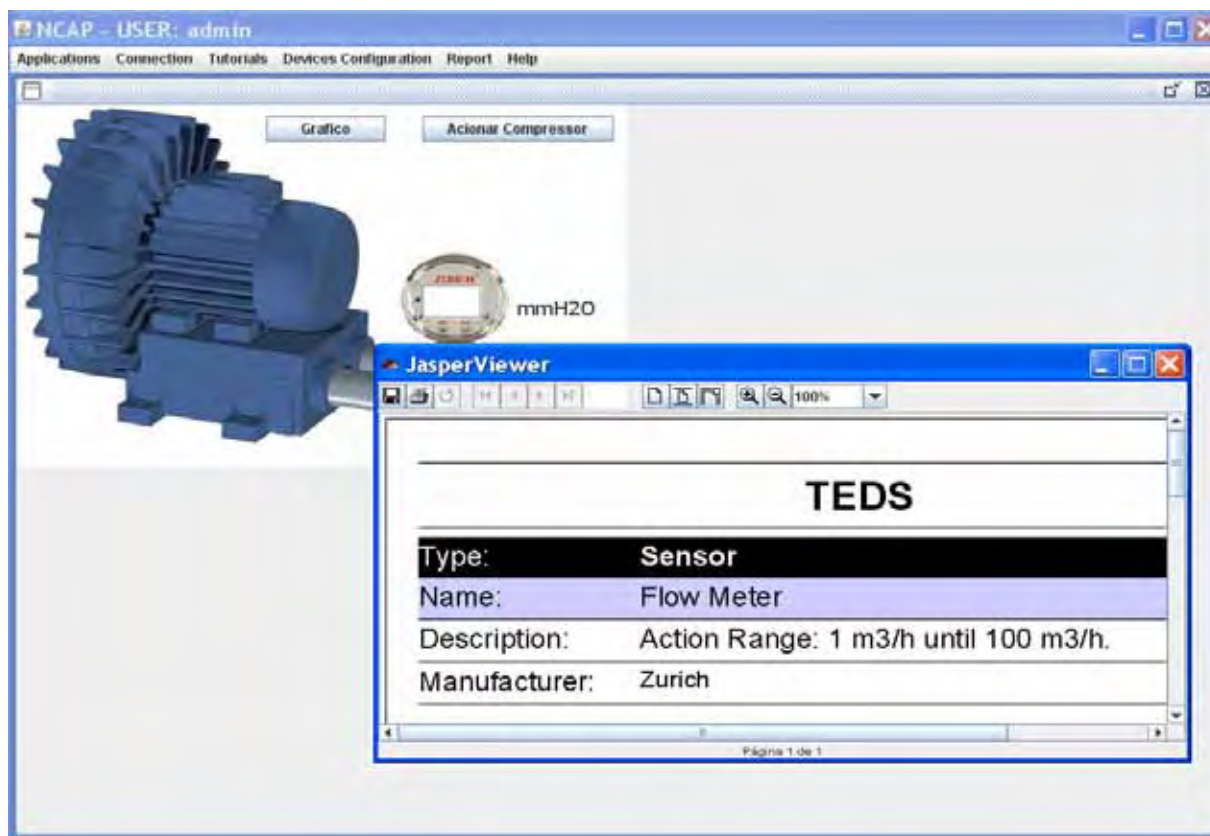


Figura 5.66: Resultado da solicitação da *TEDS* e a tela de supervisão.

A aplicação é iniciada com o acionamento do compressor, em seguida o controle da saída de fluxo do compressor foi realizada manualmente e para cada situação realizou-se a leitura da vazão e da pressão. O sistema funciona até a pressão de $300 \text{ mmH}_2\text{O}$, sendo que neste valor, o compressor desliga-se automaticamente, conforme estabelecido na lógica de controle desta aplicação.

Na Figura 5.67 verificam-se os valores de vazão e pressão coletados, conforme o esquema da Figura 5.62 demonstrados na tela do software supervisor (*NCAP*).

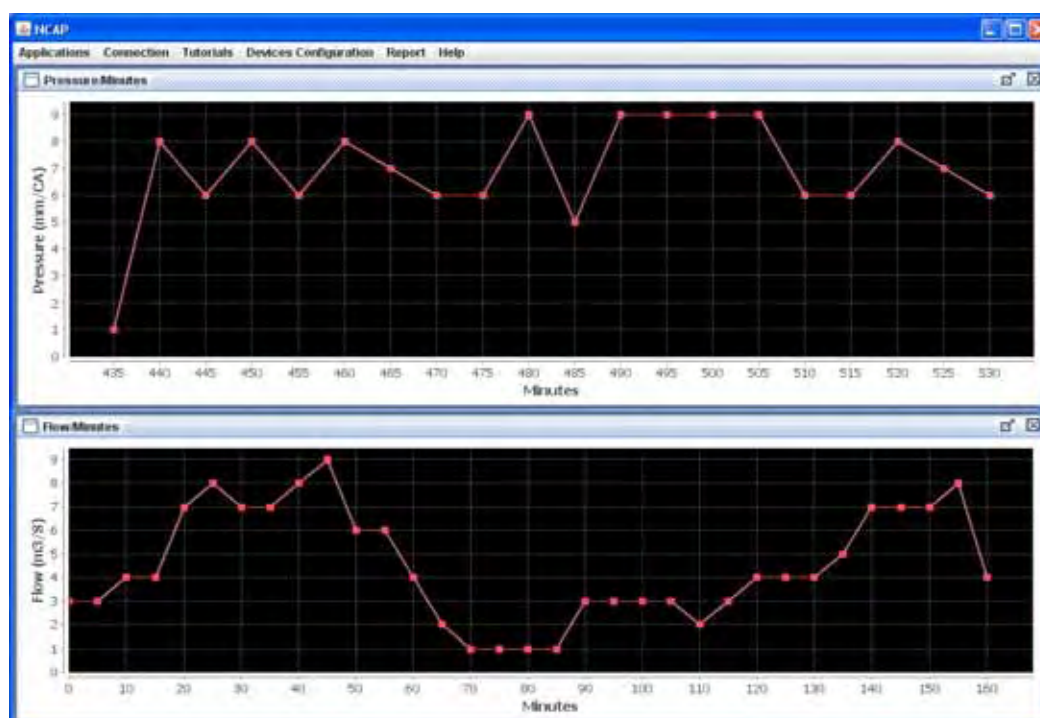


Figura 5.67: Valores coletados sem tempo real de pressão e vazão.

Na Figura 5.68 verifica-se a tela para acionamento do compressor e monitoramento da pressão e na Figura 5.69 mostra-se a interface gráfica para leitura da vazão.

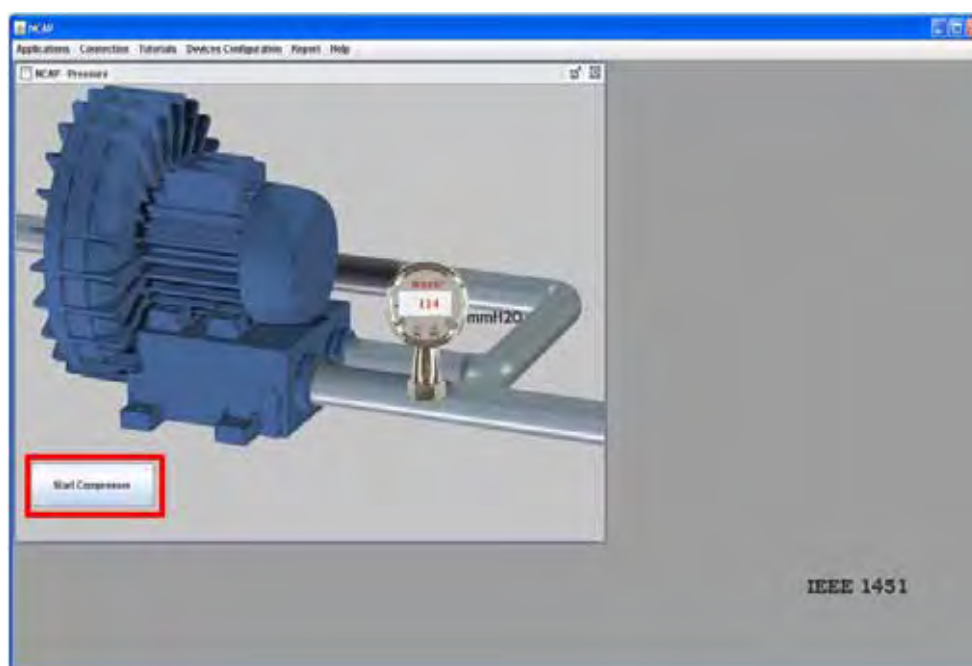


Figura 5.68: Tela para acionamento do compressor e de monitoramento da pressão.

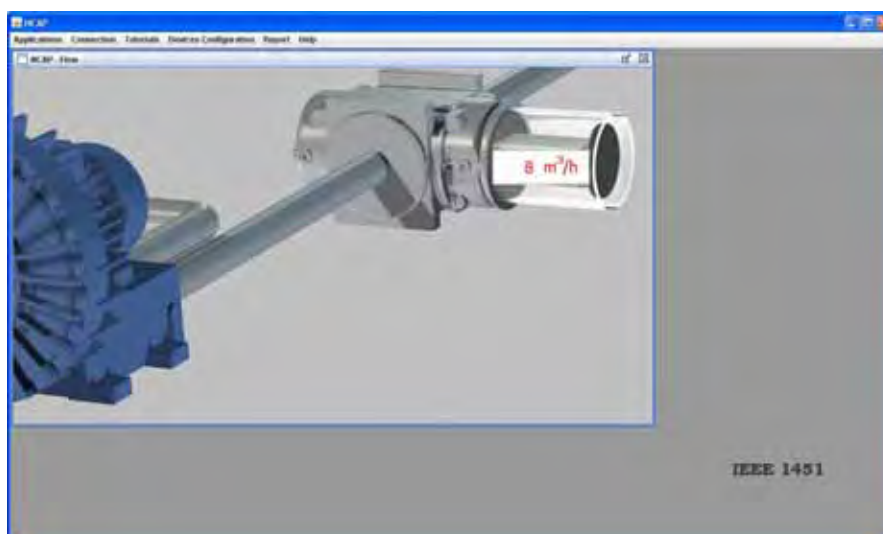


Figura 5.69: Tela de monitoramento da vazão.

5.3.3 Plataforma IEEE 1451 – Processo de Filtragem do Biogás

A rede de transdutores da linha de filtragem do Biogás é composta de dois medidores de vazão, dois manômetros e um compressor. Os manômetros com escala de 0 a 15Bar monitoram a pressão que entra nos filtros. A vazão é monitorada para auxiliar no cálculo da porcentagem de gás que está sendo filtrado conforme a norma técnica. Na Figura 5.70 mostra-se como os medidores estão instalados na linha de filtragem.



Figura 5.70: Linha de Filtragem do Biogás com os medidores instalados.

A saída dos medidores foram conectadas aos circuitos de condicionamento de sinais, que em conjunto com o *DriverTIM*, formam o módulo para transdutores da plataforma *IEEE 1451*. Na Figura 5.25 observa-se os circuitos de condicionamento de sinais e a placa *DE2* utilizados na automação do processo de filtragem do Biogás.

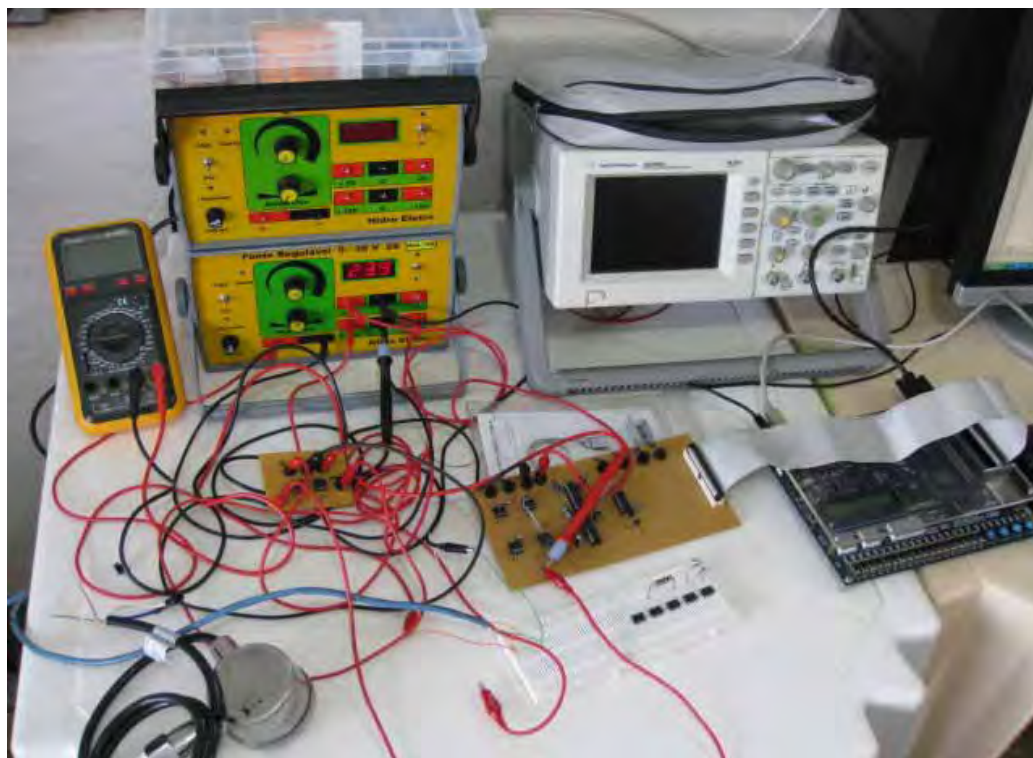


Figura 5.25: *TIM* para automação da Filtragem do Biogás

A placa de circuito de condicionamento confeccionada para esta aplicação pode ser observada em detalhes da Figura 5.71. Nesta Figura 5.71 a alimentação simétrica (-15V e 15V) é utilizada para os amplificadores operacionais (conversão da corrente para tensão) e a de 5V alimenta os conversores de sinais (A/D). A saída dos medidores são conectadas na placa conforme a identificação e os valores são enviados para a placa *DE2* através de um cabo *flat* de 40 vias.

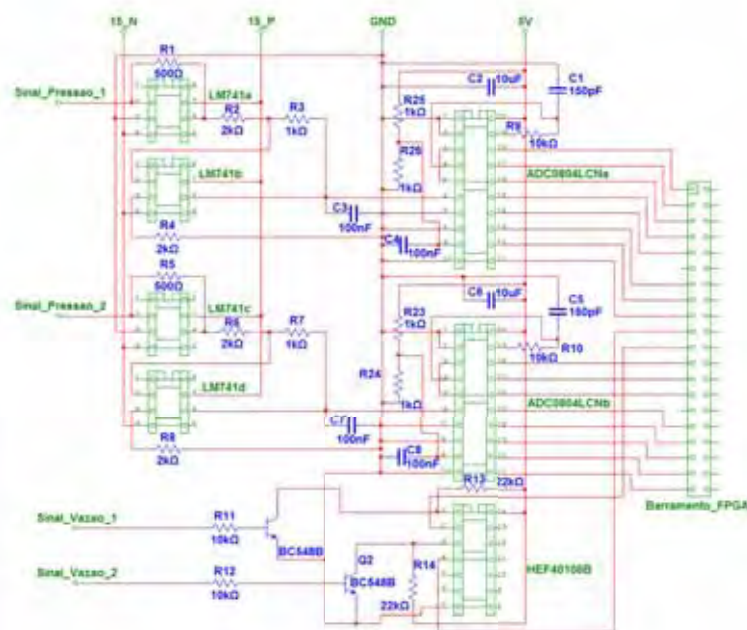
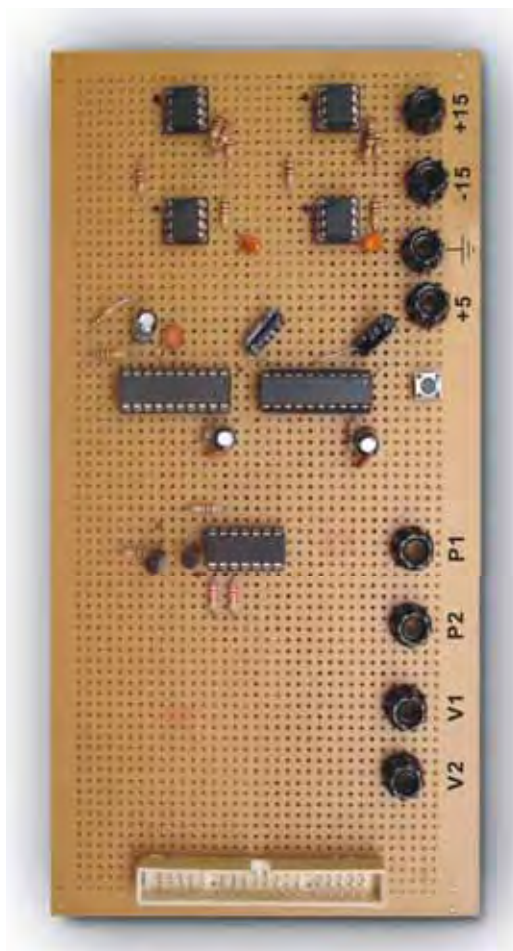


Figura 5.71: Placa e esquema elétrico para tratamento de sinais da rede de transdutores do processo de Filtragem do Biogás.

A tela do software *NCAP* para processo de filtragem foi modificada quando comparada com as utilizadas na automação do protótipo. As modificações foram realizadas para atender as necessidades de monitoramento do processo. Na Figura 5.72 observa-se a tela desenvolvida para a automação do processo de filtragem do Biogás.

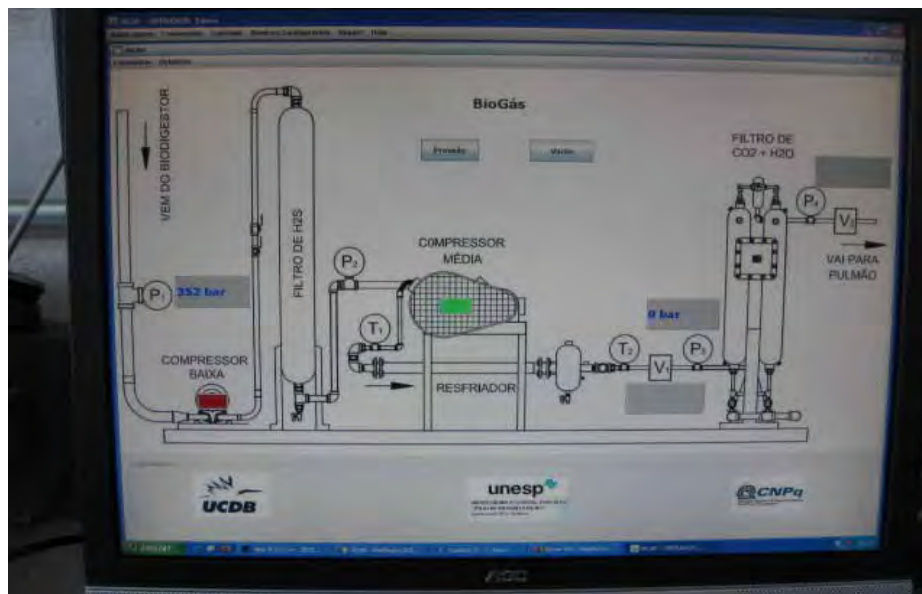


Figura 5.72: Tela do software *NCAP* para a automação do processo de Filtragem do Biogás

5.4 Comunicação do Software Eclipse com o DriverTIM

O *DriverTIM* foi testado também na automação de um kit de pneumática da Hidro Eletro Industrial, no qual inseriu-se a lógica de operação do processo de Filtragem do Biogás. Nesta aplicação utilizou-se o software *Eclipse E3* da *Eclipse Software* como software supervisor. Para que o software supervisor comunice-se com o *DriverTIM* foi realizada a configuração do *DriverASCII*, cuja descrição de funcionamento foi cedida pela empresa *Eclipse Software*. Com estas informações, foi possível configurar a comunicação do software supervisor conforme desejado. Na Figura 5.27, as variáveis *tag1* e *tag2* são elementos de escrita, isto é, são responsáveis por enviar o comando para o *DriverTIM*. No *DriverTIM*, este sinal é processado e enviado para o circuito eletrônico que consequentemente atua no kit de Pneumática, conforme a lógica da automação. Os resultados obtidos nesta aplicação demonstrou que o *DriverTIM* pode ser utilizado com diferentes softwares supervisórios e um sistema de comunicação pioneiro para empresa *Eclipse Software*. No Anexo A, apresenta-se mais informações sobre o uso do *DriverTIM* com software *Eclipse SCADA*.

5.5 Análise de desempenho dos módulos sem fio utilizados para implementar as interfaces IEEE 1451.5

No padrão *IEEE 1451.5* destaca-se a possibilidade de implementar uma interface sem fio utilizando *Wi-Fi*, *Bluetooth* e *ZigBee*. O projetista pode optar por uma das tecnologias e dentro destas possibilidades, definir qual módulo será utilizado. Neste trabalho selecionaram-se módulos sem fio de cada uma das tecnologias, sendo que estes possuem características semelhantes, conforme especificado pelos fabricantes. As principais características analisadas na seleção dos módulos foram o alcance da transmissão e a interface de comunicação. Conforme descrito no Capítulo 2, os módulos selecionados foram *INT550 ME* da *Integral* (*Wi-Fi*), *BTS1009C* da *Sunix* (*Bluetooth*) e o *XBee* da *MaxStream* (*ZigBee*). Estes módulos (um par de cada tecnologia sem fio) possuem interface de comunicação *RS232* e segundo as informações dos fabricantes, podem enviar e receber dados a uma distância de 100 metros entre os dispositivos. Na Figura 5.73 apresenta-se a estrutura básica para testar a comunicação entre os dispositivos sem fio.



Figura 5.73: Módulos utilizados para implementar a interface *IEEE 1451.5*.

Para cada aplicação testada com a plataforma *IEEE 1451*, realizou-se uma configuração com os módulos sem fio e por cabeamento, permitindo a implementação das interfaces *IEEE 1451.2* e *IEEE 1451.5*. Porém, com intuito de fornecer parâmetros aos projetistas que desejam implementar uma interface *IEEE 1451.5*, desenvolveu-se

neste trabalho um diagnóstico sobre o desempenho de cada tecnologia sem fio. Nos testes analisaram-se a capacidade de transmissão em diferentes distâncias, com linha de visada² e sem linha de visada³. A análise da eficiência da transmissão sem linha de visada é interessante para aplicações industriais, residenciais, segurança, saúde, etc. Com linha de visada pode ser atrativo para automação agrícola, monitoramento de desempenho de atletas e robô móvel.

A eficiência da comunicação foi analisada com auxílio do software *X-CTU* da *Digi* (vide Anexo A) e através da plataforma *IEEE 1451*. O software *X-CTU* possibilita a configuração dos dispositivos *Xbee* e avalia o desempenho de módulos conectados na interface *RS232* como pode ser observado na Figura 5.74. Neste software é possível entrar com a informação a ser transmitida e se a comunicação for realizada com sucesso, o dispositivo transmissor recebe de volta a mensagem enviada. Com base neste tipo de operação, o software avalia quantas mensagens foram enviadas e recebidas com sucesso, informa os dados perdidos e a porta utilizada na comunicação. O resultado da avaliação no *X-CTU* é dado em porcentagem, de acordo com o número de tentativas realizadas com sucesso.

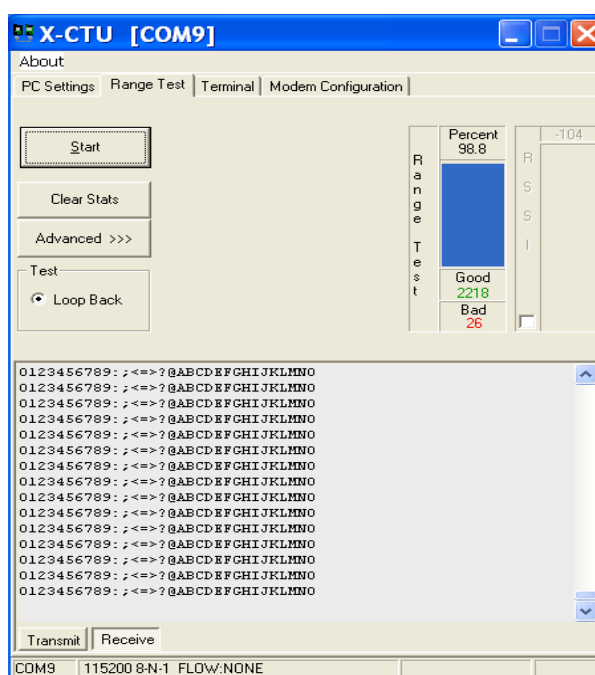


Figura 5.74: *X-CTU* parâmetros para analisar a eficiência da transmissão.

² Com linha de visada: Sem obstáculos entre os dispositivos.

³ Sem linha de visada: Com obstáculos entre os dispositivos (neste caso, piso de concreto).

5.5.1 Testes das interfaces IEEE 1451.5 com linha de visada

Nos testes realizados com linha de visada, modificou-se a distância em 5, 10, 20, 30, 45, 60 e 75 metros para os módulos. Com auxílio do software *X-CTU* enviou-se diversas vezes uma mesma informação e analisou-se a eficiência desta transmissão. Na Tabela 5.13, apresenta-se os resultados dos testes com linha de visada para o módulo *BTS1009C* (tecnologia *Bluetooth*) com taxa de transmissão de 115200. Os módulos *BTS1009C*, após a configuração, buscam a conexão entre mestre e escravo, vide Anexo A. A conexão entre os módulos *BTS1009C* é realizada quando estão na mesma área de alcance.

Tabela 5.13 : Análise da transmissão do módulo *BTS1009C* com linha de visada.

Distância	Dados Enviados com Sucesso	Pacote perdido	Eficiência da Transmissão
10 (metros)	1220	0	100%
20 (metros)	1876	0	100%
30 (metros)	1373	0	100%
45 (metros)	1139	0	100%
60 (metros)	1106	0	100%
75 (metros)	1178	0	100%

Na Tabela 5.14 mostra-se os resultados da transmissão dos módulos *Xbee* (tecnologia *ZigBee*) com linha de visada e com taxa de transmissão de 115200.

