Universidade Estadual de Londrina

VALTER LUÍS ARLINDO DE CAMARGO

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UMA PLATAFORMA PARA MONITORAMENTO ESTRUTURAL UTILIZANDO SENSORES EXTENSOMÉTRICOS CONECTADOS EM REDE

Londrina

2008

VALTER LUÍS ARLINDO DE CAMARGO

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UMA PLATAFORMA PARA MONITORAMENTO ESTRUTURAL UTILIZANDO SENSORES EXTENSOMÉTRICOS CONECTADOS EM REDE

Dissertação submetida ao Departamento de Engenharia Elétrica da Universidade Estadual de Londrina, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Fernando Cardoso Castaldo

Londrina

2008

VALTER LUÍS ARLINDO DE CAMARGO

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UMA PLATAFORMA PARA MONITORAMENTO ESTRUTURAL UTILIZANDO SENSORES EXTENSOMÉTRICOS CONECTADOS EM REDE

Esta dissertação foi julgada e aprovada para a obtenção do Título de Mestre em Engenharia Elétrica, no Programa de mestrado da Universidade Estadual de Londrina,

Londrina, 10 de Março de 2008.

BANCA EXAMINADORA

Prof. Dr. Fernando Cardoso Castaldo
Presidente

Prof. Dr. Aleksander Sade Paterno
Universidade do Estado de Santa Catarina
(UDESC)

Prof. Dr. Carlos José M. da Costa Branco
Universidade Estadual de Londrina (UEL)

Prof. Dr. José Alexandre de França
Universidade Estadual de Londrina (UEL)

Prof. Dr. Marcelo Carvalho Tosin
Universidade Estadual de Londrina (UEL)

Dedicatória

Para as minhas filhas Gabriela e Marcela.

Para a minha esposa Fátima

Agradecimentos

Ao meu orientador, Prof. Dr. Fernando Cardoso Castaldo, sou imensamente grato pela orientação e pelas sugestões que enriqueceram o trabalho.

Aos professores Dr. Ernesto Fernando Ferreyra Ramírez, Dr. Lúcio dos Reis Barbosa, Dr. Marcelo Carvalho Tosin, Dr. Robinson Hoto, Dr. Taufik Abrão e Dr. Walter Germanovix, pelo privilégio de tê-los como meus professores durante os créditos.

A professora Dra. Silvia Galvão de Souza Cervantes pelos valiosos conselhos e exemplo de profissionalismo.

Aos meus colegas do mestrado: Alexandre Fenato, Cláudio Lima Lopes Ferreira, Diogo Kaoru Takayama, Leonardo Dagui e Rodger Vitória Pereira, pelo companheirismo e amizade.

Ao colega de profissão e amigo pessoal Prof. Ms. Claiton Moro Franchi pelos incentivos, conselhos e apoio incondicional.

Aos Professores Dr. Carlos Henrique Zanelato Pantaleão, Dr. Dante Alves Medeiros Filho, Dra. Maria Madalena Dias e Ms. Evandro Cherubini Rolin, pela confiança depositada em mim através da recomendação do meu nome para o programa de mestrado.

A minha família pelo apoio durante esta jornada.

A todos que de forma direta ou indireta colaboraram para a realização deste trabalho.

Resumo

Este trabalho descreve o desenvolvimento de um sistema para medição de esforços e deformação mecânica em estruturas civis utilizando extensômetros elétricos de resistência (Strain-Gauges), implementado por circuitos de condicionamento de sinais e com capacidade para comunicação de dados em rede. A motivação deste trabalho reside no aspecto relacionado à segurança pública com o monitoramento das condições estruturais em obras da construção civil. Em sistemas convencionais, quando se deseja medir simultaneamente vários pontos espalhados espacialmente pela estrutura, utilizando um sistema de aquisição de dados centralizado, a fiação deve percorrer um longo caminho, estando sujeita à captação de radiação eletromagnética e atenuação de sinal, o que pode comprometer a confiabilidade da medida. O sistema a ser desenvolvido deve atender a critérios tais como portabilidade, facilidade de calibração e configuração, meio de transmissão confiável e de baixo custo, assegurando uma maior confiabilidade do sinal captado e redução do custo da manutenção e de instalação. Para este propósito, será empregada uma combinação de microcontroladores de baixo custo, operação através de software flexível e um sistema condicionador de sinal de alta eficiência, obtendo-se assim, uma alta taxa de coleta de dados com capacidade de comunicação em rede. O sistema desenvolvido apresenta características multifunção, ou seja, não é limitado à leitura apenas de sinais de esforços e deformações, mas pode também ser utilizado, com as devidas adaptações do sistema condicionador de sinal, para medição de sinais de outra natureza, como por exemplo, temperatura e umidade. Após uma introdução aos princípios básicos de medições utilizando extensômetros elétricos e de técnicas de condicionamento do sinal, o sistema de comunicação em rede é apresentado, seguido de uma discussão sobre o processo de aquisição de dados e calibração. Finalmente, os resultados são apresentados e discutidos.

PALAVRAS-CHAVE: Rede de Sensores; Extensômetros elétricos; *Strain-gage*s; Monitoramento de estruturas; Instrumentação eletrônica; Sistema de Aquisição de Dados.

Abstract

This paper describes the development of a system for stress and mechanical deformation measurement in civil structures using strain gauges, implemented using a signal conditioning circuit and featured for network data communication. The motivation of this work is related with the concern of public safety with the monitoring of conditions of civil structures buildings. In conventional systems, when you wish to measure simultaneously several points spatially spreaded by the structure, using a centralized data acquisition system, the wiring should go a long way and therefore subject of capturing electromagnetic radiation and signal attenuation, which can compromise the reliability of the measurement. The system being developed must meet criteria such as portability, ease of calibration and configuration, means of transmission reliable and low cost, ensuring greater reliability the signal received and reducing the cost of maintenance and installation. For this purpose, will be employed a combination of low-cost microcontrollers, operating through flexible software and a signal conditioning system, high-efficiency, obtaining thus a high rate data collection with the ability of communication in network. The system developed gives multifunction characteristics, or is not only limited to reading stress and deformations, but can also be used, with appropriate adjustments to the system of signal conditioner, for measurement signals of other kinds, such as temperature and humidity. After an introduction to the basic principles of measurements using strain gauges and techniques of signal conditioning, the network communication system is presented, followed by a discussion of data acquisition system and its calibration process. Finally, the results are presented and discussed.

KEYWORDS: Sensor Networks; Strain Gauges; Structural Health Monitoring; Electronic Instrumentation; Data Acquisition.

Lista de Figuras

1 Sistema convencional de aquisição de dados extensométricos . . . . . p. 4

2 Rede de sensores sem-fios (wireless) integrada a uma estrutura . . . . . p. 5

3 Arquitetura do sistema desenvolvido neste trabalho . . . . . p. 6

4 Extensômetro Elétrico a fio (*wire gauges*) . . . . . p. 18

5 Extensômetro elétrico laminar (*foil gauges*) . . . . . p. 18

6 Extensômetro laminar típico (*foil gauges*) . . . . . p. 19

7 Deformação de um corpo submetido à uma força externa . . . . . p. 19

8 Lei de Hooke - Tensão versus Deformação . . . . . p. 21

9 Condutor de seção transversal circular . . . . . p. 22

10 Curva dos fatores de sensibilidade de diversos materiais . . . . . p. 25

11 Diagrama de uma Ponte de Wheatstone básica . . . . . p. 27

12 Configuração utilizando 4 extensômetros ativos (*full bridge*) . . . . . p. 29

13 Extensômetro ligado em um quarto de ponte . . . . . p. 29

14 Ponte de Wheatstone contendo dois elementos ativos idênticos e ligados em braços adjacentes. . . . . p. 31

15 Ponte de Wheatstone contendo dois elementos ativos idênticos em ramos opostos e alimentada por uma fonte de tensão constante. . . . . p. 32

16 Ponte de Wheatstone contendo dois elementos ativos idênticos e alimentada por uma fonte de corrente constante. . . . . p. 33

17 Efeito da fiação na medição . . . . . p. 34

18 Arquitetura geral do sistema proposto . . . . . p. 37

19 Comunicação tipo mestre escravo (*polling*) . . . . . p. 40

20 Arquitetura tradicional de comunicação utilizando *drivers* . . . . . p. 43

21 Comunicação utilizando tecnologia OPC . . . . . p. 44
22 Mapeamento de Itens OPC . . . . . p. 45
23 Estrutura organizacional de um servidor OPC contendo uma hierarquia de grupos e itens . . . . . p. 45
24 Arquitetura sistema de aquisição de dados . . . . . p. 47
25 Diagrama em blocos do terminal de rede desenvolvido . . . . . p. 50
26 A tensão de referência é proporcional a Ve, o que elimina os erros devido a variações na alimentação (dessensibilização da tensão da alimentação). . . . p. 54
27 Amplificador de Instrumentação utilizando 3 amplificadores operacionais . . p. 55
28 Diagrama elétrico (parcial) do circuito condicionador de sinal . . . . . p. 57
29 Vista frontal esquemática do conector de parafusos utilizado para as conexões dos extensômetros . . . . . p. 58
30 Vista da implementação do conector de parafusos utilizado para as conexões dos extensômetros . . . . . p. 58
31 Diagrama de ligação de 1/4 de ponte. . . . . p. 59
32 Diagrama para ligação dos extensômetros em 1/2 ponte . . . . . p. 59
33 Diagrama para ligação dos extensômetros em ponte completa . . . . . p. 60
34 Medição da tensão de saída de uma ponte ligada diretamente a um conversor AD (sem prévia amplificação do sinal). . . . . p. 60
35 Obtenção de uma resolução de 12 bits a partir de um conversor AD de 21 bits. p. 62
36 Localização do trimpot de ajuste de zero da ponte . . . . . p. 63
37 Efeito termoelétrico formado pelas trilhas de cobre e os pinos do circuito integrado, quando as junções estão submetidas a diferentes temperaturas. . . p. 65
38 Princípio de funcionamento das pontes com alimentação CA visando o cancelamento do erro de offset. . . . . p. 66
39 Circuito de correção de desvios por combinação de software e hardware. . . . p. 66
40 Resposta de saída de um filtro digital de média móvel contendo 11 pontos em reposta a um pulso retangular de entrada contaminado com ruído aleatório (SMITH, 2007). . . . . p. 69

41 Encapsulamento do sensor de temperatura LM35 (TO-92) . . . . . p. 72

42 Esquema elétrico da parte referente à programação do endereço do terminal . p. 73

43 Localização da microchaves de endereçamento na Placa de Circuito Impresso p. 74

44 Esquema de Troca de Mensagens entre Mestre e Escravo no protocolo MODBUS . . . . . p. 76

45 Localização do protocolo MODBUS em correspondência ao modelo OSI/ISO. p. 77

46 Composição de um bilhete (mensagem) do protocolo MODBUS-ASCII . . . p. 79

47 Composição de um bilhete (mensagem) do protocolo MODBUS-RTU . . . . p. 80

48 Formato de transmissão serial de 1 byte do MODBUS-RTU com paridade . . p. 81

49 Formato de transmissão serial de 1 byte do MODBUS-RTU sem paridade . . p. 82

50 Exemplo comunicação MODBUS . . . . . p. 88

51 Exemplo de Comunicação MODBUS - Resposta . . . . . p. 88

52 Máquina de estados MODBUS-RTU TX e RX da estação escrava (MODBUS-IRDA, 2007). . . . . p. 89

53 Fotografia da primeira versão do terminal de rede implementado no microcontrolador AT89S8252 para validação do protocolo MODBUS . . . . . p. 93

54 Tela ambiente de desenvolvimento MPLAB (MICROCHIP) . . . . . p. 94

55 Diagrama esquemático do gravador JDM. . . . . p. 94

56 Placa de Circuito Impresso do terminal de aquisição . . . . . p. 95

57 Fotografia da segunda versão do terminal de rede implementado em PIC 16F876 para validação do sistema . . . . . p. 96

58 Pinagem dos conectores DB9 e DB25 na norma RS232 . . . . . p. 101

59 Utilização de repetidores (TRISTÃO, 2004). . . . . p. 104

60 Diagrama elétrico do conversor RS232/RS485 implementado . . . . . p. 110

61 Fotografia do Conversor RS232/RS485 Implementado . . . . . p. 111

62 Criação de um novo canal de comunicação no KEPServerEx - etapa1 - Fornecimento do nome do canal . . . . . p. 112

63 Criação de um novo canal de comunicação no KEPServerEx - Etapa 2 - Escolha do protocolo a ser utilizado . . . . . p. 113

64 Criação de um novo canal de comunicação no KEPServerEx - Etapa 3 - Configuração dos parâmetros de comunicação serial . . . . . p. 113

65 Criação de um novo nó na rede no software KepwareServer Modbus. . . . . p. 114

66 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 1 - fornecer o nome do dispositivo. . . . . p. 114

67 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 2 - fornecer o protocolo desejado. . . . . p. 115

68 Criação de um novo nó na rede no software Kepware Server Modbus - etapa3 - fornecer o endereço do novo terminal a ser criado. . . . . p. 115

69 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 4 - parâmetros da comunicação. . . . . p. 116

70 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 5 - banco de dados. . . . . p. 116

71 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 6 - configurar os parâmetros do protocolo Modbus. . . . . p. 117

72 Criação de um novo nó na rede no software Kepware Server Modbus - etapa 7 - definição da quantidade de registradores existentes no dispositivo escravo. p. 117

73 Criação de um novo nó na rede no software Kepware Server Modbus - etapa final - dispositivo criado com sucesso. . . . . p. 118

74 Criação de uma nova *tag* no software Kepware Server Modbus . . . . . p. 118

75 Seleção no Elipse Scada do servidor OPC a ser utilizado . . . . . p. 119

76 Assistente do Elipse Scada para importação de *tags* do servidor OPC . . . . . p. 120

77 Tela do assistente de importação de tags com a operação já concluída. . . . . p. 120

78 Tela inicial do sistema supervisório (Elipse Scada) . . . . . p. 126

79 Tela do supervisório (Elipse Scada) para a habilitação dos terminais . . . . . p. 126

80 Tela inicial do sistema supervisório (Elipse Scada) . . . . . p. 127

81 Tela de visualização dos dados do histórico (Elipse Scada) . . . . . p. 128

82 Tela do Elipse Scada para configuração das "penas" do histórico. . . . . p. 128

83 Tela do Elipse Scada para configuração dos dados do histórico. . . . . p. 129

84 Tela do Elipse Scada para configuração para recuperação de dados por consulta entre datas e horas. . . . . p. 129

85 Tela do Elipse Scada para configurar a Impressão dos dados do histórico. . . p. 130

86 Configuração da Comunicação no software MBPOLL . . . . . p. 132

87 Configuração dos Comandos no MBPOLL . . . . . p. 133

88 Tráfego de dados no software MBPOLL . . . . . p. 133

89 Nenhum erro em 14812 transações utilizando o software Modbus Poll . . . . p. 134

90 Nenhum erro em 12545 transações utilizando o software Modbus Tester Versão 0.3 (beta) . . . . . p. 134

91 Extensômetro elétrico ligado em configuração de 1/4 de ponte utilizado para o experimento . . . . . p. 136

92 Extensômetro CFG-5-120-C1-11 utilizado . . . . . p. 136

93 Configuração dos parâmetros de Comunicação do Servidor OPC MODBUS . p. 137

94 Configuração do Servidor OPC MODBUS . . . . . p. 137

95 Tela de coleta de dados logo após o sistema ter sido ligado (regime transiente) p. 138

96 Tela de coleta de dados mostrando estabilidade com o passar do tempo (regime permanente) . . . . . p. 139

97 Elemento utilizado como carga para o experimento . . . . . p. 140

98 Teste de Repetibilidade . . . . . p. 140

99 Curva de histerese para materiais elásticos . . . . . p. 141

100 Tela de Aquisição . . . . . p. 141

101 Gráfico mostrando a situação inicial do experimento . . . . . p. 142

102 Gráfico mostrando a evolução da temperatura e da sensibilidade térmica do extensômetro durante a fase do aquecimento e desaquecimento. . . . . p. 143

103 Utilização de um extensômetro inativo para eliminar os efeitos da temperatura p. 144

104 DIAGRAMA ESQUEMÁTICO COMPLETO DO NÓ DE REDE . . . . . p. 156

LISTA DE SIGLAS E ABREVIAÇÕES

API	*Application Program Interface*
CLP	Controlador Lógico Programável
CMRR	*Commom Mode Rejection Ratio*
CRC	*Cyclical Redundancy Checking*
CTS	*Clear To Send*
COM	*Component Object Model*
DSSS	*Direct Sequence Spread Spectrum*
DDE	*Dynamic Data Exchange*
ERP	*Enterprise Resources Planning*
FCC	*Federal Communications Commission*
FHSS	*Frequency Hopping Spread Spectrum*
GPRS	*General Packet Radio Service*
GPS	*Global Positioning System*
GSM	*Global System for Mobile communications*
IEEE	*Institute of Electrical and Electronics Engineers*
IHM	Interface Homem-Máquina
ISM	*Industrial, Scientific and Medical*
ISO	*International Standards Organization*
LAN	*Local Area Network*
MIPS	*Millions of Instructions Per Second*
MEMS	*Micro Electrical Mechanical Systems*
OLE	*Object Linking and Embedding*
ODBC	*Open DataBase Connectivity*
OPC	*OLE for Process Control*
OSI	*Open System Interconnection*
PLC	*Programmable Logic Controller*
RTD	*Resistance Temperature Detector*
RTS	*Request To Send*
RTUs	*Remote Terminal Units*
SCADA	*Supervisory Control and Acquisition Data System*
SI	Sistema Internacional de Unidades
SGDB	Sistema Gerenciador do Banco de Dados
UART	*Universal Asynchronous Receiver-Transmitter*
UTP	*Unshielded Twisted Pair*
WAN	*Wide Area Network*

Lista de Tabelas

1 Fatores de sensibilidade do extensômetro para alguns materiais . . . . . p. 26

2 Desempenho de sensores de temperatura . . . . . p. 71

3 Mapeamento das funções e memória do protocolo MODBUS . . . . . p. 82

4 Mapeamento de Memória do terminal de rede (3xxxxx = Input Registers) . . p. 83

5 Mapeamento de memória do terminal de rede (4xxxxx = *Holding Registers*) . p. 83

6 Distâncias máximas de acordo com a velocidade utilizada . . . . . p. 104

7 Tempo de Estabilização do sistema . . . . . p. 138

Sumário

1 INTRODUÇÃO p. 1

1.1 MOTIVAÇÃO . . . . . p. 3

1.2 CONTRIBUIÇÕES DESTE TRABALHO . . . . . p. 6

1.2.1 ASPECTOS SOCIAIS . . . . . p. 7

1.2.2 ASPECTOS ECONÔMICOS . . . . . p. 7

1.2.3 ASPECTOS AMBIENTAIS . . . . . p. 7

1.2.4 ASPECTOS CIENTÍFICOS . . . . . p. 8

1.3 OBJETIVOS . . . . . p. 8

1.3.1 OBJETIVOS GERAIS . . . . . p. 8

1.3.2 OBJETIVOS ESPECÍFICOS . . . . . p. 8

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO . . . . . p. 9

2 REVISÃO DA LITERATURA p. 11

2.1 MONITORAMENTO ESTRUTURAL . . . . . p. 11

2.2 UTILIZAÇÃO DE SENSORES INTELIGENTES PARA MONITORAMENTO ESTRUTURAL . . . . . p. 13

2.3 TRABALHOS RELACIONADOS . . . . . p. 15

2.4 CONCLUSÕES . . . . . p. 16

3 FUNDAMENTAÇÃO TEÓRICA p. 17

3.1 EXTENSÔMETROS ELÉTRICOS DE RESISTÊNCIA . . . . . p. 17

3.1.1 FATOR DE DEFORMAÇÃO (ε) . . . . . p. 19

3.1.2 TENSÃO MECÂNICA . . . . . p. 20

3.1.2.1 RELAÇÃO ENTRE TENSÃO MECÂNICA E DEFORMAÇÃO . . . . . p. 20
3.1.3 RELAÇÃO ENTRE TENSÃO ELÉTRICA E DEFORMAÇÃO . . . p. 21
3.1.3.1 Medição da variação da resistência dos extensômetros . . . p. 25
3.2 PONTE DE WHEATSTONE . . . . . p. 26
3.2.1 CONFIGURAÇÃO BÁSICA DA PONTE . . . . . p. 26
3.2.2 QUANTIDADE DE SENSORES . . . . . p. 28
3.2.2.1 Quatro elementos ativos . . . . . p. 28
3.2.2.2 Um elemento ativo . . . . . p. 29
3.2.2.3 Dois elementos ativos com respostas opostas . . . . . p. 31
3.2.2.4 Dois elementos ativos idênticos . . . . . p. 32
3.2.2.5 Ordem de grandeza . . . . . p. 33
3.2.3 EFEITO DA RESISTÊNCIA DA FIAÇÃO . . . . . p. 34
3.2.4 MEDIÇÃO DE TENSÃO MECÂNICA UTILIZANDO EXTENSÔMETROS . . . . . p. 35
3.2.5 RELAÇÃO ENTRE TENSÃO MECÂNICA E TENSÃO ELÉTRICA p. 35

4 PROJETO CONCEITUAL DO SISTEMA p. 37
4.1 TOPOLOGIA . . . . . p. 38
4.2 DEFINIÇÃO DO SISTEMA DE AQUISIÇÃO DE DADOS . . . . . p. 38
4.3 DEFINIÇÃO DO PROTOCOLO DE COMUNICAÇÃO . . . . . p. 39
4.4 ESCOLHA DO MEIO DE TRANSMISSÃO . . . . . p. 42
4.5 SERVIDOR DE COMUNICAÇÕES . . . . . p. 42
4.5.1 TECNOLOGIA OPC . . . . . p. 42
4.5.2 MESTRE MODBUS E SERVIDOR OPC . . . . . p. 46
4.6 SOFTWARE DE AQUISIÇÃO E TRATAMENTO DOS DADOS . . . . . p. 47
4.7 PLANEJAMENTO PARA FUTURA INTEGRAÇÃO . . . . . p. 48

4.8 CAPACIDADE PARA EXPANSÃO FUTURA . . . . . p. 49

5 PROJETO, MONTAGEM E TESTES DO TERMINAL DE AQUISIÇÃO p. 50

5.1 VISÃO GERAL DO TERMINAL DA REDE . . . . . p. 50

5.2 CARACTERÍSTICAS DESEJADAS . . . . . p. 51

5.2.1 ALIMENTAÇÃO . . . . . p. 51

5.2.2 COMUNICAÇÃO . . . . . p. 51

5.2.3 MEMÓRIA . . . . . p. 51

5.3 IMPLEMENTAÇÃO . . . . . p. 51

5.3.1 ALIMENTAÇÃO . . . . . p. 51

5.4 MICROCONTROLADOR . . . . . p. 51

5.5 QUESTÕES FUNDAMENTAIS SOBRE CIRCUITOS EM PONTE . . . . . p. 52

5.5.1 TENSÃO DE EXCITAÇÃO . . . . . p. 53

5.5.2 TENSÃO DE REFERÊNCIA . . . . . p. 54

5.6 CIRCUITO CONDICIONADOR DE SINAL . . . . . p. 55

5.6.1 AMPLIFICAÇÃO . . . . . p. 55

5.7 LIGAÇÃO DOS EXTENSÔMETROS . . . . . p. 56

5.7.1 LIGAÇÃO EM 1/4 DE PONTE . . . . . p. 58

5.7.2 LIGAÇÃO EM 1/2 DE PONTE . . . . . p. 59

5.7.3 LIGAÇÃO EM PONTE COMPLETA . . . . . p. 59

Exemplo de célula de carga . . . . . p. 60

5.8 CALIBRAÇÃO . . . . . p. 62

5.8.1 AJUSTE DE ZERO DO SISTEMA . . . . . p. 62

5.8.2 AJUSTE DE ZERO DA PONTE . . . . . p. 63

5.8.3 CANCELAMENTO DE DESVIOS PROVOCADOS PELA INSTRUMENTAÇÃO . . . . . p. 64

5.8.4 EFEITO TERMOELÉTRICO . . . . . p. 64

5.8.4.1 Desvio por deriva (*offset Drift*) do amplificador . . . . . p. 64
5.8.5 EXCITAÇÃO POR TENSÃO ALTERNADA . . . . . p. 65
5.8.6 SOLUÇÃO ADOTADA PARA A CORREÇÃO DO *OFFSET* . . . . p. 66
5.8.7 SOLUÇÃO ADOTADA PARA CORREÇÃO DE *DRIFTS* . . . . . p. 67
5.8.8 VALOR DE REFERÊNCIA DE CALIBRAÇÃO . . . . . p. 68
5.8.9 FILTRAGEM . . . . . p. 68
5.8.9.1 Filtro analógico . . . . . p. 68
5.8.9.2 Filtro digital . . . . . p. 68
5.8.9.3 Listagem da rotina de média móvel . . . . . p. 69
5.9 MEDIÇÃO DA TEMPERATURA . . . . . p. 71
5.9.1 SENSOR DE TEMPERATURA . . . . . p. 72
5.9.2 INTERFACE DE CONTROLE DE CALIBRAÇÃO E DE ENDEREÇAMENTO . . . . . p. 73
5.9.2.1 Endereçamento . . . . . p. 73
5.10 PROTOCOLO MODBUS . . . . . p. 75
5.10.1 DESCRIÇÃO . . . . . p. 75
5.10.2 Modelo de Comunicação . . . . . p. 76
5.10.3 Transações entre dispositivos . . . . . p. 76
5.10.4 Modbus e o Modelo OSI . . . . . p. 77
5.10.4.1 Topologia Física . . . . . p. 78
5.10.4.2 Modos de Transmissão . . . . . p. 78
5.10.4.3 Composição do bilhete MODBUS-ASCII . . . . . p. 78
5.10.4.4 RTU . . . . . p. 79
5.10.4.5 Composição do bilhete MODBUS-RTU . . . . . p. 80
5.10.4.6 Detecção de Erros . . . . . p. 81
5.10.5 MAPEAMENTO DE MEMÓRIA . . . . . p. 82
5.10.5.1 Função 0x01 - Read Coil Status . . . . . p. 83

5.10.5.2 Função 0x02 - Read Input Status . . . . . p. 84
5.10.5.3 Função 0x03 - Read Holding Registers . . . . . p. 85
5.10.5.4 Função 0x04 - Read Input Registers . . . . . p. 86
5.10.5.5 Função 0x05 - Force Single Coil . . . . . p. 86
5.10.5.6 Função 0x06 - Preset Single Register . . . . . p. 87
5.10.5.7 Exemplo de Transação Modbus . . . . . p. 87
5.11 MÁQUINA DE ESTADOS MODBUS . . . . . p. 88
5.11.0.8 Listagem da rotina principal da máquina de estados Modbus-RTU implementada . . . . . p. 89
5.11.1 ROTINA IMPLEMENTAÇÃO MÁQUINA DE ESTADOS TX . . . . p. 90
5.11.2 MÁQUINA DE ESTADOS RX . . . . . p. 91
5.12 CONSTRUÇÃO DOS PROTÓTIPOS . . . . . p. 92

6 DESENVOLVIMENTO DO SISTEMA DE AQUISIÇÃO DE DADOS p. 97
6.1 INTERFACE DE COMUNICAÇÃO . . . . . p. 98
6.1.1 TRANSMISSÃO SERIAL . . . . . p. 98
6.1.2 DESCRIÇÃO DO BARRAMENTO RS232 . . . . . p. 98
6.1.3 DESCRIÇÃO DOS SINAIS DO BARRAMENTO RS232 . . . . . p. 100
6.1.4 LIMITAÇÕES DO RS232 . . . . . p. 101
6.1.5 DESCRIÇÃO DO BARRAMENTO RS485 . . . . . p. 101
6.1.6 TIA/EIA-485 . . . . . p. 102
6.1.7 LINHAS DE COMUNICAÇÃO BALANCEADAS . . . . . p. 102
6.1.8 RESISTORES DE TERMINAÇÃO . . . . . p. 103
6.1.9 LIMITES DE DISTÂNCIA E VELOCIDADE . . . . . p. 103
6.1.10 NÚMERO MÁXIMO DE DISPOSITIVOS NA REDE RS485 . . . . p. 103
6.1.11 ATERRAMENTO / INTERLIGAÇÃO DO COMUM . . . . . p. 105
6.1.12 CABOS DE LIGAÇÃO . . . . . p. 105

6.2 CONVERSOR RS232/RS485 . . . . . p. 106
6.2.1 CONVERSOR RS232/RS485 DESENVOLVIDO . . . . . p. 107
6.3 SOFTWARE SERVIDOR DE COMUNICAÇÕES (MESTRE MODBUS) . . p. 112
6.4 CRIAÇÃO DE UM NOVO CANAL NO SOFTWARE KEPServerEX . . . . p. 112
6.5 CRIAÇÃO DE UM NOVO DISPOSITIVO NO SOFTWARE KEPServerEX . p. 113
6.6 CONEXÃO DO SUPERVISÓRIO COM O SERVIDOR MODBUS . . . . . p. 118
6.7 SISTEMAS DE SUPERVISÃO E CONTROLE . . . . . p. 121
6.7.1 INTERFACEAMENTO DOS SISTEMAS SUPERVISÓRIOS . . . . p. 122
6.7.2 COMPUTADOR . . . . . p. 124
6.8 SOFTWARE DE AQUISIÇÃO DE DADOS . . . . . p. 124
6.8.1 TELA PRINCIPAL . . . . . p. 125
6.8.2 TELA HABILITAÇÃO TERMINAIS . . . . . p. 125
6.9 TELA DE AQUISIÇÃO DOS DADOS . . . . . p. 126
6.9.1 TELA DE HISTÓRICOS . . . . . p. 127

7 RESULTADOS E DISCUSSÕES p. 131
7.1 CARACTERÍSTICAS TÉCNICAS DO SISTEMA DESENVOLVIDO . . . . p. 131
7.2 VALIDAÇÃO DA QUALIDADE DA COMUNICAÇÃO . . . . . p. 132
7.2.1 TESTES . . . . . p. 132
7.3 VALIDAÇÃO DA QUALIDADE DOS DADOS . . . . . p. 135
7.3.1 PREPARAÇÃO DO EXPERIMENTO . . . . . p. 135
7.3.2 DESVIO COM O PASSAR DO TEMPO . . . . . p. 137
7.4 REPETIBILIDADE . . . . . p. 139
7.4.1 HISTERESE . . . . . p. 140
7.4.2 DERIVA DEVIDO À TEMPERATURA . . . . . p. 141

8 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS p. 145

8.1 VIABILIDADE DO CONCEITO . . . . . p. 145
8.2 LIMITAÇÕES . . . . . p. 146
8.3 SUGESTÕES PARA TRABALHOS FUTUROS . . . . . p. 147
8.4 CONSIDERAÇÕES FINAIS . . . . . p. 148

Referências Bibliográficas p. 149

1 INTRODUÇÃO

De acordo com Arms et al. (2004), o monitoramento das condições estruturais em obras da construção civil, indústria metal-mecânica, transportes e demais setores da atividade humana, podem trazer imensos benefícios à sociedade, incluindo: menor ocorrência de falhas em sistemas e estruturas, conservação de recursos naturais, aumento da produtividade em processos de fabricação, melhora no tempo de resposta em caso de emergências e aumento da segurança pública.