Tabela 5.14: Análise da transmissão do módulo *Xbee* com linha de visada.

Distância	Dados Enviados com Sucesso	Dados perdidos	Eficiência da Transmissão
10 (metros)	1045	2	98,9%
20 (metros)	1213	25	98%
30 (metros)	1178	39	96,8%
45 (metros)	1483	46	96,9
60 (metros)	1086	45	96%
75	1221	109	91,8%

Na Tabela 5.15, observa-se os resultados dos testes com o módulo *INT 550 ME* (tecnologia *Wi-Fi*) com linha de visada para diferentes distâncias.

Tabela 5.15: Análise da transmissão do módulo *INT550 ME* com linha de visada.

Distância	Dados Enviados com Sucesso	Dados perdidos	Eficiência da Transmissão
10 (metros)	1223	0	100%

20 (metros)	1055	0	100%
30 (metros)	1003	0	100%
45 (metros)	1367	0	100%
60 (metros)	1098	0	100%
75 (metros)	1338	0	100%

5.5.2 Testes das interfaces IEEE 1451.5 sem linha de visada

Nos testes realizados sem linha de visada, optou-se por colocar um módulo no térreo e outro módulo no piso superior, sendo estes separados por concreto. Nesta etapa variou-se a distância e analisou-se a eficiência da transmissão. Na Tabela 5.16, verifica-se os resultados dos módulos *BTS1009C* sem linha de visada. Com a distância de 80 metros entre os módulos *BTS1009C* constatou-se que ocorreu um tempo muito maior para que estes módulos se encontrassem (conexão mestre-escravo).

Tabela 5.16: Análise da transmissão do módulo *BTS1009C* sem linha de visada.

Distância	Dados Enviados com Sucesso	Dados perdidos	Eficiência da Transmissão
5 (metros)	1128	0	100%
20 (metros)	1004	0	100%
35 (metros)	1274	0	100%
50 (metros)	1251	0	100%
80 (metros)	1853	41	97,8%

Na Tabela 5.17, encontram-se os resultados do desempenho do módulo *Xbee* na transmissão sem linha de visada, sendo que, a partir de 35 metros não foi possível realizar a comunicação entre os módulos *Xbee*.

Tabela 5.17: Análise da transmissão do módulo *Xbee* sem linha de visada.

Distância	Dados Enviados com Sucesso	Dados perdidos	Eficiência da Transmissão
5 (metros)	1107	19	98,3%
20 (metros)	1148	417	73,6%
35 (metros)	0	1267	0%

Na Tabela 5.18 pode-se observar a eficiência da transmissão dos módulos INT 550 ME sem linha de visada.

Tabela 5.18: Análise da transmissão do módulo *INT 550 ME* sem linha de visada.

Distância	Dados Enviados com Sucesso	Dados perdidos	Eficiência da Transmissão
5 (metros)	3170	0	100%

20 (metros)	1402	0	100%
35 (metros)	1055	0	100%
50 (metros)	1106	0	100%
80 (metros)	1903	27	97,6%

Cada módulo sem fio utilizado para implementar a interface *IEEE 1451.5*, apresentam suas particularidades que podem ser vantajosas ou não, dependendo da aplicação. O módulo *Xbee* foi o de menor custo e o *INT 550 ME* o mais caro. Porém, o *INT 550 ME* pode conectar-se em rede sem fio, tornando-o muito mais versátil em relação à comunicação que os demais módulos. O *INT550 ME* foi utilizado na automação da casa de vegetação da fazenda escola da UCDB. Neste experimento configuraram-se os módulos para operar com a Internet sem fio e monitorou as atividades da casa de vegetação com uma distância de aproximadamente 150 metros.

Os módulos *BTS1009C* e *Xbee* são configurados com maior agilidade e sem a necessidade de acessórios, ou seja, basta um computador e o software de instalação. Na configuração da tecnologia *Wi-Fi (INT550 ME)*, além do computador para executar o software é necessário um *Acess Point*. O *INT550 ME* possui suporte para comunicação via interface *RS485* e *Ethernet*, por isso, é maior em tamanho quando comparado com *Xbee* e o *BTS1009C*.

Na especificação do fabricante, os módulos *Xbee* podem trocar informações a uma distância de 100 metros com linha de visada. Nos testes realizados neste trabalho comprovaram que estes módulos conseguem trocar informações com a distância de 100 metros, porém com uma taxa de perda. Esta perda apesar de ser pequena (cerca de 3,5%) é uma desvantagem em relação aos módulos *BTS1009C* que para a mesma distância (100 metros) obteve-se 100% na eficiência da transmissão. Ou seja, os testes com os módulos *BTS1009C* comprovaram, conforme as especificações do fabricante que a uma distância de 100 metros pode-se realizar a troca de informação entre os mesmos. Os módulos *INT550 ME* quando configurados no modo *peer-to-peer* trocou informações sem perda até uma distância de 50 metros, sem linha de visada, porém quando configurado para operar com Internet sem fio estes dispositivos se comunicaram até aproximadamente 175 metros.

CAPÍTULO 6 - Conclusões e Trabalhos Futuros

Os resultados obtidos neste trabalho demonstraram que as tecnologias utilizadas no desenvolvimento da plataforma *IEEE 1451* são suficientemente robustas para ser usada em ambiente industrial e está em conformidade com o padrão *IEEE 1451*.

Estas tecnologias foram devidamente selecionadas por apresentarem características como, popularidade, gratuidade, realizam a computação reconfigurável e são padronizadas. A plataforma *IEEE 1451* baseia-se no sistema composto por software supervisor e hardware de controle, sendo que, neste caso o software supervisor desenvolve as funções da parte lógica do *NCAP* e o hardware de controle realiza as atividades do *TIM*.

Neste trabalho, inicialmente a parte lógica do *NCAP* foi definida como *SAF* (Software para Automação com *FPGA*), porém posteriormente para melhor identificação da plataforma, o software supervisor foi denominado de *NCAP*. O software *NCAP* possui hierarquia na navegação de telas, geração de relatório (*TEDS*), segurança na acessibilidade das informações e atividades remotas. Nos diferentes testes realizados podem-se gerar telas, gráficos e relatórios específicos, ou seja, de acordo com a aplicação, demonstrando a flexibilidade deste *NCAP*.

No *NCAP* da plataforma *IEEE 1451* o operador seleciona a interface que deve ser utilizada, ou seja, *IEEE 1451.2* ou *IEEE 1451.5*, sendo que as conexões físicas devem estar devidamente realizadas. Ao selecionar a interface *IEEE 1451.5* o operador pode configurar os dispositivos por meio de tutoriais disponíveis no *NCAP*. Esses tutoriais estão de acordo com as tecnologias definidas pelo padrão *IEEE 1451.5*. Como a parte lógica do *NCAP* envia e recebe dados pela interface *RS232* o hardware de controle (*TIM*) pode ser desenvolvido em um microcontrolador ou *CLP*, desde que estes módulos estejam configurados para executar o protocolo de comunicação com o *NCAP*.

Como as funcionalidades do *NCAP* podem ser exercidas via internet, existe a possibilidade de integrar nesta plataforma, clientes que possuem dispositivos móveis dotados da máquina virtual Java, aumentando o leque de atuação deste sistema.

O *TIM* da plataforma *IEEE 1451* compõe-se em duas partes, o *DriverTIM* descrito em *VHDL* no qual executa-se as atividades lógicas da aplicação e os circuitos de condicionamentos de sinais que viabilizam a integração deste módulo embarcado com a rede de transdutores. Pode-se destacar no desenvolvimento do *TIM*, a utilização

do *SOPC Builder*, pois esta ferramenta possui elementos que favorecem a construção de um hardware de forma modular e conforme as necessidades do projetista, ou seja, reduz o tempo de desenvolvimento de módulos embarcados. Outro aspecto atrativo no desenvolvimento deste *TIM* está na utilização do processador *NIOS II*, que possibilita a implementação da lógica de controle em linguagem *C/C++*.

O desenvolvimento do *template IEEE 1451* para auxiliar na programação do processador *NIOS II*, proporciona a popularização de conceitos do padrão *IEEE 1451*, além de fornecer um conjunto de bibliotecas e funções para tratamento de interrupção, comunicação *RS232* e construção da *TEDS*. Desta forma, o projetista descreve a *TEDS* através da programação do processador *NIOS II* e quando solicitado no *NCAP* estas informações aparecem junto ao relatório. Assim, na plataforma *IEEE 1451* o projetista desenvolve a lógica de controle e a *TEDS* em um mesmo ambiente de programação.

Na geração do hardware *DriverTIM* para diferentes aplicações, observou-se que a inserção de canais transdutores é flexível em relação ao tipo de canal, digital ou analógico, ficando limitado pela capacidade do *FPGA* e a placa utilizada. A configuração dos periféricos para compor um hardware favoreceu a implementação de módulo embarcado personalizado e conseqüentemente com maior aplicabilidade. Desta forma, elaborou-se um *DriverTIM* básico, contendo periféricos comuns nas aplicações testadas neste trabalho. Entre estes periféricos básicos do *DriverTIM* o fundamental é a interface *RS232*, pois o tratamento de interrupção é baseado neste sistema de comunicação, além de favorecer a implementação das interfaces *IEEE 1451.2* e *IEEE 1451.5*.

Em relação às funções implementadas no *TIM* estão em conformidade com o padrão *IEEE 1451.5*, porém, existe a possibilidade de desenvolver um *template* para cada interface *IEEE 1451*, demonstrando a expansividade desta plataforma. O *DriverTIM* serviu como base para o desenvolvimento de um *driver* para a comunicação com o software supervisor *ELIPSE E3* da empresa *ELIPSE SCADA*, na qual, implementou-se a automação de um kit de pneumática.

Nos testes realizados com a plataforma *IEEE 1451*, pode-se constatar além da automação do processo, a flexibilidade de implementar canais transdutores e a quantidade de recursos utilizados do *FPGA* da família *Cyclone II*. Os diferentes *DriverTIMs* desenvolvidos de acordo com a aplicação ocuparão cerca de 15% da capacidade de células lógicas e 47% dos bits de memória, ou seja, mesmo tratando-se de

um módulo didático, observa-se a possibilidade de aumentar o número de canais transdutores.

Nos testes realizados com os módulos para interface *IEEE 1451.5*, pode-se observar que a tecnologia *ZigBee* apresentou uma perda de aproximadamente 8% na transmissão de dados com uma distância de 75 metros. Como a especificação do fabricante estabelece uma distância de 100 metros, este módulo não é robusto para aplicação industrial. Os módulos com tecnologia *Wi-Fi* e *Bluetooth* atingiram as especificações dos respectivos fabricantes e não apresentaram perda na transmissão de dados com linha de visada. No caso dos testes sem linha de visada o módulo *ZigBee* utilizado neste trabalho, teve desempenho muito menor que os módulos *Bluetooth* e *Wi-Fi*, portanto inadequados para aplicações neste tipo de ambiente.

Neste trabalho pode-se destacar a versatilidade no desenvolvimento do sistema embarcado e a robustez do hardware reconfigurável em ambiente industrial. Como a utilização de *FPGA* para implementar as funcionalidades do *TIM* é ainda pouco explorada, a plataforma *IEEE 1451*, torna-se uma referência neste segmento. A empregabilidade da tecnologia Java no desenvolvimento do software supervisor possibilitou a reutilização de código em diferentes aplicações e a documentação em diagramas *UML* e/ou *javadoc*.

A plataforma *IEEE 1451* foi utilizada na automação do kit hidráulico (*MHI-01*), protótipo de Casa de Vegetação, sistema de filtragem para Biogás, além da comunicação do *DriverTIM* com software *Eclipse E3*.

A contribuição deste trabalho está relacionada com o desenvolvimento da plataforma *IEEE 1451*, na qual possibilita a implementação das interfaces *IEEE 1451.2* e *IEEE 1451.5* comunicando com *NCAP (IEEE 1451.1)* e gerando a *TEDS (IEEE 1451.0)*. De modo específico neste trabalho pode-se pontuar as seguintes contribuições:

- a) Aplicabilidade de hardware reconfigurável em ambiente industrial e para implementar as normas *IEEE 1451*: Atualmente a maioria das automações industriais são realizadas com *CLPs* e os principais grupos de pesquisas desenvolvem os conceitos do padrão *IEEE 1451* com microcontroladores. Portanto neste trabalho apresentaram-se os recursos de uma tecnologia pouco explorada no cenário industrial.
- b) Desenvolvimento de um firmware: Neste trabalho gerou-se *template* denominado de *IEEE 1451* para implementar os blocos funcionais do

módulo para transdutores (*TIM*). Este template pode ser utilizado em diferentes aplicações e pode servir como base para outras tecnologias, como por exemplo, em microcontroladores. O *template IEEE 1451* favorece a disseminação dos conceitos do padrão *IEEE 1451*, pois poderá ser inserida pelo fabricante no ambiente de programação do processador. Não foram encontrados relatos na literatura de configuração e de funcionalidade como a do *template IEEE 1451*.

- c) Utilização de tecnologia Java para desenvolver as funcionalidades do *NCAP*: Os blocos funcionais que compõem a parte lógica do *NCAP* devem ser desenvolvidos com linguagem orientada a objeto. Com intuito de atender este quesito, o software da plataforma *IEEE 1451* foi implementado com tecnologia Java. Este software além de atender as especificações do padrão *IEEE 1451.1* possibilita a execução de atividades de um software supervisor e possui uma configuração básica que pode ser utilizada em diferentes aplicações. A geração das especificações dos transdutores (*TEDS*) no software da plataforma *IEEE 1451* em conjunto com as funcionalidades deste software é inédita.
- d) Metodologia para descrever a *TEDS*: A descrição da *TEDS* na plataforma *IEEE 1451* é dividida basicamente em duas etapas. Na primeira o projetista deve codificar as informações com base em uma tabela proposta neste trabalho (*ProtocolTEDS.h*) e na posição (canal) de cada transdutor. Na segunda etapa ao solicitar a *TEDS* no software *NCAP*, gera-se um arquivo que pode ser impresso ou salvo em pasta. Pode-se destacar nesta metodologia a simplicidade de descrever a *TEDS* com base nesta tabela de códigos e a flexibilidade da mesma para tornar-se um modelo do padrão *IEEE 1451.0*.
- e) Criação de um *driver* para comunicação entre software *ELIPSE SCADA* e uma placa de *FPGA (DE2)*: Neste trabalho implementou-se uma comunicação entre o software *ELIPSE SCADA* e o módulo para transdutores (placa *DE2*). Como a empresa Elipse E3 não possuía driver de comunicação com placas de *FPGAs* este resultado foi pioneiro e comprovaram a flexibilidade da plataforma *IEEE 1451*.

Desta forma, a plataforma *IEEE 1451* desenvolvida neste trabalho pode ser aplicada comercialmente e gera-se um leque de projetos de pesquisa, demonstrando robustez e expansividade.

6.1 Trabalhos Futuros

Com base na plataforma *IEEE 1451*, pode-se trabalhar nos seguintes temas para projeto de pesquisa em nível de pós-graduação e iniciação científica:

- a) Desenvolvimento de múltiplos *WTIMs*, ou seja, rede sem fio de nó *IEEE 1451*: Configurar os dispositivos para operar em rede de comunicação, ou seja, cada tecnologia (*Wi-Fi*, *ZigBee* e *Bluetooth*) pode ser utilizada em sistema de comunicação com mais de dois formando assim uma rede de comunicação com múltiplos módulos para transdutores. Nesta configuração um módulo sem fio será conectado no *PC* e os demais em cada placa *DE2* correspondente. Desta forma, o software *NCAP* poderá controlar diversos processos simultaneamente.
- b) Desenvolvimento do *TIM* em controladores lógicos programáveis *CLP*: Desenvolver blocos funcionais baseado no firmware da plataforma *IEEE 1451* em *CLP* principalmente que favoreçam a implementação da *TEDS*.
- c) Comunicação entre o software *NCAP* e com diferentes hardwares de controle: Configurar *CLPs* e microcontroladores para realizar a comunicação com *NCAP* da plataforma *IEEE 1451*. Como este sistema de comunicação é baseado nos parâmetros do protocolo *UART*, deve-se utilizar dispositivos (*CLPs* e microcontroladores).
- d) Utilização de dispositivos móveis na automação de processos, ou seja, comunicando com *NCAP*: Inserir e desenvolver funcionalidades de servidor no software *NCAP* que viabilizam a comunicação com dispositivos móveis. Neste aspecto o *NetBeans* possui recursos que podem atender estas necessidades.
- e) Substituição da *API COMM* pelo pacote *RXTX*: A classe *DriverTIM.java* utiliza o pacote *javax.comm* para realizar a comunicação com a interface *RS232*. Porém, este pacote não tem suporte para desenvolver aplicações com comunicação *USB*. Uma opção é utilizar o pacote *RXTX* da *Sun*

Microsystems que além de possibilitar a comunicação *USB* e *RS232* possui portabilidade com diferentes sistemas operacionais.

- f) Inserção de lógica de controle *PID* (Proporcional Integrativo Derivativo) no *template IEEE 1451*: Inserir algoritmos de projeto de *PID* automático no *template IEEE 1451*, especificamente na biblioteca *IEEE1451.c*. Estes algoritmos podem atender um determinado tipo de planta, uma faixa de resposta transitória para uma entrada pré-estabelecida.
- g) Simulação do *DriverTIM* no software *ModelSIM* da *Mentor Graphics*: Utilizar os recursos do *ModelSIM* para simular o funcionamento do módulo embarcado, como por exemplo, a comunicação serial, armazenamento de informações, comunicação com barramento dos transdutores e a operação com display.

REFERÊNCIAS

- [1] SONG, E.Y.; LEE, K.B. Sensor network based on IEEE 1451.0 and IEEE p1451.2-RS232. In: INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE, 25., 2008, Victoria. **Automation**. Victoria: [s.n.], 2008. p. 1728-1733.

- [2] LEE, K.; SONG, E. Y. Integration of IEEE 1451 smart transducers and OGC-SWE using STWS. In: SENSOR APPLICATION SYMPOSIUM, 4, 2009, New Orleans. **Smart Sensors and Standards**. New Orleans: [s.n.], 2009. p. 298-303.

- [3] SONG, E. Y.; KANG L. Understanding IEEE 1451-networked smart transducer interface standard – what is a smart transducer. **IEEE Instrument & Measurement Magazine**, [S.l.], v. 11, n. 2, p. 11-17, Abr 2008.

- [4] IEEE 1451.0 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **IEEE 1451 website. what is IEEE 1451?** [S.l.: s.n.], 2009. Disponível em: <<http://grouper.ieee.org/groups/1451/0/>> Acesso em: 09 fev. 2009.

- [5] LEE, K.; SONG, E. Y. UML model for the IEEE 1451.1 standard. In: INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE- IMTC, 2., 2003, Vail. **Proceedings....** Vail: [s.n.], 2003. p. 1587-1592.

- [6] LEE, K. IEEE 1451: a standard in support of smart transducer networking. In: INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE- IMTC, 2., 2000, Baltimore. **Proceedings....** Baltimore: [s.n.], 2000. p. 525-528.

- [7] LEE, K. Sensor networking and interface standardization. In: INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE- IMTC, 1., 2001, Budapest. **Proceedings....** Budapest: [s.n.], 2001. p. 525-528.

- [8] LICHT, T.R. The IEEE 1451.4 proposed standard. **Instrumentation & Measurement Magazine**, [S.l.], v. 4, n.1, p. 12-18, Mar 2001.

- [9] ZIGBEE ALLIANCE. **Solutions**. [S.l.: s.n.], 2009. Disponível em: < <http://www.zigbee.org> > Acesso em: 28 jan. 2009.

[10] BLUETOOTH. **Welcome to Bluetooth**. [S.l.: s.n.], 2009. Disponível em:

<
<http://www.bluetooth.org>> Acesso em: 28 jan. 2009.

[11] WI-FI ALLIANCE. **Certified**. [S.l.: s.n.], 2009. Disponível em: <
<http://www.wi-fi.org>> Acesso em: 28 jan. 2009.

[12] SONG, E.Y.; LEE K. An implementation of the proposed IEEE implementation of the proposed IEEE 1451.0 and 1451.5 standards. In: IN SENSOR APPLICATION SYMPOSIUM, 2., 2006, [S.l.]. **Proceedings...** [S.l.: s.n.], 2006. p. 72-77.

[13] IEEE 1451.6 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **IEEE 1451 website. what is IEEE 1451?** [S.l.: s.n.], 2009. Disponível em: <<http://grouper.ieee.org/groups/1451/6/>>. Acesso em: 09 fev. 2009.

[14] IEEE P1451.7. **Transducer radio frequency identification (RFID) systems IEEE 1451.7**. [S.l.: s.n.], 2009. Disponível em: <http://www.sensorsportal.com/HTML/standard_7.htm>. Acesso em: 09 fev. 2009.

[15] IEEE 1451.1 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **IEEE 1451 working groups – group1**. [S.l.: s.n.], 2009. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 09 fev. 2009.

[16] ELIPSE SOFTWARE E3. **Elipse SCADA**. [S.l.: s.n.], 2009. Disponível em:
 <<http://www.elipse.com.br>>. Acesso em: 28 jan. 2009.

[17] ROCKWELL SOFTWARE. **RSView32 from rockwell software**. [S.l.: s.n.], 2009. Disponível em:
 <<http://www.rockwellautomation.com/rockwellsoftware/performance/view32/>>. Acesso em: 28 jan. 2009.

[18] SIEMENS. **Simatic winCC**. [S.l.: s.n.], 2009. Disponível em:
 <http://www.automation.siemens.com/hmi/html_76/products/software/wincc/index.htm>. Acesso em: 28 jan. 2009.

- [19] NIST IEEE 1451. **Real-time machine tools coolant control**. [S.l.: s.n.], 2009. Disponível em: < <http://motion.aptd.nist.gov/P1451/ISADemo.htm>>. Acesso em: 30 jan. 2009.
- [20] AGILENT TECHNOLOGIES; SUN MICROSYSTEMS. **JDDAC**. [S.l.: s.n.], 2009. Disponível em: < <https://jddac.dev.java.net/>>. Acessado em: 30 jan. 2009.
- [21] LABS. AGILENT TECHNOLOGIES. **JDDAC**. [S.l.: s.n.], 2009. Disponível em: <<http://jddac.labs.agilent.com/index.do>>. Acessado em: 30 jan. 2009.
- [22] SUN MICROSYSTEMS. **Java 2 plataforma enterprise edition**. [S.l.: s.n.], 2009. Disponível em: <java.sun.com/j2ee/overview.html>. Acesso em: 30 jan. 2009.
- [23] SUN MICROSYSTEMS. **The Java ME plataforma**. [S.l.: s.n.], 2009. Disponível em: <java.sun.com/javame/index.jsp>. Acesso em: 30 jan. 2009.
- [24] SUN MICROSYSTEMS. **Java 2 plataforma server edition**. [S.l.: s.n.], 2009. Disponível em: <java.sun.com/j2se/>. Acesso em: 30 jan. 2009.
- [25] DEITEL, H. M.; DEITEL, P. J. **Java como programar**. 3.ed. Porto Alegre: Bookman, 2001. 1164 p.
- [26] WOBSCHALL, D. Networked sensor monitoring using the universal IEEE 1451 standard. **Instrumentation & Measurement Magazine, IEEE**. New York, v. 11, [n.2](#), p. 18-22, Abr 2008.
- [27] HIGUERA, J.; POLO, J.; GASULLA, M. A ZigBee wireless sensor network compliant with the IEEE1451 standard. In: SENSOR APPLICATION SYMPOSIUM, 5., 2009, New Orleans. **SAS 2009. IEEE**. New Orleans: [s.n.], 2009. p. 309-313.
- [28] SONG, E. Y.; LEE K. Understanding IEEE 1451-Networked smar transducer interface standard – What is a smart transducer? **Instrumentation & Measurement Magazine**. New York, v. 11, [n.2](#), p. 11-17, Maio 2008.

- [29] WANG, R; MORRIS, J.; FIGUEROA, F; MORRIS, M. T. Wireless intelligent sensors supporting integrated systems health management architecture. In: SENSOR APPLICATION SYMPOSIUM, 4., 2008, Atlanta. **SAS 2008. IEEE**. Atlanta: [s.n.], 2008. p. 215-219.
- [30] WOBSCHALL, D.; STEPANENKO, A.; MAYKIV, I.; KOCHAN, R.; SACHENKO, A.; KOCHAN, V. A multi-port serial NCAP using the IEEE 1451 smart transducer standard. In: SENSORS APPLICATION SYMPOSIUM, 5., 2009, New Orleans. **SAS 2009. IEEE**. New Orleans: [s.n.], 2009. p. 293-297.
- [31] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 1.,2001, Chicago. **Sensors**. Chicago: [s.n.], 2001. Disponível em: <<http://ieee1451.nist.gov/>> Acesso em: 28 jan. 2009.
- [32] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 2., 2001, Philadelphia. **Sensors**. Philadelphia: [s.n.], 2001. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 28 jan. 2009.
- [33] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 3.,2002, San Jose. **Sensors**. San Jose: [s.n.], 2002. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 28 jan. 2009.
- [34] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 4.,2002, Boston. **Sensors**. Boston: [s.n.], 2002. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 28 fev. 2009.
- [35] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 5.,2003, Anaheim. **Sensors**. Anaheim: [s.n.], 2003. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 09 fev. 2009.
- [36] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. In: WORKSHOP ON WIRELESS SENSING, 6.,2004, Detroit. **Sensors**. Detroit: [s.n.], 2004. Disponível em: <<http://ieee1451.nist.gov/>>. Acesso em: 09 fev. 2009.
- [37] WOBSCHALL D. **Smart sensor integration with a wired network**. Buffalo:

- [s.n.], 2009. Disponível em:
<<http://ieee1451.nist.gov/S07%20Darold%20Wobschall.pdf>> Acesso em: 09 fev. 2009.
- [38] ASTILEAN, A.; FOLEA, S. Design and testing in laboratory environment of the embedded microsystem ECAM. In: AUTOMATION, QUALITY AND TESTING, ROBOTICS, 1., 2006, [S.l.]. **IEEE International Conference**. [S.l.: s.n.], 2006. p. 442-447.
- [39] NIST, A. **Proposal for IEEE P1451.5, standard for a smart transducer**. [S.l.: s.n.], 2009. Disponível em:
<<http://grouper.ieee.org/groups/1451/5/Proposals%20submitted/P1451.5-Prop2.pdf>>. Acesso em: 21 julh. 2009.
- [40] INTERNATIONAL BUSINESS MACHINES- IBM. **Research in sensor networks platforms and wireless technology**. [S.l.: s.n.], 2009. Disponível em:
<http://www.zurich.ibm.com/pdf/sys/ZRL_Sensor_Networks_2004_04_11.pdf>. Acesso em: 20 julh. 2009.
- [41] MORGAN, F.; ROCKE, P.; O' HALLORAN, M. Applied VHDL training methodology, EDA framework and hardware implementation platform. In: RECONFIGURABLE Computing and FPGAS. [S.l.: s.n.], 2005. p. 8-19.
- [42] BIRSAN, N.; SHARAD, S. Work in progress - just-in-time teaching and hands-on experimenting embedded systems for undergraduates. In: FRONTIERS IN EDUCATION CONFERENCE, 38., 2008, [S.l.]. **IEEE Computer Society**. [S.l.: s.n.], 2008. p. F1H-9-F1H-11.
- [43] SVEDA, M.; VRBA, R. Sensor networking. In: ENGINEERING OF COMPUTER BASED SYSTEMS, ECBS, 8., 2001, [S.l.]. **Proceedings...** [S.l.: s.n.], 2001. p.262-268.
- [44] DEVICETNET. **Introduction**. [S.l.: s.n.], 2009. Disponível em:
<<http://www.rtaautomation.com/devicenet/>>. Acesso em: 28 jan. 2009.
- [45] LONWORKS. **The Lon works control network is now global ISO/IEC standard**. [S.l.: s.n.], 2009. Disponível em: <<http://www.echelon.com>>. Acesso em: 28 jan. 2009.