Podem ser citados como exemplos de potenciais aplicações do sensoriamento inteligente, a utilização de sensores de esforços, deslocamentos ou corrosão, integrados de forma permanente em pontes ou viadutos de maneira a detectar possíveis deteriorações ou falhas estruturais que possam a vir causar acidentes. Outra aplicação seria na prevenção contra desastres ambientais, através de uma rede de sensores capaz de detectar corrosão, deterioração ou vazamento, instalada em dutos transportadores de produtos químicos cujo eventual vazamento seria danoso ao meio ambiente e aos seres vivos, como por exemplo, o petróleo e seus derivados. Também no monitoramento estrutural em veículos e equipamentos de missão crítica - onde falhas não podem ser toleradas, como por exemplo, aeronaves - é um item de fundamental importância para a segurança. Pode-se considerar ainda, o aumento da produtividade na indústria, a melhora na segurança pública, otimização do desempenho operacional de máquinas e equipamentos, enfim, uma ampla gama de possibilidades com alcance social e econômico que se torna realidade graças ao advento de tecnologias cada vez mais presentes em nosso cotidiano.

Particularmente, o monitoramento eletrônico inteligente de estruturas civis tem motivado diversos trabalhos na área, principalmente naqueles países sujeitos a maiores intempéries da natureza, tais como enchentes freqüentes, terremotos, tornados e outros fortuitos. A extensão dos danos causados na construção civil pode ser avaliada através de estratégias de *Structural Health Monitoring and Damage Detection* (DOEBLIN et al., 1996) que consiste basicamente no monitoramento contínuo das estruturas associadas às obras civis e monitoramento ambiental, além da implementação de planos eficazes de manutenção da infraestrutura civil. Estas ações permitem ainda a revisão das técnicas de projeto empregadas na construção civil no sentido de

adequá-las às condições ambientais vigentes.

Assim, a avaliação dos esforços e a freqüência de ocorrência contribuem para a determinação da vida útil e integridade estrutural sob as mais diversas condições de operação. Deformações de elevada magnitude e com taxas freqüentes de repetição podem levar à fadiga e ruptura do material. Ensaios de carregamento monitorados com sensores extensométricos podem ser utilizados para se estimar as cargas aplicadas na estrutura e as deformações correspondentes. Também possibilita a aquisição de dados para comprovação de modelos matemáticos empregados em análise estrutural, como por exemplo, distribui-se espacialmente em uma estrutura, uma determinada quantidade de sensores extensométricos para coleta da tensão mecânica em pontos de interesse e realizam-se ensaios. Ao final destes, os dados coletados podem ser utilizados para validação do modelo teórico através da confrontação dos valores obtidos com os esperados.

Assim, a partir da disponibilidade de tecnologias economicamente viáveis, o monitoramento de condições estruturais das obras civis pode-se tornar uma prática comum, apesar de ainda não existir no Brasil uma cultura nesse sentido. Isto devido às nossas próprias condições ambientais, relativamente estáveis. Entretanto, imposições menores tais como deslocamento e acomodação de solo, vendavais, enchentes e até tremores de terra podem infligir esforços adicionais significativos nas estruturas civis. Tais informações seriam úteis em caso de diagnósticos realizados por peritos em casos de falhas de projeto ou defeitos na construção. Por outro lado, o fator de segurança normalmente adotado pela prática da engenharia poderia ser reavaliado, o que implicaria redução de custos na indústria civil. Modelos matemáticos estruturais poderiam ser revisados com base nas condições ambientais de determinada localidade e composição do solo. Os sensores poderiam ser alocados em pontos específicos da estrutura, atuando como caixas-pretas naqueles locais onde visualmente não se percebem sinais de esforço ou deformação. Medidas corretivas ou preventivas poderiam então ser adotadas em caso de ocorrência de esforços não previstos no projeto, tais como aqueles ocasionados por intempéries ou acomodações de solo.

Para o nosso país, a exemplo do que acontece em diversos outros, novas estratégias para manutenção, inspeção e controle da infraestrutura civil poderiam ser desenvolvidas visando prolongar a vida útil destas. É sabido que o regime de trabalho das estruturas atualmente é diferente de quando foram construídas. Por exemplo, graças ao grande aumento de veículos em circulação nas últimas décadas, o tráfego de veículos sobre as pontes aumentou consideravelmente, e por conseqüência, também os esforços sofridos pelas mesmas, requerendo uma maior manutenção. Com o advento das tecnologias relacionadas ao monitoramento contínuo de estruturas, técnicas de diagnóstico das condições estruturais podem ser eficientemente aplicadas, possibilitando a intervenção quando necessário e com economia de recursos.

Entretanto, o monitoramento contínuo apresenta alguns inconvenientes operacionais, tais como cabeamento e conexões elétricas associadas, disponibilidade de energia para operação e aquisição e transmissão de forma segura dos dados coletados (HEJLL, 2007). Outras contingências referem-se ao custo de instalação e manutenção, proporcional ao número de sensores utilizados, necessários a um adequado mapeamento da aplicação sob estudo. Redes de fibra ótica, apesar de serem versáteis, apresentam como desvantagem a complexidade das conexões (LENG et al., 2005) enquanto que as soluções *wireless* apresentam elevado custo por ponto de sensoriamento (LYNCH et al., 2001). O quesito confiabilidade também é um ponto de discussão entre os usuários desta tecnologia, pois o sistema transmissor/receptor necessário para estabelecer a comunicação entre supervisor e os diversos pontos de monitoramento estará sujeito a interferências eletromagnéticas o que pode ocasionar um mau funcionamento e interpretação errônea de dados. Por outro lado, o uso de equipamentos certificados disponíveis no mercado não são economicamente viáveis para este tipo de aplicação.

1.1 MOTIVAÇÃO

No Brasil, os vários grupos de pesquisa em estruturas que necessitam realizar ensaios extensométricos em estruturas mecânicas e civis utilizam sistemas convencionais de aquisição (um único equipamento centralizado é utilizado para e condicionamento dos sinais) baseados em extensômetros elétricos de resistência. Um diagrama de aplicação típica é mostrado na Figura 1. Tais sistemas apresentam como principal inconveniente a necessidade de cabeamento ponto a ponto entre cada sensor e o sistema de aquisição, o que torna a implementação em campo bastante difícil. Além das dificuldades operacionais, a fiação que conduz o sinal elétrico proveniente de cada extensômetro está sujeita a interferências externas, tais como ruído elétrico, captação de interferências eletromagnéticas, vibrações, umidade, variações de temperatura, corrosão, entre outros. É importante ressaltar que nestes sistemas, transmitir analogicamente um sinal de alguns microvolts por alguns metros através de um cabo, alguns cuidados especiais devem ser tomados, incluindo: a blindagem do cabo, a correta disposição do mesmo através da estrutura e o adequado condicionamento do sinal (amplificação controlada, redução dos níveis de ruído e filtragem). Aparentemente, baseado em depoimentos de alguns pesquisadores e consultas a fornecedores de sistemas de aquisição de dados, estes grupos não dispõem de alternativas confiáveis e economicamente viáveis para a realização deste tipo de trabalho.

Atualmente há uma grande quantidade de pesquisas visando solucionar os problemas apontados anteriormente. A maioria delas se concentra em soluções sem-fio (*wireless*). Basicamente são utilizadas quatro tecnologias (além das proprietárias): *Bluetooth* regulada pela norma IEC

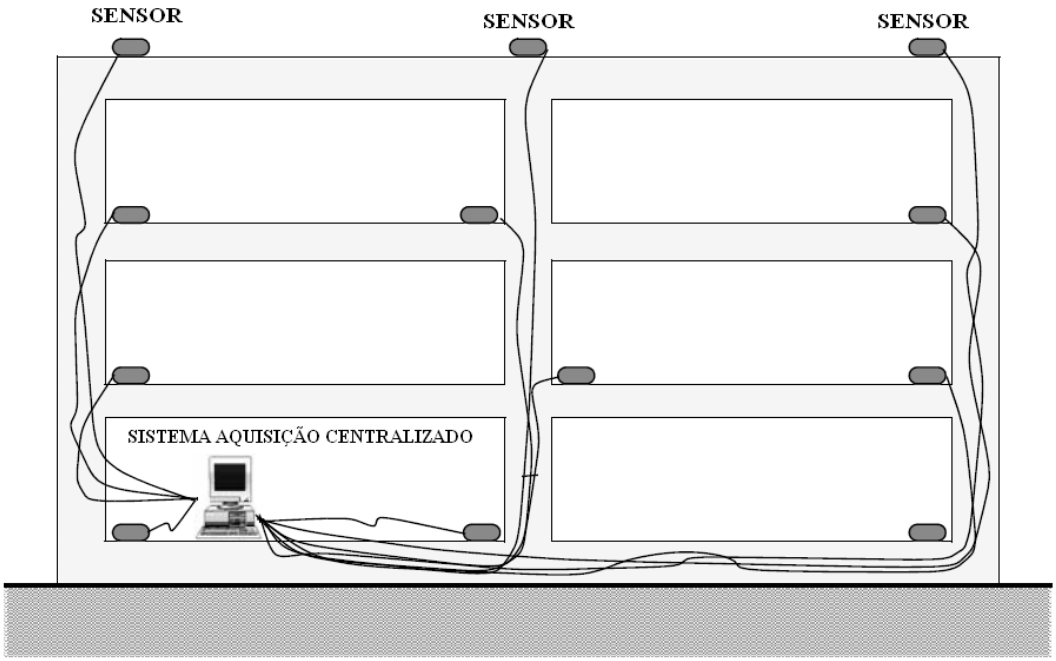

Figura 1: Sistema convencional de aquisição de dados extensométricos

802.14.1 (BLUETOOTH-ORG, 2007), *Wi-fi* norma IEC 802.11 B/G (WI-FI-ORG, 2007) , *Zigbee* norma IEC 802.14.4 (ZIGBEE-ORG, 2007) e GPRS baseado na tecnologia celular GSM. No capítulo REVISÃO DA LITERATURA serão apresentados os principais trabalhos recentes utilizando estas tecnologias para monitoramento estrutural. Estas soluções apresentam grandes vantagens, entre as quais podemos citar: a aquisição e condicionamento local dos sinais, o que torna as medidas mais exatas e livres de interferências; facilidade de implementação dos nós, uma vez que utilizam protocolos padronizados. Talvez a maior vantagem de todas seja a redução do tempo de instalação devido a sua facilidade. No entanto, uma das grandes limitações destes sistemas é a necessidade de utilização de baterias, as quais exigem uma manutenção constante. Outra limitação é a distância de alcance desses transmissores que variam de 10 metros a 100 metros em condições de visada direta, exceto para a tecnologia GPRS que possui uma potência de transmissão maior. Ainda falando da tecnologia GPRS, pode-se ver que a potência maior de transmissão é uma desvantagem para aplicações de monitoramento contínuo que utilizem sistemas alimentados por baterias, já que o consumo de energia é elevado. Outra desvantagem dessa tecnologia é que dependem do meio físico de transmissão das concessionárias de telecomunicações, cuja disponibilidade fica sujeita à existência de infraestrutura (torres de antenas) que atendam o local e além disso, deve-se pagar uma taxa mensal pelo uso do canal.

Pelas razões apresentadas anteriormente pode-se concluir que as tecnologias *wireless,* no atual estado da arte, ainda não são adequadas para o monitoramento das condições estruturais

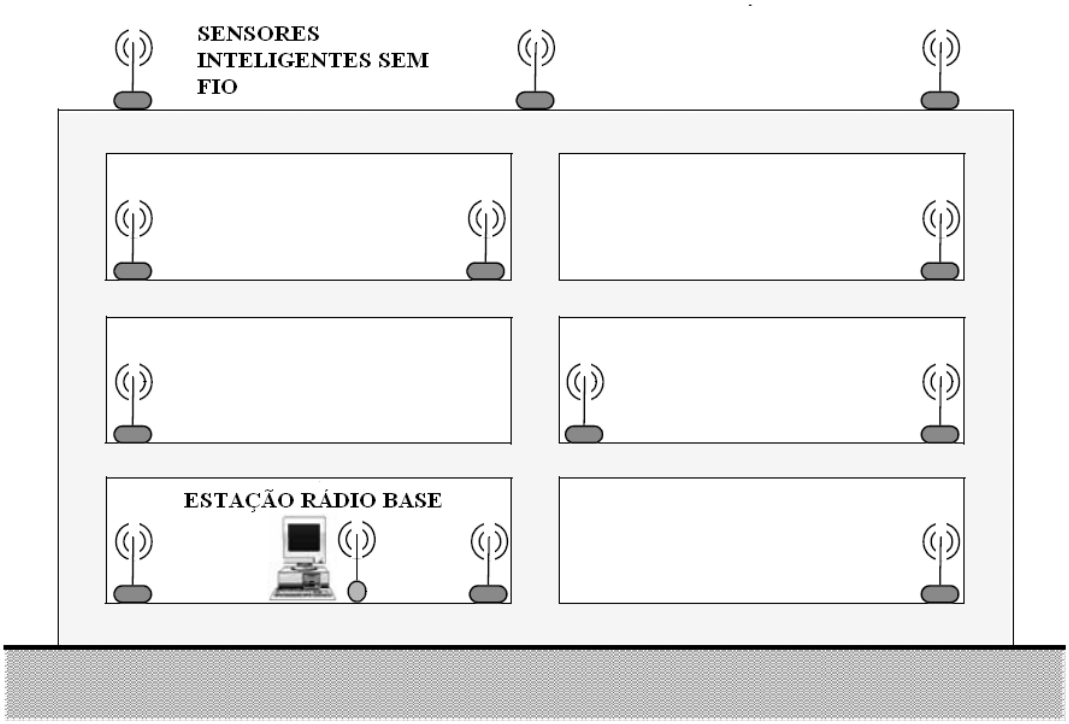

Figura 2: Rede de sensores sem-fios (wireless) integrada a uma estrutura

em grandes estruturas, como por exemplo, em pontes. Para estas, os sensores devem ser integrados de forma permanente na estrutura, já que normalmente o monitoramento deve ser contínuo e as condições de acesso aos pontos de medição nem sempre são favoráveis. Um diagrama ilustrativo de uma estrutura monitorada por sensores sem-fios é ilustrada na Figura 2.

Buscando-se prover soluções alternativas para o problema, a presente proposta visa desenvolver um sistema de monitoramento e aquisição de dados aplicado ao monitoramento de esforços (extensometria) em estruturas civis e mecânicas. O sistema é baseado em uma rede de sensores conectados através de uma linha simples de comunicação e energia e uma interface gráfica para processamento e visualização dos dados coletados. Um diagrama ilustrativo do sistema desenvolvido é mostrado na Figura 3.

Assim uma das grandes vantagens deste sistema é a sua portabilidade e possibilidade de expansão. Por utilizar um protocolo aberto, robusto e altamente difundido pode ser utilizado com um grande número de outros equipamentos já existentes, tais como softwares de supervisão e controle - os quais são, inclusive, utilizados no decorrer deste trabalho.

Em um primeiro momento, o sistema desenvolvido atenderia grupos de pesquisa em estruturas, que necessitam monitorar esforços em seus objetos de estudos. Desta forma, a partir deste projeto pretende-se contribuir para a geração de conhecimento técnico e científico, fo-

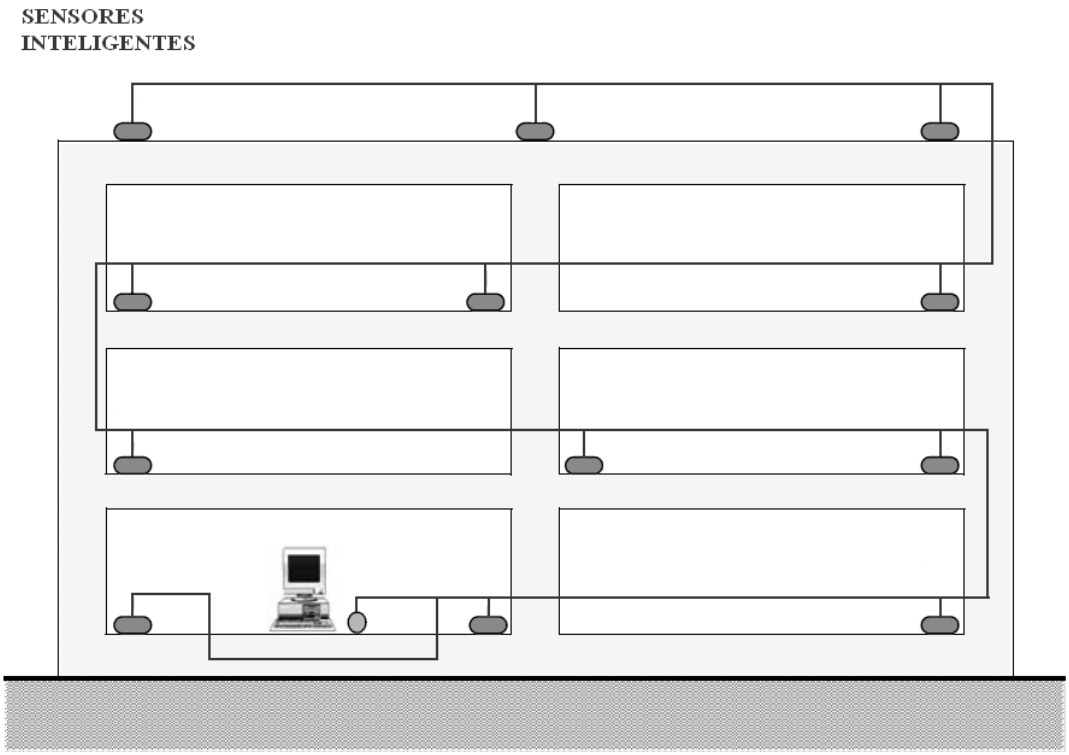

Figura 3: Arquitetura do sistema desenvolvido neste trabalho

mentar uma cultura de monitoramento de condições estruturais na construção civil e indústria metal-mecânica a exemplo do que já ocorre em outros países. Assim, entende-se que este projeto pode resultar em uma importante contribuição no sentido de prover estes grupos de pesquisa com ferramental de suporte à realização de suas atividades, bem como inserir no mercado produtos alternativos economicamente viáveis com alto índice de nacionalização, o que diminui a dependência de componentes importados.

Durante o projeto foram previstos e incorporados elementos com características apropriadas à uma futura integração, o que reduziria os custos de um sistema de monitoramento estrutural. Assim, futuramente cada nó da rede poderia ser implementado por um único Circuito Integrado (CI) com funções de medição, calibragem, processamento e comunicação. Pretende-se, portanto, desenvolver uma tecnologia básica de comunicação em rede para os sensores e projetar alguns dos circuitos constituintes do sistema, especificamente os circuitos analógicos, visando uma futura integração. Isto, obviamente, irá requerer estudos adicionais objetivando assegurar uma forte competitividade ao produto.

1.2 CONTRIBUIÇÕES DESTE TRABALHO

A realização deste projeto, no seu todo ou parte, envolve os seguintes aspectos:

1.2.1 ASPECTOS SOCIAIS

Como citado anteriormente, o monitoramento permanente pode implicar aumento na segurança pública em relação às obras civis, desde que ações preventivas e corretivas sejam tomadas imediatamente ao se detectarem anormalidades. Também as comunidades locais atendidas pelo grupo de pesquisas em estruturas podem ser beneficiadas através da prestação de serviços de realização de ensaios para as empresas e profissionais da região. Outra contribuição do projeto é o envolvimento de alunos de graduação e de mestrado assegurando a estes formação científica complementar e espírito empreendedor, com formação de mão-de-obra qualificada em projetos que envolvem tecnologia de ponta, requisito básico para que empresas de alta tecnologia se instalem localmente e gerem empregos diversos para a comunidade. Outro benefício importante é o aumento da interação universidade-empresa.

1.2.2 ASPECTOS ECONÔMICOS

A prevenção de acidentes em estruturas civis e tomada antecipada de medidas corretivas ou preventivas gera uma grande economia para a sociedade, pois propicia a ampliação da vida útil das estruturas existentes. Por outro lado, o dinheiro que seria aplicado na reconstrução de tais obras pode ser aplicado em outros setores mais carentes. Outro possível ganho vem da otimização de desempenho operacional de máquinas e equipamentos monitorados utilizados nos processos produtivos, pois possíveis falhas poderiam ser previstas antecipadamente e medidas preventivas poderiam ser tomadas, evitando-se paradas na produção. Do lado da integração universidade-empresa, há uma geração e transferência de tecnologia ao setor produtivo com um consequente incentivo ao projeto e desenvolvimento de circuitos eletrônicos no Brasil, o que diminui a dependência aos dispositivos importados, com resultados favoráveis à balança comercial brasileira.

1.2.3 ASPECTOS AMBIENTAIS

A monitoração de estruturas pode prevenir desastres ambientais, tais como o derramamento acidental de óleo em rios e riachos, vazamento de produtos tóxicos para a atmosfera e também ajudar na conservação de recursos naturais, através de substituição na construção civil de elementos não renováveis (ou de renovação demorada, como a madeira) por outros renováveis que apresentem características mecânicas equivalentes.

1.2.4 ASPECTOS CIENTÍFICOS

A principal contribuição deste trabalho consiste no desenvolvimento de uma tecnologia aplicada à instrumentação eletrônica direcionada aos sistemas de aquisição e condicionamento de sinais bem como o desenvolvimento e adaptação de protocolos de comunicação viáveis comercialmente. Os resultados destas pesquisas serão divulgados posteriormente para a comunidade científica através de publicação dos resultados em periódicos e congressos. Outra contribuição imediata é a geração de trabalhos de mestrado na área da engenharia elétrica.

1.3 OBJETIVOS

Visando delimitar o escopo deste trabalho, os objetivos foram divididos em duas categorias, a saber:

1.3.1 OBJETIVOS GERAIS

De acordo com a proposta inicialmente apresentada, o objetivo geral desta dissertação é estudar, projetar, implementar e validar uma plataforma de aquisição de dados multiponto dedicado ao monitoramento de esforços mecânicos estruturais aplicado às obras da construção civil, indústria metal-mecânica e demais setores, utilizando para tanto, equipamentos de aquisição de dados de extensômetros elétricos conectados às estruturas mecânicas. Tais equipamentos devem possuir capacidade de troca de dados com uma unidade mestre através de um protocolo de comunicação digital confiável, seguro e com taxa de resposta compatível com os tempos envolvidos neste tipo de aplicação. Também inclui o desenvolvimento do programa computacional para parametrização, captura, visualização, armazenamento e gerenciamento dos dados coletados.

1.3.2 OBJETIVOS ESPECÍFICOS

Para se atingir os objetivos gerais deste trabalho, foi necessário estabelecer objetivos mais detalhados, apresentados abaixo:

- Projeto e construção de um protótipo da unidade de rede para coleta de dados e validação.
- Projeto e construção de uma interface de comunicação entre o computador e a rede.

- Desenvolvimento de um programa computacional para parametrização, processamento e visualização de dados utilizando sistemas supervisórios.
- Implementar e testar o software para monitoramento e parametrização do terminais de aquisição de dados distribuídos em rede local;
- Implementação, testes e qualificação de uma rede básica de sensores.
- Divulgação de uma cultura associada ao monitoramento estrutural como aspecto básico relacionado à segurança pública.

Com base nas tarefas listadas acima, foi projetado um conjunto de equipamentos e programas computacionais com os quais foi possível atingir o objetivo proposto inicialmente.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

É dada a seguir uma breve descrição do conteúdo desta dissertação.

No Capítulo REVISÃO DA LITERATURA é feito um levantamento do estado atual da arte referente ao monitoramento estrutural e de redes de sensores. O objetivo é fazer uma análise dos sistemas semelhantes existentes, procurando determinar as características que concedem vantagens ou desvantagens, visando determinar as funcionalidades que deveriam ter o sistema aqui desenvolvido, de maneira que incorpore o melhor de cada tecnologia e tente superar alguma limitação destas, levando a um ganho de qualidade.

No Capítulo FUNDAMENTAÇÃO TEÓRICA é apresentado a fundamentação teórica do trabalho, composta pelos estudos realizados sobre os princípios físicos dos extensômetros elétricos, sistemas de aquisição e condicionamento de sinais típicos para este tipo de aplicação e detalhamento de seus elementos constituintes. Também é dada uma visão geral de redes industriais e uma breve descrição do protocolo Modbus.

No Capítulo PROJETO CONCEITUAL DO SISTEMA são definidas as características básicas para o sistema. São descritos os procedimentos e apresentadas as razões pela escolha da metodologia utilizada para cada um dos componentes do sistema, envolvendo um detalhamento do objeto desta dissertação.

No Capítulo PROJETO DOS NÓS DE REDE é desenvolvido o projeto em si dos nós do sistema de aquisição. São detalhados tanto o projeto teórico quanto a construção prática. Também é mostrado como é feita a calibração do sistema.

No Capítulo PROJETO DOS SOFTWARES DE AQUISIÇÃO são apresentados as descrições dos softwares desenvolvidos no computador mestre visando adquirir os dados dos terminais de rede. Seu principal foco é a modelagem e parametrização do software de monitoramento e a implementação do sistema supervisório.

No Capítulo RESULTADOS E DISCUSSÕES são apresentados e discutidos os resultados obtidos dos experimentos realizados em laboratório e da utilização dos softwares desenvolvidos.

E no último, CONCLUSÕES E TRABALHOS FUTUROS são apresentadas algumas considerações finais com relação ao trabalho desenvolvido e algumas sugestões para aperfeiçoamentos.

2 *REVISÃO DA LITERATURA*

Este capítulo aborda uma visão geral dos trabalhos relacionados ao tema desenvolvido. É uma busca para conhecer o que existe atualmente sobre o assunto, procurando determinar as principais vantagens das tecnologias existentes e também determinar suas desvantagens, de forma a propor um sistema que incorpore os benefícios das tecnologias atuais e supere, em algum grau, suas limitações.

2.1 MONITORAMENTO ESTRUTURAL

Segundo Hejll (2007), o monitoramento das condições estruturais é definido como sendo um método para monitoramento *in loco* e avaliação de desempenho de estruturas civis. Soluções tecnologicamente viáveis para monitoramento contínuo e eficaz estão sendo estudadas em diversos centros de pesquisa ao redor do mundo. A Universidade da Califórnia, campus Berkeley, desenvolveu uma plataforma de monitoramento baseada em uma arquitetura aberta de hardware e software denominada *Mote Smart Sensor* (HILL et al., 2000; NAGAYAMA, 2004) que consiste basicamente na implementação de um micro-sensor que utiliza um sistema de comunicação via rádio, o que possibilita a sua aplicação em monitoramento embarcado. Iniciativas como esta são suportadas pela DARPA[1], através de programas específicos, cujo objetivo é desenvolver um dispositivo sensor autônomo com dimensões de $1mm^3$ para implementação de redes de sensores para monitoramento de variáveis físicas, químicas e biológicas. Tais iniciativas, entretanto, dependem do adequado suporte da indústria de semicondutores para a implementação dos circuitos e dispositivos integrados, tais como os MEMS[2].