- [46] CANOPEN. **CANopen is a higher-layer protocol for the CANbus**. [S.l.: s.n.], 2009. Disponível em: <<http://www.canopen.us>>. Acesso em: 28 jan. 2009.
- [47] FIELDBUS FOUNDATION, **Fieldbus**. [S.l.: s.n.], 2009. Disponível em: <<http://www.fieldbusfoundation.org>>. Acesso em: 28 jan. 2009.
- [48] DE CASTRO, A.; CHAQUET, J. M.; MOREJON, E.; RIESGO, T.; UCEDA, J. A system- on-chip for smart sensors. In: INTERNATIONAL SYMPOSIUM INDUSTRIAL ELECTRONICS, ISIE, 2., 2002, [S.l.]. **Proceedings...** [S.l.: s.n.], 2002. p. 595-599.
- [49] WOBSCHALL, D. An implementation of IEEE 1451 NCAP for internet access of serial port-based. In: SENSORS FOR INDUSTRY CONFERENCE, 2., 2002, [S.l.]. **Sensors Applications**. [S.l.: s.n.], 2002. p.157-160.
- [50] RAMOS, H.; PEREIRA, M.; VIEGAS, V.; POSTOLACHEI, O.; GIRAO, P.
A virtual instrument to test smart transducer interface modules (STIMs). In: INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE, IMTC, 3., 2003, [S.l.]. **Proceedings...** [S.l.: s.n.], 2003. p.772-775.
- [51] WOBSCHALL, D.; WAI SING POH. A smart RTD temperature sensor with a prototype IEEE 1451.2 internet interface. In: SENSORS FOR INDUSTRY CONFERENCE, 2., 2004, [S.l.]. **Proceedings...** [S.l.: s.n.], 2004. p.183-186.
- [52] WOBSCHALL, D.; LAKSHMANAN, D. Wireless soil moisture sensor based on fringing capacitance. In: IEEE [SENSORS](#), 4., 2005, Irvine. **Sensor and Actuator Systems**. Irvine: [s.n.], 2005. p. 4.
- [53] WOBSCHALL, D. Wireless gas monitor with IEEE 1451 protocol. In: SENSOR APPLICATIONS SYMPOSIUM, 1., 2006, Houston. **Smart Sensor and Standards**. Houston: [s.n.], 2006. p. 162-164.
- [54] NI, F.L.; JIN, M.H.; XIE, Z.W.; SHI, SH.C.; LIU, Y.CH.; LIU, H.; IRZINGER, G. A. Highly integrated joint servo system based on FPGA with nios II processor. In: MECHATRONICS AND AUTOMATION INTERNATIONAL CONFERENCE , 5., 2006, [S.l.]. **Proceedings...** [S.l.: s.n.], 2006. p. 973-978.

- [55] WEIJUN X.; YAN S.; YINAN L.; QING Y. TEA: transmission error approximation for distance estimation between two zigbee devices. In: INTERNATIONAL WORKSHOP NETWORKING ARCHITECTURE AND STORAGE, 1., 2006, Shenyang. **Wireless networks and protocols**. Shenyang: [s.n.], 2006. p. 8.
- [56] MODBUS. **Modbus-IDA**. [S.l.: s.n.], 2009. Disponível em: <<http://www.modbus.org>>. Acesso em: 28 jan. 2009.
- [57] SVEDA, M.; TRCHALIK, R. ZigBee-to-internet interconnection architectures. In: INTERNATIONAL CONFERENCE- ICONS, 2., 2007, [S.l.]. **Proceedings...** [S.l.: s.n.], 2007. p. 30.
- [58] IEEE 1451.5. **The Approved draft P1451.5 (D10.8)**. [S.l.: s.n.], 2009. Disponível em: <<http://grouper.ieee.org/groups/1451/5/>>. Acesso em: 30 jan. 2009.
- [59] BERNEL, P. S. M. **Comunicação móvel: tecnologia e aplicações**. 8.ed. São Paulo: Érica, 2005. 204 p.
- [60] BOLZANI, C. A. M. **Residências inteligentes**. São Paulo: Livraria da Física, 2004. 331 p.
- [61] IEEE 802.11. [S.l.: s.n.], 2009. Disponível em: <<http://www.ieee802.org/11/>>. Acesso em: 30 jan. 2009.
- [62] WI-FI. **INT550 ME / INT550 WI**. [S.l.: s.n.], 2009. Disponível em: <[ww.integral.com.br/downloads/Manual_INT550.pdf](http://www.integral.com.br/downloads/Manual_INT550.pdf)>. Acesso em: 30 jan. 2009.
- [63] MCDEMOTT-WELLS, P. What is Bluetooth? **Potentials, IEEE**, New York, v. 23, n.5, p. 33-35, Dez 2004.
- [64] BLUETOOTH. **Technology and application**. [S.l.: s.n.], 2009. Disponível em: <<http://www.bluetooth.com/Bluetooth/SIG/>>. Acesso em: 30 jan. 2009.

- [65] PERSSON, K. E.; MANIVANNAN, D. Secure connections in bluetooth scatternets. system sciences. In: ANNUAL HAWAI INTERNATIONAL CONFERENCE, 36., 2003, [S.l.]. **Proceedings...** [S.l.: s.n.], 2003.p. 10.
- [66] SHIH, Y. C.; HSUNG, P. C.; RUI, C. C. Providing mobile LAN access capability for bluetooth devices. In: INTERNATIONAL CONFERENCE, 9., 2002, [S.l.]. **Proceedings...** [S.l.: s.n.], 2002. p. 631-636.
- [67] BLUETOOTH. **BTS1009C RS232**. [S.l.: s.n.], 2009. Disponível em: <www.sunix.com.tw/cc/en/products/Datasheet/BTS1009C.pdf>. Acesso em: 23 julh. 2009.
- [68] JIN, S. L. An Experiment on performance study of IEEE 802.15.4 wireless networks. In: IEEE CONFERENCE, ETFA, 2., 2005, [S.l.]. **Emerging Technologies and Factory Automation**. [S.l.: s.n.], 2005. p. 451-458.
- [69] GHAZYINI, M. H. F.; VAHABI, M.; RASID, M. F. A.; ABDULLAH, R. S. A. R.; MUSA, W. N. M. W. Low energy consumption MAC protocol for wireless sensor networks. In: INTERNATIONAL CONFERENCE ON SENSOR TECNOLOGIES AND APPLICATIONS, SENSORCOMM, 2., 2008, [S.l.]. **Sensor Tecnologies and Applications**. [S.l.: s.n.], 2008. p. 49-54.
- [70] YOUNG, P. H. **Técnica de comunicação eletrônica**. São Paulo: Pearson Prentice Hall, 2005. 687 p.
- [71] SERRA, M.; MARTI, P.; CARRABINA, J. Implementation of a channel equalizer for OFDM wireless LANs Rapid System Prototyping. In: IEEE INTERNATIONAL WORKSHOP, 15., 2004, [S.l.]. **Proceedings...** [S.l.: s.n.], 2004. p. 232-238.
- [72] MAXSTREAM. **XBEE**. [S.l.: s.n.], 2009. Disponível em: <<http://www.digi.com/technology/wireless/products.jsp>>. Acesso em: 26 julh. 2009.
- [73] MORAES, C. C.; CASTRUCCI, P. L. **Engenharia de automação industrial**. Rio de Janeiro: LTC, 2001. 295 p.
- [74] GONÇALVES, E. **Dominando netBeans**. Rio de Janeiro: Ciência Moderna, 2006. 384 p.

[75] PUGA, S.; RISSETTI, G. **Lógica de programação e estrutura de dados com aplicações em Java**. São Paulo: Pearson Prentice Hall, 2003. 272 p.

[76] BATISTA, E.A.; SILVA, A.C.R.; GONDA, L.; PEREIRA, M.C.; DA SILVA, I.; FERREIRA, W.M.; D'ELIA, F. IEEE1451-based multi-interface module (I2M) for industrial processes automation. In: [SENSORS APPLICATIONS SYMPOSIUM- SAS, 4., 2008, \[S.l.\]. IEEE...](#) [S.l.: s.n.], 2008. p. 220-224.

[77] UNIVERSIDADE FEDERAL DO AMAZONAS- UFAM. **Sistemas embarcados**. [S.l.: s.n.], 2009. Disponível em: <<http://www.dcc.ufam.edu.br/~rbarreto/se/slides/ISE-04- PbD.pdf>>. Acesso em: 25 julh. 2009.

[78] ALTERA. **Literature: SOPC builder**. [S.l.: s.n.], 2009. Disponível em: <<http://www.altera.com/literature/lit-sop.jsp>>. Acesso em: 31 jan. 2009.

[79] ALTERA. **Literature: NIOS II processor**. [S.l.: s.n.], 2009. Disponível em: <<http://www.altera.com/literature/lit-nio2.jsp>>. Acesso em: 31 jan. 2009.

[80] ALTERA. **Quartus II development software literature**. [S.l.: s.n.], 2009. Disponível em: <<http://www.altera.com/literature/lit-qts.jsp>>. Acesso em: 31 jan. 2009.

[81] BATISTA, E. A.et. al. Implementação de um módulo embarcado para automação de processos industriais utilizando software elipse SCADA e FPGA com processador NIOS II. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 17.,2008, Juiz de Fora. **Sistemas de Automação**. Juiz de Fora: [s.n.], 2008. 1 CD/ROM.

APÊNDICE A – CONFIGURAÇÕES MÓDULOS SEM FIO

Módulos ZigBee

Os módulos *XBee* podem ser conectados em placas que possuem interface *RS232* ou *USB*, propiciando a construção de um sistema embarcado sem fio. Cada módulo *XBee* possui vinte pinos e um acoplador de antena. Na Figura 1A apresenta-se as características físicas de um módulo *XBee*.

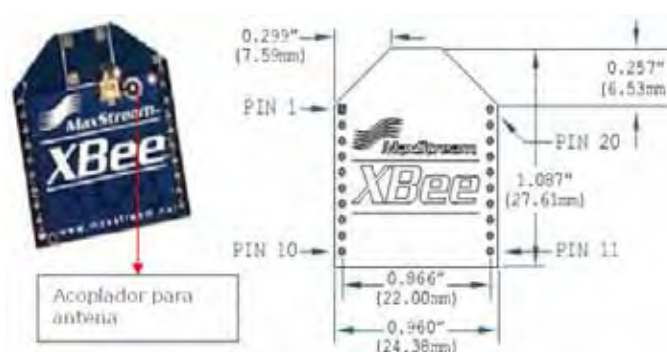


Figura 1A: Características de um Módulo *XBee* da *MaxStream*.

Fonte: <http://www.digi.com/products/wireless/point-multipoint/xbee-series1-mobile.jsp>

Na Figura 2A, apresenta-se um modelo de uma placa, dotada de um suporte para chip *XBee* e de um conector *RS232*.

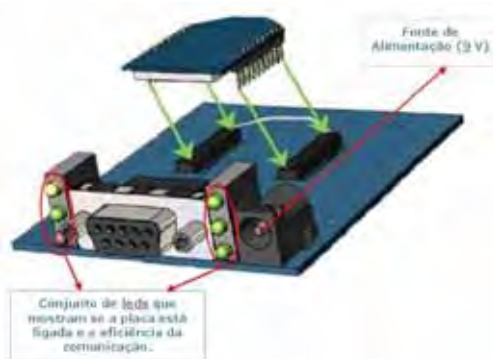


Figura 2A: Uma placa de desenvolvimento para conexão de módulos *XBee*.

Fonte: <http://www.digi.com/products/wireless/point-multipoint/xbee-series1-mobile.jsp>.

Para realizar as atividades de comunicação, os módulos devem ser configurados através do software *X-CTU* da *DIGI*⁴. O software *X-CTU*, que pode ser obtido

⁴ *DIGI* empresa especializada na produção de módulos sem fio.

gratuitamente no site www.maxstream.com, possui uma interface amigável, que possibilita ao usuário configurar desde a comunicação serial até a função do módulo na rede. Para testar a comunicação entre dois módulos *XBee*, basta realizar as seguintes etapas:

1) Após a instalação do software *X-CTU*, conecte uma das placas na porta *RS232* do *PC*;

2) Abra o software *X-CTU*, na janela *Modem Configuration*, clique em *Read*, neste momento, com a placa devidamente conectada ao *PC*, vai aparecer a descrição do módulo *XBee*. Modifique os campos de endereçamento *Destination Address Low (DL)*, *Detination Address High (DH)* e *Source Address (MY)*, conforme pode se observar na Figura 3A. O comando *Read* (do software *X-CTU*) reconhece qual modem está conectado no *PC*, além de especificar o padrão de comunicação (*IEEE 802.15.4*) e a versão do *Xbee*;

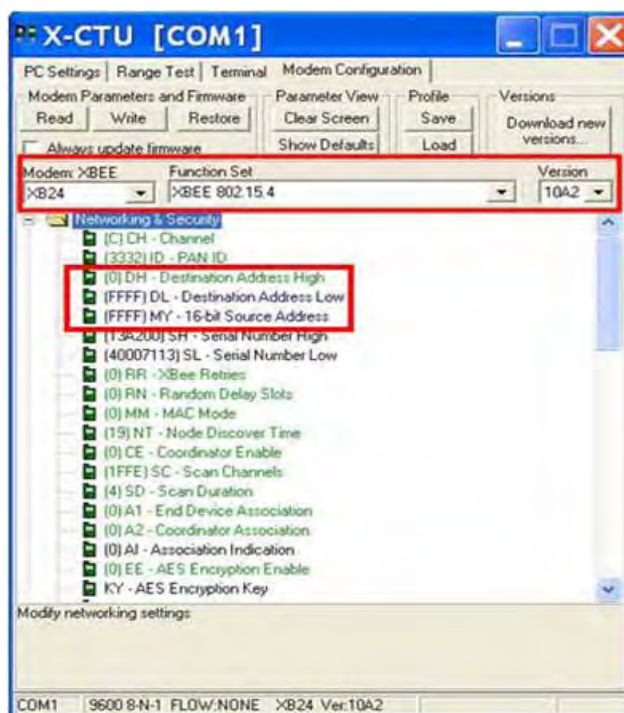


Figura 3A: Configuração dos módulos *XBee*.

3) Em seguida, execute os passos de *Restore* e *Write* (clikando respectivamente os botões no menu *Modem e Configuration*);

4) Retire o módulo do *PC* e conecte o outro módulo;

5) Execute as etapas de 1 a 4 novamente;



6) Na Janela *PC Settings* do software *X-CTU*, o usuário pode definir os parâmetros para a transferência de dados, como se observa na Figura 4A;

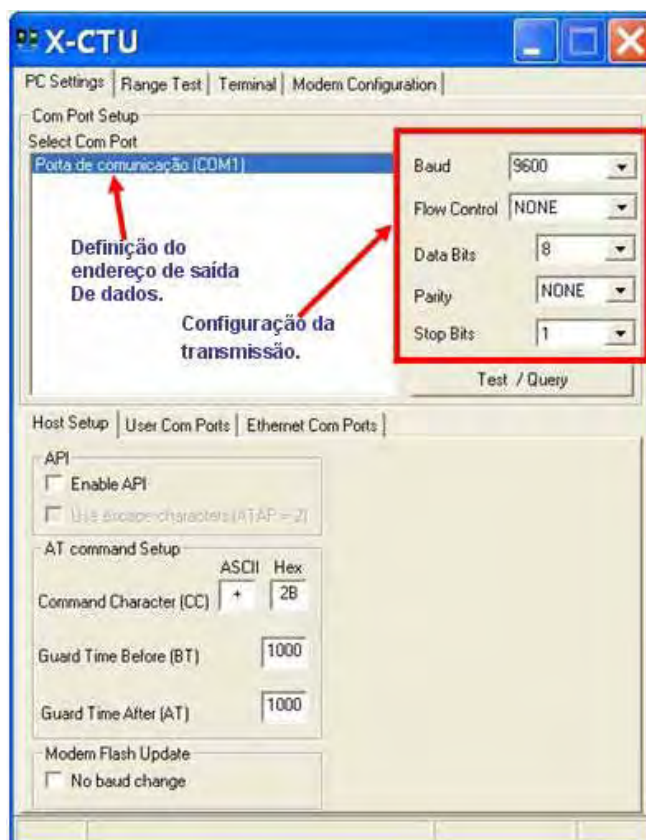


Figura 4A: Parâmetros para a transferência de dados no *X-CTU*.

7) No módulo que não está conectado no *PC*, insira um fio entre os pinos 2 e 3 da *RS232*;

8) Finalmente na Janela *Range Test*, o usuário pode definir os dados a ser enviados, inserindo no campo disponível no *menu transmit* e em seguida clicando em *START*.

O software *X-CTU* possui uma importante propriedade que é a de demonstrar a eficiência da comunicação como pode ser observado na Figura 5A.

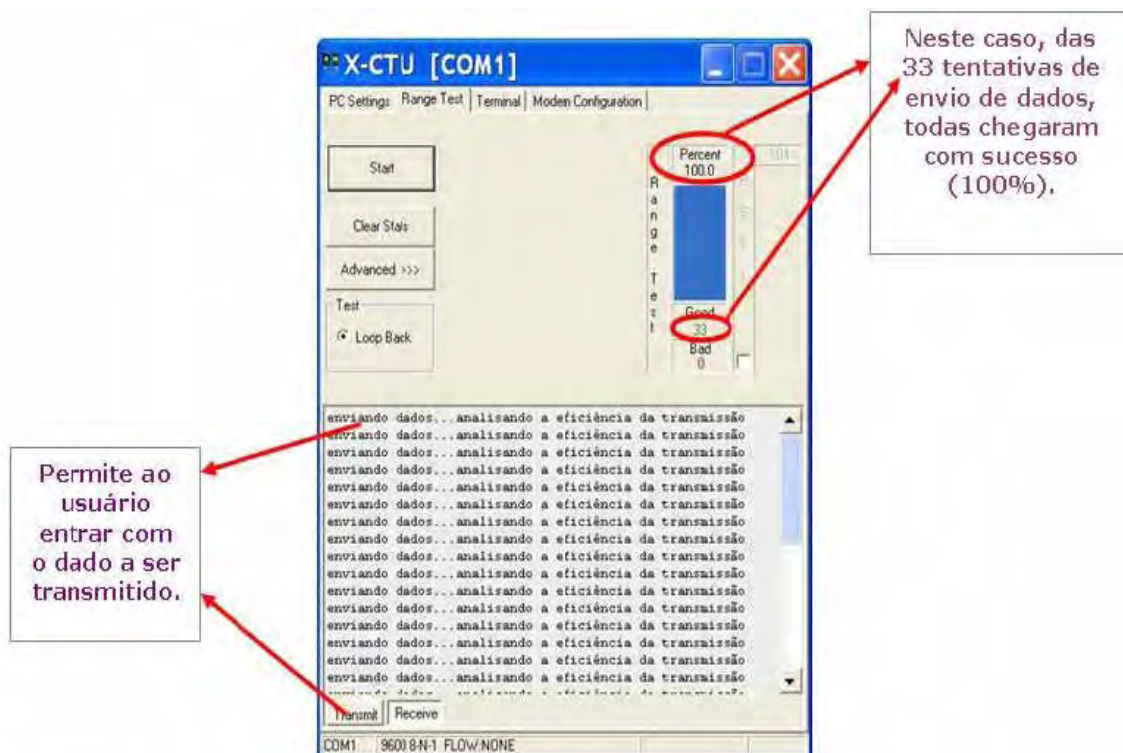


Figura 5A: Transmissão de dados entre dois módulos *XBee*.

Neste projeto entre as atividades do *NCAP* está à configuração dos dispositivos que são utilizados para desenvolver a interface *IEEE 1451.5*.

Módulo Wi-Fi

A configuração dos módulos *INT550WI* para a comunicação ponto-a-ponto, deve ser por meio das seguintes etapas:

- 1) Ligue os módulos na tensão nominal especificada (127 V);
- 2) Execute o programa *Digi Device Discovery* (CD de instalação);
- 3) O número *IP* do equipamento pode ser obtido por *Dynamic Host*

Configuration Protocol (DHCP), *Auto-IP* ou definido e/ou modificado através da opção “*Configure network settings*”, como observa-se na Figura 6A. Este procedimento deve ser realizado para os dois dispositivos;

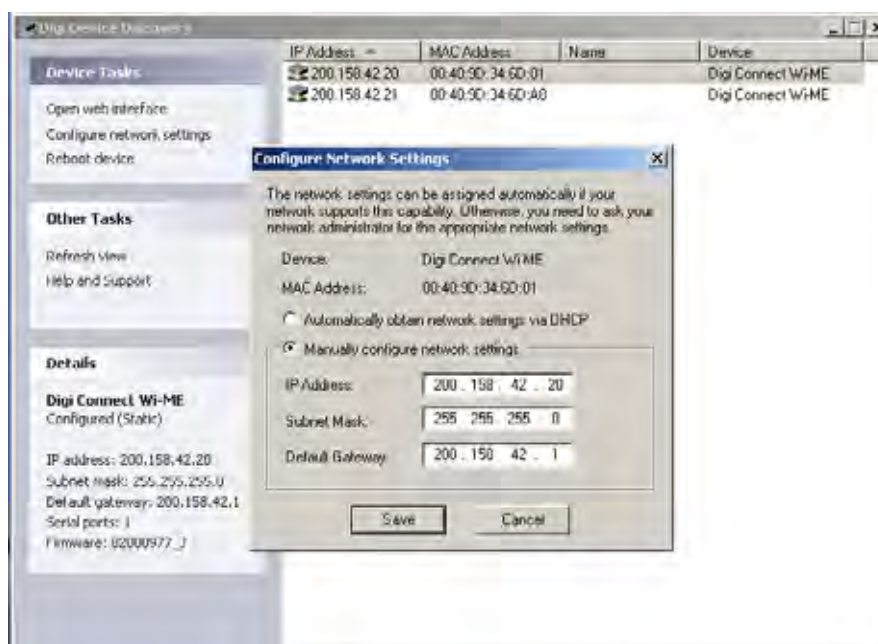


Figura 6A: Definição de *IP* para os módulos *INT550*.

Fonte: Manual do *INT 550WI*.

4) Os parâmetros de transmissão de dados são definidos individualmente,

sendo necessário um cadastro de usuário no próprio software. Para realizar este cadastro, deve-se clicar na opção *Open web interface* em destaque na Figura 7A;

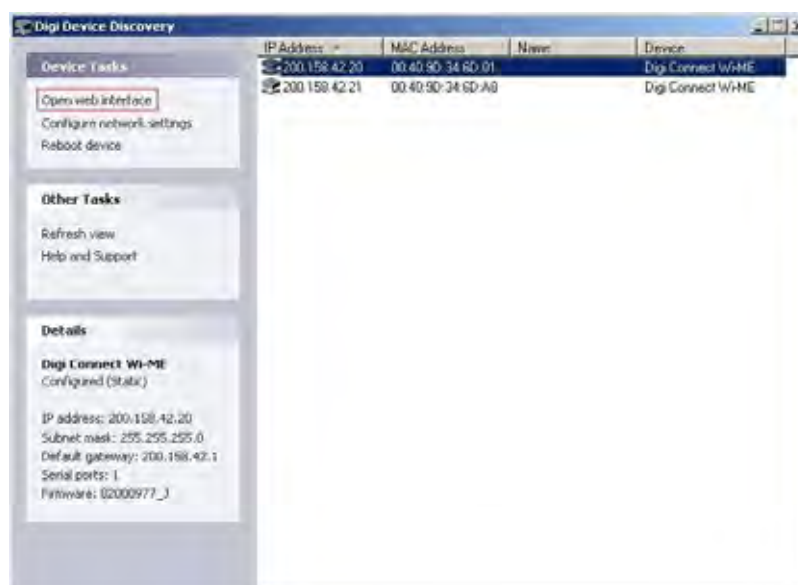
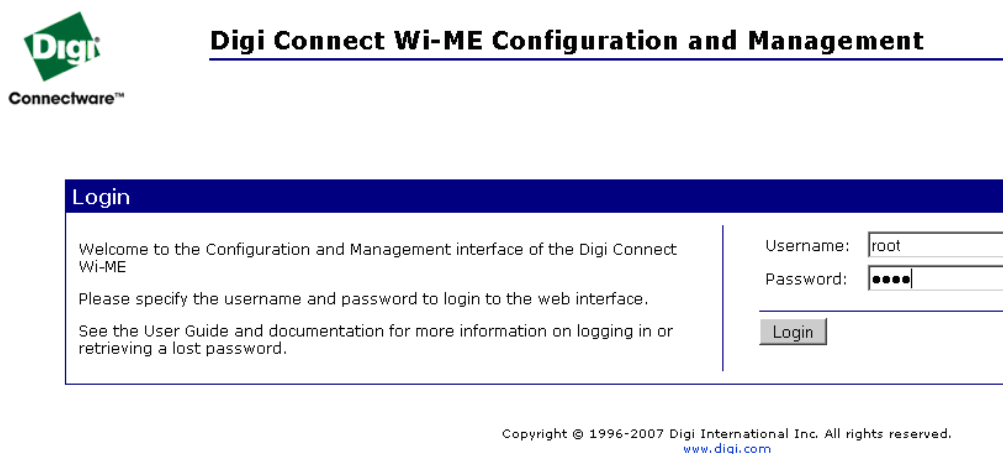


Figura 7A: Opção para iniciar a definição de parâmetros para transmissão.

Fonte: Manual do *INT 550WI*.

- 5) Deve-se cadastrar um *login* e senha padrão como mostra-se na Figura
- 8A. A senha padrão é *dbps*;



The screenshot shows the login interface for the Digi Connect Wi-ME Configuration and Management web interface. The page has a blue header with the Digi Connectware logo on the left and the title "Digi Connect Wi-ME Configuration and Management" on the right. Below the header, there is a "Login" section with a blue bar. The main content area is divided into two columns. The left column contains a welcome message and instructions: "Welcome to the Configuration and Management interface of the Digi Connect Wi-ME", "Please specify the username and password to login to the web interface.", and "See the User Guide and documentation for more information on logging in or retrieving a lost password." The right column contains a login form with fields for "Username:" (containing "root") and "Password:" (containing "dbps"), and a "Login" button. At the bottom of the page, there is a copyright notice: "Copyright © 1996-2007 Digi International Inc. All rights reserved. www.digi.com".

Figura 8A: Tela de cadastro para configuração de parâmetros de transmissão.

Fonte: Manual do *INT 550WI*.

Observação A1: Para estas configurações iniciais recomenda-se desabilitar temporária de qualquer software de *firewall*.

Observação A2: Caso o *browser* esteja configurado para conexões através de Servidor Proxy este deve ser desabilitado temporariamente.

- 6) Após abrir o *browser* de configuração do módulo desejado, deve-se
- iniciar a definição dos parâmetros de comunicação na opção *Serial Ports* como verifica-se em destaque na Figura 9A;



Figura 9A: Opção para definir os parâmetros de transmissão.

Fonte: Manual do INT 550WI.

- 7) Em seguida deve-se selecionar o tipo de *profile* desejado para o módulo,
na opção *Change Profile*;
- 8) Na opção *Profile* defini-se “TCP Sockets” e em seguida clique na opção
Apply;
- 9) Selecione a opção *Automatically establish TCP connections* e digite o IP
correspondente ao dispositivo no qual se deseja realizar a comunicação,
sendo que a porta padrão é 2101, como pode-se observar na Figura 10A;

Alarms
System
Remote Management
Users

Applications
Ekahau Client

Management
Serial Ports
Connections

Administration
File Management
Backup/Restore
Update Firmware
Factory Default Settings
System Information
Reboot

Logout

TCP Server Settings

Connect directly to the serial device using the following TCP ports on the network.

☒ Enable Telnet access using TCP Port: ☐ Enable TCP Keep-Alive

☒ Enable Raw TCP access using TCP Port: ☐ Enable TCP Keep-Alive

☒ Enable Secure Shell (SSH) access using TCP Port: ☐ Enable TCP Keep-Alive

☒ Enable Secure Socket access using TCP Port: ☐ Enable TCP Keep-Alive

TCP Client Settings

Automatically establish bi-directional TCP connections between the serial device and a server or other networked device.

☒ Automatically establish TCP connections

Establish connection under one of the following conditions:

☒ Always connect and maintain connection

☐ Connect when data is present on the serial line

Match string:

☐ Strip string before sending

☐ Connect when DCD (Data Carrier Detect) line goes high

☐ Connect when DSR (Data Set Ready) line goes high

Establish connection to the following network service:

Server (name or IP):

Service:

TCP Port:

☐ Enable TCP Keep-Alive

Figura 10A: Definição de *TCP* dos módulos *INT550*.

Fonte: Manual do *INT 550WI*.

10) Em *Basic Serial Settings*, realizam-se as definições dos parâmetros para a comunicação, sendo que na aba *Flow Control* selecionou-se a opção *none*;

11) Para testar a comunicação deve-se reiniciar os dispositivos.

Com as configurações desenvolvidas e os dispositivos devidamente conectados no *PC* e na placa *DE2*, pode-se realizar a comunicação de acordo com os limites da transmissão desta tecnologia.

APÊNDICE B – DOCUMENTAÇÃO DO SOFTWARE NCAP

Neste Apêndice, apresenta-se a documentação do software *NCAP* utilizando diagrama de classe (*UML*) e o *javadoc*, ambas geradas no *NetBeans 6.1*. Nas Figuras 1B a 6B, apresentam-se as etapas para gerar o diagrama de classe do software *NCAP*.

Para gerar um diagrama em *UML* no *NetBeans IDE 6.1* deve-se entrar no *menu* de *Arquivo* e selecionar *Novo projeto*, como mostra-se na Figura 1B.

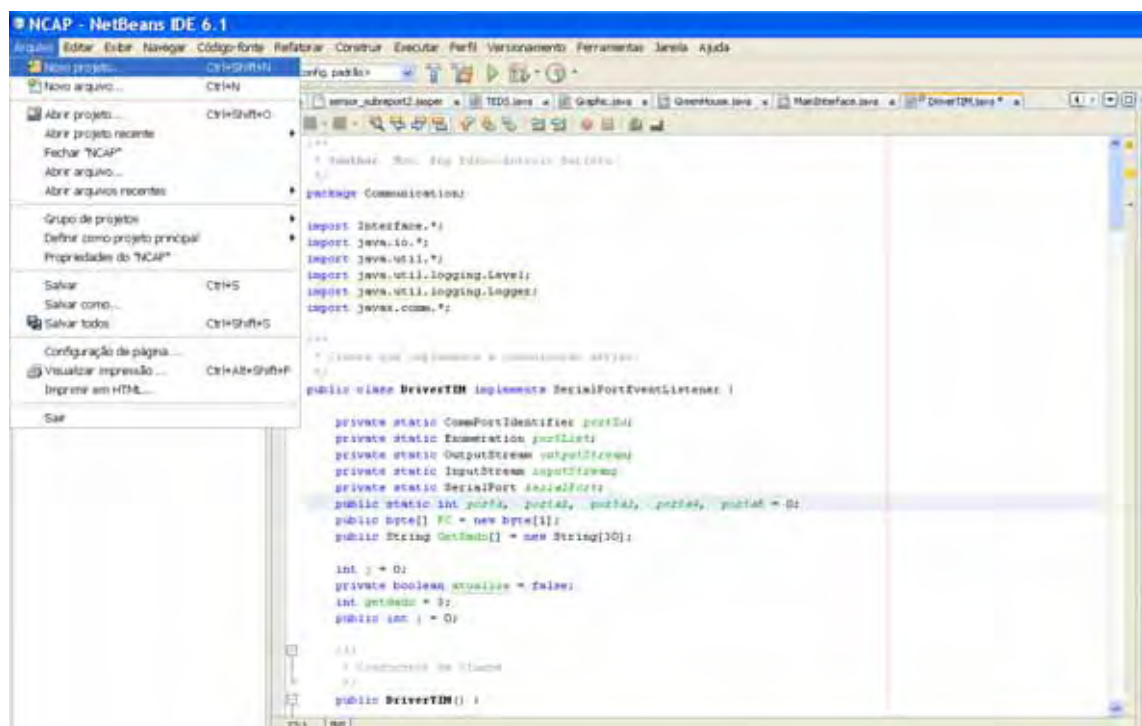


Figura 1B: Primeira etapa para gerar um diagrama *UML* no ambiente *NetBeans IDE 6.1*.

No *menu* *Novo Projeto* o usuário deve selecionar *Modelo de plataforma Java com engenharia reversa*, como mostra-se na Figura 2B.

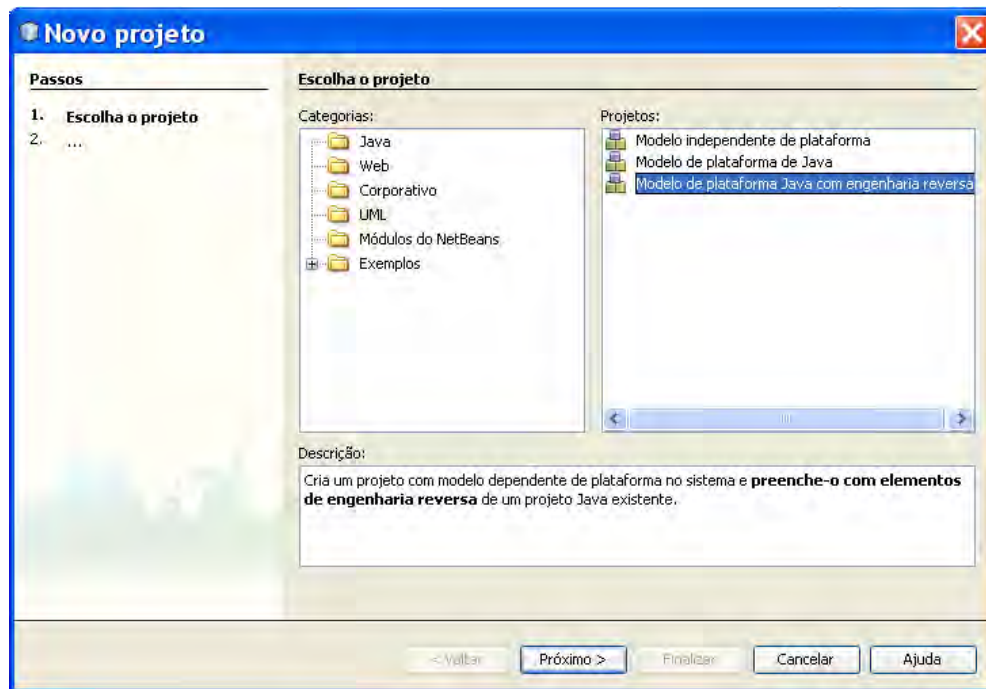


Figura 2B: Segunda etapa para gerar um diagrama *UML*.

Em seguida, o usuário pode nomear o projeto, selecionar o local onde o diagrama deve ser salvo e finalizar esta etapa, como pode-se observar na Figura 3B.

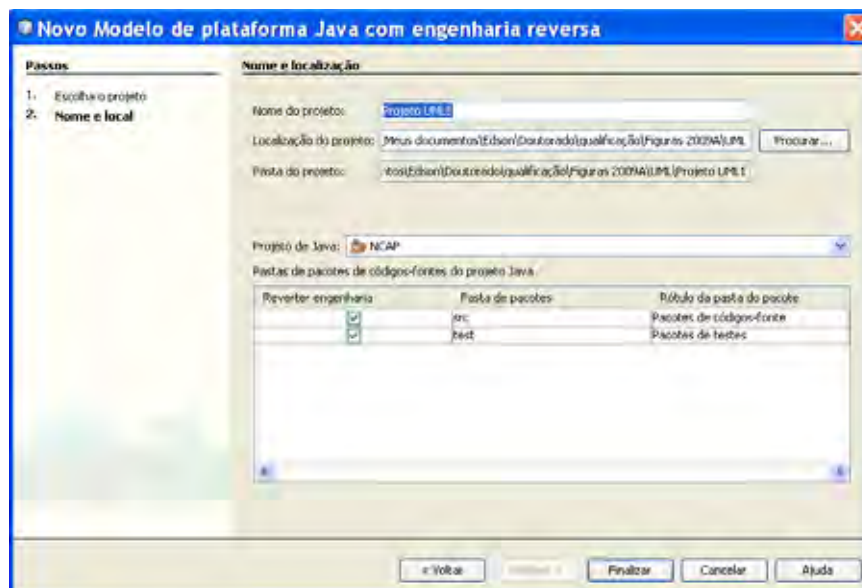


Figura 3B: Terceira etapa para gerar um diagrama *UML*.

O usuário deve selecionar todos os pacotes e classes que compõe a aplicação, em seguida clicar com o botão direito e selecionar a operação *Criar diagrama a partir de elementos selecionados*, como na Figura 4B.

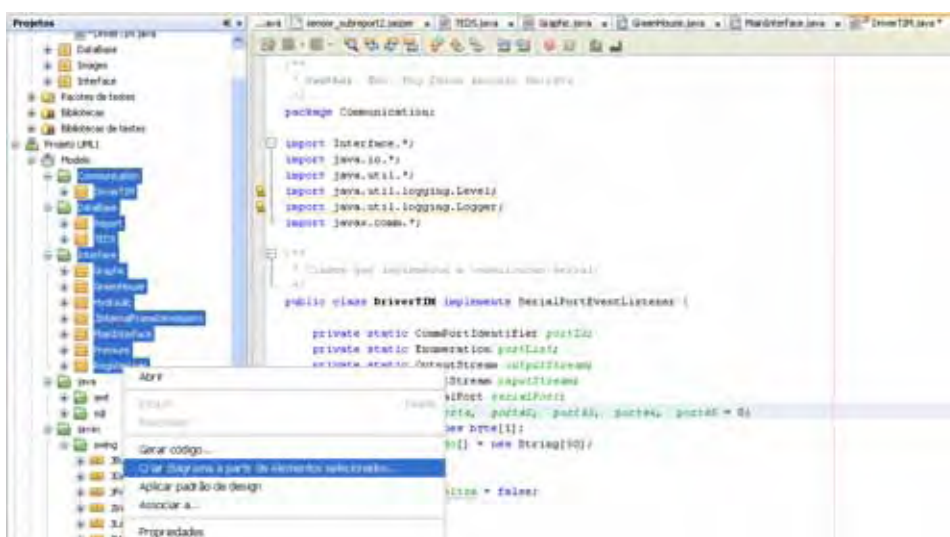


Figura 4B: Quarta etapa para gerar um diagrama *UML*.

No menu *Criar novo diagrama* o usuário deve selecionar o tipo de diagrama que deseja gerar e finalizar esta etapa, como mostra-se na Figura 5B.

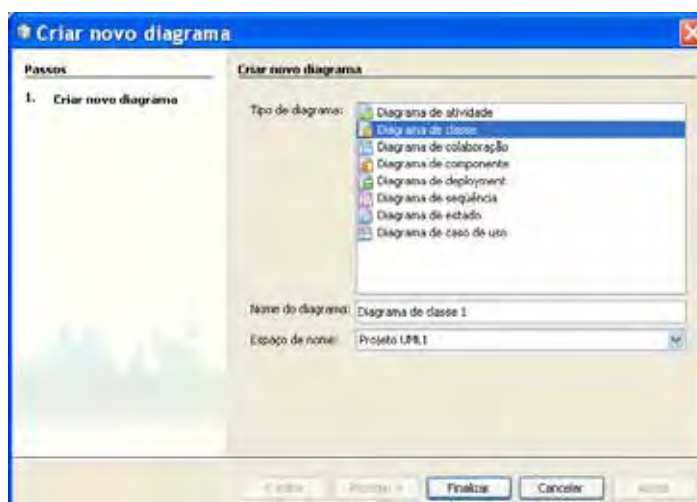


Figura 5B: Quinta etapa para gerar um diagrama *UML*.

Após finalizar a etapa de *Criar novos projetos*, o diagrama correspondente estará disponível na tela de trabalho, como apresenta-se na Figura 6B. Na Figura 6B pode-se observar que o diagrama de classe do software *NCAP*.

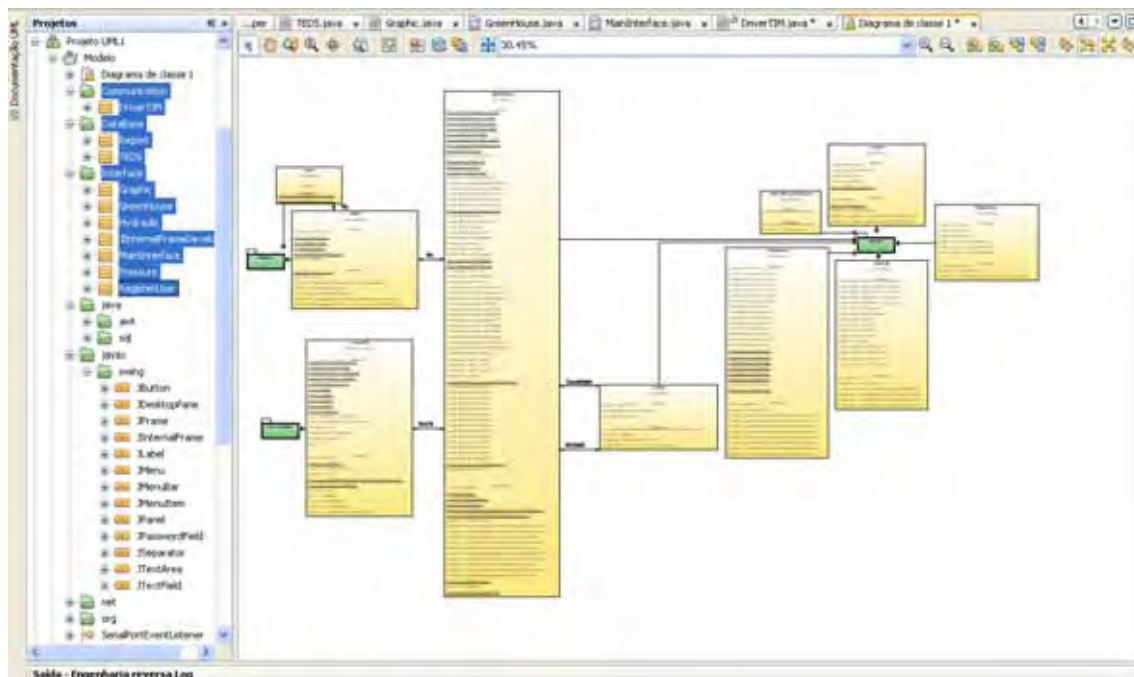
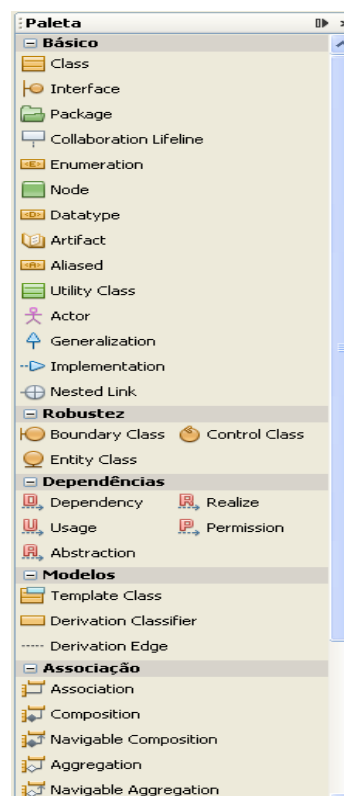


Figura 6B: Diagrama de classe do software *NCAP* gerado no *NetBeans 6.1*.

Na Figura 7B, pode-se observar o conjunto de simbologias disponíveis para desenvolver e interpretar um diagrama em *UML* no ambiente *NetBeans 6.1*.



Figuras 7B: Simbologias para diagrama *UML*.

Na Figura 8B, observa-se uma generalização e agregação de atributos entre duas classes, sendo que a classe Hidráulica herda os atributos da classe *DriverTIM*. A classe *DriverTIM* possui um *link* direto com a interface *Communication* e a classe *Hidráulica* está implementada a interface denominada de *Interface*, como se pode verificar na Figura 8B.

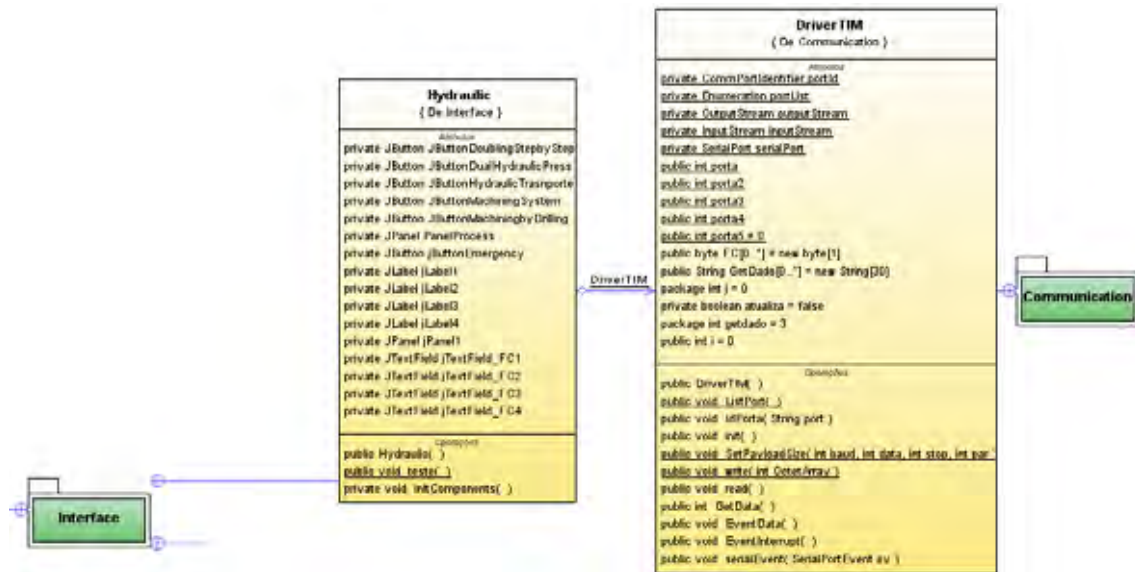


Figura 8B: Associação UML das classes *Hidráulicas* e *DriverTIM*.

Na Figura 9B, apresentam-se os tipos de associação *UML* das classes *Login*, *TEDS* e *Graphic* com as interfaces *Interface* e *Database*. Observa-se nesta Figura 9B que a classe *Login* possui uma agregação e generalização com a classe *TEDS* e a classe *Graphic* implementa a *Interface*.

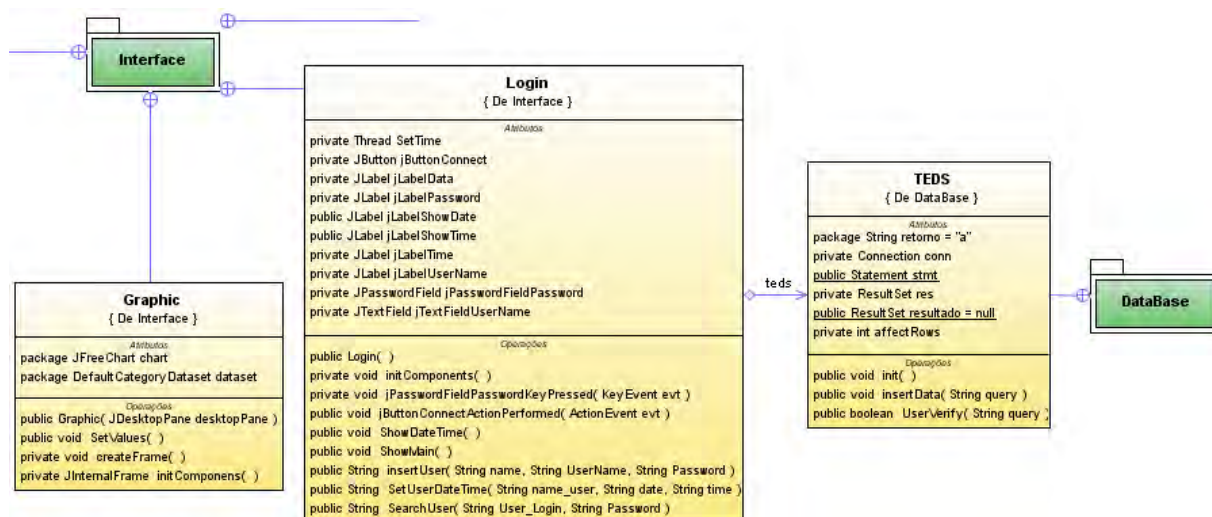


Figura 9B : Correlação entre as classes *Graphic*, *Login*, *TEDS* com as interfaces *Interface* e *Database*.

Na Figura 10B verificam-se as associações *UML* da classe *MainInterface* com a interface *Interface* e com a classe *Graphic*.