Na Universidade de Keio no Japão (LUI e TOMIZUKA, 2003), pesquisadores desenvolveram sensores de deformação e de tensão mecânica que não necessitam de alimentação contínua e podem ser diretamente envolvidos em concreto, sendo a transmissão da informação feita através de sinais de rádio. Outro trabalho na mesma linha é de Williams et al. (1998) onde é

[1]DARPA = *US Defense Advanced Research Projects Agency*
[2]MEMS = *Micro Electrical Mechanical Systems*

apresentado uma nova idéia - bastante interessante - de que os sensores integrados à estrutura sejam auto-suficientes, ou seja, capazes de gerar a própria energia de que necessitam. A própria vibração da estrutura seria utilizada para gerar a energia necessária para os elementos.

Sensores adaptados a estruturas civis monitoram valores máximos de deformação elástica, aceleração, deslocamento, vibração, energia absorvida e deformação plástica acumulada, no sentido de se avaliar as condições de estabilidade e danos causados às obras civis. Como resultado destas pesquisas, uma gama de sensores aplicados na construção civil é relatada na literatura, além do extensômetro padrão. Em Lin et al. (2003), por exemplo, é apresentado uma rede de sensores distribuída para monitoramento das condições estruturais de uma ponte. O princípio utilizado é o da reflectometria elétrica no domínio do tempo ETDR[3] . Esta técnica consiste na detecção do tempo de reflexão de um pulso aplicado à uma estrutura. Conforme tempo de resposta e o formato do sinal refletido, pode-se inferir os esforços a que a estrutura está submetida, detectar descontinuidades (por exemplo, rachaduras) e determinar precisamente o ponto destas ocorrências. Assim não só se detectaria a existência do problema, mas também sua localização.

Em Symans e Kelly (1999) é proposto um método utilizando técnica *neuro-fuzzy* para minimizar os danos causados a estruturas em caso de terremotos. Sua proposta consiste na instalação de amortecedores semi-ativos controlados por um sistema computacional que detecta a ocorrência do abalo sísmico, determina sua amplitude e atua eficazmente sobre os amortecedores. Sua proposta visa diminuir os efeitos nocivos dos terremotos através do monitoramento contínuo das estruturas.

Em Elgamal et al. (2005) é apresentado um sistema de aquisição de dados utilizando um computador, o qual é responsável pela coleta das informações e pela detecção de possíveis problemas, através da programação de um limiar de anormalidade fornecido previamente. Em caso de anormalidade é enviado um alarme, através da internet, para os órgãos responsáveis pela manutenção das estruturas (pontes, no caso específico daquele estudo) para que tomem as providências necessárias. Uma série de sensores é distribuída pela estrutura e a tarefa de decidir a normalidade ou não da condição atual é decidida por algoritmos implementados no software residente do computador. Segundo os autores, nos Estados Unidos existe uma rede de infraestruturas que atinge o montante aproximado de 1 trilhão de dólares. Evidentemente soluções que detectem ou diminuam os danos causados por desastres naturais contribuem para a vitalidade da economia daquele país, considerando os altos valores envolvidos. Foi criado um portal na internet (UCSD, 2007) onde são disponibilizados ferramentas para análise das condições de estruturas.

[3]ETDR = *Electrical Time Domain Reflectometry*

Schulz e Sundaresan (2006) apresentaram um relatório de testes para determinação das condições de fadiga dos diversos materiais utilizados em estruturas de turbinas eólicas. Estas são compostas por centenas de unidades que se movimentam constantemente e sujeitas a esforços variados, conforme a força, sentido e direção do vento. Portanto a monitoração permanente destas estruturas aumenta em muito a segurança das instalações e diminui os custos de manutenção, já que esta pode ser feita imediatamente ao se detectar qualquer anormalidade, impedindo que o problema se agrave e cause falha sistêmica, caso em que os prejuízos seriam maiores tanto pelo custo da substituição dos componentes danificados quanto pelos lucros cessantes durante o período em que a turbina ficaria parada para manutenção.

Na tese de doutorado de Hejll (2007), são combinadas diversas técnicas probabilísticas com a utilização de equipamentos de monitoração em tempo real. Uma ação é sugerida quando são comparados os dados atuais com as informações estatísticas anteriores da estrutura sendo monitorada.

Visando desenvolver um estudo comparativo entre o modelo teórico e os resultados obtidos, Gomes e Calil Júnior (2005) desenvolveram um sistema para aquisição de esforços em silos horizontais. Os dados experimentais foram utilizados para validação do modelo teórico proposto pelos autores.

Existem inúmeros outros trabalhos relativos ao tema. Um trabalho que faz uma revisão bibliográfica aprofundada destes pode ser encontrado em Sandoval (2004).

2.2 UTILIZAÇÃO DE SENSORES INTELIGENTES PARA MONITORAMENTO ESTRUTURAL

Pereira et al. (2003) apresentam as características de várias tecnologias sem fio e suas aplicações. São discutidos os vários aspectos de uma rede de sensores e suas aplicações no mundo real. Nesse trabalho, é mostrado as definições existentes, as principais características de uma rede de sensores, as métricas de desempenho, a arquitetura, os modelos de comunicação e envio de dados existentes, os protocolos, a segurança entre outros pontos relevantes.

São objetos de estudo do trabalho de Sandoval (2004) os chamados “Sensores inteligentes” ou *smart-sensors* - composto por um microcontrolador e um transceptor sem fio. Sua pesquisa se concentra na plataforma *Mote* desenvolvida pela Universidade da Califórnia, campus Berkeley, que é uma plataforma de desenvolvimento para aplicações de monitoramento estrutural utilizando sensores inteligentes. É composta por um ambiente de *software* e *hardware* que seguem a filosofia de código aberto. Foram desenvolvidas, para efeito de validação e testes,

unidades de rede sensoras, chamadas de *Tadeo*, compostas por um acelerômetro, um microfone, um termistor e um foto-resistor.

Raghuvanshi (2006) desenvolveu uma unidade sensora inteligente comparável à plataforma *Mote*, referenciada no parágrafo anterior. O protocolo de comunicação implementado foi o *Zigbee*, com taxa de transmissão de 250 kbps. Na validação do produto foi utilizada uma bateria de 580 AH para alimentar uma rede de sensores que funcionou por 254 dias.

Bu (2005) apresenta um novo conceito de rede de controle chamada de *Wireless ad-hoc control networks (WACNets)*. Tais sistemas consistem de um grande números de nós contendo sensores e/ou atuadores, inteligência local e controle e componentes para processamento de dados e comunicação. O tamanho, número, densidade, capacidade e dependência do local de instalação destes nós são determinados pela aplicação específica em que são utilizados. Os protocolos e algoritmos existentes nos nós possuem a capacidade de auto-organização e intercooperação. Naquele trabalho é feita uma implementação utilizando sensores inteligentes compatíveis com a norma IEEE 1451 e uma rede sem fio *Bluetooth* foi utilizada. Os nós desenvolvidos contêm controladores digitais locais e uma rede digital de comunicação que liga as unidades de rede a um computador central, responsável pelas tarefas de supervisão e sincronismo do sistema.

Em Portugal, Morais et al. (2005) descrevem o desenvolvimento de um sistema automático de medição utilizado na realização de ensaios de ancoragens. O equipamento de medição utilizado é baseado em sensores inteligentes (*Smart-Sensor*). A filosofia do sistema descrito apresenta algumas semelhanças com o proposto neste trabalho, ou seja, unidades de aquisição descentralizada, condicionamento local dos sinais e posterior transmissão dos dados para uma unidade de aquisição mestre, responsável pelo processamento das informações coletadas.

Chaves (2001) apresenta uma aplicação de coleta de amostras em associação com receptor GPS para determinar a localização espacial do sensor.

Em Xu et al. (2004) é proposto uma rede de sensores sem fio, batizada de *Wisden*, para monitoramento estrutural. Neste trabalho são abordadas as vantagens da aquisição de dados de forma descentralizada. O trabalho consiste em melhorar a qualidade do sinal transmitido por transmissores de RF de baixa potência utilizando técnica de *Wavelets*.

Em Hill (2003) é desenvolvido um sistema operacional, chamado de *TinyOS*, e três plataformas de hardware visando atender ao monitoramento estrutural através de uma rede de comunicação sem fios. Uma das plataformas desenvolvidas é baseada no desenvolvimento de um circuito integrado que contém os principais elementos do terminal de aquisição, ou seja, a máquina de estados do protocolo de comunicações e a interface de comunicação física.

Em Arms et al. (2004) é descrita uma metodologia utilizada para desenvolver um sensor de fio modular capaz de ser reprogramado através de um telefone celular a distância. Foi produzida uma estação base com uma interface para um telefone celular. Isto permite o acesso remoto aos sensores de qualquer lugar onde haja comunicação via celular disponível, mesmo a quilômetros de distância. Os parâmetros de operação, tais como: limiar de disparo, taxa de aquisição de dados, duração da coleta de amostras e número de canais ativos podem ser todos reprogramados remotamente. A estação base pode enviar comandos tipo *broadcast* tais como, sair do modo de baixo consumo (*sleep*), inicie a coleta de dados, ou volte ao modo de baixo consumo.

Dysdenborg (2004) desenvolveu, na sua tese de doutorado, um sistema de comunicação de sensores orientada à conexão, interligados em um sistema sem fio baseado na tecnologia *Bluetooth*.

Em Brownjohn et al. (2004) é apresentado um caso de aplicação real, onde um viaduto de uma via expressa de Singapura foi instrumentado com sensores estáticos. Nesta aplicação, os dados são gerenciados através de uma rede sem fio e internet.

Ilarionov et al. (2005) apresenta um sistema microcontrolado que faz a aquisição de temperatura e pressão utilizando microcontroladores PIC16F873 (MICROCHIP).

2.3 TRABALHOS RELACIONADOS

Embora exista uma profusão de trabalhos relativos à rede de sensores, foram procurados outros semelhantes a este aqui desenvolvido, escritos nas línguas inglesa, portuguesa e espanhola e não foi encontrado nenhum trabalho acadêmico que seja exatamente igual a este.

O trabalho com o tema mais similar encontrado foi o de Simunic et al. (2001) que descreve a implementação de um novo sistema digital de aquisição de dados de extensômetros elétricos utilizados no monitoramento da ponte Maslenica na Croácia em substituição a um sistema de aquisição analógico anteriormente existente na mesma. A troca foi proposta devido às limitações do sistema analógico. O sistema de aquisição é baseado em CLPs e a rede escolhida foi a PROFIBUS DP. Escolha que segundo os autores ocorreu após um minucioso estudo das opções existentes. O monitoramento analógico convencional foi comparado ao novo monitoramento digital e as vantagens da utilização de sistemas de aquisição digital ficaram evidentes. Dentre estas os autores destacam:

- sistema de medição padronizado (equipamentos industriais)
- grande número de canais de medida (mais de 100)

- sistema aberto (possibilidade de expansão e desenvolvimento)
- Sinais padronizados
- Compatibilidade com sensores de sistemas convencionais
- maiores distâncias entre os sensores e a unidade de aquisição de dados.
- redução do cabeamento
- aplicabilidade de testes em campo.

Na dissertação de mestrado de Mello (2007), sob o título "Um estudo da aplicação de redes de sensores para monitoração da proteção catódica em dutos" é feita uma aplicação de rede sem fio para monitoramento do grau de corrosão de dutos utilizados para transporte de petróleo ou de seus derivados. Neste trabalho são avaliados e comparados os protocolos para transmissão sem-fio *Zigbee* e *Directed Diffusion*. O autor optou pela utilização do último. Um dos objetivos desse trabalho foi o de construir um sistema de baixíssimo consumo, já que a alimentação dos mesmos seria a pilhas. Como era de se esperar, uma de suas limitações é a necessidade de manutenção periódica para troca das pilhas e baixo alcance dos transmissores.

Na tese de doutorado de Penteado Neto (2005), é apresentado um sistema de rede de sensores extensométricos visando detectar a ruptura de cabos ou isoladores da rede de distribuição de energia elétrica. Em cada ponto monitorado é colocado uma unidade de rede microcontrolada que condiciona os sinais e transmite os dados através de RF (Rádio Freqüência). Uma das limitações deste projeto é a dificuldade da transmissão utilizando estes módulos de RF sujeitos a grandes interferências e ruídos. Outra é a necessidade de manutenção periódica para troca das pilhas.

2.4 CONCLUSÕES

Nota-se pelos trabalhos citados neste capítulo que há uma grande quantidade deles utilizando soluções sem-fios para a tarefa de monitoramento estrutural. O tema é atual e existem vários grupos de pesquisas trabalhando para otimizar e viabilizar a utilização de redes de sensores em larga escala. No entanto, algumas questões ainda não foram satisfatoriamente sanadas, tais como a autonomia das baterias, confiabilidade e qualidade do meio de transmissão, padronização dos protocolos de acesso e custos envolvidos - ainda significativamente altos.

3 *FUNDAMENTAÇÃO TEÓRICA*

Neste capítulo são descritos os fundamentos teóricos e os conceitos básicos necessários para o desenvolvimento do tema objeto deste trabalho. Sua leitura é útil para a compreensão dos procedimentos, técnicas e métodos utilizados.

3.1 EXTENSÔMETROS ELÉTRICOS DE RESISTÊNCIA

William Thomson, também conhecido como Lord Kelvin, foi o primeiro a relatar em 1856 que os condutores metálicos submetidos a esforços mecânicos mudam as características da sua resistência elétrica. Kelvin observou que a resistência de um fio varia deterministicamente quando comprimido ou tracionado. Desta forma, se um condutor é conectado a uma estrutura mecânica de maneira que a mudança em comprimento da estrutura é igual a mudança de comprimento do condutor, então a variação resistência do fio é diretamente proporcional ao esforço sofrido (ANDOLFATO et al., 2007).

Fundamentalmente, os extensômetros são projetados para converter deformações lineares em sinais elétricos. Na prática, os extensômetros elétricos de resistência são mais conhecidos pelo seu nome em inglês, ou seja, *strain gauges*.

Existem vários tipos de extensômetros. Um extensômetro típico é mostrados na Figura 4.

Os extensômetros laminares (*foil gauges*), como ilustrados nas Figuras 5 e 6, são os mais utilizados para aplicações rotineiras.

Constituem-se de filme metálico com uma geometria de dobras e montado sobre uma base polimérica. Alguns podem ser montados sobre uma base cerâmica. Em geral, os extensômetros a fios são para aplicações em altas temperaturas.

Os extensômetros laminares apresentam as seguintes características (MADGETECH, 2007):

- Alta estabilidade.

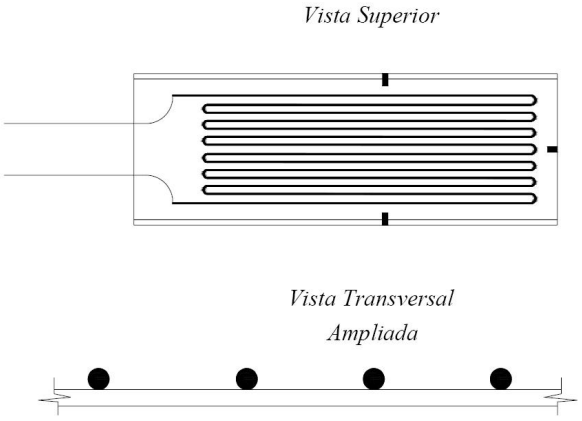

Figura 4: Extensômetro Elétrico a fio (*wire gauges*)

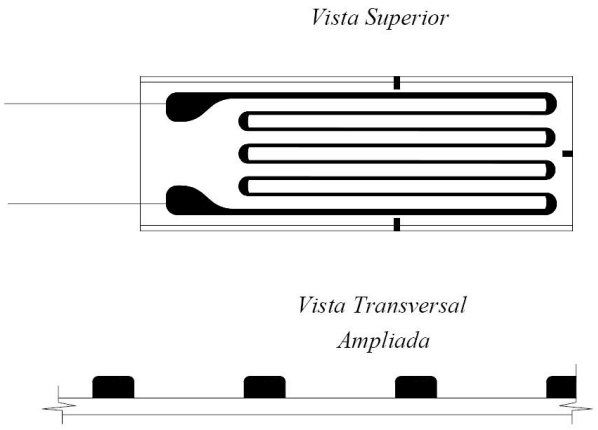

Figura 5: Extensômetro elétrico laminar (*foil gauges*)

- Bom fator de proporcionalidade.
- Preços razoáveis.
- Baixa tensão elétrica de saída
- necessita de amplificação.

O processo de fabricação deste formato é feito com processos de foto corrosão (*photoetching*). Como este processo é versátil, uma grande quantidade de formatos está disponível para as mais variadas aplicações. O extensômetro mais curto disponível é da ordem de 0,2 mm; e o mais longo é cerca de 100 mm. Os valores de resistências padrões são de 120 e 350 Ohms. Outros valores para aplicações especiais de 500, 1000, e 5000 Ohms são também disponíveis.

Para a compreensão deste princípio será feita a análise que se segue.

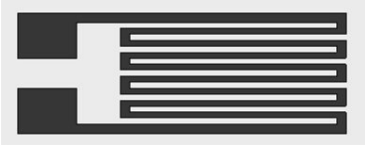

Figura 6: Extensômetro laminar típico (*foil gauges*)

3.1.1 FATOR DE DEFORMAÇÃO (ε)

Todos os corpos se deformam quando estão sob a ação de forças externas. Se um fio é submetido a um esforço de tração, ele vai se tornar ligeiramente mais longo e sua área de seção transversal é reduzida. Da mesma forma, se um fio é submetido a um esforço compressivo, sua área de seção transversal aumenta. A Figura 7 ilustra esse fenômeno.

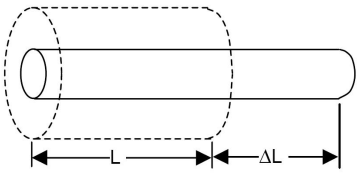

Figura 7: Deformação de um corpo submetido à uma força externa

O comprimento inicial L atingirá o valor L_F após a aplicação da força. O fator de deformação (ε) é calculado através da equação:

$$\varepsilon = \frac{L_F - L}{L} = \frac{\Delta L}{L}$$

Verifica-se que o fator de deformação (ε) é uma quantidade adimensional, no entanto, é comum expressar o fator de deformação como a razão de duas unidades de comprimento, por exemplo, como m/m ou in/in. Como a tensão é adimensional, não há necessidade de conversão de unidade.

A região elástica dos materiais varia entre 0,05% a 0,2% ($\varepsilon = 0,0005$ a $\varepsilon = 0,002$). Como os valores típicos para os fatores de deformação são pequenos e menores que $0,002$, são normalmente expressos em unidades de micro-deformação (*micro-strain*) : **Micro-deformação = deformação x 10^6.**

Por exemplo, um fator de deformação de $0,001$ será expresso por 1000 $\mu m/m$ ou $1mm/m$. Em geral, o menor valor encontrado na prática situa-se na faixa de unidades de $\mu m/m$.

3.1.2 TENSÃO MECÂNICA

Para uma distribuição uniforme das forças resistivas internas em um determinado corpo, a tensão mecânica (*stress*) pode ser calculada dividindo-se a força aplicada (F) pela unidade de área (A), matematicamente:

$$\sigma = \frac{F}{A} \tag{3.1}$$

onde:

F = força aplicada em Newtons (SI);

A = área da seção transversal da barra em m^2 (SI)

σ = tensão mecânica em $N/m^2 = Pa$ (Pascal) (SI)

3.1.2.1 RELAÇÃO ENTRE TENSÃO MECÂNICA E DEFORMAÇÃO

A relação entre a tensão aplicada e a deformação provocada é obtida através da lei de Hooke:

$$E = \frac{\sigma}{\varepsilon}$$

A constante de proporcionalidade (E) entre a tensão e o fator de deformação é chamado de módulo de elasticidade (também conhecido como módulo de Young).

Assim, a lei de Hooke estabelece uma relação linear entre a tensão e a deformação, ou seja,

$$\sigma = E\varepsilon$$

Deve-se mencionar que esta lei é válida apenas para a região elástica dos materiais, ou seja, àquela na qual a deformação do material não é irreversível. Acima do limite elástico, denominada de deformação plástica, o material se deforma de maneira definitiva, onde a lei de Hooke não mais se aplica. Se a tensão continuar aumentando, existirá um certo valor em que o material vai se quebrar ou cisalhar. A Figura 8 mostra um gráfico de tensão versus fator de deformação típico (ANDOLFATO, apud FURMAN, 2003).

Então, desde que não se ultrapasse o limite de elasticidade do material, pode-se obter uma relação direta entre uma tensão mecânica e a deformação do material.

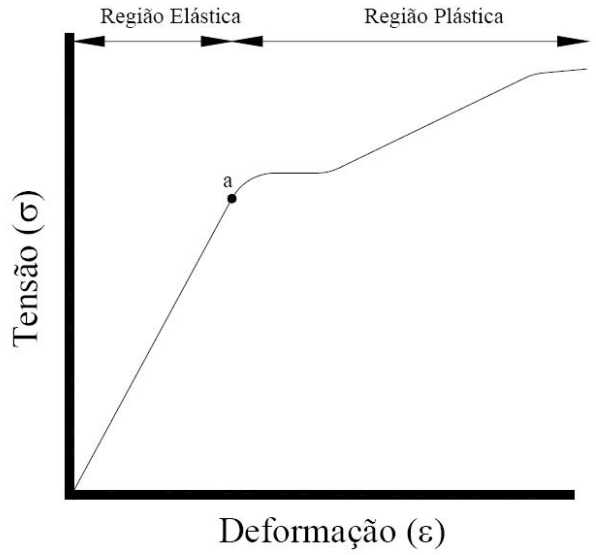

Figura 8: Lei de Hooke - Tensão versus Deformação

3.1.3 RELAÇÃO ENTRE TENSÃO ELÉTRICA E DEFORMAÇÃO

Os valores da deformação, da tensão mecânica e da força aplicada a um corpo de prova estão intimamente relacionados. Existem várias maneiras de se medir deformações. Dentre estas, será utilizado no presente trabalho, os extensômetros de resistência elétrica.

A equação da resistência R de um fio condutor é dada por:

$$R = \frac{\rho L}{A} \tag{3.2}$$

onde

ρ= resistividade do condutor - resistência específica do metal (Ωm)

L= comprimento do condutor (m)

A= área da seção transversal do condutor (m^2)

diferenciando a equação 3.2 resulta

$$dR = \frac{\partial R}{\partial \rho} d\rho + \frac{\partial R}{\partial L} dL + \frac{\partial R}{\partial A} dA \tag{3.3}$$

$$\frac{\partial R}{\partial \rho} = \frac{L}{A}$$

$$\frac{\partial R}{\partial L} = \frac{\rho}{A}$$

$$\frac{\partial R}{\partial A} = -\frac{\rho L}{A^2}$$

e dividindo todos os membros da equação 3.3 por R

$$\frac{dR}{R} = \frac{\frac{L}{A}}{R}d\rho + \frac{\frac{\rho}{A}}{R}dL + \frac{-\frac{\rho L}{A^2}}{R}dA$$

substituindo R por $\frac{\rho L}{A}$, fica:

$$\frac{dR}{R} = \frac{\frac{L}{A}}{\frac{\rho L}{A}}d\rho + \frac{\frac{\rho}{A}}{\frac{\rho L}{A}}dL + \frac{-\frac{\rho L}{A^2}}{\frac{\rho L}{A}}dA$$

$$\frac{dR}{R} = \frac{L}{A}\frac{A}{\rho L}d\rho + \frac{\rho}{A}\frac{A}{\rho L}dL - \frac{\rho L}{A^2}\frac{A}{\rho L}dA$$

e finalmente:

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \frac{dL}{L} - \frac{dA}{A} \tag{3.4}$$

Notar que esta equação relaciona variações de resistência elétrica do condutor com variações de resistividade (o chamado termo piezoresistivo), com a deformação axial do condutor ($\varepsilon_a = dL/L$) e o termo $\frac{dA}{A}$ representa a variação relativa na área da seção transversal do condutor devido a tensão exercida sobre o mesmo. Será visto a seguir que dA/A e dL/L estão relacionados. Assim, se a variação de resistividade do condutor é pequena, pode-se pensar em medir a deformação de um condutor metálico medindo-se a variação de sua resistência elétrica.

Será mostrado a seguir como a deformação axial e a variação da área transversal se relacionam.

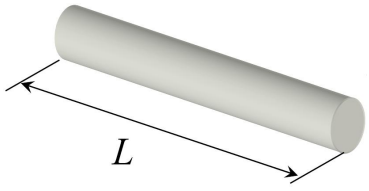

Figura 9: Condutor de seção transversal circular

Para um condutor de seção transversal circular, como mostrado na Figura 9, a área é dada por:

$$A = \frac{\pi D^2}{4}$$

$$dA = \frac{\pi D}{2} dD$$

$$\frac{dA}{A} = \frac{\frac{\pi D}{2} dD}{\frac{\pi D^2}{4}} = \frac{\pi D}{2} dD \frac{4}{\pi D^2} = 2\frac{dD}{D}$$

Para o caso de uma tensão axial, tem-se que

$$\varepsilon_a = \frac{dL}{L}$$

$$\varepsilon_t = \frac{dD}{D}$$

onde

ε_a= fator de deformação axial no condutor

ε_t= fator de deformação transversal no condutor

A razão entre os fatores das deformações transversal e axial é o chamado módulo de Poisson (ν). É importante ressaltar que o módulo de Poisson, da mesma forma que a resistividade e que o módulo de elasticidade, é uma propriedade do material. Por exemplo, para o aço varia de 0,25 a 0,3. Matematicamente, o coeficiente de Poisson é dado por:

$$\nu = -\frac{fator-de-deformacao-transversal}{fator-de-deformacao-axial} = -\frac{\varepsilon_t}{\varepsilon_a}$$

$$\varepsilon_t = \frac{dD}{D} = -\nu\varepsilon_a \tag{3.5}$$

$$\frac{dA}{A} = 2\frac{dD}{D}$$

$$\frac{dA}{A} = -2\nu\varepsilon_a \tag{3.6}$$

Se considerarmos a elongação de um fio, $L \rightarrow L + \Delta L$, pelo efeito de Poisson vai haver também uma redução da área seccional, $A \rightarrow A - \Delta A$. A partir da Equação 3.2 pode ser visto

que ambos os efeitos contribuem para o aumento da resistência.

Substituindo as equações 3.5 e 3.6 na equação 3.4, temos

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \varepsilon_a - 2\varepsilon_t$$

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \varepsilon_a - 2(-\nu\varepsilon_a)$$

Desta forma, então, relacionamos a variação de resistência elétrica do condutor com a deformação axial:

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \varepsilon_a\,(1 + 2\nu)$$

Há ainda a considerar a variação relativa da resistividade e do módulo de Poisson, mas estas são influências secundárias se o material não estiver sendo submetido a carregamentos extremos (por exemplo, oscilando em alta frequência, o que pode resultar em aquecimento do elemento), isto é, estes termos devem ser constantes na faixa de carregamento do material.

Rearranjando, obtém-se:

$$\frac{dR}{R} = \varepsilon_a(\frac{d\rho}{\rho}\frac{1}{\varepsilon_a} + 1 + 2\nu)$$

$$K = \frac{d\rho}{\rho}\frac{1}{\varepsilon_a} + 1 + 2\nu$$

onde K é o fator de sensibilidade do extensômetro, também conhecido pelo seu nome original da língua inglesa *Gauge Factor (GF)*.

O primeiro termo à direita do sinal de igualdade é o termo piezoresistivo, o qual se espera manter constante durante o carregamento do material. Assim sendo, chega-se a:

$$\frac{dR}{R} = \varepsilon_a K \tag{3.7}$$

Esta expressão significa que a variação na resistência é diretamente proporcional ao fator de deformação axial da amostra.

O fator de sensibilidade do extensômetro (K) depende das características de fabricação deste tais como: materiais utilizados, formas construtivas e tamanho, dentre outras. Este fator

é determinado pelo fabricante pela medida $\Delta R/R$ para uma amostra extraída de cada lote de produção e esta informação é fornecida junto com o extensômetro. O valor de K é aproximadamente constante para a maioria dos tipos de extensômetros e estudos experimentais mostram que apresentam valores na faixa de 2 à 4**.** Além disso a quantidade $(1 + 2\nu)$ é aproximadamente igual á 1,6 para a maior parte destes materiais, o que significa que o termo $\frac{d\rho}{\rho}\frac{1}{\varepsilon_a}$ contribui com um valor entre 0,4 à 2,4 (AGILENT TECHNOLOGIES, 1999).

Também estão disponíveis extensômetros de semicondutor. Estes apresentam uma alta sensibilidade negativa (isto é, a resistência diminui com a tensão aplicada) da ordem de -50 à -200, mas por outro lado, são altamente não lineares.

Na Figura 10 pode-se observar que o fator de sensibilidade do extensômetro corresponde à taxa de inclinação da curva.