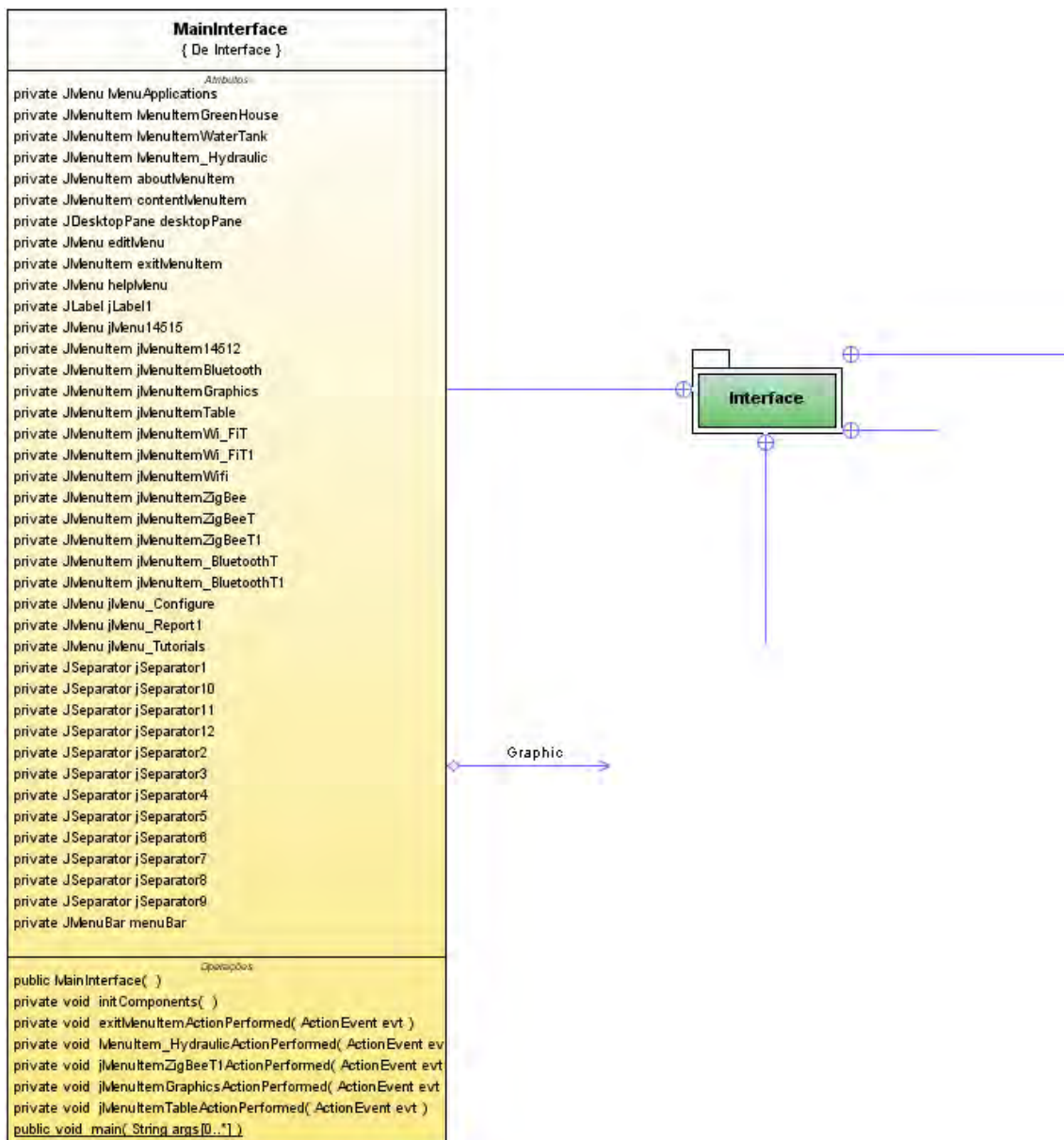


Figura 10B: Associação UML da classe *MainInterface* com a interface *Interface* e com a classe *Graphic*.

Na Figura 11B observa-se parte do código fonte do software NCAP, especificamente da etapa relacionada à geração da *TEDS*.

Na Figura 13B mostra-se a tela do software *NCAP* após a compilação e as respectivas aplicações disponíveis.

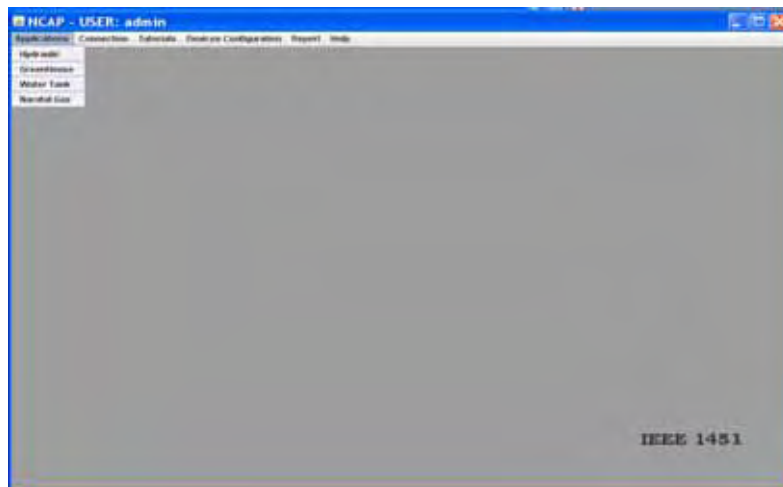
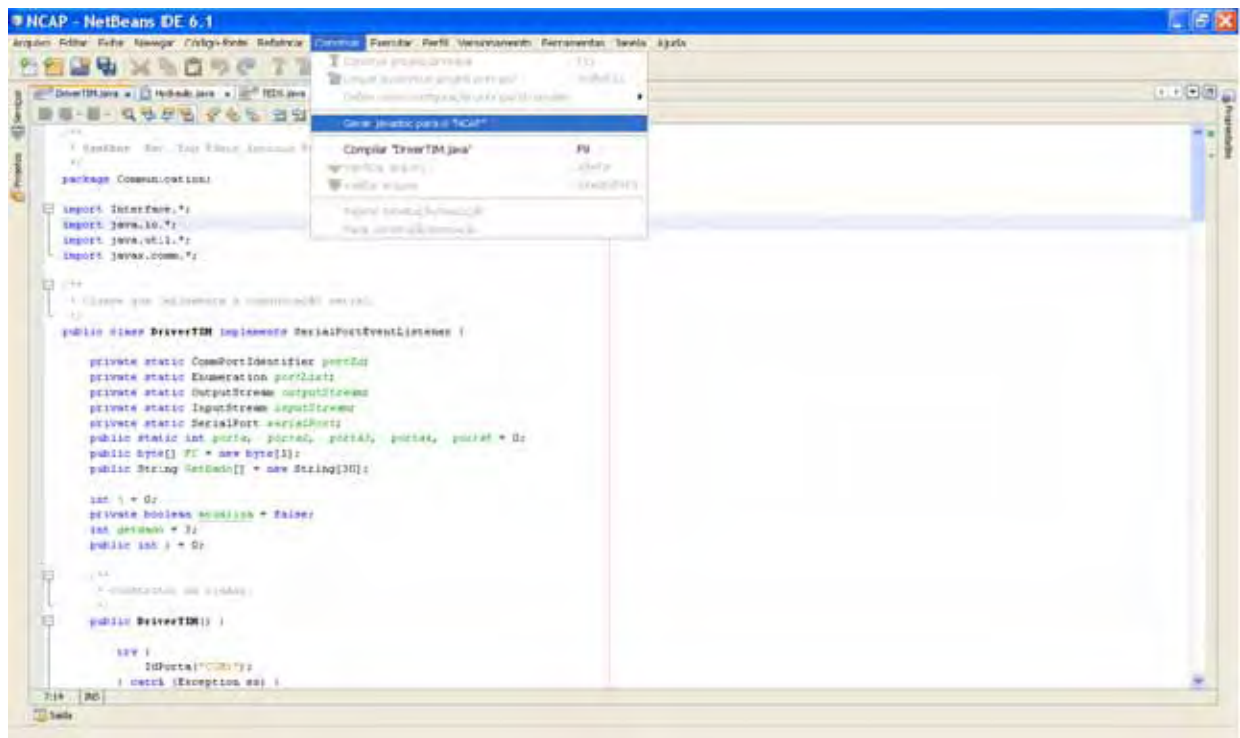


Figura 13B: Tela do *NCAP* para seleccionar a aplicação desejada.

Outra forma de realizar a documentação de uma aplicação no *NetBeans 6.1* é pelo *javadoc*. No *javadoc* a documentação é baseada na composição e descrição dos pacotes, classes e métodos. Este tipo de documentação é semelhante à utilizada pela *Sun Microsystems*. Para gerar o *javadoc* do software *NCAP*, inicialmente acionou-se o menu *Construir* e seleccionou-se a opção *Gerar javadoc para o NCAP*, como demonstrado na Figura 14B.



6.1.

Com a solicitação de geração de uma documentação *javadoc*, o projetista pode verificar no *prompt* o ambiente *NetBeans* se a operação foi realizada com sucesso, como observa-se na Figura 15B.

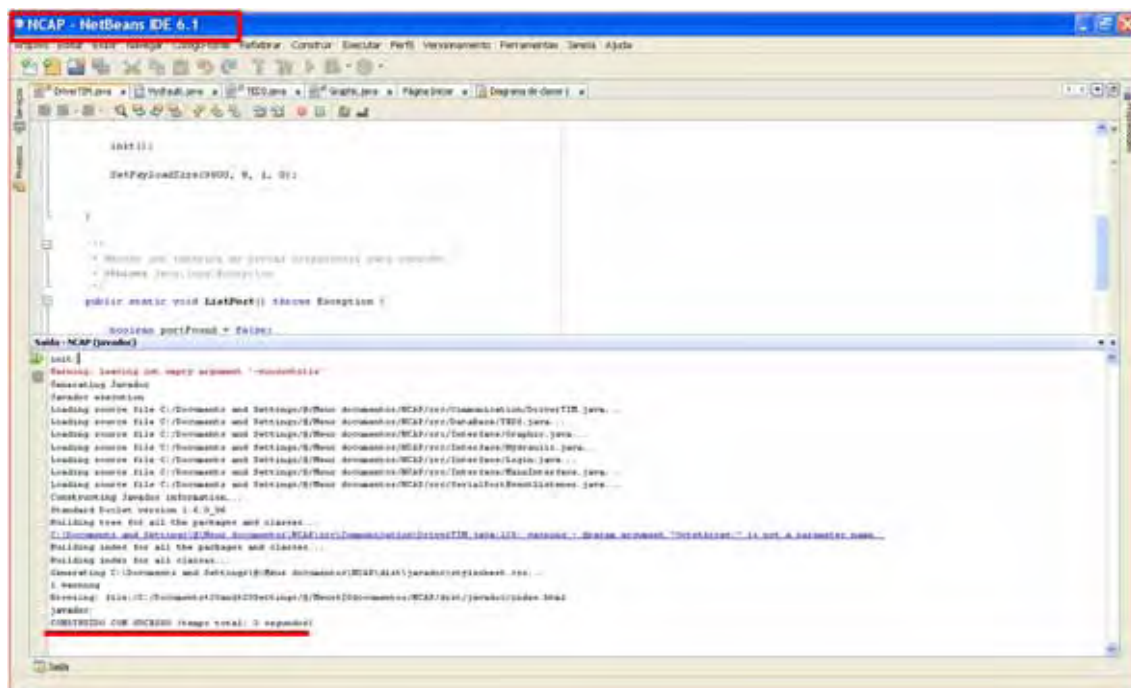


Figura 15B: Geração com sucesso do *javadoc* do software *NCAP*.

Na Figura 16B, mostra-se parte da documentação do *javadoc* do software *NCAP*, especificamente da *DriverTIM*. Nesta Figura 16B pode-se observar a descrição de cada método definido pelo projetista.

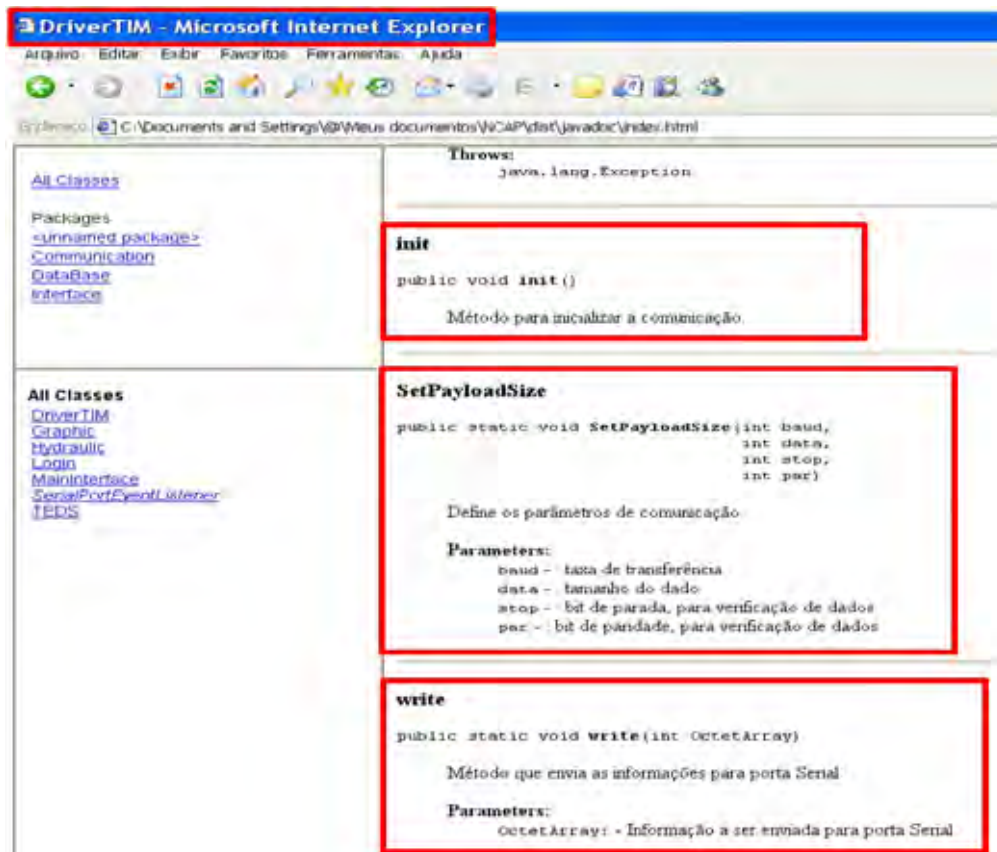


Figura 16B: Descrição dos métodos da classe *DriverTIM* na documentação *javadoc*.

Na Figura 17B, apresentam-se as descrições dos métodos que compõem a classe *Login*, com base na documentação *javadoc*.

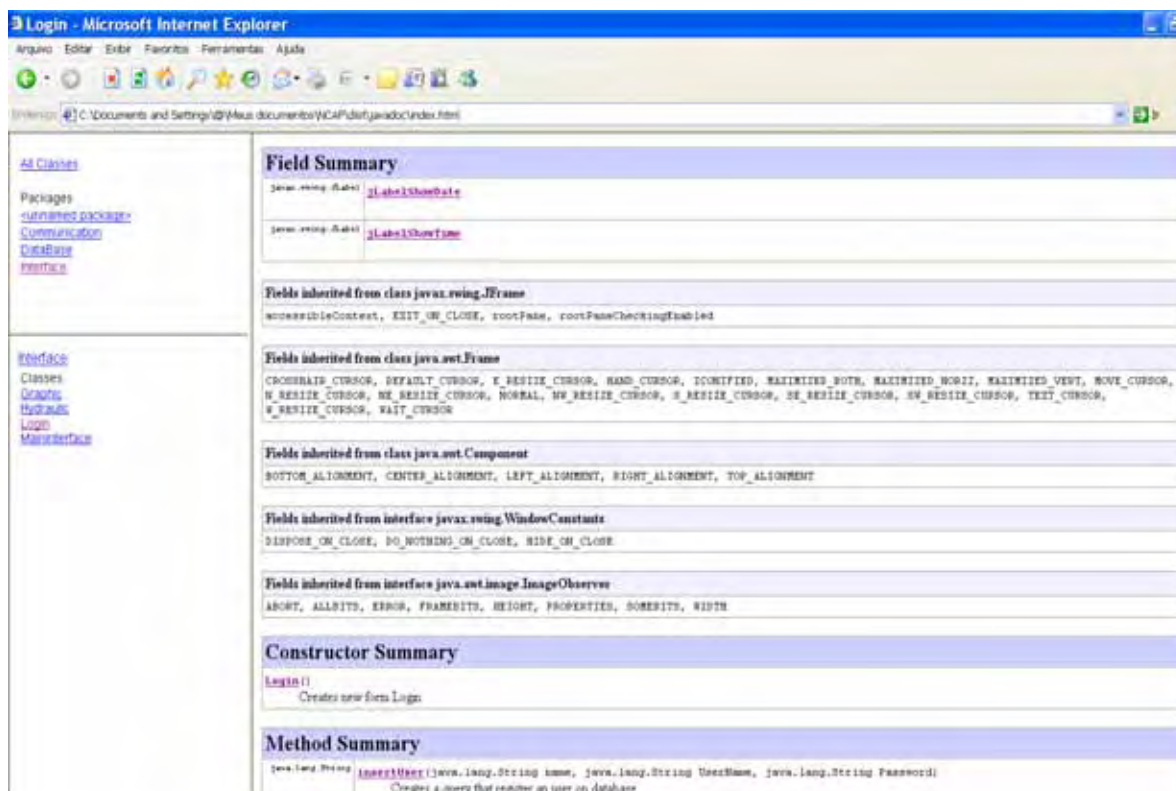


Figura 17B: Parte das descrições dos métodos da classe *Login*.

Na Figura 18B, encontra-se parte do *javadoc* da classe *Report*, pertencente ao software *NCAP*. Nesta classe utiliza-se o método da classe *TEDS* para disponibilizar as informações da rede de transdutor na tela de trabalho.

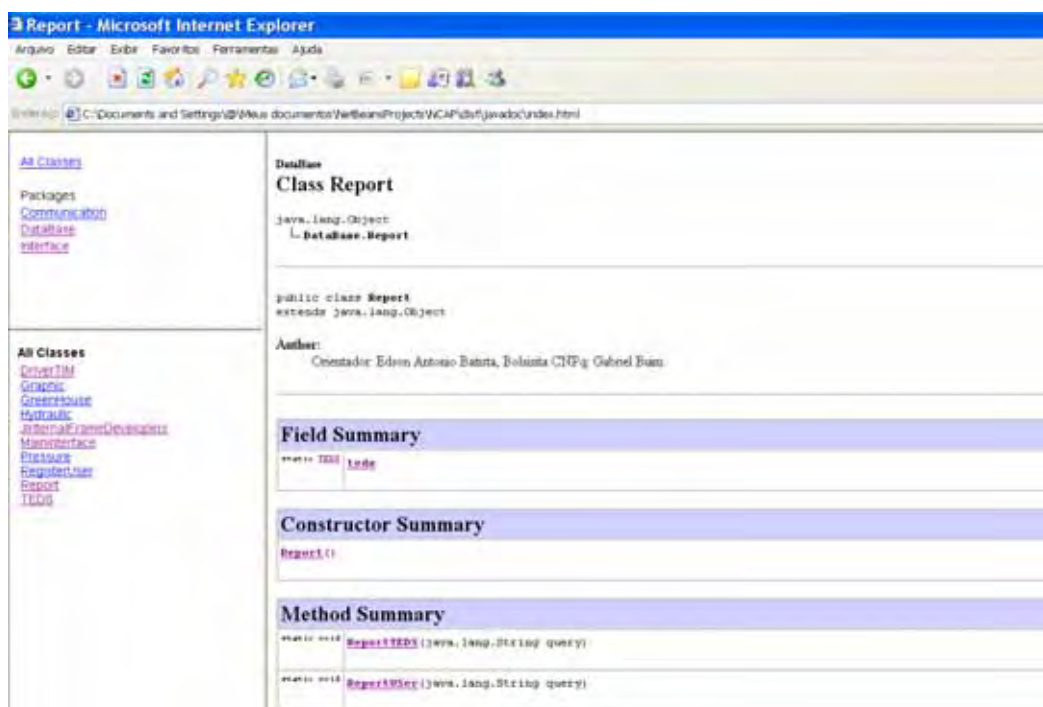


Figura 18B: Documentação da classe *Report* desenvolvida no software *NCAP*.

NA Figura 19B, apresenta-se o diagrama de classes, contendo todas as classes do software *NCAP*.

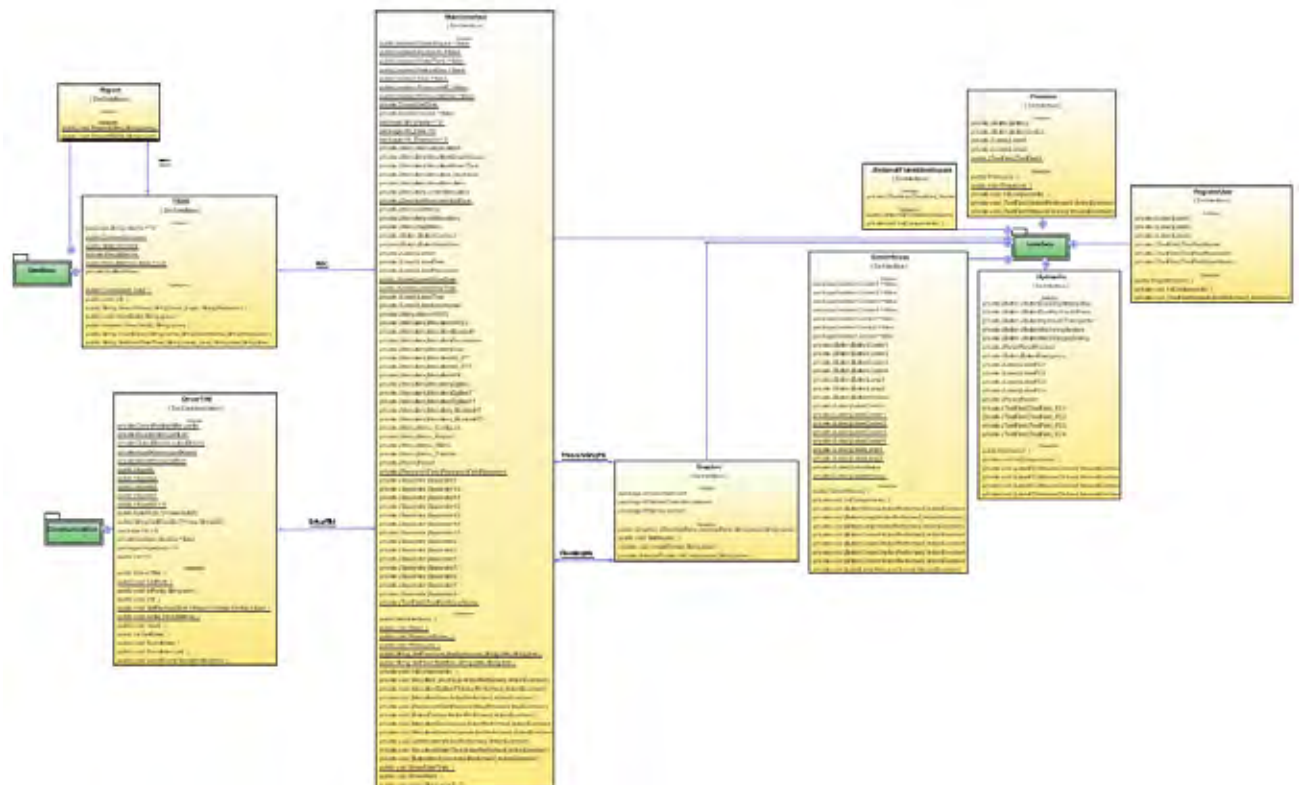


Figura 19B: Diagrama de classe contendo todas as classes do software *NCAP*.

APÊNDICE C – DESENVOLVIMENTO DE HARDWARE COM NIOS II

Para gerar um hardware utilizando o *SOPC Builder*, é necessário iniciar um projeto no *Quartus II* pelo modo gráfico e em seguida solicitar o auxílio do *SOPC*, ou seja, o *SOPC* é uma ferramenta do *Quartus II*. No *SOPC* gera-se um módulo em *VHDL* ou *Verilog*, que pode ser instanciado como as *MegaFunctions* da *Altera*.

O *SOPC Builder* permite ao usuário gerar um hardware em alto nível de abstração, porém é necessário ter conhecimento das conexões existentes entre sistemas de controle, barramento Avalon e periféricos e as configurações de todos os dispositivos utilizados no projeto. Na Figura 1C, ilustra-se a tela inicial do ambiente *SOPC Builder* e os recursos para desenvolvimento de hardware.

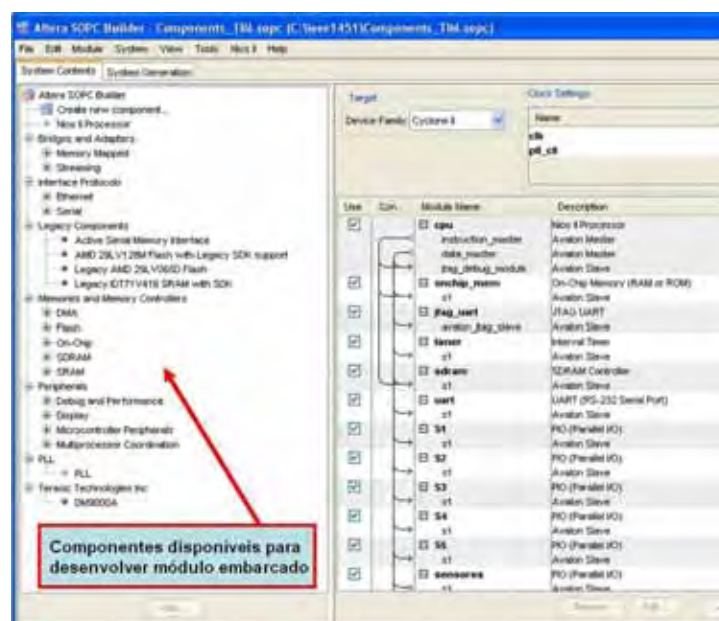


Figura 1C: Tela inicial do sistema integrado de desenvolvimento *SOPC Builder*.

O projetista seleciona os componentes através de “duplo clic” sobre o mesmo. Os componentes deverão estar interligados ao barramento *Avalon*, de acordo com a Figura 2C. Porém, conforme o projetista seleciona os componentes, o software (*SOPC*) fornece algumas conexões padrão.

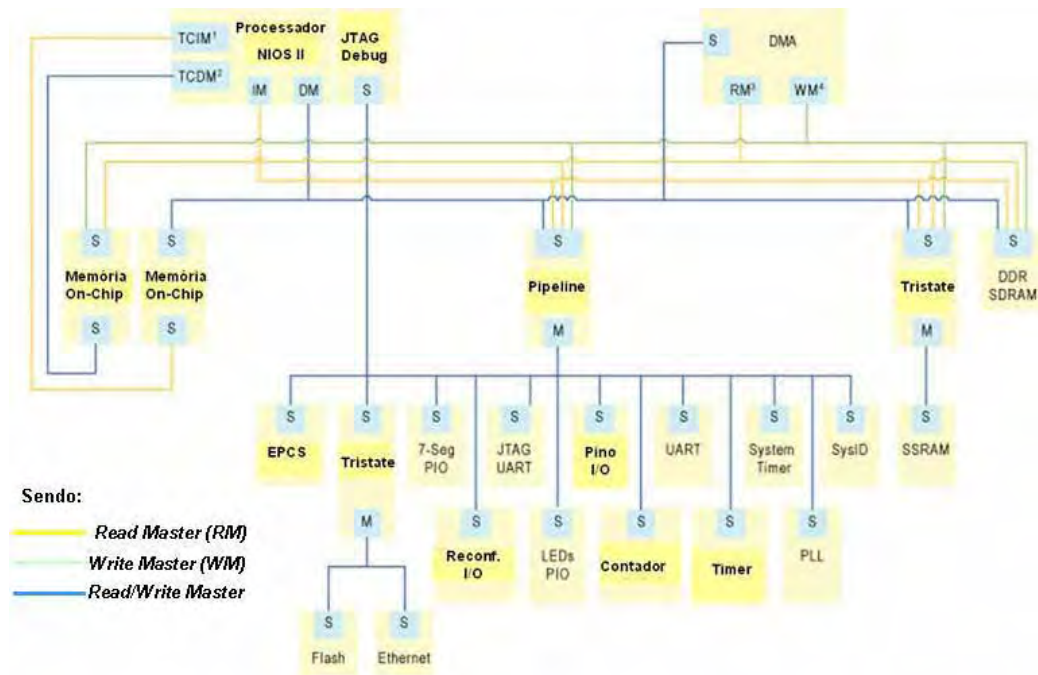


Figura 2C: Esquema de conexão entre periféricos e processador *NIOS II*.