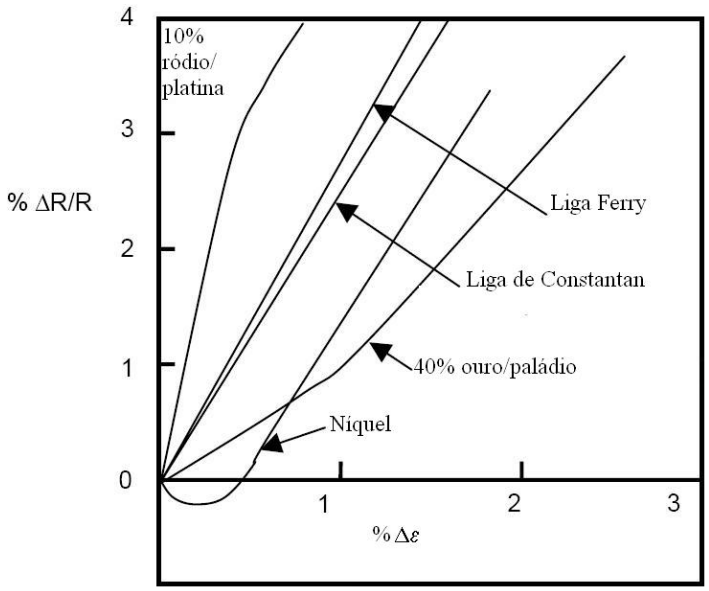

Figura 10: Curva dos fatores de sensibilidade de diversos materiais

Uma lista da ligas mais comuns empregadas na fabricação de extensômetros elétricos, juntamente com a sua sensibilidade, é mostrada na Tabela 1. Os extensômetros mais comumente utilizados são os fabricados com liga de cobre-níquel, conhecidos como Constantan.

3.1.3.1 Medição da variação da resistência dos extensômetros

A resolução limite para a maioria dos extensômetros elétricos é em torno de 1 $\mu m/m$ (1 $\mu\varepsilon$). Com um extensômetro de resistência típica de 120Ω e um fator do extensômetro (*gauge factor*) igual a $2,0$, implica que temos que ser capazes de detectar uma variação na resistência de $\Delta R = 120 \times 2,0 \times 10^{-6}\Omega = 0,00024\Omega$. Enquanto que alguns multímetros digitais de precisão são capazes de medir tal ordem de resistências, estes apresentam a desvantagem do

Tabela 1: Fatores de sensibilidade do extensômetro para alguns materiais

	Baixa Deformação	Alta Deformação	Elongação máxima (%)
Cobre	2,6	2,2	0,5
Constantan	2,1	1,9	1,0
Níquel	-12	2,7	–
Platina	6,1	2,4	0,4
Prata	2,9	2,4	0,8
40% Ouro/Paládio	0,9	1,9	0,8
Semicondutor	-100	-600	–

Fonte: DATAFORTH (2007)

alto custo. Uma maneira bem mais simples e barata de se conseguir o mesmo resultado é inserir o extensômetro em um circuito que seja capaz de detectar pequenas variações nas resistências, em vez de valores absolutos. Estes circuitos convertem a variação ΔR em uma tensão elétrica proporcional a esta variação. O circuito mais comumente utilizado para esta tarefa é a uma configuração chamada de "Ponte de Wheatstone".

3.2 PONTE DE WHEATSTONE

A ponte de Wheatstone foi desenvolvida nos primórdios da eletrônica como uma maneira de medir com precisão valores de resistores sem necessitar de uma tensão de referência ultra estável ou um ohmímetro de alta impedância. Embora as pontes resistivas sejam raramente utilizados para seu propósito original, elas continuam sendo largamente empregadas nas aplicações de sensoriamento. Nesta seção serão discutidas algumas considerações fundamentais na utilização dessa topologia.

3.2.1 CONFIGURAÇÃO BÁSICA DA PONTE

A Figura 11 mostra o diagrama de uma ponte de Wheatstone clássica, onde a tensão de saída (Vo) é a diferença de tensão entre Vo^+ e Vo^-.

Quando utilizada em um sensor, o valor de um ou mais resistores vai se alterar de acordo com a intensidade da propriedade sendo medida. Estas mudanças na resistência causam também uma alteração no valor da tensão de saída. A Equação 3.8 mostra que a tensão de saída, Vo, é uma função da tensão de excitação e de todos os resistores componentes da ponte.

$$V_{CB} = Vo^- = \frac{R3}{R3 + R4} Ve$$

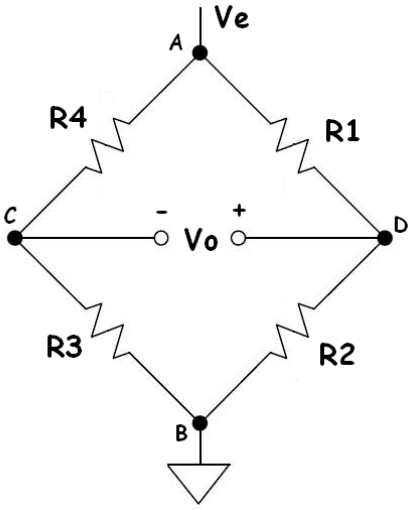

Figura 11: Diagrama de uma Ponte de Wheatstone básica

$$V_{DB} = Vo^{+} = \frac{R2}{R1 + R2} Ve$$

$$Vo = Vo^{+} - Vo^{-}$$

$$Vo = Ve \left(\frac{R2}{R1 + R2} - \frac{R3}{R3 + R4} \right) \tag{3.8}$$

Embora a Equação 3.8 não seja muito elegante, ela pode ser simplificada para a maioria das pontes comumente utilizadas. A saída da ponte são mais sensíveis a alterações na resistência quando Vo^{+} e Vo^{-} são iguais à metade de Ve. Esta condição é facilmente atingida utilizando-se quatro resistores iguais de valor R. Alterações na resistência causadas pela variação da propriedade sendo medida são consideradas pelo termo ΔR. Resistores com o termo ΔR são chamados de resistores "ativos". Nos quatro casos que se seguem, todos os resistores possuem o mesmo valor nominal R. Um, dois ou quatro resistores estarão ativos (terão um termo ΔR). Na derivação destas equações, ΔR é assumido ser positivo. Se a resistência realmente diminui então o termo $-\Delta R$ é utilizado. No casos especiais abaixo, a magnitude de ΔR é a mesma pra todos os resistores ativos.

Como regra geral, pode-se verificar que: alterações de iguais valores em braços adjacentes cancelam-se mutuamente.

3.2.2 QUANTIDADE DE SENSORES

Os extensômetros podem ser dispostos em diferentes configurações: $1/4$ de ponte, $1/2$ ponte ou ponte completa. Para utilização na configuração em $1/4$ de ponte, deve-se fornecer os outros 3 elementos resistivos da ponte de forma a completá-la. Temos então as seguintes possibilidades de uso:

- 4 extensômetros (ponte completa) - Todos os ramos da ponte são extensômetros. É necessário apenas fornecer a tensão de excitação e medir a saída da ponte.
- 1 extensômetro ($1/4$ ponte) - Somente um extensômetro é fornecido. O circuito de instrumentação deverá fornecer os resistores necessários para completar a meia ponte e além disso, instalar um resistor adicional, do mesmo valor do extensômetro, no ramo adjacente ao do extensômetro.
- 2 extensômetros ($1/2$ ponte) - Somente dois dos extensômetros são fornecidos. O circuito de instrumentação deve fornecer os outros dois resistores necessários para completar a ponte.

3.2.2.1 Quatro elementos ativos

No primeiro caso, todos os quatro elementos da ponte são ativos. A Figura 12 ilustra este caso. As resistências de $R2$ e $R4$ aumentam com a intensidade da propriedade sendo medida, enquanto que o valor das resistências $R1$ e $R3$ diminuem. Este caso é típico das células de carga que utilizam quatro extensômetros. A orientação física dos extensômetros determinam se seus valores irão aumentar ou diminuir quando a força for aplicada. A Equação 3.9 mostra que esta configuração produz uma relação linear simples entre a tensão de saída (Vo) e a alteração na resistência (ΔR). Esta configuração também fornece o maior sinal de saída. Deve-se notar que a saída não é uma função linear de ΔR, mas sim da relação $\Delta R/R$. Isto é uma diferença sutil, porém importante, porque a mudança na resistência na maioria dos elementos sensores é proporcional à suas resistências nominais.

Da Equação 3.8, fazendo que $R2 = R4 = R + \Delta R$ e $R1 = R3 = R - \Delta R$ tem-se que:

$$Vo = Ve\frac{\Delta R}{R} \tag{3.9}$$

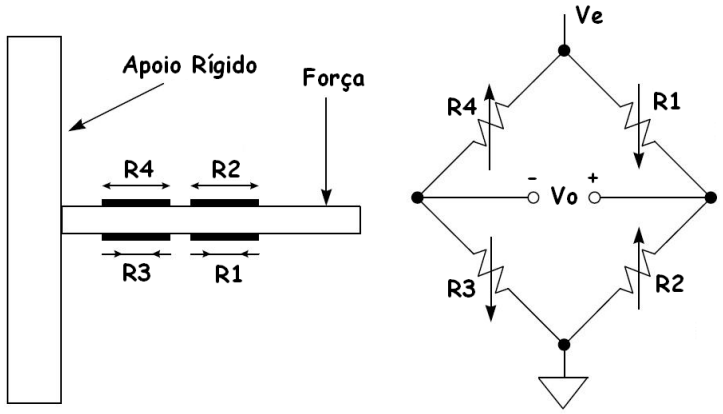

Figura 12: Configuração utilizando 4 extensômetros ativos (*full bridge*)

3.2.2.2 Um elemento ativo

O segundo caso é um simples elemento ativo. Esta configuração é utilizada frequentemente quando o custo ou a fiação é mais importante do que a amplitude do sinal. A Figura 13 ilustra esta situação.

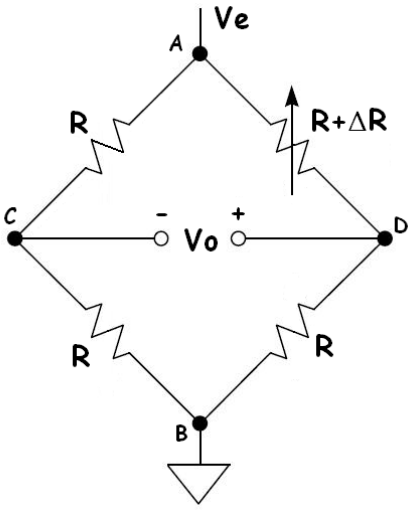

Figura 13: Extensômetro ligado em um quarto de ponte

Para a análise que se segue, será considerado que a tensão fornecida (Ve) é constante, que a impedância de entrada do voltímetro é infinita e que a ponte está perfeitamente balanceada no início ($\Delta R = 0$). Nestas condições, tem-se que a tensão de saída inicialmente é nula ($V_o = 0$).

A corrente que flui de A até B passando por D (ramo direito da ponte) é dada por:

$$I_{ADB} = \frac{Ve}{2R + \Delta R}$$

A diferença de potencial sobre o extensômetro de resistência $R + \Delta R$ é:

$$V_{AD} = (R + \Delta R) \frac{Ve}{2R + \Delta R}$$

A diferença de potencial entre os pontos A e C é dada por:

$$V_{AC} = \frac{Ve}{2}$$

Por conseqüência, a tensão de saída do sistema (V_o) é dada por:

$$V_o = V_{AD} - V_{AC}$$

$$V_o = \frac{Ve\,(R + \Delta R)}{2R + \Delta R} - \frac{Ve}{2}$$

$$V_o = Ve \frac{(2R + 2\Delta R - 2R - \Delta R)}{4R + 2\Delta R}$$

$$V_o = Ve \frac{\Delta R}{4R + 2\Delta R} \tag{3.10}$$

Tipicamente a alteração na resistência é muito pequena comparada com o valor da resistência original ($\Delta R \ll 1\%$), de maneira que podemos aproximar $4R + 2\Delta R \cong 4R$. Assim, obtém-se:

$$V_o = \frac{1}{4} \times Ve \times \frac{\Delta R}{R} \tag{3.11}$$

Lembrando que $\frac{\Delta R}{R} = K \times \varepsilon_a$, obtém-se, finalmente, a expressão para determinar o fator de deformação axial:

$$\varepsilon_a = 4 \times \frac{V_o}{Ve} \times \frac{1}{K} \tag{3.12}$$

Observando-se a Equação 3.11 verifica-se que a saída desta configuração tem apenas $1/4$ da amplitude que teria se tivesse quatro elementos ativos. Outra consideração importante é que a Equação 3.11 foi obtida supondo pequenas variações de resistências. Caso não seja este o caso, a saída é não linear, causada pela existência do termo ΔR no denominador da Equação

3.10. No entanto, esta não linearidade é pequena e determinística e caso necessário, pode ser corrigida através de software.

3.2.2.3 Dois elementos ativos com respostas opostas

O terceiro caso, mostrado na Equação 3.13, tem dois elementos ativos. Por exemplo, na Figura 14 é ilustrada uma barra onde um elemento da ponte é montado na região de tensão ($R + \Delta R$) e o outro em compressão ($R - \Delta R$). Desta maneira, suas resistências variam em direções opostas (ΔR e $-\Delta R$). Ambos os resistores são colocados no mesmo lado da ponte ($R1$ e $R2$, ou $R3$ e $R4$). Como esperado, a sensibilidade é o dobro da configuração que utiliza apenas um elemento ativo e metade daquela que utiliza quatro elementos ativos. A saída desta configuração é uma função linear de $\Delta R/R$.

$$Vo = \frac{Ve}{2}\frac{\Delta R}{R} \tag{3.13}$$

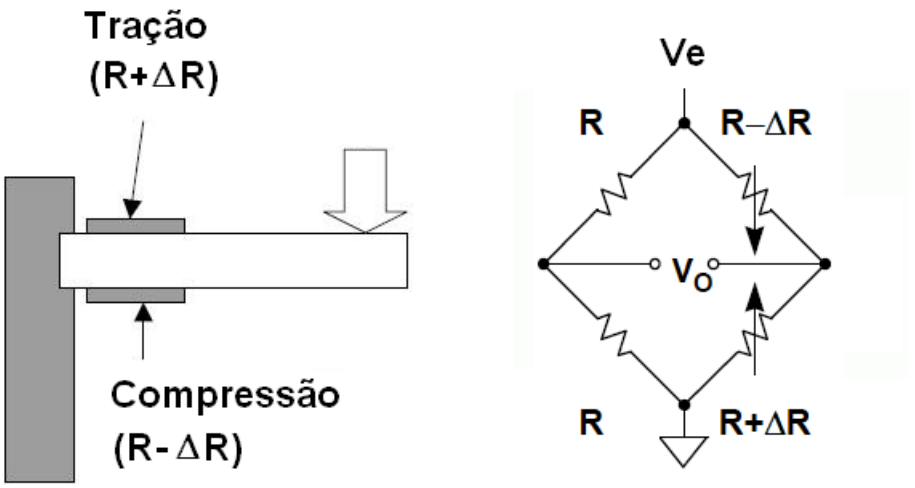

Figura 14: Ponte de Wheatstone contendo dois elementos ativos idênticos e ligados em braços adjacentes.

Tanto no segundo, quanto no terceiro caso apresentado acima, somente metade da ponte é ativa. A outra metade simplesmente fornece uma tensão de referência que é igual à metade de Ve. Conseqüentemente, não é realmente necessário que todos os quatro resistores da ponte tenham o mesmo valor nominal. É importante somente que ambos os resistores do lado esquerdo da ponte sejam iguais um ao outro e que ambos os resistores do lado direito da ponte também sejam iguais um ao outro.

3.2.2.4 Dois elementos ativos idênticos

O quarto caso também utiliza dois elementos ativos, porém têm uma resposta parecida, ou seja, ambos aumentam ou diminuem de valor simultaneamente. Este é o caso quando se utilizam dois extensômetros idênticos montados adjacentes um ao outro e com seus eixos em paralelo. Para ser efetivo, estes resistores devem estar nas diagonais da ponte ($R1$ e $R3$, ou $R2$ e $R4$). A Figura 15 ilustra esta situação.

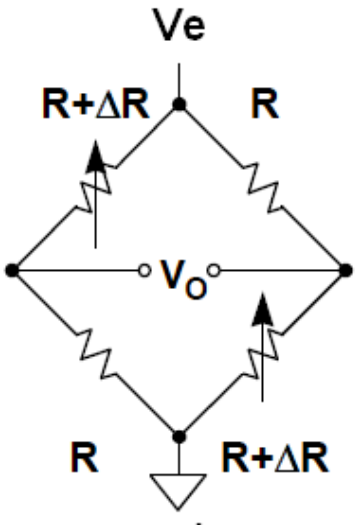

Figura 15: Ponte de Wheatstone contendo dois elementos ativos idênticos em ramos opostos e alimentada por uma fonte de tensão constante.

A vantagem óbvia desta configuração é que o mesmo tipo de elemento sensor pode ser utilizado em ambas as posições. A desvantagem é o resultado não linear da saída causado pelo termo ΔR no denominador da equação 3.14. No entanto, esta não linearidade é determinística e pode ser removida por software.

$$Vo = Ve\left(\frac{\Delta R}{2R + \Delta R}\right) \tag{3.14}$$

A não linearidade também pode ser eliminada alimentando-se a ponte com uma fonte de corrente constante em vez de uma fonte de tensão constante, conforme ilustrado na Figura 16.

Neste caso a tensão de saída é dada por

$$Vo = \frac{Ie}{2}\Delta R \tag{3.15}$$

Onde Ie é a corrente de excitação.

Deve se observar que Vo na Equação 3.15 é uma função em termos de ΔR, não da razão

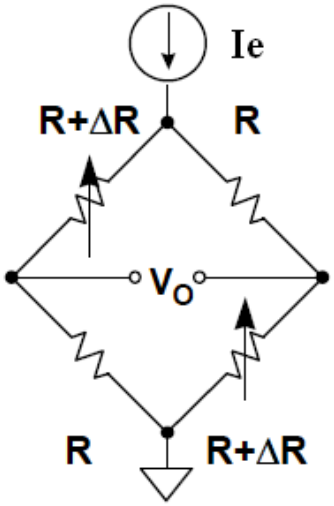

Figura 16: Ponte de Wheatstone contendo dois elementos ativos idênticos e alimentada por uma fonte de corrente constante.

$\Delta R/R$ como nos casos anteriores.

O entendimento dos quatro casos especiais apresentados anteriormente é útil quando se trabalha com elementos sensores individuais. Muitas vezes, no entanto, o sensor possui uma ponte interna com uma configuração desconhecida. Nestas situações não é realmente importante saber a configuração exata. O fabricante irá fornecer as informações necessárias, tais como sensibilidade, erro de linearidade, tensão em modo comum e demais informações relevantes.

3.2.2.5 Ordem de grandeza

O exemplo dado a seguir tem a finalidade de ilustrar a ordem de grandeza das unidades envolvidas.

Exemplo: Um extensômetro tem resistência nominal de 120Ω e um fator K = 2,06. Está instalado em configuração de $^1/_4$ de ponte. Os demais resistores são de 120Ω. Qual será a tensão de saída da ponte com uma deformação de 1000 $\mu strain$=1mm/m se a alimentação da mesma é de 5 Volts?

Inicialmente, se temos todos os resistores iguais na ponte, então:

$$V_o = \frac{1}{4} \times Ve \times \frac{\Delta R}{R}$$

Lembrando que também que $\frac{\Delta R}{R} = \varepsilon_a K$

Assim,

$$V_o = \frac{1}{4} \times Ve \times \varepsilon_a \times K$$

então,

V_o= $\frac{1}{4} \times 5 \times (1000 \times 10^{-6}) \times 2,06$ = $2,575$ mV

3.2.3 EFEITO DA RESISTÊNCIA DA FIAÇÃO

As equações fornecidas anteriormente para configurações de extensômetros em $^1/_4$, $^1/_2$ e ponte completa assumem que a resistência da fiação é desprezível. Enquanto que ignorar estes efeitos pode ser benéfico para se entender o princípio básico de medição utilizando pontes de Wheatstone, fazer isto na prática pode ser uma grande fonte de erro.

Quando os extensômetros estão distantes do local da instrumentação, o efeito da resistência da fiação adicionada ao circuito pode desbalancear a ponte. Por exemplo, para 30 m de fio de cobre de bitola 26 AWG ($133,86\Omega/km$) irá produzir uma resistência adicional de: $R = \frac{133,86}{1000} \times 30 \times 2 = 8,03\Omega$, que é muito maior do que a variação de resistência esperada devido a deformações por esforço para um extensômetro de 120Ω. Além de adicionar um erro por desvio intolerável, a resistência da fiação também dessensibiliza a saída da ponte. Além disso, variações na temperatura dos fios também podem ser significativas.

Utilizando uma conexão a 6 fios pode-se eliminar os efeitos da fiação porque as resistências afetam igualmente dois braços adjacentes da ponte. Como pode ser observado na Figura 17, alterações na resistência da fiação não alteram a razão dos braços da ponte. Desse modo, são canceladas quaisquer alterações nas resistências devido à fiação ou a variação temperatura.

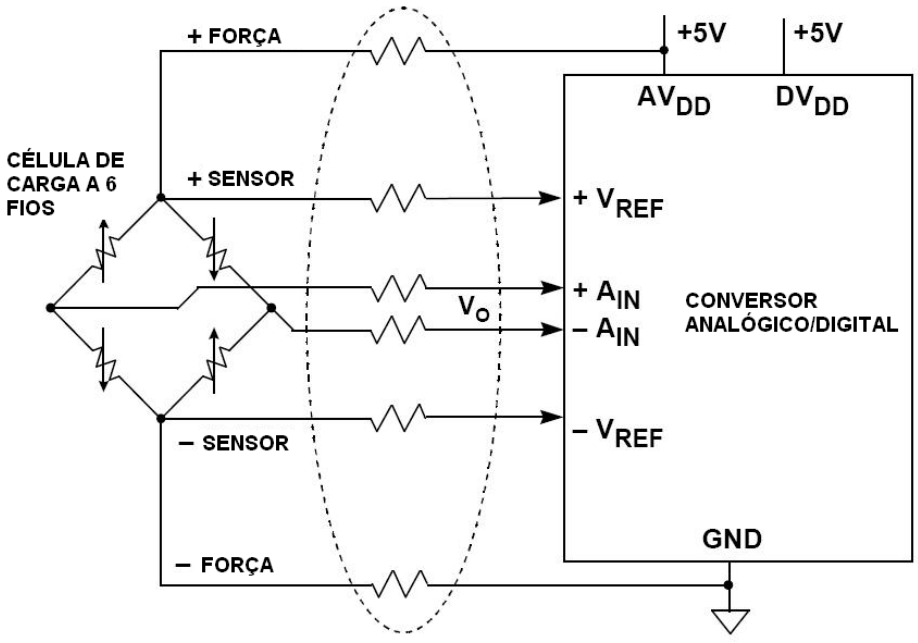

Figura 17: Efeito da fiação na medição

3.2.4 MEDIÇÃO DE TENSÃO MECÂNICA UTILIZANDO EXTENSÔMETROS

Inicialmente deve-se selecionar o extensômetro dentre os ofertados por fabricantes. A variável básica é o fator do extensômetro, K, fornecido nos catálogos. Este fator é a razão entre a variação relativa da resistência e a deformação axial. Depois de escolhido o mais adequado para a aplicação, o extensômetro deve ser instalado (colado) no material a ser testado e ligado ao circuito eletrônico (ponte de Wheatstone). O material deverá ser submetido ao carregamento e a variação relativa da resistência será medida e então a deformação axial poderá ser calculada. Finalmente a tensão mecânica é determinada utilizando se a lei de Hooke.

3.2.5 RELAÇÃO ENTRE TENSÃO MECÂNICA E TENSÃO ELÉTRICA

Lembrando que, para o caso de 4 elementos ativos, a equação da tensão da saída é $Vo = Ve\frac{\Delta R}{R}$, e que $\frac{dR}{R} \approx \frac{\Delta R}{R}$ para pequenas variações, temos que $\frac{\Delta R}{R} = \varepsilon_a K$. Substituindo obtém-se:

$$Vo = Ve \times \varepsilon_a \times K$$

Lembrando ainda que, $\varepsilon_a = \frac{\sigma}{E}$, obtém-se finalmente:

$Vo = \frac{Ve \times K}{E}\sigma$ ou ainda $\sigma = \frac{E}{Ve} \times \frac{1}{K} \times Vo$

Exemplo 1: Um extensômetro de fator $K = 2$ está montado em uma barra de aço retangular que tem módulo de elasticidade $E = 200GPa$. A barra tem $3cm$ de largura e $1cm$ de altura e está sob a ação de uma força de tração de $30kN$. Determine a variação de resistência do extensômetro se sua resistência sem carga for de 120Ω.

Primeiro o cálculo da tensão,

$\sigma = F/A$,

$\sigma = 30kN/(3 \times 10^{-2}m \times 1 \times 10^{-2}m)$

$\sigma = 1,0 \times 10^5 kN/m^2 = 100MPa$

após, o cálculo da deformação com a equação de Hooke,

$\varepsilon_a = \frac{\sigma}{E} = \frac{1,0 \times 10^2 MPa}{200 \times 10^3 MPa}$

$\varepsilon_a = 5,0 \times 10^{-4} m/m$ = 500 $\mu strain$ = $0,5mm/m$

A variação relativa da resistência ($\Delta R/R$) é o produto do fator de deformação (ε_a) pelo

fator do extensômetro (K), matematicamente:

$$\frac{\Delta R}{R} = \varepsilon_a K = 5,0 \times 10^{-4} \times 2,0$$

$$\frac{\Delta R}{R} = 1,0 \times 10^{-3} \Omega/\Omega$$

$$\Delta R = 0,12 \Omega$$

4 PROJETO CONCEITUAL DO SISTEMA

Neste capítulo são mostrados os materiais, procedimentos, técnicas e métodos utilizados para o projeto a nível global. Nos capítulos posteriores são detalhadas cada unidade aqui apresentada. Assim, o objetivo específico deste capítulo é apresentar uma visão geral do sistema, incluindo:

1. OS REQUISITOS BÁSICOS
2. ESTRUTURA GLOBAL
3. DEFINIÇÃO DOS COMPONENTES INTEGRANTES

O projeto desenvolvido consiste, basicamente, em um sistema de aquisição de dados multi ponto dedicado ao monitoramento estrutural, conforme pode ser visto na Figura 18.

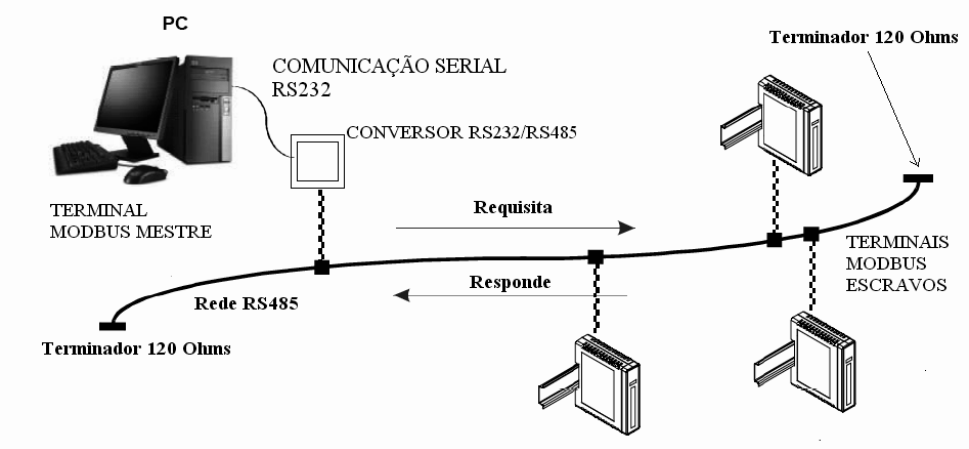

Figura 18: Arquitetura geral do sistema proposto

4.1 TOPOLOGIA

A idéia principal é que todos os sinais elétricos gerados pelos extensômetros devido aos esforços mecânicos presentes na estrutura sejam condicionados localmente através de circuitos analógicos dedicados presentes nos terminais da rede (nó de rede). O sinal condicionado é então digitalizado e transmitido para a unidade gerenciadora, através de protocolo específico com verificação de redundância, para se evitar erros de transmissão de dados. Uma das vantagens da transmissão de sinais digitalizados com modulação diferencial é que estes são muito menos suscetíveis a ruídos externos do que a transmissão de sinais analógicos da ordem de μV por dezenas de metros de fiação, como acontece nos sistemas convencionais. Assim, os tênues sinais gerados pelos sensores não necessitarão fluir através do cabeamento, não estando, portanto, sujeitos a ruídos externos.

Assim, cada unidade de rede contém a instrumentação eletrônica associada à aquisição, tratamento do sinal, calibração do sistema de medida, conversão analógico-digital e comunicação com uma unidade mestre remota, responsável pelo gerenciamento do sistema de aquisição de dados. Toda a conexão entre os diversos nós da rede e o mestre é feita através de um único cabo tipo lógico UTP (*Unshielded Twisted Pair*) padrão utilizado em redes Ethernet, que transporta dados e energia para os nós da rede. Portanto, os nós serão autônomos, ou seja, não necessitarão de fonte de energia disponível *in loco* (embora também possam ter). Como cada unidade da rede recebe a energia a partir de um único ponto central, não há necessidade de manutenção periódica nos pontos de monitoramento instalados, os quais são freqüentemente de difícil acesso. Esta característica torna o sistema de aquisição extremamente versátil para fins de monitoramento permanente de estruturas.

A partir da unidade mestre, uma comunicação será estabelecida com um computador pessoal para o processamento e tratamento gráfico dos sinais coletados a partir dos diversos pontos monitorados.

4.2 DEFINIÇÃO DO SISTEMA DE AQUISIÇÃO DE DADOS

Diferentemente das tecnologias *wireless* que operam em rede, onde cada nó transmite simultaneamente as informações, causando importante congestionamento no espectro eletromagnético, além de cada ponto necessitar de fonte individual de energia, o que gera custo extra de manutenção, o sistema proposto opera sob outra filosofia. Os pontos de monitoramento são permanentemente conectados à estrutura civil ou mecânica e se comunicam individualmente

com a unidade remota mestre em intervalos de tempo exclusivos (*time frames*), o que garante a otimização do consumo de energia.

Cada ponto de medida (nó da rede) incluirá uma interface para sensoriar um extensômetro utilizado para monitoramento de esforços e instrumentação eletrônica associada à aquisição, tratamento do sinal, calibração do sistema de medida, conversão analógico-digital e comunicação com uma unidade mestre.

Embora os conversores analógico/digital do tipo delta-sigma popularizados recentemente possam ser utilizados de maneira simples e com sucesso na presente aplicação, optou-se por implementar o projeto utilizando amplificadores operacionais comuns. Duas das principais razões para essa escolha são a facilidade de aquisição e seu baixo custo, fatores essenciais para este projeto, já que um dos objetivos propostos é o desenvolvimento de uma plataforma de baixo custo. Um dos maiores esforços, portanto, consistiu em conseguir um desempenho satisfatório do sistema de condicionamento de sinais utilizando componentes que apresentam ruídos intrínsecos maiores do que os dos amplificadores de instrumentação comerciais, valor de offset relativamente grande, deriva térmica considerável e baixa estabilidade com o passar do tempo. Todos estes fatores, a princípio, parecem contra-indicar a utilização destes componentes. No entanto, este desafio foi aceito e demandou uma grande carga de trabalho de pesquisa teórica e experimental. A solução encontrada para corrigir estas imperfeições, foi uma combinação de estratégias de *hardware* e de *software*. Uma vez que o sistema dispõe de um microcontrolador, este foi utilizado para a tarefa de implementar os algoritmos de correção dos eventuais desvios. Na seção que trata dos terminais de rede serão detalhados os procedimentos e os códigos de programação utilizados. Importante ressaltar que foram construídos protótipos para se validar as técnicas propostas e, como será visto no Capítulo RESULTADOS E DISCUSSÕES, encontrou-se uma solução satisfatória para o problema.

4.3 DEFINIÇÃO DO PROTOCOLO DE COMUNICAÇÃO

Um aspecto importante considerado neste projeto, refere-se ao meio físico de transmissão dos sinais digitais através de um barramento de comunicação, como por exemplo, a máxima taxa permissível de transmissão de dados, o número admissível de pontos de rede e o comprimento efetivo de cabo que pode ser utilizado. Com a finalidade de tornar o sistema de aquisição viável economicamente, pretende-se utilizar cabos tipo UTP na conexão entre nós da rede e unidade mestre. A adequação entre a informação a ser transmitida (sinais digitais) e o canal de comunicação (par de fios) é essencial para o correto desempenho do sistema de aquisição projetado. Estes resultados garantem a funcionalidade do sistema de aquisição. Os tempos de

resposta dos diversos circuitos operando na rede devem ser adequados para a capacidade de transmissão da linha de comunicação de forma a assegurar que o mestre e os nós da rede se comuniquem corretamente. Também foi uma preocupação, o consumo e a distribuição eficiente de energia.

O protocolo de comunicação utilizado é constituído por camadas lógicas que empregam verificação de redundância e controle de fluxo *(handshakings)* entre unidades de rede e mestre. Basicamente, o mestre se comunica com a rede e indica qual a unidade alvo que deve transmitir os dados. Assim, a comunicação será do tipo *polling* mestre-escravo. Esta forma de comunicação é ilustrada na Figura 19.

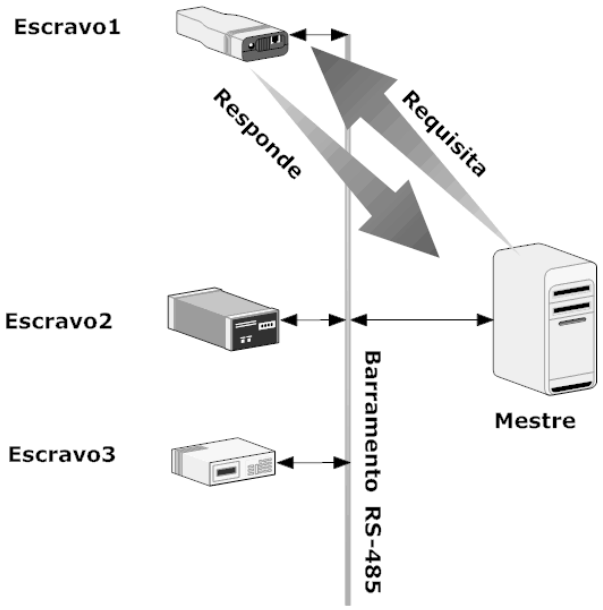

Figura 19: Comunicação tipo mestre escravo (*polling*)

Considerando que a linha de transmissão suporta uma taxa razoável de transmissão de sinais digitais, uma leitura completa dos dados de uma rede composta por 31 pontos é feita em cerca de um segundo, valor que pode ser considerado adequado para monitoramento de estruturas estáticas. Obviamente, o estudo técnico posterior deste sistema deverá indicar quais as próprias limitações com relação ao número de pontos (carregamento) e velocidade de operação (taxa de transmissão e tráfego). Inúmeras são as possibilidades, que inclusive poderiam ser selecionadas via software.

Um dos principais esforços na realização deste projeto foi a implementação de uma interface de comunicação confiável entre unidade mestre e unidades de rede, bem como a implementação de um protocolo de alto nível, baseado na adequação de protocolos existentes, de forma assegurar uma comunicação eficaz e imune a interferências externas.

Existem muitos protocolos eficientes para tal tarefa, dentre eles podemos citar: Ethernet, Fieldbus Foundation, Profibus, Devicenet, Canopen, Modbus, etc. Cada um desses foi originalmente desenvolvido por um determinado fabricante, que posteriormente abriu o protocolo em resposta à exigência do mercado por conectividade.

Visando escolher o protocolo mais adequado para ser implementado neste projeto, foram consultados (SNOWDON, 2002), onde os principais protocolos são comparados quando ao meio físico: CAN, RS-485, Ethernet, I2C 1-Wire. Também os protocolos de camada mais alta são comparados: DeviceNet, CANOpen e CanKingdom.

Também (DOEBELIN, 2004), apresenta em seu livro um capítulo que trata exclusivamente de redes de comunicação utilizadas para aquisição de sinais, considerando tanto as analógicas quanto as digitais. São apresentadas também as redes de campo básicas.

Outros trabalhos também contribuíram para o entendimento das características dos protocolos e análises comparativas. Dentre os trabalhos, podemos citar: (TRISTÃO, 2004), (BOARETTO, 2005) e (ROSSI,2005).

Depois de uma pesquisa das características dos principais protocolos existentes, optou-se pela utilização do protocolo MODBUS RTU, pelas seguintes razões:

- Determinístico;
- Permite monitorar até 247 pontos de rede;
- Padronizado;
- Protocolo aberto com fácil acesso à documentação;
- Não é necessário o pagamento de *Royalties*;
- Seguro e Confiável;
- Adequado para tratar os tempos de comunicação entre elementos envolvidos;
- Largamente difundido e utilizado, principalmente em ambientes industriais;
- Fácil implementação em microcontroladores de baixo custo.

Inclusive, poderia ter se desenvolvido um novo protocolo para tal operação. No entanto, a grande vantagem de se utilizar um protocolo padrão largamente difundido é a existência no

mercado de acessórios que permitem a sua integração e expansão. No caso do protocolo MODBUS, pode-se utilizá-lo juntamente com uma ampla gama de produtos disponíveis comercialmente, tais como: softwares supervisórios, *gateway* Ethernet TCP/IP e inclusive *wireless*, já que existem equipamentos que fazem a conversão de MODBUS RTU (RS485) para redes sem fio.

4.4 ESCOLHA DO MEIO DE TRANSMISSÃO

Como já mencionado no capítulo anterior, o protocolo MODBUS está na camada equivalente à camada de aplicação (camada 7) do modelo OSI/ISO e não especifica o meio de transmissão. Para o protocolo MODBUS dois principais meios poderiam ser utilizados: serial ou TCP/IP. Por facilidade de implementação e baixo custo do produto final - objetivos fundamentais do projeto - foi escolhida a comunicação serial.

Note-se que, trabalhos posteriores poderão incluir o meio de transmissão via TCP/IP mudando apenas a pilha da camada física do protocolo ora implementado.

O padrão RS485 é normalmente utilizado, por ser muito robusto em relação à imunidade contra interferências (por ser baseado em modulação diferencial de sinais), boa velocidade de transmissão (até 12 MB/s), bom alcance (1200 metros em 9600 BPS). Pelas características apresentadas de desempenho - mais do que suficientes para esta aplicação - o meio físico de transmissão escolhido foi o RS-485.

4.5 SERVIDOR DE COMUNICAÇÕES

Para uma melhor compreensão do princípio de funcionamento do servidor de comunicações utilizado neste projeto, nas seções que se seguem serão apresentados detalhes da tecnologia OPC, a qual é utilizada para troca de dados com o sistema de aquisição.

4.5.1 TECNOLOGIA OPC

Um dos grandes problemas de se interfacear equipamentos e sistemas no chão de fábrica reside em se compatibilizar os protocolos da camada de aplicação. Imagine a tarefa de interfacear um sistema supervisório com um CLP há alguns anos atrás. Vamos supor que este supervisório fosse o *Elipse E3* e que o CLP fosse uma CPU da família 5 da Rockwell. O *Elipse E3* era fornecido em várias versões. O CLP 5 pode se comunicar com diversas redes diferen-

tes, por exemplo, com uma rede com protocolo DH+ (proprietário da Rockwell). O PC pode utilizar cartões de comunicação Rockwell, Sutherland-Schultz ou outro. O número de combinações é muito grande. Na prática, teria que se utilizar ou desenvolver um *drive* que atendesse perfeitamente à combinação: Sistema SCADA (existem dezenas)/sistema operacional (várias opções), cartão de comunicação PC/CLP (várias fontes e possibilidade de rede). Isto significava a existência de centenas de *drives* de comunicação, que só atendiam a versões específicas da combinação de fatores apresentada acima. Um arranjo típico para esta solução é apresentado na Figura 20.

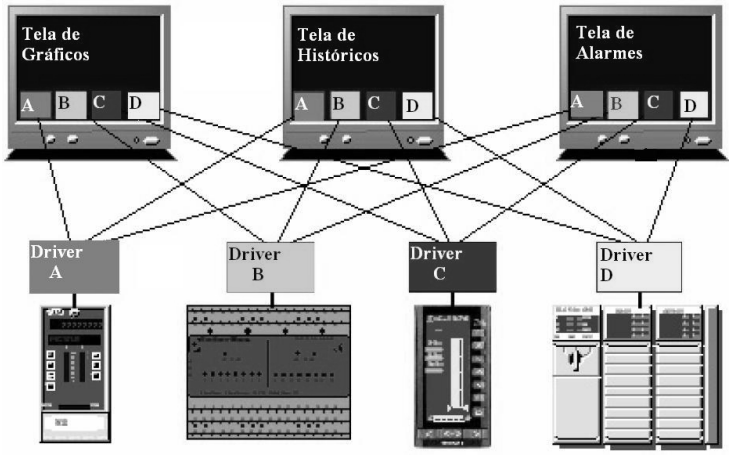

Figura 20: Arquitetura tradicional de comunicação utilizando *drivers*

No caso da arquitetura tradicional, servidores ou equipamentos têm interfaces/drivers diferentes para cada aplicação cliente. (Cada seta representa um software de *driver* ou interface). Embora o protocolo de um equipamento ou servidor não possa mudar, as arquiteturas das aplicações clientes (fornecida por diversos vendedores) são diferentes. Isto leva a um aumento de trabalho, custo e tempo. Se o protocolo do equipamento ou do servidor é alterado ou atualizado, então a aplicação cliente também necessita ser modificada.

Fica evidente a dificuldade encontrada na época pelos profissionais do setor de automação: conhecer e ter acesso a dezenas de produtos de fornecedores diferentes, todos incompatíveis entre si.

Pela necessidade urgente de padronização, foi criado o padrão OPC (*OLE for Process Control*) que é baseado no modelo de comunicação entre aplicativos criado pela Microsoft e denominado OLE (*Object Linking and Embedding*), que é uma maneira eficiente de se estabelecer interfaces para aplicações em substituição às chamadas de procedimento e as DLL usadas inicialmente para encapsular uma aplicação. O projeto foi inicialmente liderado pela Microsoft e, posteriormente, foi criada uma organização independente chamada de *OPC Foundation*, responsável pelo desenvolvimento de uma plataforma específica para componentes de automação

de processos. Na página (da internet) da fundação podem ser encontradas as especificações do protocolo OPC. Esta tecnologia é hoje o padrão de fato da indústria. O OPC elimina a situação apontada anteriormente, pois um fabricante de CLP sempre fornecerá com o seu equipamento um servidor OPC. O fabricante de SCADA também fornecerá o cliente OPC. O mesmo acontece com um fornecedor de inversores, de relés inteligentes ou de qualquer outro dispositivo de aquisição de dados ou atuador. A arquitetura de uma plataforma utilizando tecnologia OPC é mostrada na Figura 21.

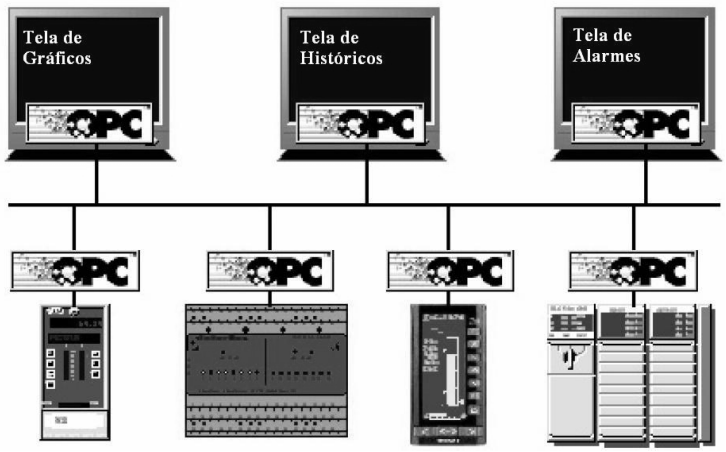

Figura 21: Comunicação utilizando tecnologia OPC

O OPC otimiza a interface entre aplicações cliente e servidor fornecendo um mecanismo padrão para comunicar dados de uma fonte de dados para qualquer aplicação cliente. Em outras palavras, o OPC é a ferramenta *Plug and Play* para IHM e Automação. Para tanto, faz uso de um protocolo universalmente aceito e conhecido para envio de dados entre as fontes de dados (Servidor) e as aplicações (Clientes), eliminando o uso de *drivers* e conversores, geralmente proprietários, que criavam uma dependência entre o usuário e o fornecedor daquele sistema proprietário muitas vezes custosa e ineficiente.

As aplicações precisam apenas saber como buscar dados de um servidor OPC, ignorando a implementação do dispositivo e o servidor precisa fornecer dados em um formato único: servidor OPC. A estrutura organizacional de um servidor OPC típico pode ser observada nas Figuras 22 e 23.

O servidor OPC fornece dados de tempo real proveniente de sensores (temperatura, pressão, etc.), comandos de controle (abrir, fechar, ligar, desligar, etc.), status de comunicação, dados de performance e estatística do sistema, etc.

Cada transação pode ter de 1 a milhares de itens de dados, o que torna o protocolo muito eficiente. O OPC não resolve o problema de nomes globais. Deve-se saber exatamente em que servidor uma dada variável pode ser encontrada.

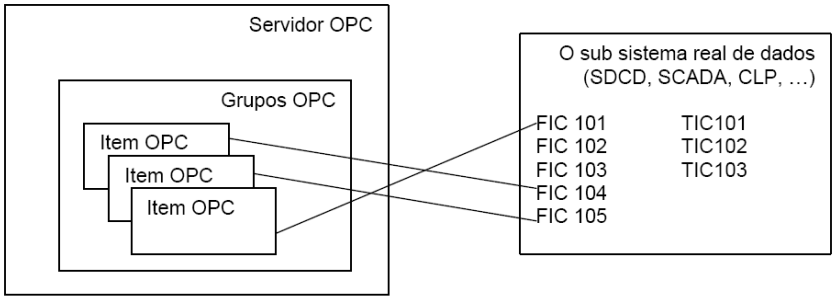

Figura 22: Mapeamento de Itens OPC

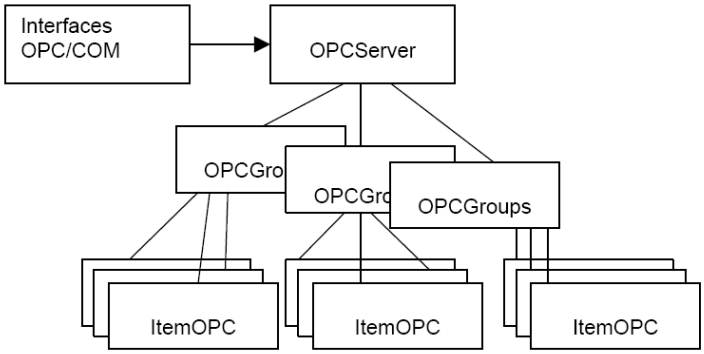

Figura 23: Estrutura organizacional de um servidor OPC contendo uma hierarquia de grupos e itens

Entre suas funções principais ele permite à aplicação cliente:

- Gerenciar grupos: Criar, clonar e deletar grupos de itens, renomear, ativar, desativar grupos.
- Incluir e remover itens em um grupo.
- Navegar pelas tags existentes (*browser interface*).
- Ver os atributos ou campos associado a cada tag.
- Definir a linguagem de comunicação (país) a ser usada.
- Associar mensagens significativas a códigos de erro
- Obter o status de funcionamento do servidor
- Ser avisada, caso o servidor saia do ar.

O grupo de dados constitui uma maneira conveniente da aplicação organizar os dados de que necessita. O grupo pode ser ativado ou desativado como um todo. Cada tela sinótica, receita,

relatório e outros, pode usar um ou mais grupos. A interface de grupo permite à aplicação cliente:

- Adicionar e remover itens dos grupos.
- Definir a taxa de leitura do dado no grupo.
- Ler e escrever valores para um ou mais itens do grupo.
- Assinar dados do grupo por exceção.

Cada item é um objeto OPC que proporciona uma conexão com uma entrada física de dados. Cada item fornece ao cliente informação de: valor, *time stamp*, qualidade do dado e tipo de dado.

As leituras de dados podem ser de três tipos: leitura cíclica onde uma unidade é lida periodicamente (*polling*), leitura assíncrona (o cliente é avisado quando a leitura se completa) e por exceção (assinatura). As duas primeiras trabalham sobre listas (subconjuntos) de um grupo e o serviço de assinatura envia aos clientes qualquer item no grupo que mudar de valor. Cada grupo de dados pode ter uma taxa de leitura específica.

4.5.2 MESTRE MODBUS E SERVIDOR OPC

Resumindo: a comunicação entre programas aplicativos de diferentes fabricantes ficou muito facilitada com o surgimento do padrão OPC e portanto, ao se utilizar um produto que segue este padrão não é mais necessário conhecer como os dados são armazenados internamente nos produtos de cada fabricante, só é necessário saber como acessá-los.

Atualmente existem dezenas de fabricantes de servidores OPC, todos com qualidades equiparáveis, já que devem ser certificados pelo órgão responsável pela tecnologia (*OPC Foundation*). Assim, o critério de escolha recai na facilidade de acesso ao produto e custo de aquisição. Para este projeto foi escolhido o servidor OPC KEPServerEx, fabricado pela KEPWARE, que funciona como uma unidade mestre MODBUS, responsável por controlar a transferência das informações do sistema supervisório para os terminais da rede e vice-versa.

A idéia é utilizar um servidor de comunicações Modbus para comunicar-se com os elementos da rede de sensores e que este servidor de comunicações também seja um servidor OPC, possibilitando assim, trocar dados com outros aplicativos clientes OPC. Desta maneira, o tratamento dos dados pode ser feito, por exemplo, utilizando-se um dos produtos do pacote *Office*® da Microsoft, tais como: Excel, Word ou Access.

Também na área de pesquisa das engenharias é muito comum a utilização de softwares de simulação e captura de dados, tais como o Matlab/Simulink®, LabView® e DasyLab®. Foi pesquisado a possibilidade de se desenvolver aplicações clientes OPC com estes para coletar e tratar dos dados recebidos. Concluiu-se que todos estes podem ser utilizados, já que clientes OPC podem ser construídos a partir deles.

4.6 SOFTWARE DE AQUISIÇÃO E TRATAMENTO DOS DADOS

Na área da engenharia de controle e automação industrial é comum a utilização de sistemas supervisórios para realizar a tarefa de aquisição e tratamento de dados. Este tipo de software possui capacidade gráfica para visualização em tempo real dos dados capturados a partir dos elementos sensores. Quase todos os sistemas supervisórios modernos dispõem de recursos para se comunicarem com servidores OPC. Portanto, qualquer um poderia ser utilizado, dependendo apenas da disponibilidade.

Por facilidade de acesso do autor ao produto, optou-se neste trabalho pela utilização do software supervisório Elipse Scada, que funcionará como um cliente OPC. A figura 24 ilustra a arquitetura do sistema proposto.

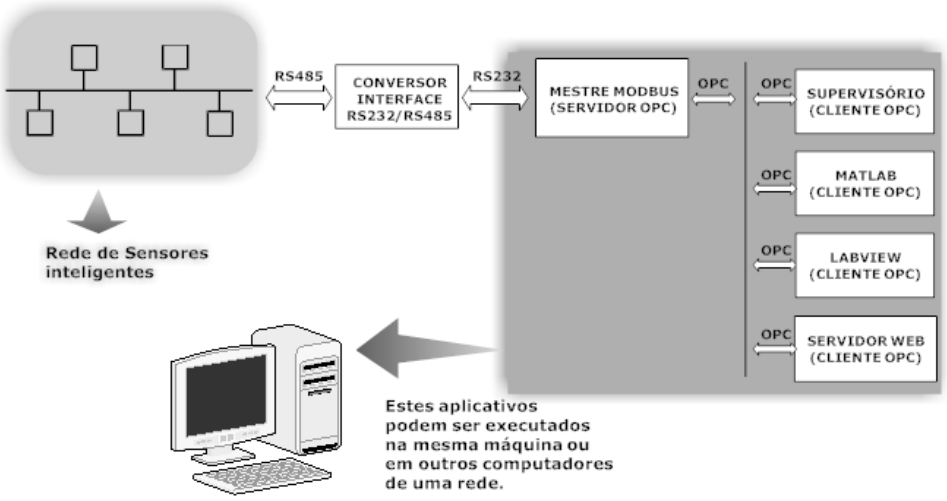

Figura 24: Arquitetura sistema de aquisição de dados

O princípio de funcionamento para a leitura dos nós da rede é o seguinte: O servidor Modbus solicita periodicamente os valores de temperatura e de fator de deformação das unidades escravas conectadas na rede (*polling*) e armazena estas informações no banco de dados do servidor OPC. Quando o supervisório necessitar de qualquer informação de uma unidade escrava,

por exemplo, a temperatura do terminal 17, o sistema supervisório solicita esta informação para o servidor OPC que vai devolver o valor da última leitura efetuada.

Quando o supervisório desejar enviar uma determinada informação para uma unidade escrava, vai enviá-la para o servidor OPC que se encarregará de enviar a solicitação ao mestre Modbus. Este por sua vez, enviará a informação para a unidade escrava. Notar que a codificação e decodificação do bilhete no formato Modbus é feita automaticamente pelo servidor de comunicações. Assim, o supervisório só precisa saber o nome da variável que deseja acessar e não detalhes de como a informação é obtida.

4.7 PLANEJAMENTO PARA FUTURA INTEGRAÇÃO

Visando uma redução do tamanho do nó da rede e redução dos custos para a produção de um sistema de monitoramento estrutural como o proposto, pretende-se, em um momento posterior, fazer que os nós de rede sejam implentados em um único circuito integrado. Assim, o *front-end* analógico será objeto de investigação para futura integração e assim contribuir para a redução do tamanho físico do nó da rede. Um dispositivo *dummy strain-gauge* será utilizado *in loco* para fins de calibração e compensação térmica. Circuitos compensados termicamente baseados em referências de tensão do tipo *band-gap* (ALLEN et al., 1990), realizáveis em tecnologia CMOS convencional, serão utilizados para assegurar a estabilidade da operação do sistema. Amplificadores diferenciais de ganho programável com alta taxa de rejeição de modo comum (CMRR) e a subseqüente filtragem para restrição de banda e incremento da resolução compõem a etapa final da interface analógica. Esta deverá ser integrada através do programa MOSIS (MOSIS, s.d.) de fabricação de circuitos integrados, o qual esta universidade dispõe de acesso (UEL-04379, s.d.). Após a captura e condicionamento dos sinais analógicos, os mesmos são digitalizados, novamente condicionados para adequação dos níveis de tensão e transmitidos através da linha. Um microcontrolador sincronizado com a unidade mestre comanda a seqüência de envio dos dados capturados, bem como recebe ordens da unidade remota, como por exemplo, para realizar a calibração do sistema. Assim, assegura-se uma maior confiabilidade na leitura dos dados bem como a integridade da própria transmissão da informação, pois a aquisição e condicionamento do sinal são realizados localmente e a informação digitalizada antes de ser transmitida.

A integração total dos dispositivos constituintes do nó de rede será considerada, para fins de estudos de viabilidade técnica e comercial do produto. Neste caso, a digitalização dos dados deverá ser realizada por um conversor AD a ser projetado e as funções do microcontrolador realizadas através de máquinas de estado síncronas simplificadas. Obviamente, esta etapa final

de integração demandará estudos adicionais não previstos para o período vigente deste projeto, o que sugere uma continuidade para o mesmo.

4.8 CAPACIDADE PARA EXPANSÃO FUTURA

Posteriormente, partindo-se do suporte de hardware disponível e do protocolo desenvolvido, outras variáveis poderão ser igualmente monitoradas, tais como umidade e corrosão, bastando-se adequar o sensor à interface de rede. Outros dispositivos além do extensômetro elétrico de resistência, como sensores de vibração ou deslocamento também poderão ser igualmente conectados permitindo o monitoramento de outras variáveis, desde que a taxa de aquisição não exceda a capacidade de transmissão do protocolo.

5 *PROJETO, MONTAGEM E TESTES DO TERMINAL DE AQUISIÇÃO*

Neste capítulo são detalhadas as estruturas de *hardware* e *software* desenvolvidas para implementar a formulação proposta no Capítulo INTRODUÇÃO. São mostrados os aspectos construtivos da solução proposta bem como suas características técnicas.

5.1 VISÃO GERAL DO TERMINAL DA REDE

Na Figura 25 é ilustrado o modelo do terminal de rede implementado. Sua estrutura é composta por um circuito de condicionamento de sinais, um sensor de temperatura, um conversor analógico para digital, um microcontrolador e uma interface de comunicação. Também faz parte de cada unidade, um sistema conversor de energia de Corrente Contínua (CC).