Fonte: <http://www.altera.com/education/univ/materials/manual/unv-lab-manual.html>.

No ambiente *SOPC Builder*, o projetista tem as seguintes possibilidades:

- Escolher a família de *FPGA* que deseja trabalhar dentro das opções que suportam o *NIOS II*.
- Definir o *clock* de operação do hardware.
- Selecionar os componentes que devem compor o hardware.
- Visualizar as conexões entre os componentes.
- Definir as prioridades das interrupções.

Na Figura 3C observam-se as possibilidades que um projetista tem quando trabalha com *NIOS*.

versão do processador. Na Figura 4C mostra-se os parâmetros de configuração do processador *Nios II*.

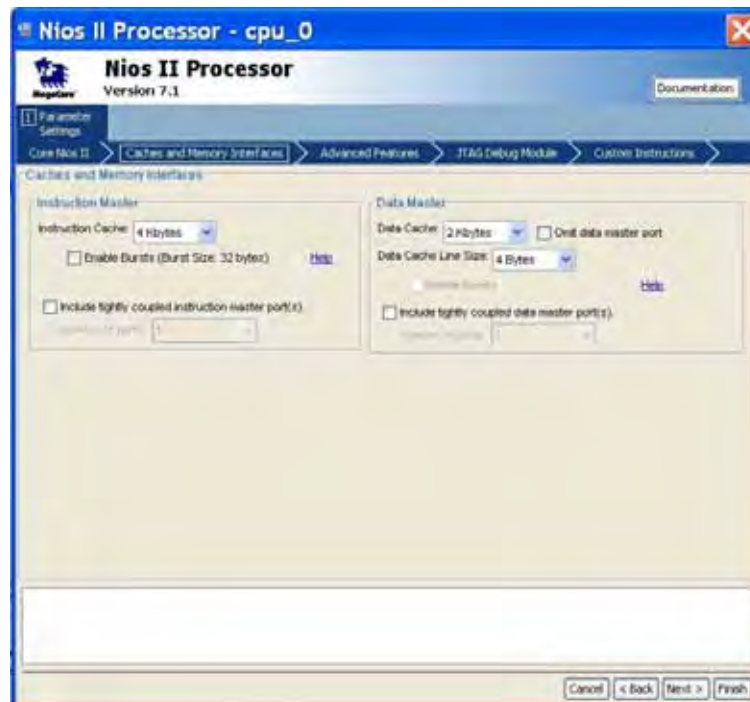


Figura 4C: Definição do tamanho das instruções e dados no *Nios II*.

Entre os componentes do *SOPC Builder*, encontra-se o Protocolo para Interfaces (*Interface Protocols*) no qual, está inserido o UART (*RS232 Serial Port*), este componente possibilita a construção de um *driver* serial para a comunicação entre *PC* e *FPGA*, utilizando o sistema *RS232*. Ao selecionar o componente *UART RS232* o projetista tem a possibilidade de configurar a comunicação, como por exemplo, a taxa de transmissão, o bit de paridade, o *stop bit* e o tamanho do dado. Na Figura 5C, apresenta-se uma tela configuração do *UART RS-232*.



Figura 5C: Configuração da *UART RS232* no *SOPC Builder*.

NIOS II

O processador *softcore NIOS II* refere-se à segunda geração de um microprocessador para módulos embarcados da *Altera*. O *NIOS II* é um processador de 32 bits que possui arquitetura de computador com um Conjunto Reduzido de Instruções, *Reduced Instruction Set Computer (RISC)*.

O *NIOS II* possui as seguintes versões :

- Econômica ou *NIOS II/e*: esta versão é a mais simples e possui 600 a 700 elementos lógicos com um desempenho de até 5 *Dhrystone Millions of Instruction per Second (DMIPS)*, implementado em *Cyclone II*⁶. Não possui memória *cache* nem *pipeline* na decodificação das instruções.
- Padronizada ou *NIOS II/s*: possui de 1200 a 1400 elementos lógicos, com desempenho de até 25 *DMIPS*, quando implementado em *Cyclone II*. Possui *cache* de dados e *pipeline* na decodificação das instruções.
- Rápida ou *NIOS II/f*: possui de 1400 a 1800 elementos lógicos, com desempenho de até 51 *DMIPS* quando implementado em *Cyclone II*. É a versão mais completa, incluindo *cache* de dados e de instrução, 6 níveis de *pipeline* na

⁶ *Cyclone II* é uma família de dispositivo de lógica programável.

decodificação das instruções. Na Figura 6C apresentam-se as opções e características de um processador *NIOS II* baseadas no *FPGA Cyclone II*.

Nios II/e	Nios II/s	Nios II/f
RISC 32-bit	RISC 32-bit Cache de Instruções Tecnologia de Previsão de Saltos Multiplicação de Hardware Divisão de Hardware	RISC 32-bit Cache de Instruções Previsão de Saltos Multiplicação de Hardware Divisão de Hardware Deslocador Programável Cache de Dados Previsão Dinâmica de Saltos
Até 5 DMIPS 600-700 LEs 2 M4Ks	Até 25 DMIPS 1200-1400 LEs 2 M4Ks+cache	Até 51 DMIPS 1400-1800 LEs 3 M4Ks+cache

Figura 6C: Versões e características do *NIOS II* para o *Cyclone II*.

As principais características do processador *NIOS II* são :

- 32 bits de instrução.
- 3 formatos de instrução.
- 32 bits de dados de endereçamento.
- 32 interrupções prioritárias.
- 32 registradores.

Como o *NIOS II* permite o desenvolvimento de sistemas complexos e trata-se de um sistema embarcado, este processador pode ser utilizado em áreas como: aeroespacial, TV digital, domótica, automação industrial, etc. As aplicações do *NIOS II* segundo a *Altera* são para os seguintes sistemas:

- Que necessitem de alta capacidade de processamento.
- Que possam ser reconfigurados em campo.
- Com múltiplas *CPUs*.
- Que implementam as funções de um processador *RISC* de 32 *bits*.

Ambiente NIOS II IDE

O ambiente *NIOS II IDE (Integrated Development Environment)* é utilizado para desenvolver o software do processador *NIOS II* em conformidade com hardware gerado no *SOPC Builder*. Atualmente a programação neste *IDE*, pode ser desenvolvida pelas linguagens *C*, *C++* ou *assembly*. Existe a possibilidade de utilizar a tecnologia *Java* na programação das atividades do processador *NIOS II*, sem a necessidade de instalação de *RTOS (Real Time Operating System)*, tornando este *softcore* mais flexível.

No ambiente *NIOS II IDE* são disponibilizados *templates* e bibliotecas para o projetista utilizar como base no desenvolvimento do software. O projetista pode selecionar o template conforme as necessidades da aplicação ou optar por criar seu próprio template. Na Figura 7C ilustra-se as definições necessárias para iniciar o desenvolvimento de uma aplicação no *NIOS II IDE*. Pode-se observar na Figura 7C que em (1) encontra-se os templates disponíveis pelo fabricante, em (2) apresenta-se uma breve descrição das principais funções do template correspondente, em (3) está o template selecionado e em (4) mostra-se o diretório contendo o hardware que terá o processador programado.

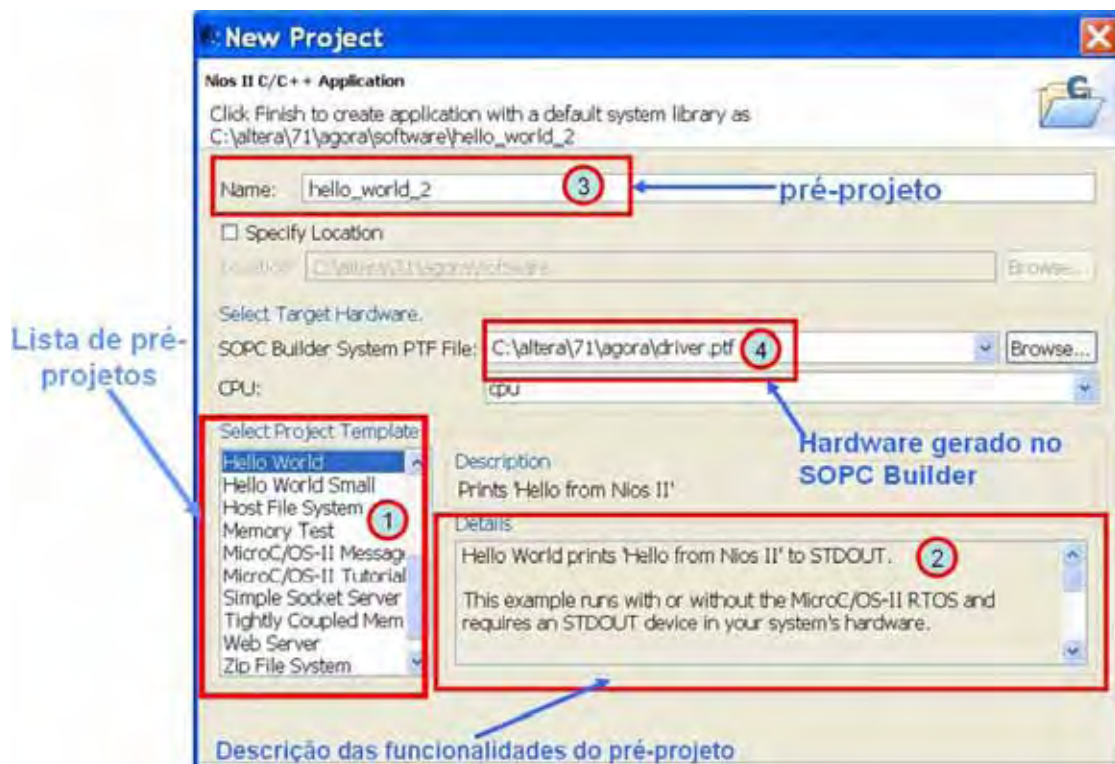


Figura 7C: Definição de parâmetros para iniciar projeto no *NIOS II IDE*.

Para criar um template e disponibilizá-lo na caixa de seleção *Select Project Template*, deve-se executar as seguintes etapas:

1. Criar uma pasta no diretório \72\nios2eds\examples\software, sendo que o nome da pasta corresponderá ao nome do *template*.
2. Inserir-se nesta pasta os programas que irão compor o *template*.
3. A descrição das funcionalidades do *template* que aparece na área de texto é inserida em um arquivo *XML*.

Os templates devem fornecer funções para reduzir e/ou facilitar a programação do *softcore NIOS II*. Na Figura apresenta-se o ambiente de programação *NIOS II IDE*, sendo que ao iniciar uma programação do processador deve-se informar o diretório no qual se encontra o arquivo *.ptf* e o *template* relacionado. O projetista pode selecionar diferentes *template* para uma mesma *CPU*, como observa-se nesta Figura 8C.

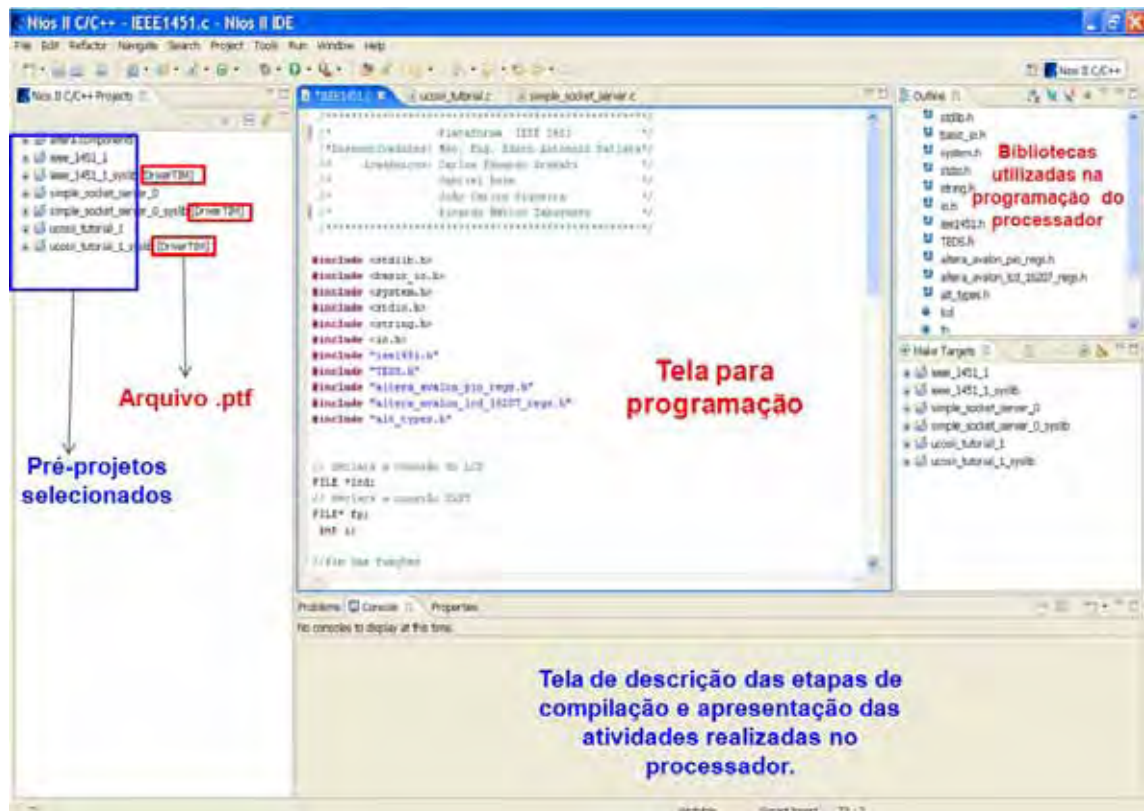


Figura 8C: Ambiente de programação *NIOS II IDE*.

As bibliotecas pertencentes ao *NIOS II IDE* estão codificadas em linguagem *C* e fornecem informações importantes para acessar os periféricos do módulo embarcado. Na Figura 9C mostra-se parte das bibliotecas *system.h* e *basic_io.h* geradas de acordo com os componentes selecionados no *SOPC Builder*, sendo que na biblioteca *system.h* apresenta-se as configurações e endereço dos componentes e na biblioteca *basic_io* encontra-se as funções para comunicação com periféricos.

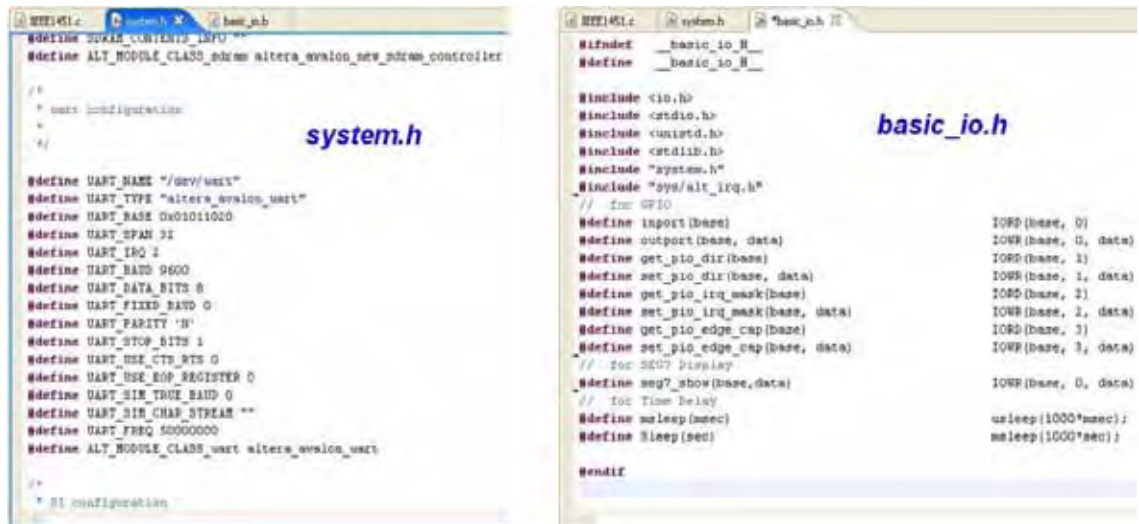


Figura 9C: Bibliotecas *system.h* e *basic_io.h*.

O projetista deve informar as propriedades da parte software do sistema embarcado, como por exemplo, onde será alocada a programação (memória: *SDRAM* ou *on-chip* ou *flash*), se a aplicação necessita de Sistemas Operacionais de Tempo Real, *Real-Time Operating Systems (RTOS)*, etc. Estas propriedades são definidas no *menu Properties*, como mostra-se na Figura 10C.

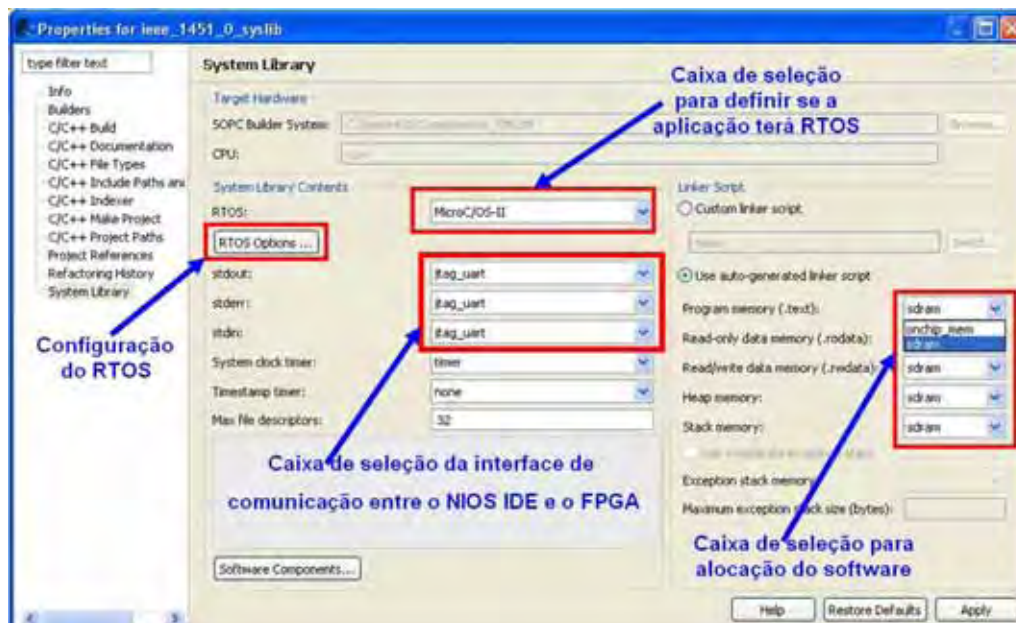


Figura 10C: Configurações do projeto no ambiente *NIOS II IDE*.

O ambiente *NIOS IDE* possui um terminal que informa aos projetistas detalhes da compilação, descreve os erros ocorridos na programação, serve para interagir com as atividades do módulo embarcado. Na Figura , pode-se observar a localização do terminal e as mensagens informando o processo de compilação de um software.

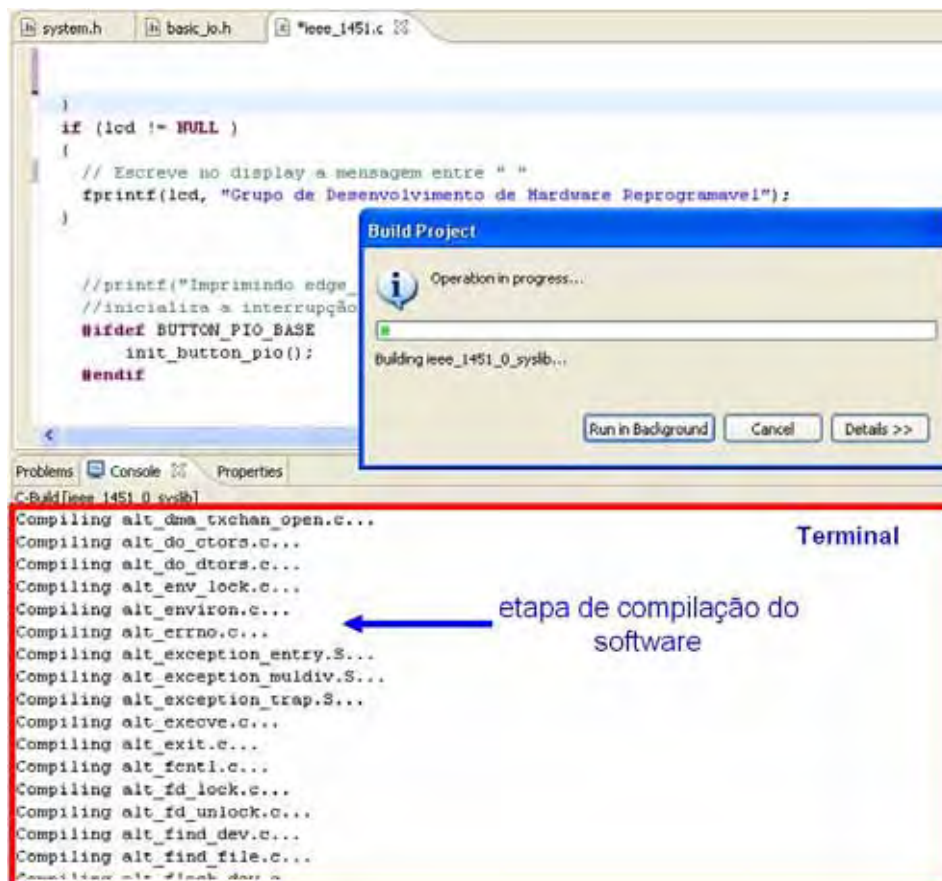


Figura 11C: Etapa de compilação demonstrada no terminal *NIOS II IDE*.

Na Figura observam-se no terminal, as mensagens, as fontes e os locais dos erros ocorridos durante a etapa de compilação.



Figura 12C: Mensagens de Erros apresentadas no terminal *NIOS II IDE*.

Na Figura verifica-se a interação entre o terminal *NIOS II* com uma aplicação, sendo que em: (1) destaca-se o template utilizado na programação do hardware; (2) aparece uma mensagem comum para aplicações que utilizam a licença gratuita, esta etapa só é possível se a compilação do software não apresentar erro; (3) apresenta-se a interação programada entre o terminal e a aplicação.

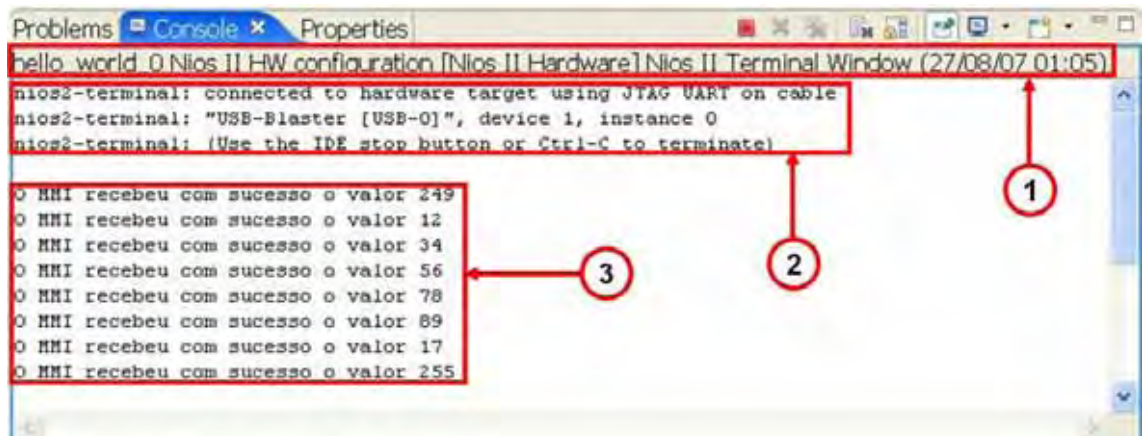


Figura 13C: Interação entre terminal *NIOS* e uma aplicação.

APÊNDICE D – INFORMAÇÕES COMPLEMENTARES

Com intuito de complementar as informações do texto deste trabalho, neste apêndice mostra-se diversas figuras dos experimentos e circuitos eletrônicos utilizados no desenvolvimento da plataforma *IEEE 1451*.

Nas Figuras 1D, 2D e 3D mostram-se algumas etapas de desenvolvimento do circuito eletrônico, responsável por viabilizar o acionamento dos solenóides ($24V$) do *MHI-01* a partir de uma entrada de $5V$ da placa *DE2*.



Figura 1D: Circuitos eletrônicos utilizados para os atuadores do *MHI-01* implementados em *protoboard*.



Figura 2D: Circuitos eletrônicos utilizados para os atuadores do *MHI-01* implementado em placa de circuito impresso.



Figura 3D: Circuitos eletrônicos utilizados para os atuadores do *MHI-01* protegidos por caixa de acrílico.

Nas Figuras 4D e 5D apresentam-se algumas configurações realizadas para testar a comunicação sem fio.