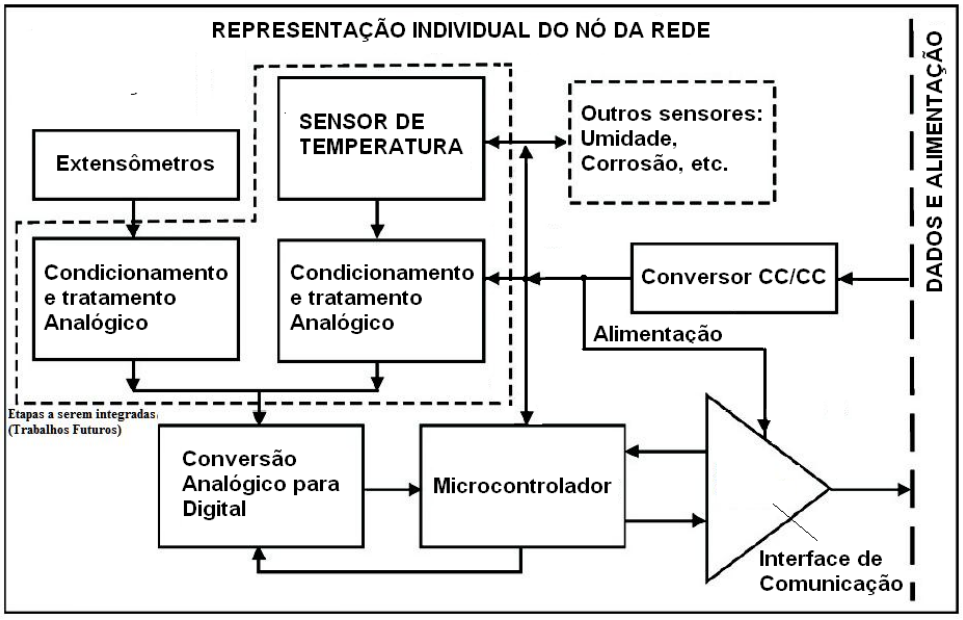

Figura 25: Diagrama em blocos do terminal de rede desenvolvido

5.2 CARACTERÍSTICAS DESEJADAS

5.2.1 ALIMENTAÇÃO

Todos os nós devem ser alimentados pelo mesmo cabo que faz a conexão de rede. Assim, o sistema recebe uma tensão de 12V e deve convertê-la em 5V que serão utilizados pelo microcontrolador e pelos elementos de condicionamento de sinais.

5.2.2 COMUNICAÇÃO

Conforme já discutido anteriormente no Capítulo Projeto Conceitual, a comunicação utilizará o barramento RS-485 multi-ponto padronizado. Sua topologia de conexão será do tipo barramento comum.

5.2.3 MEMÓRIA

Devem existir elementos suficientes de armazenamento para as variáveis temporárias ou de execução, para implementação do filtro de média móvel e memória EEPROM para armazenamento dos valores de calibração do sistema.

5.3 IMPLEMENTAÇÃO

5.3.1 ALIMENTAÇÃO

Para a alimentação do sistema, foi utilizada uma fonte de alimentação de microcomputador. Sua escolha se deu pelos seguintes motivos:

- Facilidade de Aquisição;
- Baixo custo;
- Boa capacidade de fornecimento de energia.

5.4 MICROCONTROLADOR

Todo o funcionamento do sistema é comandado e supervisionado por um microcontrolador. Ele é responsável pela implementação das rotinas do protocolo de comunicação, controle dos

circuitos de transmissão e de recepção de sinais, digitalização (conversão analógico/digital), detecção de modo unívoco do endereço do terminal (programado através de micro-chaves), armazenamento dos dados de calibração em memória permanente do tipo EEPROM.

Entre diversas opções do mercado foi escolhido o microcontrolador PIC16F876 fabricado pela MICROCHIP. Suas principais características são:

- Desempenho de até 5 MIPS quando utilizados com cristal de freqüência de 20 MHz. Nos microcontroladores PIC o sinal do clock é internamente dividido por quatro. Assim, ao se utilizar um cristal de 4 MHz, a freqüência do barramento interno será de 1 MHz, ou seja, cada ciclo de máquina dura $1\mu s$. Os microcontroladores PIC executam uma instrução a cada $1\mu s$, exceto para as instruções de desvio que consomem dois ciclos de máquina. Portanto, utilizando-se um cristal oscilador externo de 4 MHz obtém-se um desempenho próximo de 1 MIPS, o que é adequado para esta aplicação.
- 8 kWords de memória de programa
- 368 bytes de memória de dados tipo RAM (memória volátil).
- 192 bytes de memória de dados tipo EEPROM (memória não volátil)
- Portas de entrada e saída - um total de 18 I/Os (entradas e saídas) separadas em 4 grupos denominados de portas, codificadas de A,B,C e D.
- 1 temporizador/contador de 16 bits e 2 de 8 bits.
- 4 canais de conversores analógico-digitais de 10 bits
- 1 USART - ou Receptor/Transmissor Assíncrono Universal. Este elemento constitui o coração da comunicação serial a ser implementada. Internamente é implementado em hardware que é utilizado tanto na comunicação da transmissão quanto na recepção. Sua principal função é converter bytes em um fluxo de bits correspondente e vice-versa, acrescentando alguns pulsos de controle para permitir a detecção de início e término de transmissão (modo assíncrono) e também de erros, quando utilizando detecção de paridade.

5.5 QUESTÕES FUNDAMENTAIS SOBRE CIRCUITOS EM PONTE

Existem várias considerações a serem feitas quando se está lidando com pontes de baixo sinal. As mais importantes questões serão discutidas a seguir.

5.5.1 TENSÃO DE EXCITAÇÃO

Os extensômetros elétricos requerem uma tensão de excitação para gerar uma tensão que represente o esforço medido. Pode-se utilizar o método de tensão constante, corrente constante ou, ainda, por alimentação alternada.

Enquanto pouco utilizada atualmente, por muitos anos a excitação CA de pontes resistivas foi um meio efetivo de se remover desvios de corrente contínua devido aos componentes eletrônicos (MAXIM-DALLAS, 2004). Se o sinal fornecido à ponte é uma tensão alternada, a tensão de saída da ponte também será um sinal alternado. Este sinal pode ser acoplado capacitivamente, amplificado e ter seu nível deslocado e a amplitude do sinal CA final obtido será independente de qualquer tensão contínua de desvio devido aos componentes eletrônicos. A amplitude dos sinais alternados podem então ser medidos utilizando-se técnicas específicas para medição de tensões alternadas. No entanto, são extremamente sensíveis a capacitâncias parasitas, o que torna esse método inadequado quando os extensômetros estiverem instalados longes do sistema condicionador de sinal, uma vez que as capacitâncias provocadas pela fiação são significativas. Outra questão que desfavorece seu uso é a necessidade de manter o sinal de excitação CA extremamente estável em amplitude, sendo que a exatidão das medidas está diretamente relacionada com esta estabilidade. Para se conseguir isto, normalmente são necessários circuitos adicionais que tornam mais complexo o circuito condicionador. Estes circuitos são responsáveis por manter constante a amplitude do sinal mesmo em variações significativas de temperatura (o que não é trivial) e também com o passar do tempo. No entanto, alguns de seus conceitos que a tornam atrativas, são aproveitados neste trabalho (de forma adaptada).

Para o presente projeto foi definida a utilização de tensão constante para a excitação da ponte. Lembrando que as deformações nos extensômetros são detectadas como uma variação na tensão de saída do circuito em ponte, percebe-se que a sensibilidade é diretamente proporcional à tensão de excitação. Pode parecer à primeira vista que se aplicarmos uma tensão grande o suficiente na ponte, poderia-se aumentar a sensibilidade das medidas. No entanto, a potência limite para estes elementos está em torno de 25 mW. Devido à essa limitação são utilizadas baixas tensões para alimentação da ponte (entre 5 V e 10 V).

Para este projeto a alimentação da ponte de Wheatstone foi fixada em 5V. Esse valor foi escolhido por atender ao compromisso entre uma aceitável relação sinal-ruído e uma corrente máxima limitada, que não cause excesso de dissipação de potência no extensômetro, o que poderia causar o efeito indesejável de aquecê-lo.

Como conseqüência disto, o sinal de saída é muito tênue e é necessário a utilização amplificadores para aumentar a amplitude deste, antes de convertê-lo para o formato digital.

5.5.2 TENSÃO DE REFERÊNCIA

Uma questão com a ponte é que a saída é um sinal diferencial com um sinal em modo comum igual à metade da tensão de alimentação. Freqüentemente o nível deste sinal diferencial deve ser deslocado e convertido para um sinal referenciado ao terra antes de ir para o conversor AD.

A Equação 3.8 mostra que a saída da ponte é diretamente proporcional à tensão de alimentação. Desta maneira, o circuito deve manter constante a tensão na mesma proporção da exatidão desejada, ou deve compensar as variações na tensão de alimentação. A maneira mais simples de compensar as alterações na tensão de alimentação é derivar a referência do conversor AD diretamente dos ramos de excitação da ponte. Na Figura 26 a tensão de referência do conversor AD vem do divisor de tensão colocado em paralelo com a ponte. Isto faz com que as variações na tensão de alimentação sejam rejeitadas porque a tensão de referência do conversor AD vai se alterar junto com a sensibilidade da ponte.

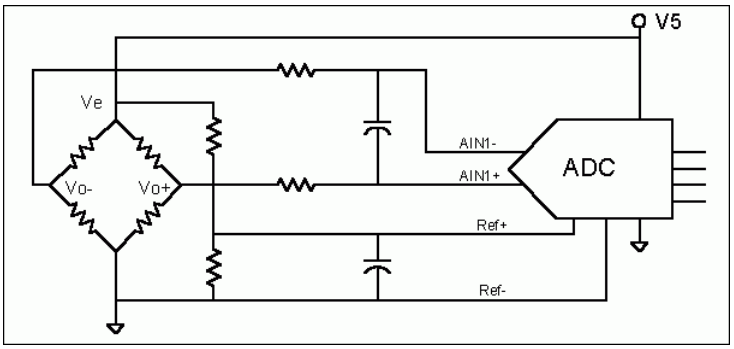

Figura 26: A tensão de referência é proporcional a Ve, o que elimina os erros devido a variações na alimentação (dessensibilização da tensão da alimentação).

Uma abordagem alternativa poderia utilizar um canal adicional no conversor AD para medir a tensão de excitação da ponte. Neste caso, as alterações de tensão da ponte seriam compensadas por software.

A Equação 5.1 mostra a tensão de saída corrigida (Voc), como uma função da tensão de saída medida (Vom), a tensão de excitação medida (Vem), e a tensão de excitação quando foi feita a calibração (Veo).

$$Voc = Vom\frac{Veo}{Vem} \tag{5.1}$$

5.6 CIRCUITO CONDICIONADOR DE SINAL

Na prática, a medida de deformações raramente envolvem quantidades maiores do que alguns mili-strains (1×10^{-3}). Por exemplo, suponha um corpo de prova submetido a um esforço que gera uma deformação de 500 $\mu\varepsilon$. Utilizando-se um extensômetro de fator de sensibilidade igual a 2 vai exibir uma alteração na resistência elétrica de apenas $2 \times 500 \times 10^{-6} = 1 \times 10^{-3} =$ 0,1%. Para um extensômetro de 120 Ω, isto significa uma alteração de apenas 0,12 Ω.

Assim, o sistema condicionador de sinal deve ser capaz de medir variações muito pequenas nas resistências. Desta forma, a seleção adequada e uso da ponte, condicionamento do sinal, fiação e componentes da aquisição de dados devem ser considerados para medidas confiáveis. Vários requisitos de condicionamento de sinais devem ser considerados, conforme descrito a seguir.

5.6.1 AMPLIFICAÇÃO

Os extensômetros elétricos fornecem tipicamente baixos níveis de sinais, da ordem de microvolts. Desta forma, é importante que o sinal seja primeiramente amplificado, antes de ser digitalizado.

O circuito mostrado na Figura 27, funciona como um amplificador de instrumentação.

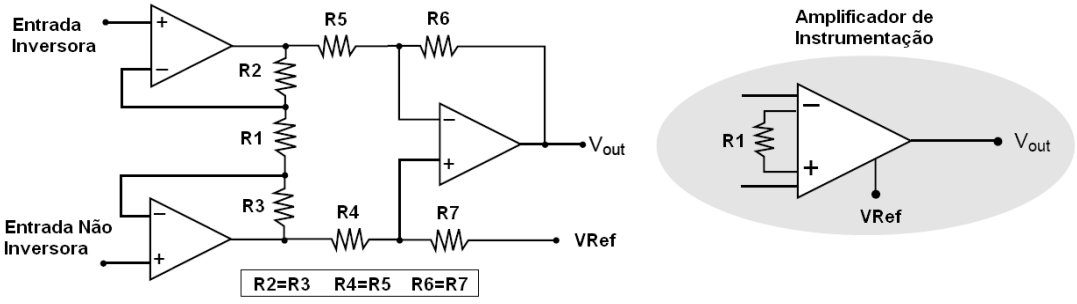

Figura 27: Amplificador de Instrumentação utilizando 3 amplificadores operacionais

A finalidade desta configuração é obter uma alta impedância de entrada (idealmente infinita) e uma boa taxa de rejeição em modo comum (*Commom Mode Rejection Ratio - CMRR*). Deve-se salientar ainda que a melhor taxa de rejeição é obtida quando o valor de R2 for igual ao de R3 (FRADEN, 2004).

Para amplificar o sinal proveniente da ponte foi utilizado um amplificador operacional de baixo custo (LM324) na configuração de amplificador de instrumentação. Esta configuração é formada pelos amplificadores operacionais IC4A a IC4C e o ganho desta primeira etapa é

controlado pelo resistor R1.

Observar que o amplificador operacional da etapa de saída funciona como um subtrator utilizando o amplificador operacional IC4D, ou seja, faz a diferença entre os sinais amplificados na etapa anterior. Circuitos subtratores permitem que se obtenha na saída uma tensão igual à diferença entre os sinais aplicados, multiplicada por um ganho (SEDRA e SMITH, 2004). A saída de IC4D é conectada diretamente à porta analógica do microcontrolador.

De acordo com Webster (1999), se R_2=R_3; R_4=R_5 e R_6=R_7, o ganho de tensão do amplificador pode ser calculado por :

$$A_v = \left(2\frac{R_2}{R_1} + 1\right)\left(\frac{R_6}{R_5}\right) \tag{5.2}$$

onde A_v é o ganho de tensão do estágio.

Reportando se à Figura 28 e considerando a Equação 5.2, tem-se que o ganho do estágio de amplificação do circuito desenvolvido é dado por:

$$A_v = \left(2\frac{100k}{1k} + 1\right)\left(\frac{100k}{100k}\right) = 201$$

Este valor foi escolhido devido à sensibilidade típica das células de carga que é de 2 mV/V. Como o circuito é alimentado por 5 Volts, isto significa que o valor máximo de variação do sinal coletado na ponte será de 10 mV. Quando isso ocorrer, o valor de fundo de escala será de 2,01 Volts. Como o valor da tensão de referência é fixada em 2,1 V, teremos como limite superior de 4,12 Volts e inferior de 0,99 Volts - valores satisfatórios para esta aplicação.

5.7 LIGAÇÃO DOS EXTENSÔMETROS

Para a descrição das ligações dos extensômetros nas diversas configurações será utilizado como referência o diagrama elétrico (parcial) mostrado na Figura 28. Observa-se que para que seja possível as três configurações referidas anteriormente (¼, ½ e ponte completa) serão utilizados 3 resistores internos para completar os outros "braços" da ponte de Wheatstone.

Para a utilização da configuração em ¼ de ponte deve ser fornecido um resistor adicional para simular o outro extensômetro adjacente. Para tanto é utilizado um resistor de 121 Ω (R18) em paralelo com um outro de 12 $k\Omega$ (resistência equivalente de 119,79 Ω). O valor ótimo destes foi determinado experimentalmente. Um dos lados dessa associação de resistores é ligada ao terra e o outro no pino 2 do conector de parafusos (veja as Figuras 29 e 28).

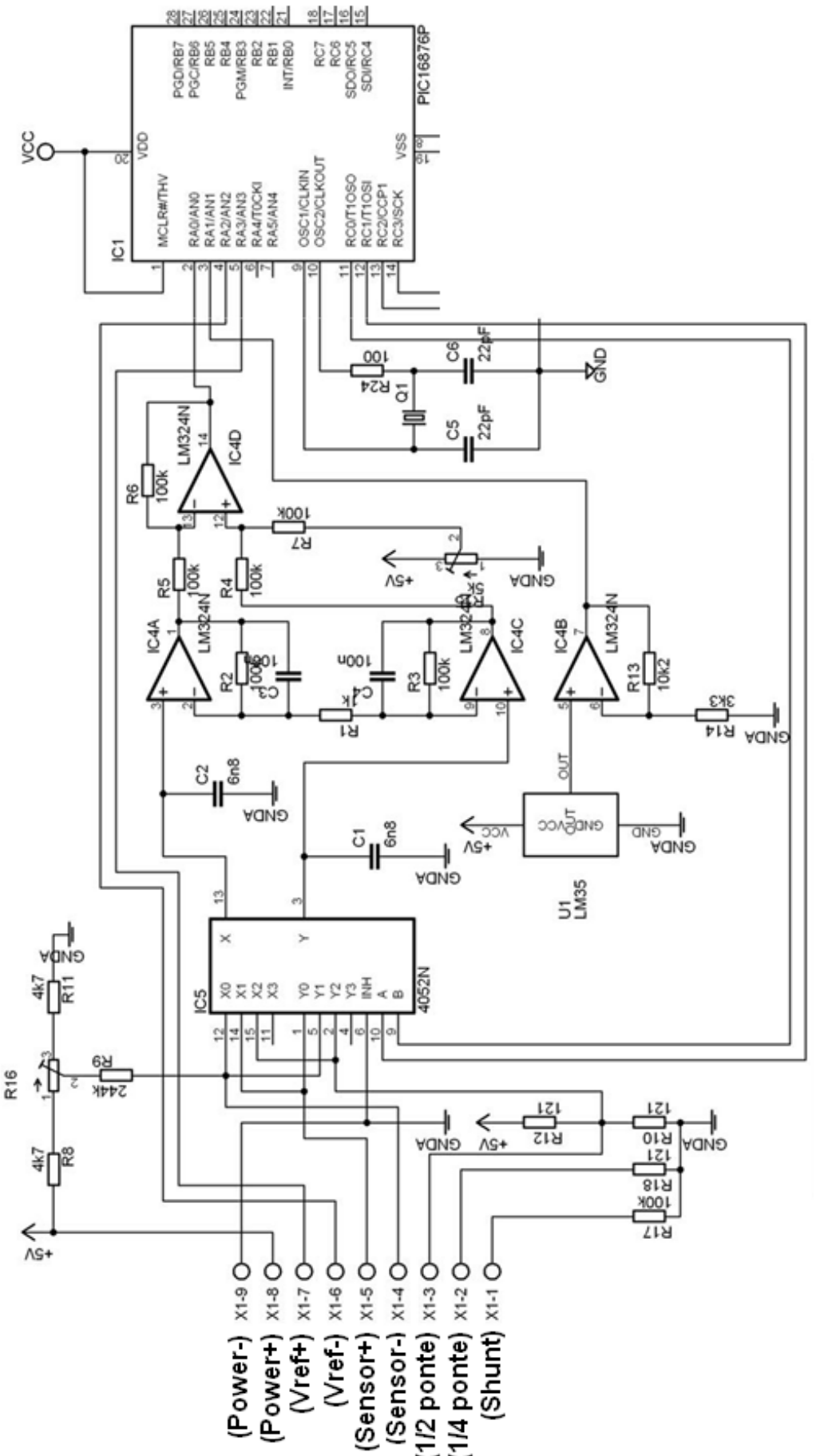

Figura 28: Diagrama elétrico (parcial) do circuito condicionador de sinal

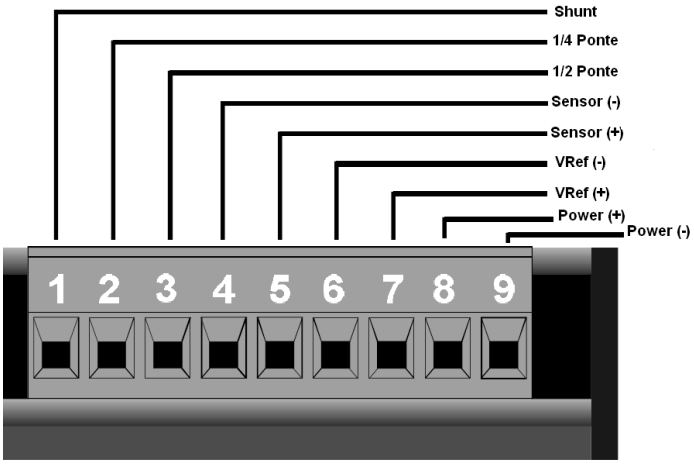

Figura 29: Vista frontal esquemática do conector de parafusos utilizado para as conexões dos extensômetros

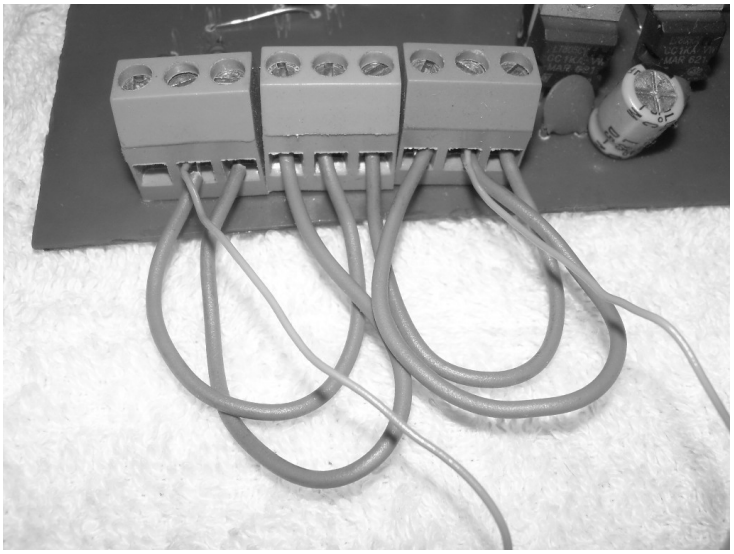

Figura 30: Vista da implementação do conector de parafusos utilizado para as conexões dos extensômetros

Para fornecer os outros dois 'braços' da ponte são utilizados os resistores (R10 e R12) de $1k\Omega$ (filme metálico de 0,1% tolerância). Estes devem ser utilizados para completar a ponte quando for utilizado somente um extensômetro (¼ de ponte) ou dois extensômetros (½ ponte).

Para a descrição das ligações físicas dos extensômetros nas diversas configurações será utilizado como referência o diagrama mostrado na Figura 29. Na Figura 30 pode-se observar o mesmo conector na placa já implementada.

5.7.1 LIGAÇÃO EM 1/4 DE PONTE

Esta configuração pode ser observada na Figura 31.

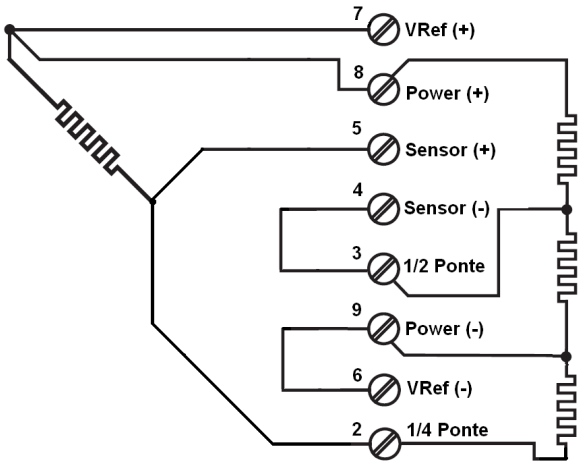

Figura 31: Diagrama de ligação de 1/4 de ponte.

5.7.2 LIGAÇÃO EM 1/2 DE PONTE

Esta configuração pode ser observada na Figura 32.

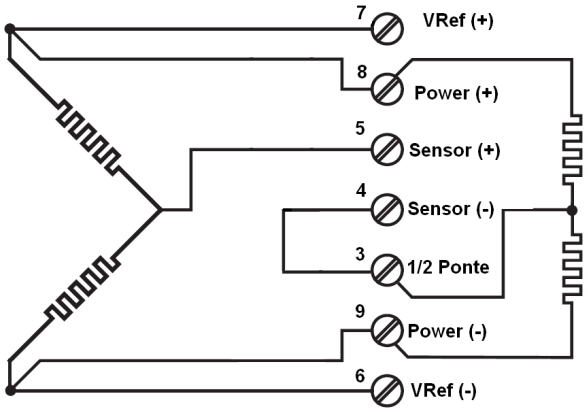

Figura 32: Diagrama para ligação dos extensômetros em 1/2 ponte

5.7.3 LIGAÇÃO EM PONTE COMPLETA

Esta configuração pode ser observada na Figura 33.

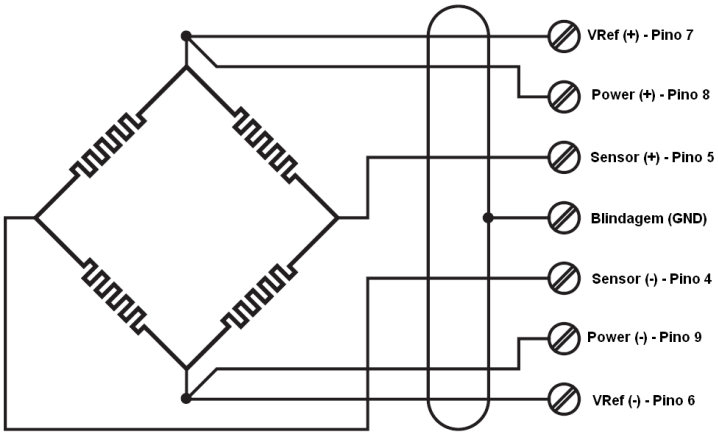

Figura 33: Diagrama para ligação dos extensômetros em ponte completa

Exemplo de célula de carga

Um exemplo comum de uma ponte resistiva é uma célula de carga com quatro elementos ativos. Quatro extensômetros elétricos são organizados em uma configuração em ponte e fixados a uma estrutura rígida que se deforma levemente quando uma força é aplicada. A sensibilidade para uma célula de carga típica é de $2mV/V$. A partir da equação 3.9 podemos ver que isto é equivalente a uma alteração da resistência em toda faixa de atuação de apenas 0,2% (ou 2000 ppm). Demonstração:

$\frac{Vo}{Ve} = \frac{\Delta R}{R}$

$\frac{\Delta R}{R}$= $2mV/V = 0,002 = 0,2\% = 2000$ ppm.

Se a saída da célula deve ser medida com uma resolução de 12 bits, então alterações na resistência de 0,5 ppm devem ser precisamente medidas ($2\times10^{-3}/2^{12}$= $4,88\times10^{-7} \cong 0,5$ppm).

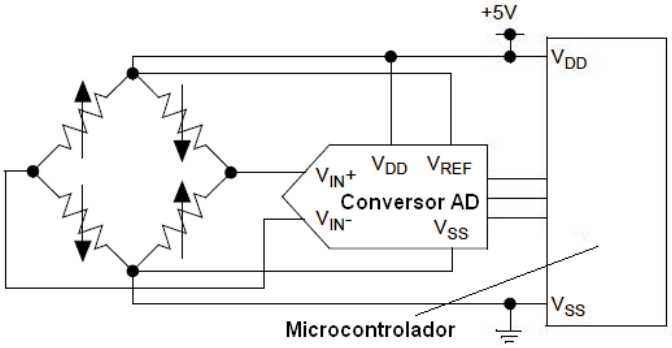

Figura 34: Medição da tensão de saída de uma ponte ligada diretamente a um conversor AD (sem prévia amplificação do sinal).

Medir estas alterações utilizando uma ponte (alimentada por 5V) ligada diretamente a um

conversor AD (conforme mostrado na Figura 34), sem amplificação prévia do sinal, requer que este seja de pelo menos 21 bits de resolução. Demonstração:

Se a tensão da ponte é de 5 V e a sensibilidade da célula de carga é de $2mV/V$, então a faixa de variação da tensão de saída é de $10mV$ ($5V \times 2mV/V$).

Se a resolução desejada é de 12 bits, então a tensão mínima a ser detectada é de:

$$\frac{10mV}{2^{12}} = 2,44\mu V$$

Então, ao se utilizar um conversor AD alimentado com 5 Volts este deverá ter esta resolução. Assim,

$$\frac{5V}{x} = 2,44\mu V$$

onde x é a faixa de valores do conversor. Então esta deve ser:

$$x \geq \frac{5}{2,44 \times 10^{-6}}$$

$$x \geq 2.048.000$$

O próximo passo é determinar a resolução em termos de bits. Lembrando que x é uma potência de um número binário. Então

$$2^b \geq 2.048.000$$

onde b é o número de bits.

$$log(2^b) \geq log(2.048.000)$$

$$b \geq \frac{log(2.048.000)}{log(2)}$$

$$b \geq 20,96$$

Esta situação é ilustrada na Figura 35.

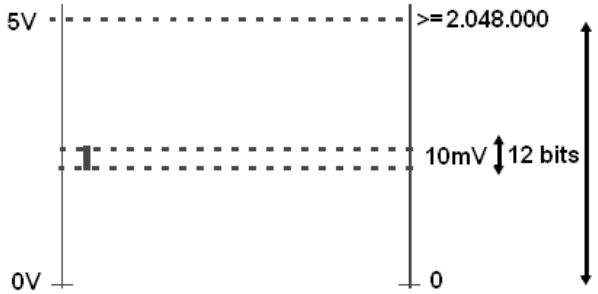

Figura 35: Obtenção de uma resolução de 12 bits a partir de um conversor AD de 21 bits.

Além da necessidade de um conversor AD de alta resolução, a tensão de referência do conversor AD precisa ser ultra estável. Ela não pode-se alterar mais do que 0,5 ppm em toda faixa da temperatura de operação.

Os resistores da célula de carga variam não apenas com a tensão mecânica aplicada à estrutura. A expansão térmica da estrutura em que ela está acoplada e o coeficiente de variação térmica do extensômetro vão causar alterações na resistências. Estas alterações indesejáveis podem ser tão grandes, até mesmo maiores, do que as alterações provocadas pelo esforço mecânico. No entanto, se as alterações indesejadas ocorrerem igualmente em todos os ramos da ponte de resistores seus efeitos podem ser desprezados porque as alterações cancelam-se mutuamente. O valor de R pode dobrar que a tensão de saída permanecerá a mesma, desde que ΔR também tenha dobrado seu valor.

O exemplo acima mostra como uma ponte pode tratar facilmente a medida de pequenas variações em resistência. As seções seguintes cobrem os principais questões quando utilizando circuitos em ponte.

5.8 CALIBRAÇÃO

Para a exatidão dos valores é necessária a calibração dos nós da rede. Alguns destes procedimentos são feitos de forma manual enquanto outros são feitos automaticamente via *software*. A seguir são detalhados os principais procedimentos para calibração.

5.8.1 AJUSTE DE ZERO DO SISTEMA

Dependendo do braço da ponte que foi desbalanceado, vai existir uma tensão resultante que pode assumir valores positivos ou negativos. Por sua vez, o microcontrolador não atua com sinais negativos, razão pela qual se faz necessária a instalação de um trimpot R15, que

promove na saída do IC4D um deslocamento no nível de tensão do sinal (*offset*), de modo a atender os requisitos de tensão do microcontrolador. Este *trimpot* é ajustado para se obter uma tensão de 2,100 V na saída do amplificador IC4D. Para se realizar este procedimento, deve ser comutada uma micro-chave que vai habilitar a execução de uma rotina de *software*, cuja finalidade é comandar a chave analógica para curto-circuitar a entrada do amplificador de instrumentação e armazenar em memória o valor ajustado e também a temperatura em que o sistema foi calibrado. Estes valores serão utilizados posteriormente para realizar a auto-calibração do sistema, removendo derivas devido à temperatura ou com o passar do tempo.

5.8.2 AJUSTE DE ZERO DA PONTE

Os sistema deve ter uma posição de repouso (posição de referência). Esta será tomada quando os extensômetros estiverem em condição normal (sem aplicação de carga). Neste caso um potenciômetro deverá ser ajustado até que não haja sinal de saída da ponte, ou seja, deverá ser medido 2,100 Volts nesta posição. Para este ajuste, foi inserido no circuito o trimpot (R16) que funciona adicionando ou removendo resistência em um dos braços da ponte até obter a condição balanceada.

Na verdade, o valor nulo citado no parágrafo anterior é relativo ao valor de referência de 2,100 Volts, ou seja, deve-se ajustar o *trimpot* até que se obtenha este valor. A Figura 36 mostra a posição do *trimpot* R16.

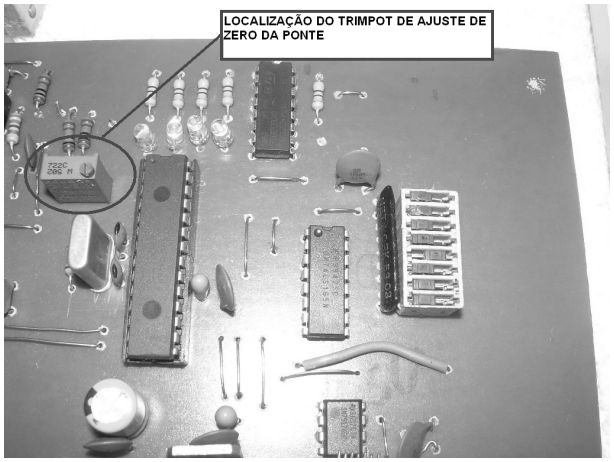

Figura 36: Localização do trimpot de ajuste de zero da ponte

5.8.3 CANCELAMENTO DE DESVIOS PROVOCADOS PELA INSTRUMENTAÇÃO

De acordo com Kester (2003), alguma fonte de erro de *offset* é inevitável em um sistema. Desvios (*offset*) provindos da ponte ou do condicionador de sinais desvia o sinal desejado do seu nível de referência. Compensar estas variações é fácil durante a calibração já que o sinal permanece dentro da faixa de trabalho do sistema. Se o sinal diferencial da ponte está sendo convertido para um outro sinal referenciado ao terra, o desvio da ponte e amplificadores pode facilmente criar um sinal que é teoricamente abaixo do terra. Quando isto ocorre ele cria um ponto morto. A saída do conversor AD vai permanecer marcando zero até que a saída da ponte se torne positiva o suficiente para superar todos os desvios negativos do sistema. Para evitar isto, um intencional desvio positivo deve ser previsto no circuito. Este desvio assegura que a saída vai estar na sua região de trabalho, mesmo se a ponte e os componentes associados tiverem efeitos de desvios negativos. Um problema que surge com essa solução é a redução da faixa dinâmica de trabalho. O ajuste dos desvios pode ser feito através de potenciômetros mecânicos, potenciômetros digitais ou ainda através de software.

5.8.4 EFEITO TERMOELÉTRICO

O efeito termoelétrico pasasita vai existir quando diferentes tipos de materiais forem conectados, como por exemplo, os pinos dos circuitos integrados e as trilhas de cobre da placa de circuito impresso, e cujas temperaturas de ambas as junções estejam em valores diferentes. O valor da tensão gerada pode ser de dezenas de microvolts para uma variação de apenas 1 $°C$. O diagrama da Figura 37 mostra uma junção típica com efeito termopar formado entre as trilhas de cobre da placa de circuito impresso e os pinos de um circuito integrado amplificador. A tensão gerada por efeito termoelétrico fica em torno de 35 $\mu V/°C$. Considerando os baixos valores de tensão tratados pelos circuitos amplificadores, pode-se verificar que este efeito não é desprezível.

5.8.4.1 Desvio por deriva (*offset Drift*) do amplificador

Desvio por deriva (*offset drift*) e ruídos são os maiores problemas associados aos circuitos utilizando pontes. Na célula de carga do exemplo acima, a saída em plena escala da ponte é 2 mV/V e a resolução desejada é de 12 bits. Se a célula de carga é alimentada com uma fonte de 5 V, então o valor de final de escala será de 10 mV e a exatidão da medida deve ser de 2,5 μV ou melhor. Dizendo de outra forma, um desvio de apenas 2,5 μV irá criar um

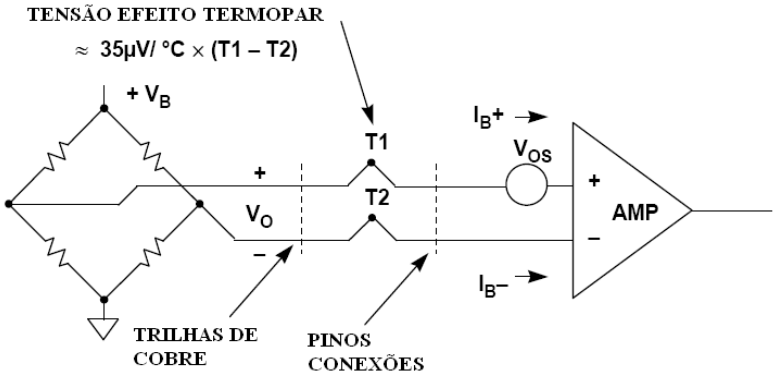

Figura 37: Efeito termoelétrico formado pelas trilhas de cobre e os pinos do circuito integrado, quando as junções estão submetidas a diferentes temperaturas.

erro de 1 bit. Isto é um grande desafio para os amplificadores convencionais de alta qualidade. O Amplificador OP07, por exemplo, tem um desvio máximo por temperatura de 1,3 $\mu V/°C$ e uma deriva máxima devido ao tempo de funcionamento de 1,5 μV por mês. Para manter estes valores extremamente baixos de desvios por temperatura ou por tempo de funcionamento, algum tipo de ajuste é necessário. Isto pode ser feito via *hardware*, *software*, ou através de uma combinação de ambos.

A tensão de *offset* e a corrente de polarização são outra fonte de erro. Qualquer desbalanceamento na corrente de polarização vai produzir erro de *offset*. Somando-se a isso, a tensão de *offset* e corrente de polarização são uma função da temperatura. Idealmente para a tarefa de condicionamento de sinal de ponte de Wheatstone, devem ser utilizados amplificadores de instrumentação de alto desempenho, baixo *offset*, baixa deriva térmica, baixa corrente de polarização e baixo ruído.

5.8.5 EXCITAÇÃO POR TENSÃO ALTERNADA

A excitação de uma ponte por tensão alternada pode remover eficazmente a tensão de *offset* do circuito em série com a ponte até o conversor analógico/digital. O conceito é simples. A tensão de saída da ponte é medida em duas condições diferentes, conforme ilustrado na Figura 38. Na primeira medida obtém-se o valor V_A, onde este é valor da saída da ponte somado com o erro de offset da rede E_{OS}. A polaridade da tensão de excitação da ponte é revertida e uma segunda medida V_B é tomada. Subtraindo V_B de V_A leva a $2V_O$, e o termo E_{OS} que representa o erro de offset é cancelado, conforme mostrado na Figura 38.

No entanto, a inversão da polaridade da tensão de excitação da ponte requer componentes adicionais especialmente desenvolvidos para essa finalidade, que devem possuir baixas perdas

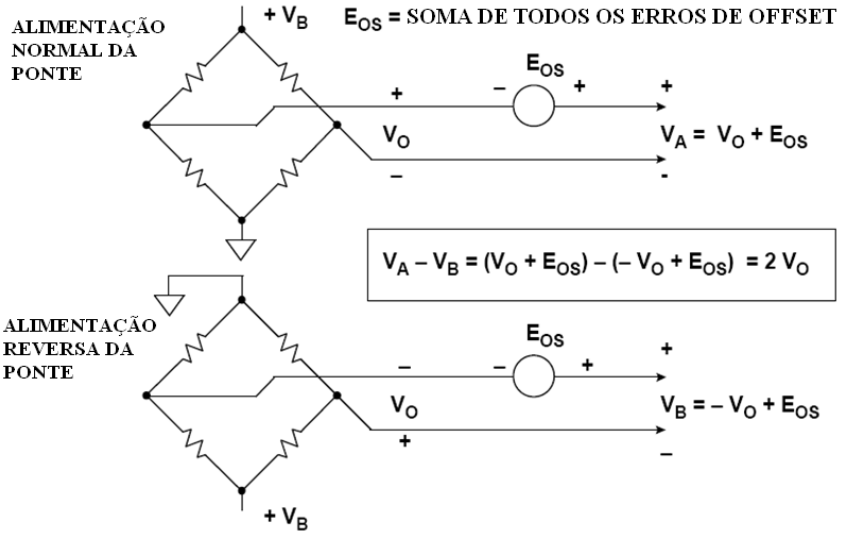

Figura 38: Princípio de funcionamento das pontes com alimentação CA visando o cancelamento do erro de offset.

de comutação e baixo ruído. O autor tentou esta solução inicialmente e não conseguiu acesso a estes componentes no mercado nacional.

5.8.6 SOLUÇÃO ADOTADA PARA A CORREÇÃO DO *OFFSET*

A solução adotada foi a proposta por Kester (2003), que apresenta uma forma de correção dos desvios utilizando uma chave de dois pólos, dois contatos (DPDT) entre a ponte e o conversor, conforme pode ser visto na Figura 39.

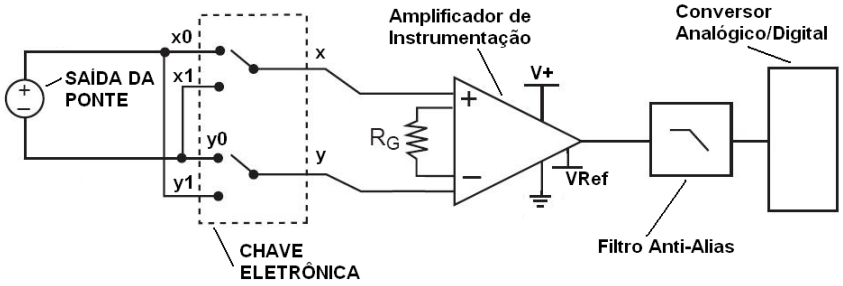

Figura 39: Circuito de correção de desvios por combinação de software e hardware.

Utilizando-se um circuito multiplexador pode-se implementar um sistema de auto-calibração utilizando uma combinação de *hardware* e *software*.

Utilizando-se o mesmo conceito das pontes CA discutidas anteriormente, tem-se o seguinte princípio de funcionamento: uma leitura do conversor AD é feita quando a chave está na posição ($x = xo$ e $y = y0$) e subtraída de outra leitura feita quando está na posição ($x = x1$ e $y = y1$). O resultado é $2 \times Vo \times Ganho$.

Antes dos sinais irem para o amplificador, foi inserido um circuito multiplexador, cuja finalidade é funcionar como uma chave analógica.

Na rotina de software são feitas duas medições. A primeira capturando os sinais normais (reporte-se à Figura 28 para a descrição que segue). Isto é feito colocando-se ambos os sinais de controle A e B em nível lógico baixo. Nesta condição, as saídas x e y receberão o sinal das entradas xo e yo, respectivamente, que correspondem aos sinais normais.

Uma segunda medição é feita, agora com A em nível lógico alto e B em nível lógico baixo, as saídas x e y receberão os sinais yo e xo, respectivamente, que correspondem aos sinais invertidos.

Observe que a tensão de entrada (v_{in}) vai aparecer a cada medida em uma das entradas do amplificador de instrumentação com a polaridade trocada. Assim na primeira leitura teremos: $V_1 = v_{in} + v_{os}$, enquanto que na segunda leitura teremos $V_2 = -v_{in} + v_{os}$. Se estas duas amostras são subtraídas, obtém se:

$$Vo = V_1 - V_2$$

$$Vo = (v_{in} + v_{os}) - (-v_{in} + v_{os}) = 2v_{in} \tag{5.3}$$

Conforme pode ser observado na Equação 5.3, o valor obtido com esse procedimento é igual a $2v_{in}$. Se dividirmos este valor por dois, obtém-se o sinal original. O resultado é um valor livre da tensão de *offset* provocada por efeitos termoelétricos ou de outra natureza. Assim, os problemas com desvios são simplesmente eliminados. Esta técnica também melhora a relação sinal ruído por um fator de $\sqrt{2}$. A figura 39 ilustra o princípio de funcionamento deste estágio.

Existe uma rotina de software que contínua e periodicamente amostra a entrada e se auto-ajusta para manter uma diferença mínima entre os pinos de entrada. Como este ajuste é contínuo, desvios com o tempo ou com a temperatura torna-se uma função do circuito de correção.

5.8.7 SOLUÇÃO ADOTADA PARA CORREÇÃO DE *DRIFTS*

Na Figura 28, o multiplexador IC5 funciona como uma chave analógica e é utilizado para desconectar a saída da ponte do amplificador ao mesmo tempo que curto-circuita as entradas deste. Deixando o outro lado da ponte conectado à entrada do amplificador, a tensão de entrada em modo comum é mantida, eliminado desta maneira qualquer erro que possa ser causado em função do erro por tensão em modo comum. Curto-circuitando-se as entradas do amplificador permite que uma medida do desvio de tensão (*offset*) seja feita. Esta leitura é então subtraída

das leituras normais subseqüentes para remover o desvio. No entanto, ele não remove o desvio da ponte em si, mas apenas o dos componentes eletrônicos entre a ponte e o conversor AD.

A saída da ponte é medida em um determinado estado, sem carga, por exemplo. Então uma carga é aplicada à célula e outra medida é feita. A diferença entre estas duas medidas é devido apenas ao estímulo aplicado. Tomando-se a diferença entre as leituras remove-se não somente os desvios internos devido aos componentes, mas também o desvio da própria ponte.

5.8.8 VALOR DE REFERÊNCIA DE CALIBRAÇÃO

Valor de referência de calibração (*shunt*) é uma maneira prática de se certificar que o sistema está operando corretamente. Quando a resistência de calibração é habilitada, uma resistência é adicionada ou removida de um dos braços da ponte. Isto tem o efeito equivalente a uma deflexão forçada do extensômetro, já que irá alterar a sua resistência. Conhecendo-se o valor da resistência introduzida, pode-se calcular a tensão de saída esperada. Através da leitura, pode-se comparar o valor esperado e o real e verificar se estão em conformidade.

Embora os valores de calibração possam ser armazenados em memória permanente do tipo EEPROM, deve-se considerar que estes são muito sensíveis as condições ambientais do momento da calibração, e portanto, o reaproveitamento desse valores encontra limitações de praticidade.

5.8.9 FILTRAGEM

5.8.9.1 Filtro analógico

Com a finalidade de filtrar as freqüências altas, foram adicionados capacitores de 100 nF em paralelo com os resistores de realimentação dos amplificadores operacionais IC4A e IC4B, cuja a freqüência de corte é dada por:

$f_c = \frac{1}{2\pi RC}$

Com os valores dos componentes utilizados, esta é de 15,9 Hertz.

5.8.9.2 Filtro digital

De acordo com Smith (2007) o filtro de média móvel é o tipo mais comum implementado em processadores digitais de sinais. Uma das principais razões é sua facilidade de implementação e de uso. No entanto, apesar de sua simplicidade este tipo de filtro é o mais indicado

para remoção de ruídos aleatórios, devendo ser a técnica de primeira escolha para filtragem no domínio do tempo. Em termos de equação, pode ser escrito como:

$$y[i] = \frac{1}{N} \sum_{k=0}^{N-1} x(i-k)$$

onde $y[]$é o sinal de saída, $x[]$ é o sinal de entrada e N é o número de pontos discretos da janela.

Quanto maior a janela, maior será a relação sinal/ruído, já que pode-se demonstrar que este é melhorado por um fator equivalente a $\sqrt{N}$. Por exemplo, se forem utilizados 25 pontos para a janela do filtro, o ruído vai ser reduzido por um fator de 5.

A Figura 40 ilustra a resposta do filtro de média móvel com janela de 11 pontos, para o processamento de um sinal de entrada contaminado com ruído aleatório.

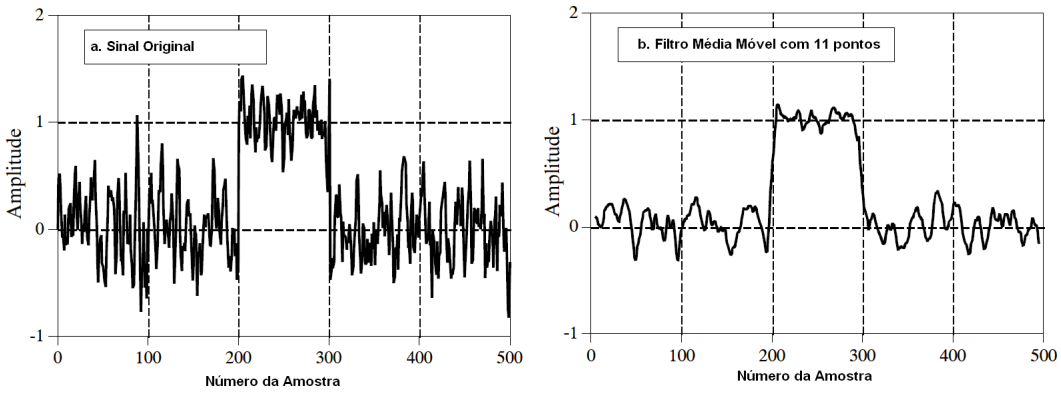

Figura 40: Resposta de saída de um filtro digital de média móvel contendo 11 pontos em reposta a um pulso retangular de entrada contaminado com ruído aleatório (SMITH, 2007).

Para o presente projeto, foi implementada uma rotina de filtro de média móvel conforme pode ser visto na listagem que se segue. Para tanto foi utilizado um *buffer* circular de 32 posições. Embora tenham sido reservadas 32 posições, o tamanho da janela é parametrizável. Por questões de velocidade de processamento, é recomendável que o tamanho da janela seja uma potência de dois, ou seja, 2, 4, 8, 16 ou 32. Isto porque os compiladores utilizam, de forma muito eficiente, a função de deslocamento de bits para a direita (*shift bit right*) para fazer as divisões.

5.8.9.3 Listagem da rotina de média móvel

```
if (ucWindow_Size != gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL])
        {
                ucWindow_Size = (UCHAR) gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL];
                ACC = num_amostras = pos_amostra_atual = 0;
```

```
}

if ( (ucWindow_Size < 1) || (ucWindow_Size > 32) )
{
        gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL] = ucWindow_Size = 8;
}
/*********************************************/
if (LENDO_RECALIBRACAO)
{
        acc_cal += ustemp;
        LENDO_RECALIBRACAO=FALSE;
        LENDO_TEMPERATURA=TRUE;
        FIRST_READING=TRUE;
        Le_ADC();
}
else if (LENDO_TEMPERATURA) // trata o canal 'temperature'
{
        acc_temp += ustemp;
        LENDO_TEMPERATURA=FALSE;
}
else  // trata o canal 'strain'
{
        if ( ucWindow_Size == 1) // não está utilizando filtro média móvel
        {
                gusRegInputRegister[REG_INPUT_STRAIN_VALUE] = ustemp;
        }
        else  // está utilizando filtro média móvel
        {
                if (num_amostras < ucWindow_Size)
                {
                        num_amostras++;
                }
                else
                {
                   ACC -=  gusRegInputRegister[pos_amostra_atual+OFFSET_SAMPLE];
                   // elimina amostra mais antiga
                }
                gusRegInputRegister[pos_amostra_atual+OFFSET_SAMPLE] = ustemp ;
                // duas leituras, uma só amostra

                ACC += ustemp; // soma a amostra atual

                gusRegInputRegister[REG_INPUT_STRAIN_VALUE] = ACC/num_amostras;
                if (SISTEMA_CALIBRADO && gusRegHolding[RECALIBRATION_ENABLED])
                {
                   gusRegInputRegister[REG_INPUT_STRAIN_VALUE] += diferenca_offset;
                }
                if (++pos_amostra_atual == ucWindow_Size)
                        pos_amostra_atual = 0;
        }
        if (gusRegHolding[RECALIBRATION_ENABLED])
        {
                if (RECALIBRANDO)
                {
                        LENDO_RECALIBRACAO=TRUE; // AGORA VAI LER A recalibração
                        FIRST_READING=TRUE;
                        Le_ADC();
                }
                cont_aux_temp++;
                if (cont_aux_temp == 250)
                {
                        cont_aux_temp = 0;
                        acc_cal=0;
                        acc_temp=0;
                        RECALIBRANDO=TRUE;
                }
                else if (cont_aux_temp == 8) // 8 AMOSTRAS
                {
                        if (SISTEMA_CALIBRADO)
                        {
                          offset_atual = (acc_cal >> 3);
                          diferenca_offset = (signed int16)
                             (gusRegHolding[VALOR_REFERENCIA_CALIBRACAO] - offset_atual);
                          gusRegInputRegister[REG_OFFSET_DIF] = diferenca_offset;
```

```
                        }
                        temperatura_atual= (acc_temp >> 3);
                        gusRegInputRegister[REG_INPUT_TEMPERATURE_VALUE] = temperatura_atual ;
                        RECALIBRANDO=FALSE;
                }
        }
        else
                cont_aux_temp = 0;
}
```

5.9 MEDIÇÃO DA TEMPERATURA

Como mostrado no Capítulo FUNDAMENTAÇÃO TEÓRICA, os extensômetros são bastante sensíveis a variações de temperatura, especialmente quando ligados na configuração de ¼ de ponte. Visando compensar estes efeitos (através de *software*), foi instalado um sensor de temperatura ambiente na placa de circuito impresso do terminal de aquisição. A temperatura ambiente é comparada com a de calibração e, em caso de desvio, um fator é aplicado visando compensar o efeito dessa variação.

Tabela 2: Desempenho de sensores de temperatura

Tipo do Sensor	Linearidade	Faixa	Sensibilidade
Termistor	pobre	-80 a +150°C	ótima
RTD	boa	-260 a +850°C	pobre
LM35	ótima	0 a +100°C	boa

Fonte: (LEONG, 2001)

Para se medir temperatura, várias opções podem ser consideradas. Um dos elementos mais comuns utilizados são os termistores tipo NTC (*negative temperature coefficient*). Outro componente bastante utilizado é o elemento RTD (*Resistance Temperature Detector*) que é na verdade um termistor PTC (*Positive Temperature Coefficient*). Por exemplo, o PT100 é um termistor de resistência igual a 100 Ω na temperatura de 0 °C. Uma grande desvantagem da utilização de termistores é a necessidade de circuitos adicionais, tais como fornecimento de uma fonte de corrente constante insensíveis a variações de temperatura e amplificadores de instrumentação de baixo ruído e também baixa deriva térmica para condicionamento do sinal. Isto logicamente torna o sistema mais complexo e, portanto, mais dispendioso.

Outra opção é a utilização de circuitos integrados especialmente projetados para medição de temperatura, como por exemplo, os tipos LM 335 e LM35 fabricados pela National Semiconductor e o tipo AD592 da Analog Devices.

A seleção foi feita baseada nos seguintes parâmetros:

- Linearidade

- Faixa de operação
- Sensibilidade
- Facilidade de Implementação
- Custo
- Facilidade de Aquisição

Um comparativo de desempenho de algumas características destes pode ser visto na Tabela 2. Por razões de custo, simplicidade de implementação, ótima linearidade e faixa de operação compatível, optou-se pelo sensor LM35, o qual será descrito brevemente a seguir.

5.9.1 SENSOR DE TEMPERATURA

O sensor de temperatura LM35 é um dispositivo de 3 terminais que fornece uma tensão de saída diretamente proporcional a temperatura. A tensão de saída é de 10 mV/°C, referenciado a partir de 0 °C. Por exemplo, se a temperatura for de 30 °C a saída será de 300 mV. A exatidão do dispositivo é de 0,5 °C para a faixa de 0 a 70 ° C. O LM35 pode ser alimentado com uma tensão que pode variar entre 5 Volts e 30 Volts. Este componente pode ser encontrado em diversos encapsulamentos, incluindo o tipo plástico TO-92 o qual foi utilizado neste projeto. A pinagem do LM35 é vista na figura 41.

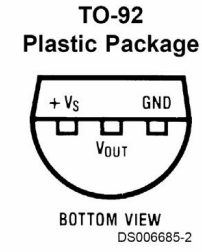

Figura 41: Encapsulamento do sensor de temperatura LM35 (TO-92)

O microcontrolador PIC16F876 possui um conversor AD de 10 bits conectado a um multiplexador analógico interno que permite que cada pino da porta A seja utilizada como entrada analógica (desde que configurado para tal). Para a faixa de valores de 0 a 5 Volts, o conversor de 10 bits fornece uma resolução de 5 mV (5V/1023) que é compatível com a precisão do LM35 (0,5 °C = 5 mV).

A função para leitura da temperatura é mostrado na listagem a seguir:

```
double getTemperature(int adcValue)
{
//10 bit adc
return (adcValue/1023)*5; // cálculo da Temperatura
}
```

5.9.2 INTERFACE DE CONTROLE DE CALIBRAÇÃO E DE ENDEREÇAMENTO

5.9.2.1 Endereçamento

O circuito do terminal possui também chaves para a seleção de endereço do módulo para identificação do mesmo pela unidade mestre. O endereçamento é feito através de SW1, que é um conjunto de microchaves (*dip-switch*) de 8 posições, permitindo endereçar teoricamente até 256 endereços diferentes. No caso específico do MODBUS são utilizados, no máximo, 247 destes endereços. A Figura 42 ilustra como foi implementado eletricamente, enquanto que a Figura 43 mostra a sua localização física na placa de circuito impresso.