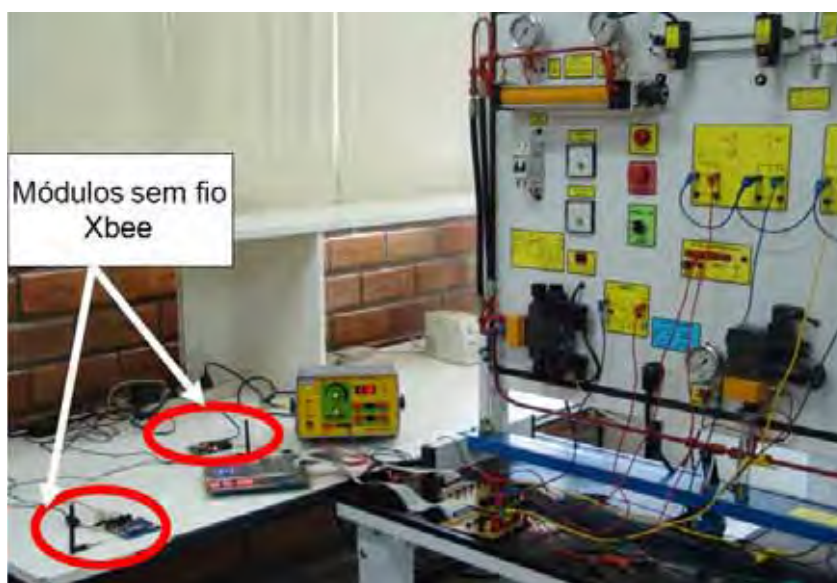


Figura 4D: Configuração para automação sem fio utilizando a tecnologia *ZigBee*.



Figura 5D: Configuração para automação sem fio utilizando a tecnologia *Wi-Fi*.

Os primeiros testes realizados para analisar a comunicação serial e a transmissão sem fio montou-se uma configuração (*SAF0*) que possibilitasse a inserção de um número inteiro por meio de uma interface gráfica e o mesmo deveria ser devidamente transmitido. Esta transmissão de dados foi baseada no sistema de comunicação cliente/servidor. Na Figura 6D mostra-se a arquitetura do servidor utilizada para testar a transmissão sem fio com tecnologia *ZigBee*.

Figura 6D: Configuração do servidor na transmissão sem fio com base na tecnologia *ZigBee*.

Na Figura 7D, apresenta-se a configuração cliente com tecnologia *ZigBee*.

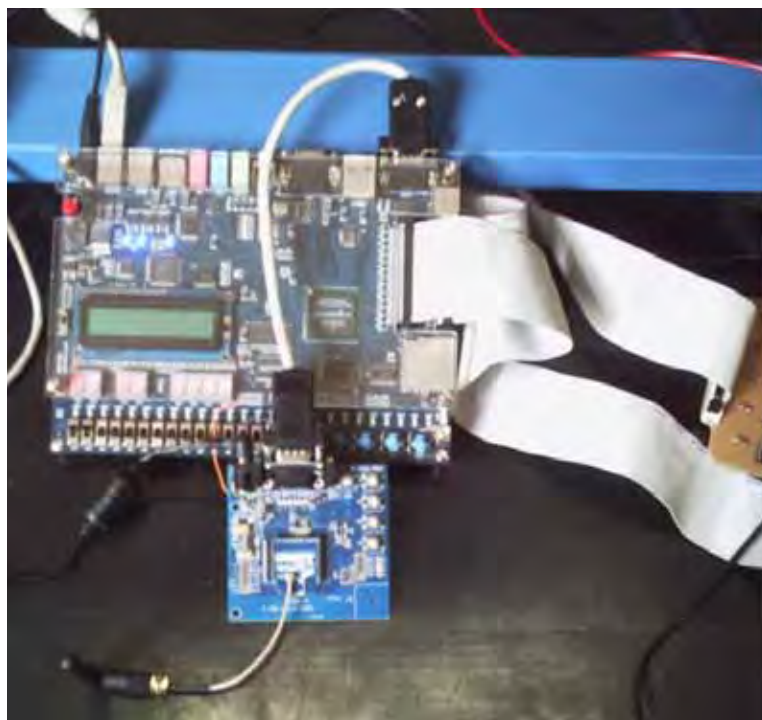


Figura 7D: Configuração do cliente na transmissão sem fio com base na tecnologia *ZigBee*.

Na automação do processo de filtragem de Biogás, testada neste trabalho, uma das etapas foi capturar a vazão de um fluido através de pulso de tensão. Estes pulsos

foram enviados para placa *DE2* que processa as informações e envia para o *PC*. A sincronia da frequência dos pulsos gerados (recebido pela *DE2*) e de recepção deve estar próxima.

Desta forma, montou-se um sistema que permita a placa *DE2*, operar com uma frequência externa a placa para captar os pulsos como se mostra na Figura 8D. O osciloscópio que aparece na Figura 8D monitorou a forma de onda enviada ao *PC*.

Figura 8D: Configuração utilizada para simular a captação de pulsos do medidor de vazão.

Sendo que:

- 1) Osciloscópio demonstrando as ondas geradas pelo gerador de função (2), no qual simula os pulsos do medidor de vazão.
- 2) Gerador de função simulando os pulsos gerados pelo medidor de vazão que opera a 1m^3 por pulso é conectado na porta paralela da placa *FPGA* (5) por um cabo *flat*.
- 3) Notebook utilizando o software *Nios II* para visualização da frequência gerada pelo gerador de função (2), no qual, simula os pulsos do medidor de vazão, com essa frequência realiza-se um cálculo para obtenção do valor real da vazão do gás.

- 4) Gerador de função simulando a frequência de base (1 *KHz*) necessária para realizar os cálculos desenvolvidos em *VHDL* e implementados na placa *FPGA* (5), conecta-se um fio simples a entrada de *clock* externo da placa.
- 5) Placa *FPGA* onde desenvolve um hardware e então é importado à placa para realizar os cálculos de obtenção da vazão, no qual, por meio dos *leds* são representados estes valores.

Na Figura 9D, pode-se observar a configuração utilizada na automação do processo do Biogás, sendo que:

- 1) Fonte A é ajustada em 24 V e serve para alimentar o manômetro, medidor de vazão e o circuito para conversão Corrente-Tensão.
- 2) Fonte B ajustada em 12 V serve para alimentar o relé para acionamento do compressor.
- 3) Gerado de Função serve para gerar a frequência de base (1 *KHz*) na realização de cálculos desenvolvidos em *VHDL* e importados na placa *FPGA*. Esta frequência é conectada na entrada de *clock* externo da placa *DE2*. Frequência de base tem a principal função de um timer, ou seja, como um tempo para que possa ser analisado cada pulso gerado pelo medidor de vazão e realizar as operações aritméticas para obter os valores da vazão.
- 4) Osciloscópio monitora os pulsos do medidor de vazão.



Figura 9D: Instrumentação de auxílio para viabilizar parte da automação do processo de Filtragem de Biogás.

A plataforma *IEEE 1451* foi testada também na automação de kit Nível de Líquido da empresa *Bit9*. Neste processo monitoraram-se os níveis de líquido de cada tanque com uma configuração, conforme mostra-se na Figura 10D.



Figura 10D: Configuração utilizada na automação do Nível de Líquido da Bit9.

Na Figura 11D apresenta-se uma configuração para monitoramento da pressão via transmissor de pressão e com o software *NCAP*. Este sistema foi apresentado em uma feira de ciências realizada na *UCDB*.



Figura 11D: Medição da pressão via *NCAP* (gráfico) e através do transmissor de pressão.

Na Figura 12D mostra-se o medidor de pressão instalado no biodigestor da estação de tratamento da Água Guariroba. Este medidor ficou monitorando a pressão do biodigestor durante quatro dias ininterruptamente, sendo que, os dados foram devidamente armazenados no software *NCAP* e posteriormente avaliados por uma equipe técnica.



Figura 12D: Medidor de pressão instalado no biodigestor da Estação de tratamento da Águas Guariroba.

Na Figura 13D, mostra-se parte da configuração para testar a automação sem fio do processo de filtragem do Biogás em conformidade com a *plataforma IEEE 1451*.



Figura 13D: Configuração utilizada para testar a automação sem fio do processo de Filtragem do Biogás.

Plataforma IEEE 1451 - Automação de uma Maquete de Casa de Vegetação

A utilização de casa de vegetação é um segmento importante da agroindústria que busca a otimização na produção de mudas. Sendo assim, o controle de parâmetros como temperatura, luminosidade, nutriente e umidade são fundamentais em uma casa de vegetação. Neste trabalho desenvolveu-se uma maquete para representar as atividades de uma Casa de Vegetação, sendo composta por duas lâmpadas incandescentes, quatro coolers, um sensor de temperatura e um motor de passo.

As lâmpadas são de $150W$ e $12V$ e tem a função de aquecer o ambiente, três *coolers* funcionam como ventilador, um *cooler* atua como exaustor e o motor de passo controla o posicionamento de uma janela com intuito de refrigerar o ambiente. Na automação desta maquete foi necessária a utilização de conversores *A/D*, circuito eletrônico para controlar o funcionamento de motor de passo, além dos circuitos para acionar as lâmpadas e os *coolers*. Assim, o número de canais para compor o *TIM* são de 8 (oito), sendo 1 (um) canal de entrada e 7 (sete) de saída. Porém, o sensor deve ocupar 3 pinos do barramento de expansão da *DE2* e o motor de passo 4 pinos do barramento de expansão, os demais utilizam um pino de saída do módulo embarcado. Na Figura 14 D visualiza-se o *DriverTIM* que compõe o *TIM* para automação da Maquete de uma Casa de Vegetação.

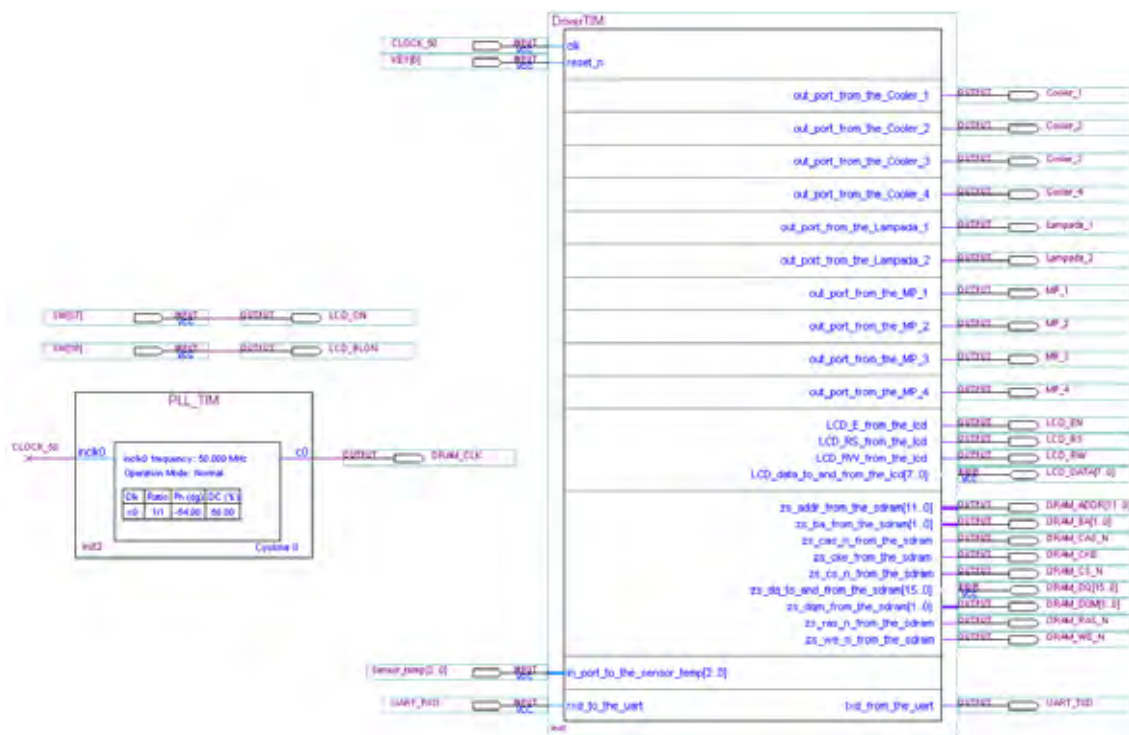


Figura 14D: *DriverTIM* para automação da Casa de Vegetação.

Os recursos utilizados na implementação do módulo embarcado que viabiliza a automação desta maquete são apresentados na Tabela :

Tabela 1D: Recursos do *TIM*, para automação do kit de hidráulica no *Cyclone II*.

Logic Cell	Dedicated Logic Register	Memory Bit	Total de Pinos
4277, 13%	2422, 7%	228736, 47%	74, 16%

Para acionar as lâmpadas e os *coolers* implementou-se uma placa de circuito impresso semelhante à utilizada no acionamento dos solenóides. A movimentação do motor de passo foi controlada com auxílio do *CI UNL2003* e na leitura da temperatura utilizou-se um conversor *A/D CI0804*. Os circuitos eletrônicos utilizados na automação da Maquete de uma Casa de Vegetação são apresentados na Figura .



Figura 15D: Placas de circuito impresso para automação da Casa de Vegetação.

Estas placas de circuitos eletrônicos foram devidamente fixadas na maquete com a finalidade de tornar o sistema mais versátil.

Para realizar a automação da maquete de uma Casa de Vegetação montou-se uma configuração conforme apresentada na Figura 16D, sendo que as proteções das lâmpadas com placas de alumínio são para minimizar a intensidade luminosa para os observadores. Nesta Figura 16D tem-se:

- 1) Conjunto de *cooler*.
- 2) Sensor de temperatura.
- 3) Tecnologia *ZigBee*.
- 4) Fonte para alimentar o relé, o *CI LM* e o conversor *A/D*.
- 5) Fonte para alimentar os módulos *ZigBee* e a placa *DE2*.
- 6) Placa *DE2*.
- 7) *PC* utilizado na automação.
- 8) Placas de circuito impresso para condicionamento e conversão de sinais.
- 9) Multímetro utilizado para monitorar a tensão de saída do sensor de temperatura.
- 10) Localização do motor de passo para movimentar a janela.

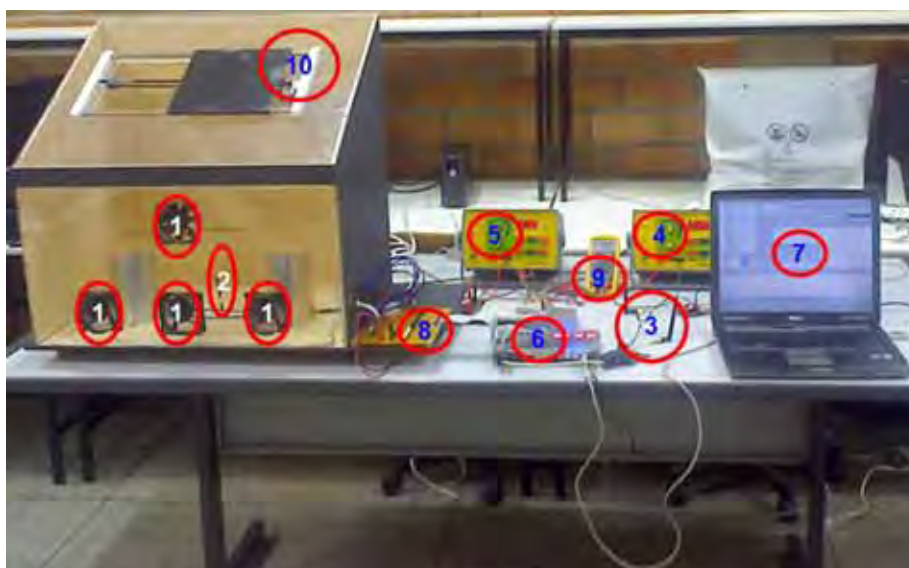


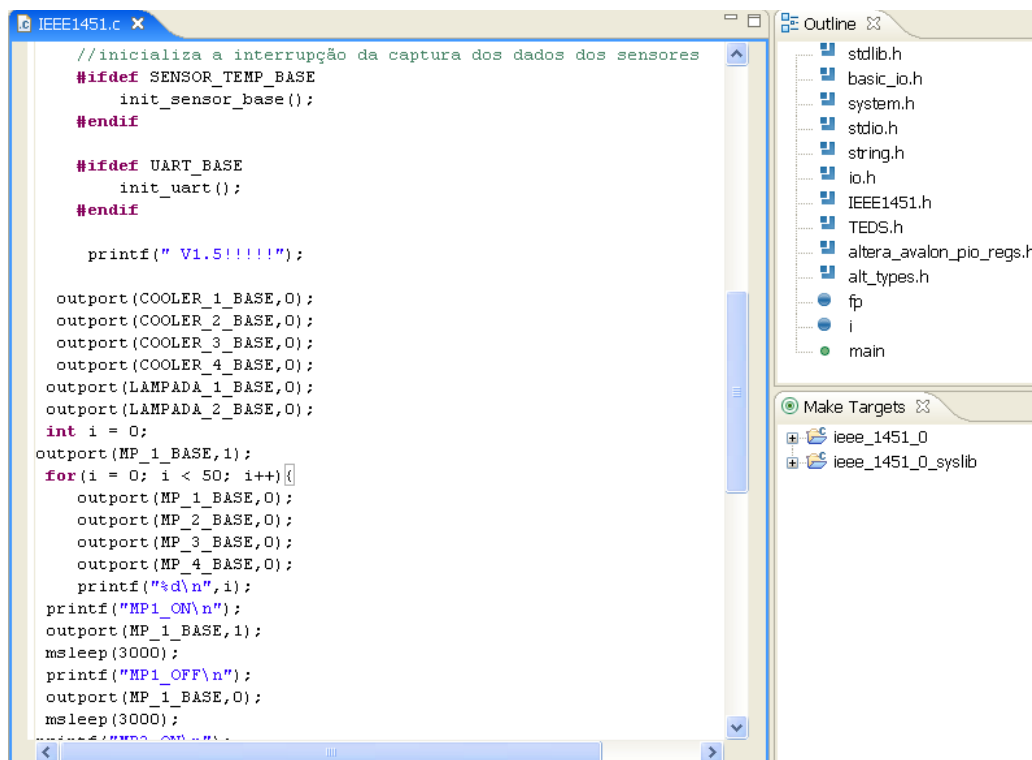
Figura 16D: Configuração utilizada para testar a automação da maquete de uma casa de vegetação.

Na Figura 17D apresenta-se outra configuração utilizada na automação da maquete de casa de vegetação, neste caso, a conexão é realizada com módulos *Wi-Fi*, para desenvolver a interface *IEEE 1451.5*.



Figura 17D: Automação de uma Maquete Casa de Vegetação.

Na Figura apresenta-se parte do código que controla as atividades do processador *NIOS II*.



```
//inicializa a interrupção da captura dos dados dos sensores
#ifdef SENSOR_TEMP_BASE
    init_sensor_base();
#endif

#ifdef UART_BASE
    init_uart();
#endif

printf(" V1.5!!!!");

outport(COOLER_1_BASE,0);
outport(COOLER_2_BASE,0);
outport(COOLER_3_BASE,0);
outport(COOLER_4_BASE,0);
outport(LAMPADA_1_BASE,0);
outport(LAMPADA_2_BASE,0);
int i = 0;
outport(MP_1_BASE,1);
for(i = 0; i < 50; i++){
    outport(MP_1_BASE,0);
    outport(MP_2_BASE,0);
    outport(MP_3_BASE,0);
    outport(MP_4_BASE,0);
    printf("%d\n",i);
    printf("MP1_ON\n");
    outport(MP_1_BASE,1);
    msleep(3000);
    printf("MP1_OFF\n");
    outport(MP_1_BASE,0);
    msleep(3000);
}
```

Figura 18D: Lógica de controle no *NIOS IDE* – Casa de Vegetação.

Na Figura , mostra-se as informações de uma das lâmpadas que compõe a Maquete da Casa de Vegetação.

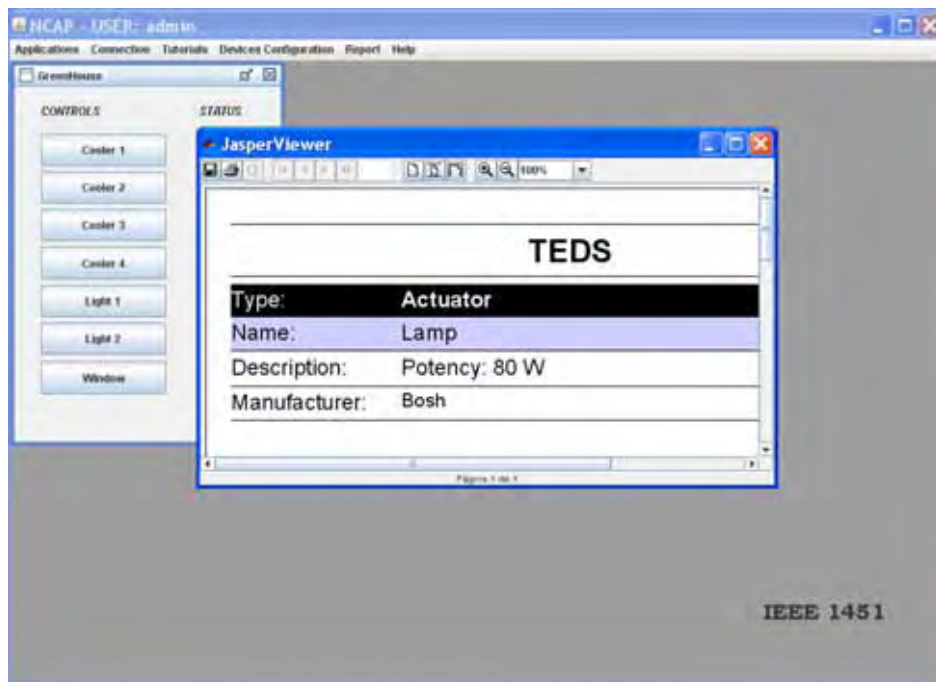


Figura 19D: TEDS gerada no software NCAP – Maquete da Casa de Vegetação.

APÊNDICE E – FIRMWARE DO DRIVERTIM

Neste Apêndice apresentam-se parte da programação do processador *NIOS II* utilizados nas diferentes aplicações da plataforma *IEEE 1451*. Nas Figuras apresentadas neste Apêndice destaca-se a empregabilidade do template desenvolvido neste trabalho.

Na Figura 1E mostra-se parte do código fonte da programação do processador *NIOS II* utilizado na lógica de controle para o monitoramento da pressão e vazão.

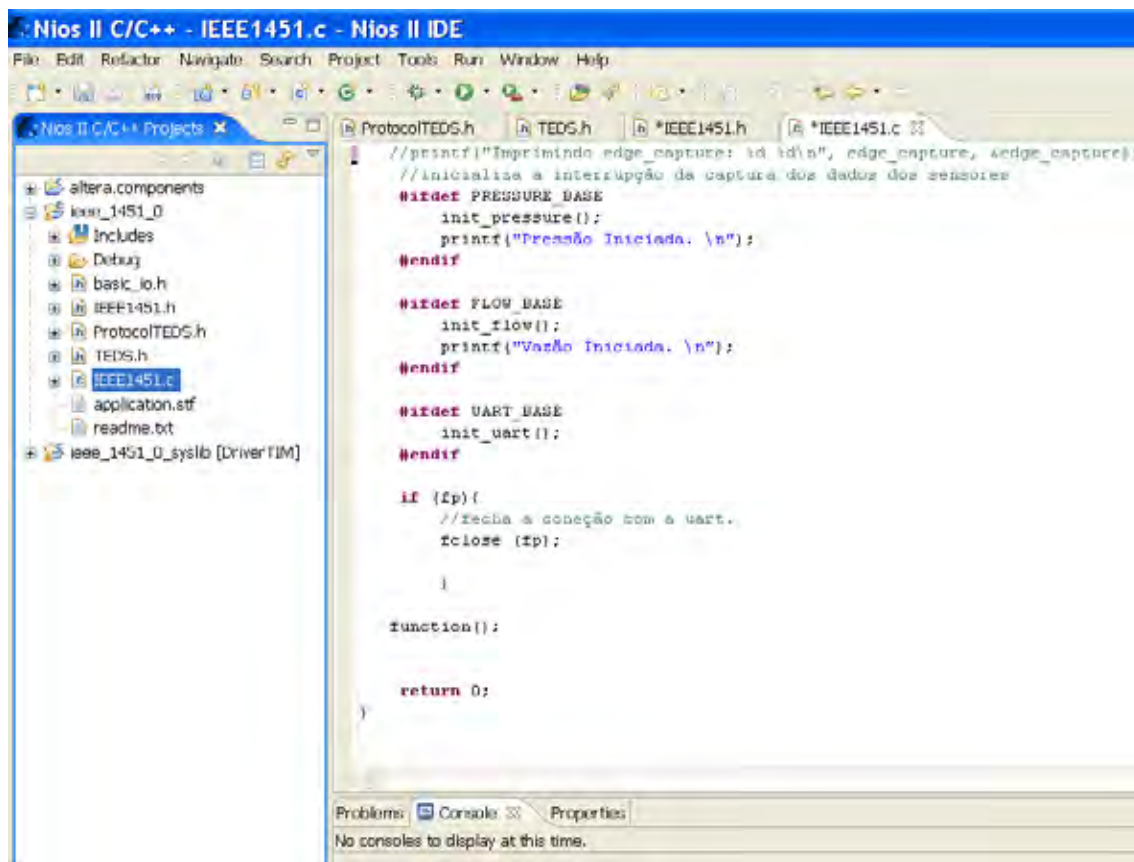


Figura 1E: Parte da lógica de controle para o monitoramento da vazão e pressão.

Na Figura 2E, mostra-se parte do código fonte do arquivo *IEEE1451.h*, utilizado na CPU do *DriverTIM* para exercer as funcionalidades da automação da Casa de Vegetação.

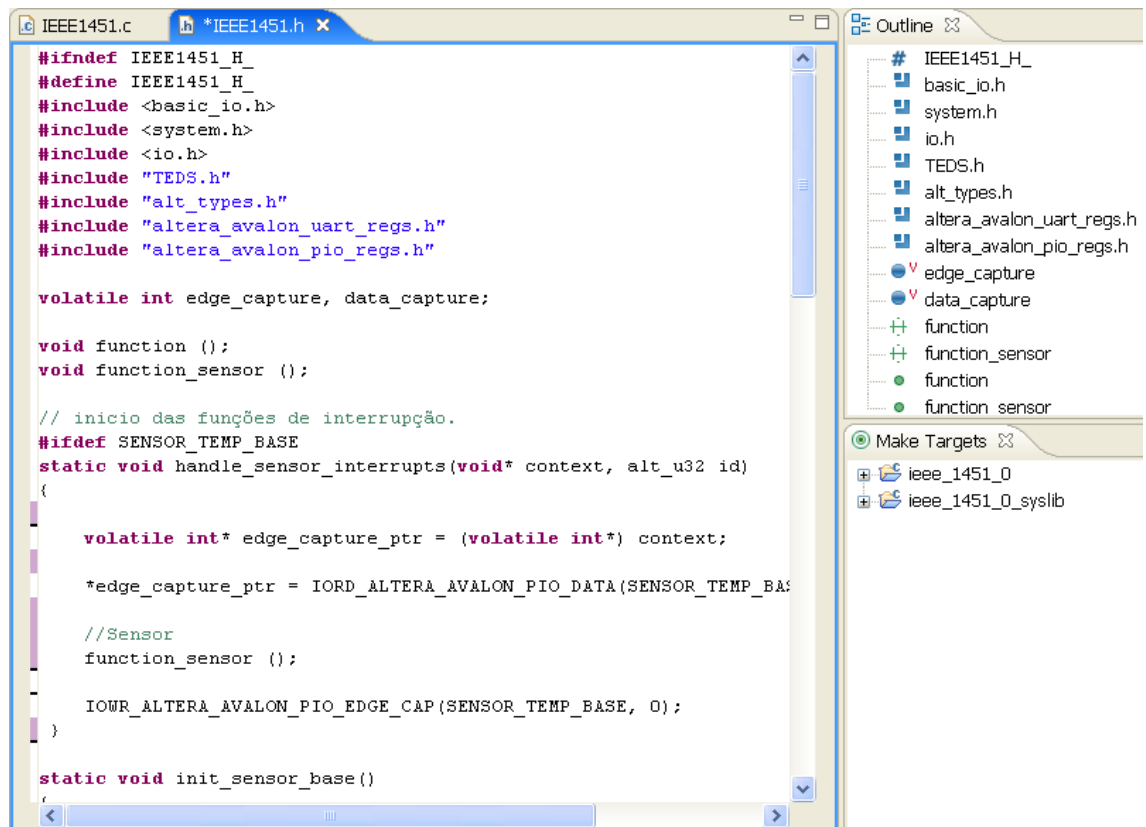


Figura 2E: Programação da CPU do *DriverTIM* - Casa de Vegetação.

Nas Figuras 3E e 4E apresentam-se parte do código da programação do processador *NIOS II*, utilizada na automação do *MHI-01*.

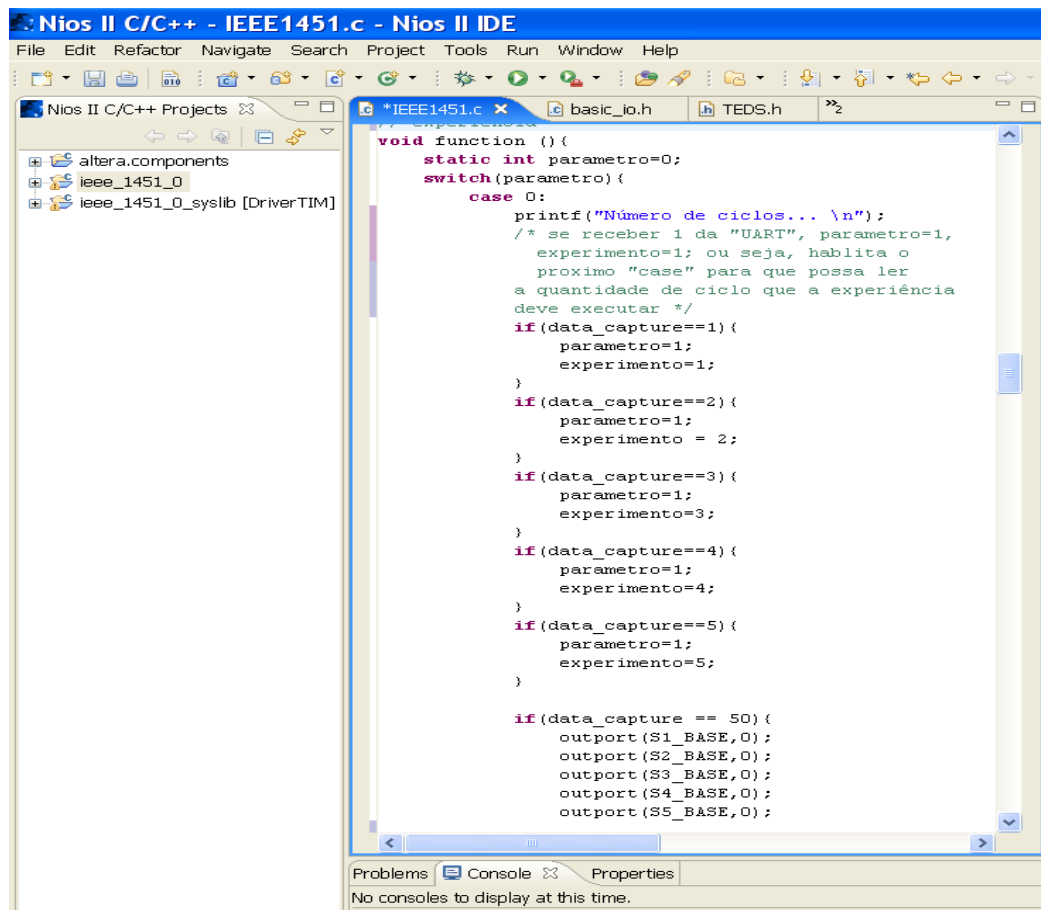
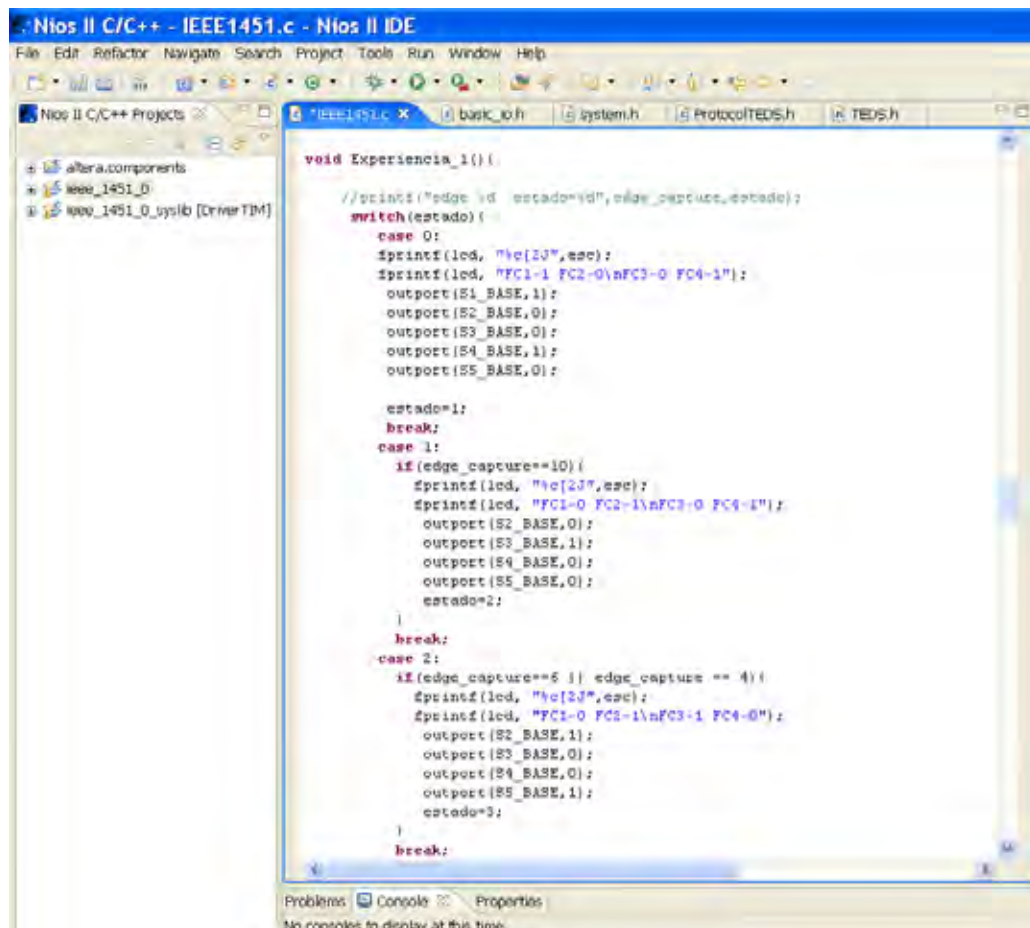


Figura 3E: Programação do processador *NIOS II* para automação do *MHI-01*.



```
void Experiencia_1() {  
    //printf("edge %d estado=%d", edge_capture, estado);  
    switch(estado) {  
        case 0:  
            sprintf(led, "%c[23]", estado);  
            sprintf(led, "FC1-1 FC2-0\nFC3-0 FC4-1");  
            output(S1_BASE, 1);  
            output(S2_BASE, 0);  
            output(S3_BASE, 0);  
            output(S4_BASE, 1);  
            output(S5_BASE, 0);  
  
            estado=1;  
            break;  
        case 1:  
            if(edge_capture==10) {  
                sprintf(led, "%c[23]", estado);  
                sprintf(led, "FC1-0 FC2-1\nFC3-0 FC4-1");  
                output(S2_BASE, 0);  
                output(S3_BASE, 1);  
                output(S4_BASE, 0);  
                output(S5_BASE, 0);  
                estado=2;  
            }  
            break;  
        case 2:  
            if(edge_capture==5 || edge_capture == 4) {  
                sprintf(led, "%c[23]", estado);  
                sprintf(led, "FC1-0 FC2-1\nFC3-1 FC4-0");  
                output(S2_BASE, 1);  
                output(S3_BASE, 0);  
                output(S4_BASE, 0);  
                output(S5_BASE, 1);  
                estado=3;  
            }  
            break;  
    }  
}
```

Figura 4E: Parte da lógica de controle a ser realizada pelo processador *NIOS II* na automação do *MHI-01*.

Na Figura 5E mostra-se os códigos propostos neste trabalho para descrever a *TEDS*.

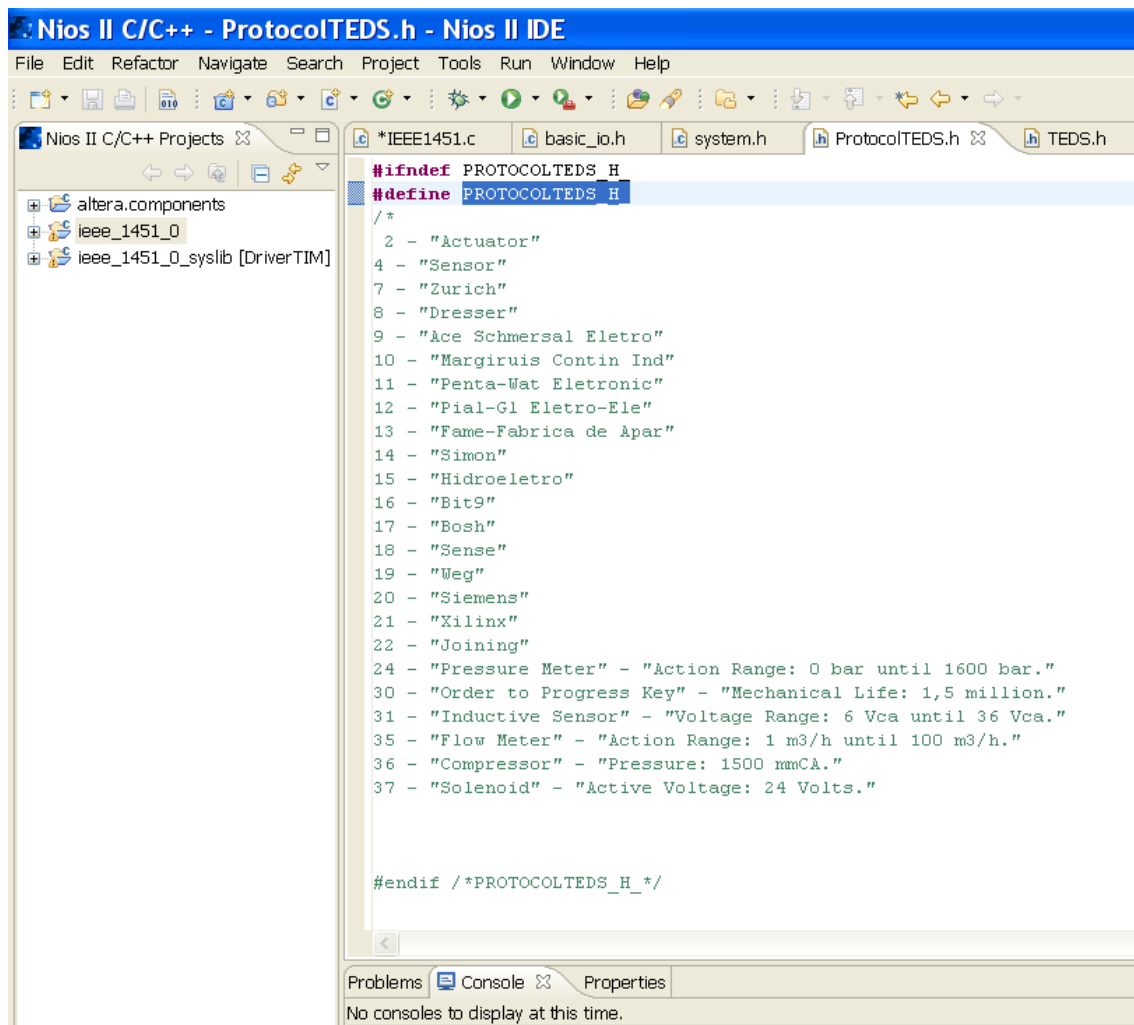


Figura 5E: Código de informações propostas neste trabalho para desenvolver a TEDS.

Na Figura 6E, observam-se as propriedades utilizadas neste trabalho para programar o processador *Nios II*, sendo que em:

- 1) Mostra-se o *Select Template Project* (template) utilizado como base na programação.
- 2) Encontra-se a localização e nome do hardware desenvolvido no *SOPC Builder*.
- 3) Apresenta-se o tipo de memória utilizada para armazenar a programação do processador *Nios II*.
- 4) Mostram-se as características da programação definidas pelo projetista.

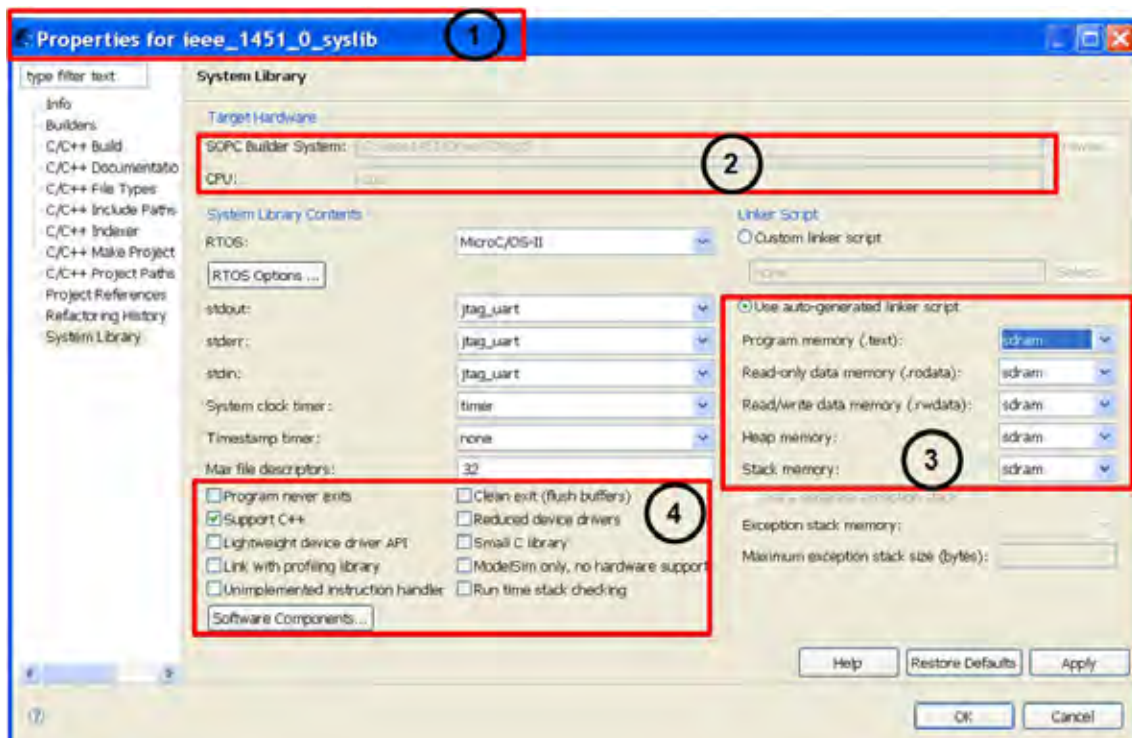


Figura 6E: Propriedades utilizadas neste trabalho na programação do processador *NIOS*

II.

ANEXO A – COMUNICAÇÃO DO ELIPSE E3 COM DRIVERTIM

Neste Anexo descreve-se a parte da implementação da automação do kit de pneumática da *Hidro Eletro Industrial*, utilizando o software *Elipse E3* com *DriverTIM*. Os circuitos eletrônicos que compõem o *TIM* nesta aplicação são iguais aos utilizados na automação do *MHI-01*.

Na automação do kit de pneumática a lógica de controle e consequentemente as telas do software supervisor estão em conformidade com o funcionamento do processo de filtragem de Biogás.

Na Figura 1A, mostra-se a tela do software supervisor que representa uma das etapas do processo de filtragem do Biogás. Para facilitar a compreensão do funcionamento, as válvulas foram animadas de forma que quando estiverem abertas fiquem verdes e quando estiverem fechadas fiquem vermelhas.

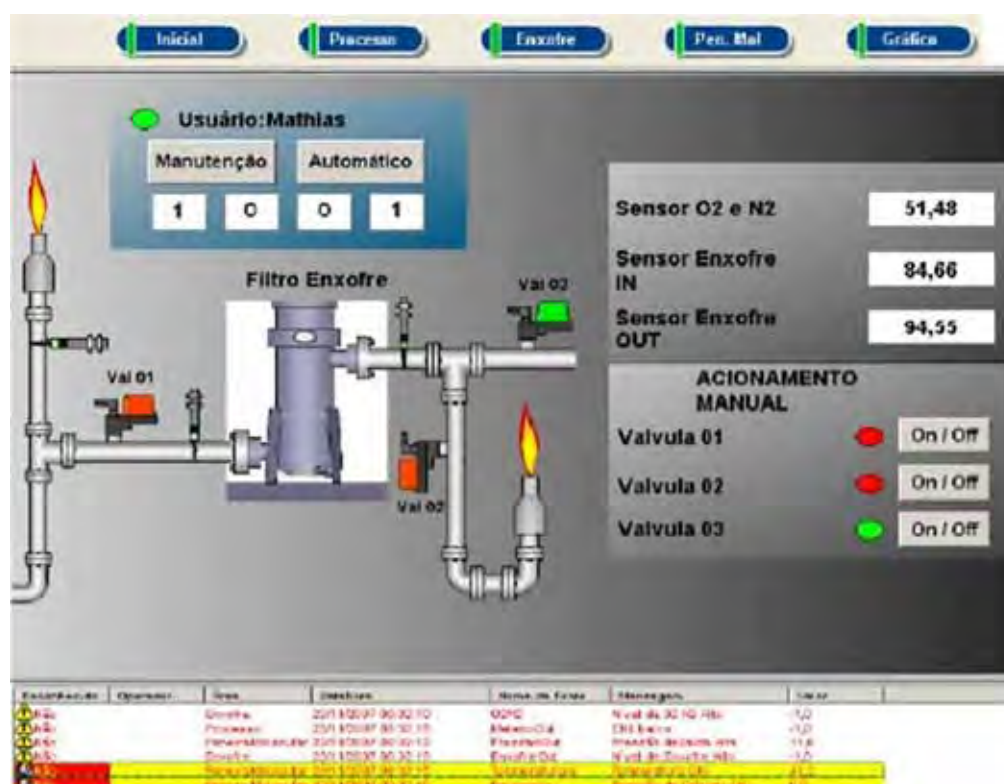


Figura 1A: Tela do software supervisor representando uma das etapas do processo de Filtragem do Biogás.

Na Figura 2A, as variáveis *tag1* e *tag2* são elementos de escrita, isto é, são responsáveis por enviar o comando para o *MMI*.

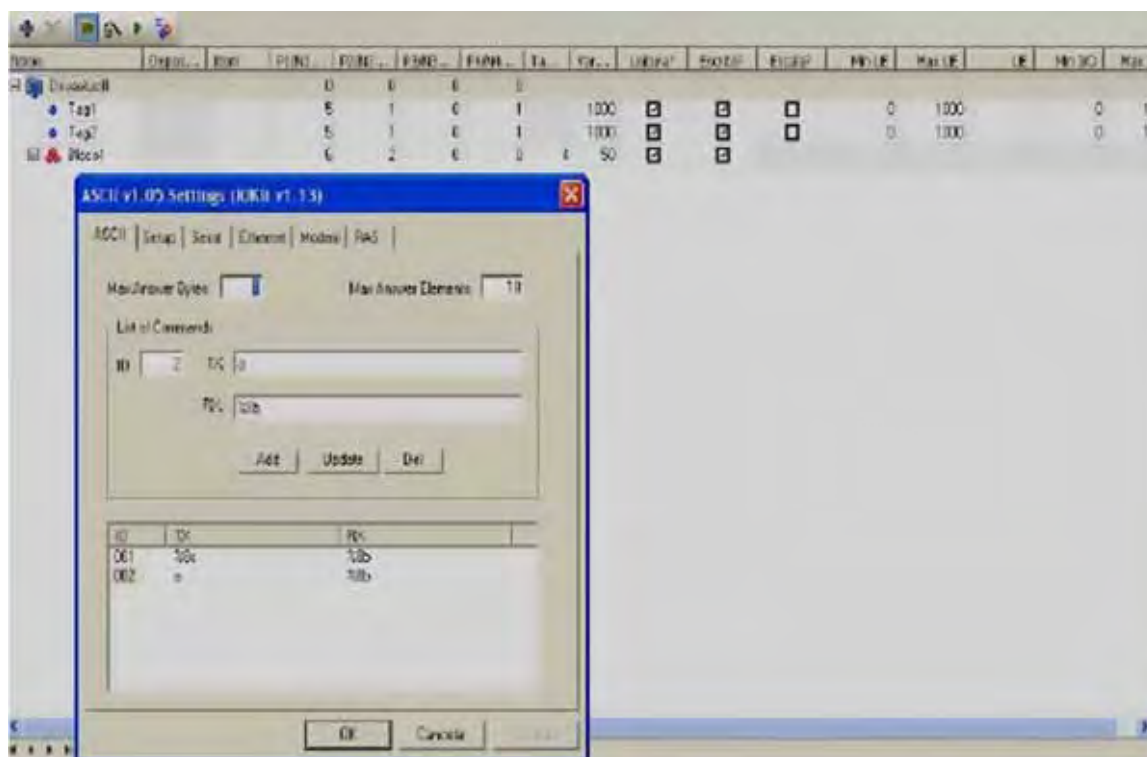


Figura 2A. Tela de configuração do *DriverASCII*.

Na Figura 3A, mostra-se o painel do kit de pneumática utilizado para simular a automação de um sistema de filtragem de Biogás utilizando software supervisor E3 da *Elipse* com o *MMI*.

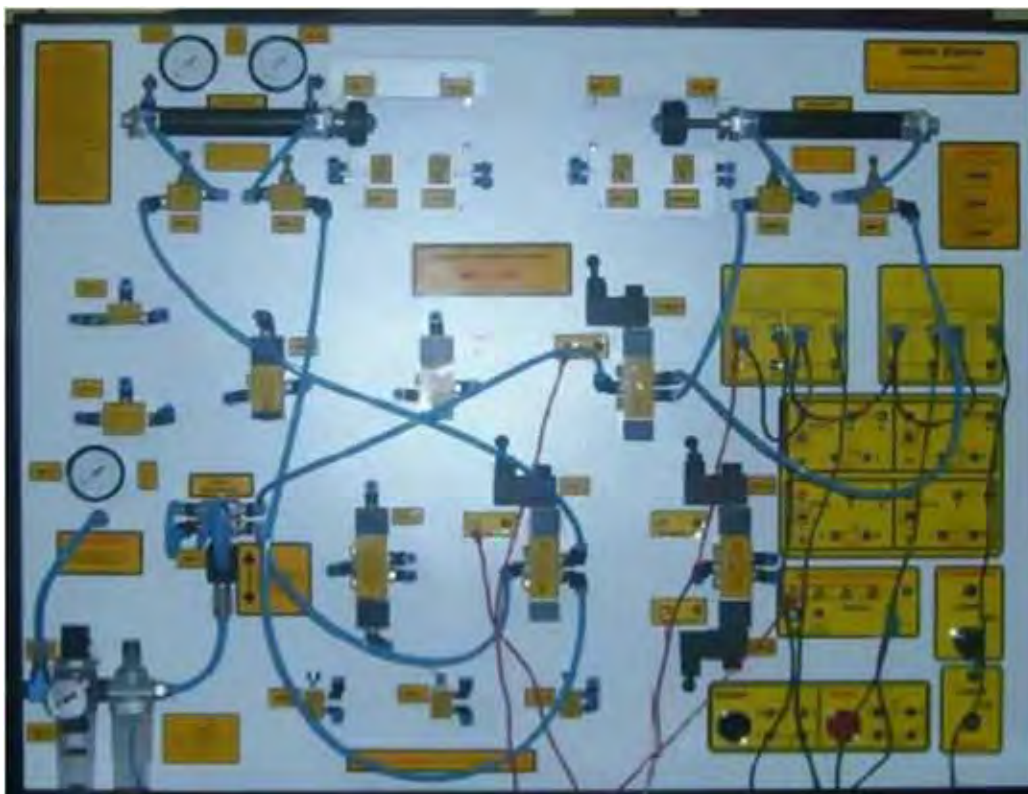


Figura 3A. Painel do kit de Pneumática.