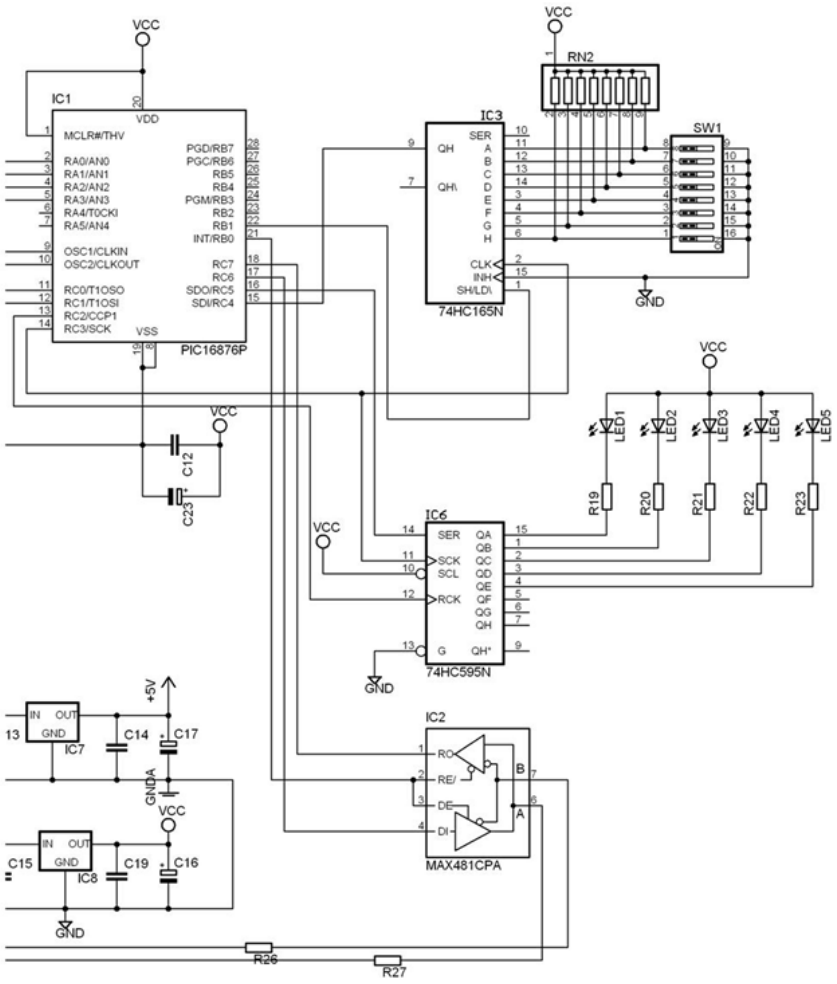

Figura 42: Esquema elétrico da parte referente à programação do endereço do terminal

Para interfacear a chave DIP com o microcontrolador foi utilizado um conversor paralelo/serial tipo 74HC165 (IC3). Sua utilização se fez necessária devido a limitações no número de pinos do microcontrolador. A aquisição do conteúdo das chaves é feita ao se inicializar o sistema, via protocolo SPI. Este foi utilizado porque também pode ler e escrever em outros componentes do sistema, tais como o conversor serial para paralelo 74HC195 (IC6) e também para se comunicar com um conversor analógico/digital externo de maior resolução, ou seja, o sistema já está preparado para futuros aperfeiçoamentos, sendo que a inclusão de um conversor A/D de maior resolução, externo ao microcontrolador, é naturalmente, a primeira coisa a ser feita em caso de continuidade do projeto. A implementação foi feita através do seguinte trecho de código:

```
var_input = spi_le_byte(); // lê DIP SWITCH
gucThisStationAddress = (var_input & 31);
if (gucThisStationAddress == 0)
     gucThisStationAddress = 31; //começa com 31
```

Como pode ser observado, nesta implementação o número de nós foi limitado em 31 unidades. Isto devido ao circuito integrado utilizado para a comunicação RS485 que permite no máximo 32 unidades. Caso seja necessário implementar uma rede maior, deve-se trocar o CI de comunicação. Existem hoje no mercado circuitos integrados que podem se comunicar com até 256 unidades simultaneamente.

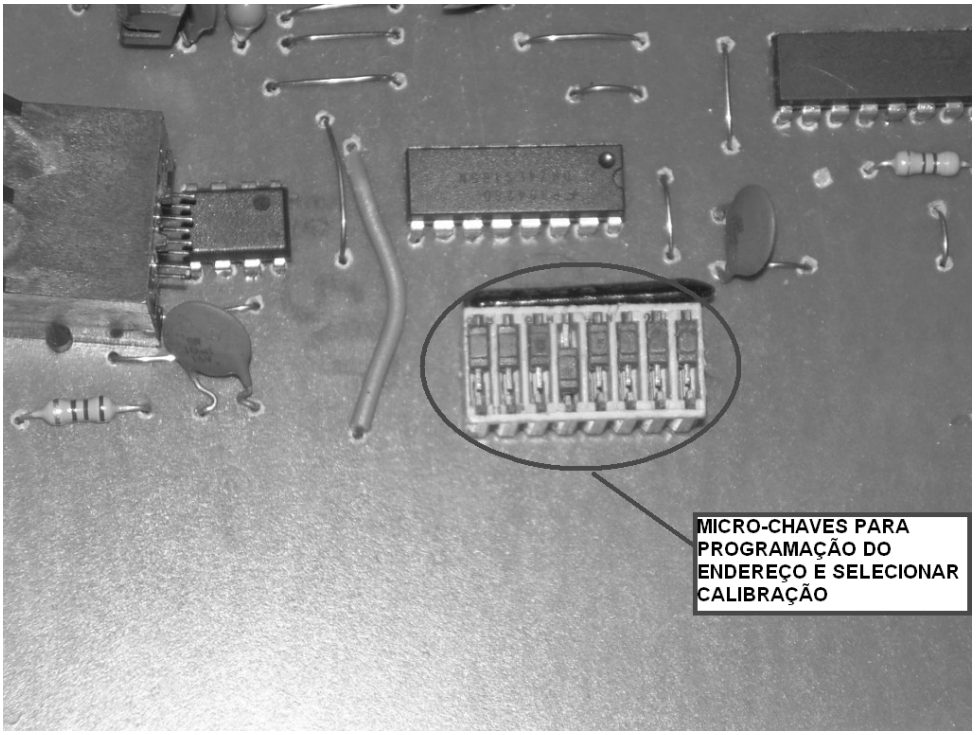

Figura 43: Localização da microchaves de endereçamento na Placa de Circuito Impresso

Para o sistema implementado, a velocidade selecionada foi de 9600 BPS, 8 BITS e SEM PARIDADE (Valor padrão do Modbus-RTU).

Existe também um pino no transmissor que é utilizado para a sua habilitação, o qual é controlado pelo microcontrolador, no momento em que se faz necessária a transmissão de dados. Dessa forma, a transmissão não acontece de forma permanente, mas somente no momento em que for acionada pelo microcontrolador, reduzindo consumo de energia a aumentando a confiabilidade. O trecho de código a seguir ilustra como é feito esse controle.

```
#define HABILITA_TX disable_interrupts(INT_RDA); output_high(PIN_B0);
#define HABILITA_RX output_low(PIN_B0); enable_interrupts(INT_RDA);
```

Observar que a macro criada possibilita o envio de HABILITA_TX quando se desejar transmitir e HABILITA_RX quando se desejar voltar ao estado padrão, que corresponde ao de ficar monitorando o meio de transmissão.

5.10 PROTOCOLO MODBUS

Esta seção tem a finalidade de descrever:

1. O protocolo Modbus Padrão.
2. Como um dispositivo solicita o acesso a outro dispositivo.
3. Como este responderá.
4. Como os erros serão detectados e informados.
5. Os comandos necessários a este projeto.

Maiores informações podem ser obtidas em (MODBUS-IDA, 2007a), (MODBUS-IDA, 2007b) e (MODICON, 2007).

5.10.1 DESCRIÇÃO

O protocolo Modbus foi desenvolvido em 1979 pela *Modicon Industrial Automation Systems*, hoje *Schneider Electric*, para comunicar um dispositivo mestre com outros dispositivos escravos (SEIXAS FILHO, 2006).

É um protocolo de especificação aberta desde o princípio. Este é talvez o protocolo de mais larga utilização em automação industrial, pela sua simplicidade e facilidade de implementação. Estas características fizeram-no se tornar um padrão amplamente utilizado, adotado por vários fornecedores, sendo aplicado aos mais diversos níveis de aplicações, inclusive sistemas de supervisão.

Foi criada uma associação, Modbus-IDA, com fins não lucrativos agrupando usuários e fornecedores de dispositivos de automação que visam a adoção do pacote de protocolos Modbus e a evolução da arquitetura de endereçamento para sistemas de automação distribuídos em vários segmentos de mercado.

Modbus-IDA fornece a infra estrutura para obter e compartilhar informação sobre os protocolos, suas aplicações e a certificação de dispositivos visando simplificar a implementação pelos usuários.

5.10.2 Modelo de Comunicação

O protocolo Modbus é baseado em um modelo de comunicação mestre-escravo, onde um único dispositivo, o mestre, pode iniciar transações. O demais dispositivos da rede (escravos) respondem, suprindo os dados requisitados pelo mestre ou executando uma ação por ele comandada. Os papéis de mestre e escravo são fixos, quando se utiliza comunicação serial, mas em outros tipos de rede, um dispositivo pode assumir ambos os papéis, embora não simultaneamente.

5.10.3 Transações entre dispositivos

As mensagens de consulta e de resposta também são chamadas de bilhetes. Na mensagem de consulta, o código de função informa ao dispositivo escravo com o respectivo endereço, qual a ação a ser executada. Os *bytes* de dados contêm informações como qual o registrador inicial e a quantidade de registros a serem lidos. O campo de verificação de erro permite ao escravo validar os dados recebidos. A Figura 44 ilustra uma transação típica.

Figura 44: Esquema de Troca de Mensagens entre Mestre e Escravo no protocolo MODBUS

Na mensagem de resposta, o código de função é repetido de volta para o mestre. Os *bytes* de dados contêm os dados coletados pelo escravo ou um código informando um possível erro ocorrido. Se um erro ocorre, o código de função é modificado para indicar que a resposta é de erro e os *bytes* de dados contém um código que descreverá o erro. A verificação de erro permite

ao mestre validar os dados recebidos.

Durante a comunicação em uma rede Modbus, o protocolo determina como o dispositivo conhecerá seu endereço, como reconhecerá uma mensagem endereçada para ele, como determinar o tipo de ação a ser tomada e como extrair o dado ou outra informação qualquer contida na mensagem. Se uma resposta é necessária, como o dispositivo construirá uma mensagem e a enviará.

O mestre pode endereçar mensagens para um escravo em particular ou enviar mensagens para todos (*broadcast*). Os escravos retornam uma mensagem somente para as consultas endereçadas especificamente a ele. As mensagens *broadcast* não geram respostas.

5.10.4 Modbus e o Modelo OSI

O protocolo de comunicação de uma rede é um conjunto de regras e convenções de linguagem que permite a conversação e troca de informação entre sistemas (TANEMBAUM, 1997). Os protocolos, atualmente adotados em redes industriais, são baseados em documento desenvolvido pela ISO (*International Standards Organization*). Esse documento, denominado OSI (*Open System Interconnection*) é um modelo de referência para o desenvolvimento de protocolos de comunicação. A estrutura do modelo de redes OSI é baseada em sete camadas. Softwares desenvolvidos com base nesse modelo são ditos abertos, pois qualquer fabricante de equipamentos pode usá-los para desenvolvimento de produtos para serem empregados em rede.

O protocolo MODBUS pode ser enquadrado na camada 7 (Camada de Aplicação) do modelo OSI. A Figura 45 mostra a localização do protocolo Modbus em relação as camadas do modelo OSI.

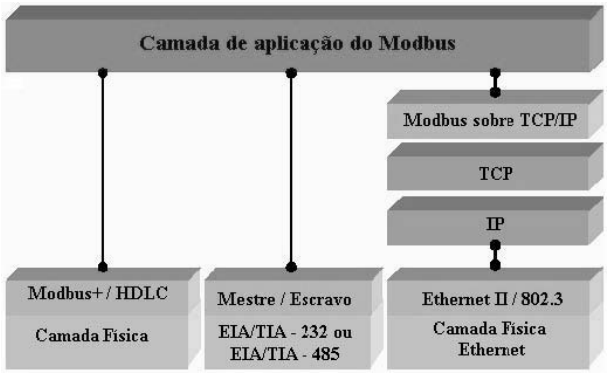

Figura 45: Localização do protocolo MODBUS em correspondência ao modelo OSI/ISO.

Algumas características do protocolo Modbus são fixas, como o formato da mensagem, funções disponíveis e tratamento de erros de comunicação. Outras características são selecio-

náveis como o meio de transmissão, velocidade, timeout, bits de parada e paridade e o modo de transmissão (RTU os ASCII). A seleção do modo de transmissão define como os dados serão codificados. Define também uma estrutura de mensagens de comunicação usadas para transferir dados discretos e analógicos entre dispositivos microprocessados com detecção e informação sobre possíveis erros ocorridos durante a transmissão.

5.10.4.1 Topologia Física

Referindo-se, ainda, à Figura 45, pode-se observar que a camada 1 contém o meio físico de transmissão (cabos condutores, fibra óptica ou atmosfera) e a interface de rede, que converte a grandeza física em valor digital.

Observar que, para o caso do protocolo Modbus, sua camada física de transporte pode ser as mais diversas, embora seja utilizada normalmente sobre conexões seriais padrão RS-232/RS-485. Também pode ser utilizado o TCP/IP sobre Ethernet, conhecido como MODBUS TCP/IP.

Por sua vez, as conexões seriais podem ser:

- Ponto a Ponto com RS-232.
- Barramento Mutiponto com RS-485.

5.10.4.2 Modos de Transmissão

Existem dois modos de transmissão:

1. ASCII (*American Code for Information Interchange*) e
2. RTU (*Remote Terminal Unit*).

Que são selecionados durante a configuração dos parâmetros de comunicação. Nos protocolos MODBUS Plus e TCP/IP as mensagens são colocadas em quadros *(frames)*, não sendo necessária a definição do modo de transmissão, usando sempre o modo RTU. O modo RTU também é chamado de Modbus-B ou Modbus Binário e é o modo preferencial sugerido pela norma.

5.10.4.3 Composição do bilhete MODBUS-ASCII

A composição de um bilhete MODBUS-ASCII é mostrada na Figura 46.

Início	Endereço	Função	Dados	LRC	Término
':'	2 caracteres	2 caracteres	N caracteres	2 Caracteres	CR LF

Figura 46: Composição de um bilhete (mensagem) do protocolo MODBUS-ASCII

Cada *byte* da mensagem (bilhete) é enviado como dois caracteres ASCII. Durante a transmissão, intervalos de até um segundo entre caracteres são permitidos, sem que a mensagem seja truncada. Algumas implementações fazem uso de tais intervalos de silêncio como delimitadores de fim de mensagem, em substituição à seqüência CR (*Carriage Return*) + LF (*Line Feed*). Cada *byte* contém um caractere ASCII entre 0 e 9, A a F. Para a verificação de erros são utilizados os *bytes* LRC (*Longitudinal Redundancy Check*)

- Definição do início e fim da mensagem
  - ASCII: Inicia com o caractere ":" e termina com os caracteres CR seguido de LF .

10 bits por byte

- 1 *start bit*
- 7 bits de dados LSB enviado primeiro
- 1 bit de paridade (par/ímpar)
- 1 *stop bit* 0 bit de paridade
- 2 *stop bits*

5.10.4.4 RTU

A mensagem deve ser transmitida de maneira contínua, já que pausas maiores que 3,5 caracteres indicam o término da transmissão. Os bytes são transmitidos em dígitos hexadecimais. Para a verificação de erros são utilizados os bytes do CRC (*Cyclic Redundance Check*)

- Definição do início e fim da mensagem
  - Tempos de silêncio de 3,5 caracteres.

11 bits por byte , sendo:

- 1 *start bit*

- 8 bits de dados LSb enviado primeiro
- 1 bit de paridade (par/ímpar)
- 1 *stop bit* 0 bit de paridade
- 2 *stop bits*

Como a comunicação a ser utilizada neste trabalho é em modo RTU, vamos nos concentrar nesta forma de comunicação.

5.10.4.5 Composição do bilhete MODBUS-RTU

A composição de um bilhete MODBUS-RTU é mostrada na Figura 47.

Início	Endereço	Função	Dados	CRC	Término
Silêncio > 3,5 caractere	8 bits	8 bits	n $\times$ 8 bits	16 bits	Silêncio > 3,5 caractere

Figura 47: Composição de um bilhete (mensagem) do protocolo MODBUS-RTU

onde:

- Endereçamento (1 *byte*)
  - 0 : Usado para *broadcast*.
  - 1 a 247 : Usados pelos escravos.

    A faixa de endereços válidos vai de 0 a 247 (0x00 a 0xF7 hexadecimal), sendo que os dispositivos recebem endereços de um a 247. O endereço zero é reservado para *broadcast*, ou seja, mensagens com esse valor de endereço são reconhecidas por todos os elementos da rede. Quando o mestre envia uma mensagem para os escravos, este campo contém o endereço do escravo. Quando o escravo responde, coloca seu próprio endereço neste campo.
- Código da Função (1 *byte*)
  - Estabelece a ação a ser efetuada.
    - 0 a 127 : Funções
    - 128 a 255 : Informe de erro na transmissão.

Varia de 1 a 255 (0x01 a 0xFF), mas apenas a faixa de um a 127 (0x01 a 0x7F) é utilizada, já que o bit mais significativo é reservado para indicar respostas de exceção. Normalmente, uma resposta inclui o código de função da requisição que lhe deu origem. No entanto, em caso de falha, o bit mais significativo do código é ativado para indicar que o conteúdo do campo de dados não é a resposta esperada, mas sim um código de diagnóstico.

- *Bytes* de Dados
  - Informação adicionais necessárias.
  - Endereços de memória

    Tamanho e conteúdo do campo de dados variam com a função e o papel da mensagem, requisição ou resposta, podendo mesmo ser um campo vazio.
- Quantidade de itens transmitidos
- Quantidade de *bytes* do campo
- Verificação de Erros (2 *bytes*) CRC.

5.10.4.6 Detecção de Erros

- Há dois mecanismos para detecção de erros no protocolo Modbus serial:
- bits de paridade em cada caractere e o *frame check sequence* ao final da mensagem.
- A verificação de paridade é opcional em ambos os modos de transmissão, ASCII e RTU. Um bit extra é adicionado a cada caractere de modo que ele contenha um número par ou ímpar de bits ativos, caso sejam adotadas paridade par ou ímpar respectivamente. A principal desvantagem desse método é sua incapacidade de detectar números pares de inversões de bit. Caso não seja utilizado, o bit de paridade é substituído por um *stop bit* adicional. As Figuras 48 e 49 mostram os formatos dos bilhetes com paridade e sem paridade, respectivamente. O modo RTU utiliza como verificador de erro um valor 16 bits, o CRC, utilizando como polinômio, $P(x) = x^{16} + x^{15} + \ldots + x^2 + 1$. O registro de cálculo do CRC deve ser inicializado com o valor 0xFFFF.

MODBUS **RTU**, 8 BITS, **C/** PARIDADE, **1** STOP BIT										
START	1	2	3	4	5	6	7	8	PAR	STOP

Figura 48: Formato de transmissão serial de 1 byte do MODBUS-RTU com paridade

Observações:

MODBUS **RTU**, 8 BITS, **S/** PARIDADE, **2** STOP BIT									
START	1	2	3	4	5	6	7	8	STOP	STOP

Figura 49: Formato de transmissão serial de 1 byte do MODBUS-RTU sem paridade

- A verificação de erro é efetuada opcionalmente pela paridade de cada byte transmitido e obrigatoriamente pelo método CRC sobre toda a mensagem.
- O modo ASCII permite intervalos de tempo de até um segundo entre os caracteres sem provocar erros, mas sua mensagem típica tem um tamanho duas vezes maior que a mensagem equivalente usando o modo RTU.
- O modo RTU transmite a informação com um menor número de bits, mas a mensagem deve ter todos os seus caracteres enviados em uma seqüência contínua.
- O dispositivo mestre espera uma resposta por um determinado tempo antes de abortar uma transação (*timeout*).
- O tempo deve ser longo o suficiente para permitir a resposta de qualquer escravo. Se ocorre um erro de transmissão, o escravo não construirá a resposta para o mestre. Será detectado um *timeout* e o mestre tomará as providências programadas.

5.10.5 MAPEAMENTO DE MEMÓRIA

Todo dispositivo em uma rede Modbus deve ter a sua memória dividida em registradores de 16 bits numerados conforme o modelo apresentado na Tabela 3.

Tabela 3: Mapeamento das funções e memória do protocolo MODBUS

Endereço registrador	**Dispositivo**	**Função Leitura**	**Função Escrita**
0xxxx	Solenóides Saídas Discretas	1	5 ou 15
1xxxx	Entradas Digitais	2	-
3xxxx	Entradas Analógicas	4	-
4xxxx	Memória Registradores	3	6 ou 16

A divisão é baseada na estrutura de memória do dispositivo:

- Saídas discretas para os atuadores ON-OFF utilizam um bit. Cada registrador comporta 16 saídas.
- Entradas discretas para os sensores ON-OFF utilizam um bit. Cada registrador comporta 16 entradas.

- Entradas analógicas utilizam registradores de 16 bits para os valores obtidos por conversores A/D a partir dos sinais dos sensores analógicos.
- Registradores de Memória com 16 bits para os valores utilizados internamente no dispositivo.

Para o projeto aqui desenvolvido, as funções MODBUS correspondentes podem ser acessadas pelos endereços mostrados nas Tabelas 4 e 5.

Tabela 4: Mapeamento de Memória do terminal de rede (3xxxxx = Input Registers)

Endereço	**Função**
300001	REG_INPUT_STRAIN_VALUE
300002	REG_INPUT_TEMPERATURE_VALUE
300003	REG_OFFSET_DIF
300004	reservado
300005	OFFSET_SAMPLE
300006...300038	AMOSTRA_MEDIA_MOVEL[32]

Tabela 5: Mapeamento de memória do terminal de rede (4xxxxx = *Holding Registers*)

Endereço	**Função**
400001	VALOR_REFERENCIA_CALIBRACAO
400002	VALOR_OFFSET_REFERENCIA_MAXIMO
400003	VALOR_TEMPERATURA_CALIBRACAO_SISTEMA
400004	VALOR_TEMPERATURA_CALIBRACAO_PONTE
400005	VALOR_TEMPERATURA_CALIBRACAO_SHUNT
400006	VALOR_SHUNT_CALIBRACAO
400007	NUMERO_PONTOS_JANELA_MEDIA_MOVEL
400008	REVERSE_READING_ENABLED
400009	RECALIBRATION_ENABLED
400010	VALOR EXTENSÔMETRO (120,350)
400011	FATOR DO EXTENSÔMETRO (GAGE-FACTOR)
400012	TIPO LIGAÇÃO (1/4, 2/4 ou 4/4)

A identificação dos comandos (funções) de leitura e escrita são diferentes de acordo com o tipo de dado a ser lido ou escrito.

5.10.5.1 Função 0x01 - Read Coil Status

- Finalidade: Efetua a leitura do estado das saídas discretas.
- Formato da Requisição:

  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x01
  - Endereço inicial .....2 bytes
  - Número de saídas.....2 bytes
  - CRC .....2 bytes

- Formato da Resposta:

  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x01
  - Tamanho da resposta .....1 byte,
  - n Resposta..... n bytes
  - CRC ..... 2 bytes

- 1 Em valores de 2 bytes, o byte mais significativo é por convenção representado primeiro.
- 2 Cada grupo de oito saídas é representado por um byte, onde cada saída corresponde a um bit individual (1 = ligada, 0 = desligada). Esses bits são ordenados de acordo com os endereços das saídas, com o endereço inicial no bit menos significativo. Caso haja mais de oito saídas os múltiplos bytes de resposta são também ordenados por endereço, com os bytes representando os menores endereços primeiro.

5.10.5.2 Função 0x02 - Read Input Status

- Finalidade: efetua a leitura do estado das entradas discretas.
- Formato da Requisição:

  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x02
  - Endereço inicial .....2 bytes
  - Número de entradas .....2 bytes
  - CRC ..... 2 bytes

- Formato da Resposta:

  - Endereço do dispositivo...1 byte

  - Código de função.....1 byte, 0x02
  - Tamanho da resposta .....1 byte,
  - n Resposta.....n bytes
  - CRC .....2 bytes

- Byte mais significativo representado primeiro. 2 A representação é semelhante àquela das saídas, ou seja, cada entrada corresponde a um bit, sendo esses bits agrupados de oito em oito em bytes. Bits e bytes são ordenados por endereço, de forma crescente a partir do bit menos significativo do primeiro byte.

5.10.5.3 Função 0x03 - Read Holding Registers

- Finalidade: Efetua a leitura dos valores dos registradores de memória.
- Formato da Requisição:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x03
  - Endereço inicial .....2 bytes
  - Número de registradores..2 bytes
  - CRC .....2 bytes
- Formato da Resposta:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x03
  - Tamanho da resposta .....1 byte,
  - n Resposta.....n bytes
  - CRC .....2 bytes (Byte mais significativo representado primeiro).
- Os registradores são representados por ordem crescente de endereço. Cada registrador armazena normalmente um valor de 16 bits, dois bytes, dos quais o mais significativo vem representado primeiro. Registradores de 32 bits ocupam dois endereços consecutivos, com os 16 bits mais significativos contidos no primeiro endereço.

5.10.5.4 Função 0x04 - Read Input Registers

- Finalidade: Efetua a leitura dos valores das entradas analógicas.
- Formato da Requisição:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x04
  - Endereço inicial .....2 bytes
  - Número de registradores..2 bytes
  - CRC ..... 2 bytes
- Formato da Resposta:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x04
  - Tamanho da resposta .....1 byte,
  - n Resposta.....n bytes
  - CRC ..... 2 bytes (Byte mais significativo representado primeiro)

5.10.5.5 Função 0x05 - Force Single Coil

- Finalidade: Efetua a escrita de uma única saída discreta.
- Esse valor permanece constante enquanto não for alterado por uma nova operação de escrita ou pela programação interna do dispositivo.
- Formato da Requisição:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x05
  - Endereço .....2 bytes
  - Valor .....2 bytes (Ligado = 0xff00, desligado = 0x0000)
  - FCS .....1 ou 2 bytes 1
- Formato da Resposta:
  - Cópia da requisição.

5.10.5.6 Função 0x06 - Preset Single Register

- Finalidade: Efetua a escrita de um valor em um registrador de memória. Assim como acontece para as saídas, o valor no registrador permanece constante enquanto não for alterado por operações de escrita ou pelo próprio dispositivo.
- Formato da Requisição:
  - Endereço do dispositivo...1 byte
  - Código de função.....1 byte, 0x06
  - Endereço .....2 bytes
  - Valor .....2 bytes
  - CRC .....2 bytes
- Formato da Resposta:
  - Cópia da requisição. 1 Byte mais significativo representado primeiro.

5.10.5.7 Exemplo de Transação Modbus

O Mestre solicita uma leitura dos registradores 40105 a 40107 do terminal da rede número 6.

O primeiro byte a ser enviado é o endereço do dispositivo, neste exemplo igual a 6.

Observar que o endereço lógico dos registradores começa em 1, enquanto que o endereço físico começa em 0. Assim há uma diferença de 1 dígito entre o endereço lógico e o endereço físico do dispositivo. Por exemplo, o primeiro registrador é o “40001”, mas é endereçado como “40000”. Portanto, no exemplo dado, o endereço inical desejado é “40105”, assim devemos endereçá-lo como “104”, que transformado em hexadecimal será “0x0068”. Note que os registradores utilizam 16 bits para codificar a informação. Estes 16 bits são enviados em dois bytes separados (HIGH BYTE e LOW BYTE).

O próximo byte é a quantidade de registros a serem lidos. Neste caso, 3.

E, por último, são enviados os dois bytes de CRC na ordem inversa dos anteriores ou seja, primeiro o LOW BYTE e, depois, o HIGH BYTE.

A composição deste bilhete pode ser observada na Figura 50.

O escravo repete o código da função indicando uma resposta normal, conforme mostrado na Figura 51.

Solicitação			
Nome do Campo	**Exemplo (HEX)**	**ASCII**	**RTU**
Cabeçalho		’:’	
Endereço	06	‘’0’ ’6’	0000 0110
Código da Função	03	’0’ ’3’	0000 0011
Endereço Inicial (HI)	00	’0’ ’0’	0000 0000
Endereço Inicial (LO)	68	’6’ ’8’	0110 1000
Núm. Registros (HI)	00	’0’ ’0’	0000 0000
Núm. Registros (LO)	03	’0’ ’3’	0000 0011
Verificação de erro		LRC (2)	CRC(2)
Terminador		CR LF	
Total de Bytes		17	8

Figura 50: Exemplo comunicação MODBUS

Resposta			
Nome do Campo	**Exemplo (HEX)**	**ASCII**	**RTU**
Cabeçalho		’:’	
Endereço	06	‘’0’ ’6’	0000 0110
Código da Função	03	’0’ ’3’	0000 0011
Qtde. Bytes	03	’0’ ’3’	0000 0011
Dado (HI)	02	’0’ ’2’	0000 0010
Dado (LO)	2B	’2’ ’B’	0010 1011
Dado (HI)	00	’0’ ’0’	0000 0000
Dado (LO)	00	’0’ ’0’	0000 0000
Dado (HI)	00	’0’ ’0’	0000 0000
Dado (LO)	63	’6’ ’3’	0110 0011
Verificação de erro		LRC (2)	CRC (2)
Terminador		CR LF	
Total de Bytes		23	11

Figura 51: Exemplo de Comunicação MODBUS - Resposta

5.11 MÁQUINA DE ESTADOS MODBUS

O diagrama de estados Modbus-RTU é mostrado na Figura 52.

A transição do "estado inicial" para o estado de "repouso" precisa de um intervalo de silêncio do barramento de 3,5 vezes o tempo de cada caractere (t3,5). Isto assegura que, se o sistema tornar-se ativo durante a transmissão de um quadro, este bilhete (provavelmente incompleto) seja descartado.

No modo RTU, o estado da conexão é declarado em "repouso" se não for detectada nenhuma atividade de transmissão no barramento em um intervalo de tempo equivalente a 3,5 caracteres.

Quando a conexão está no estado de repouso, cada caractere de transmissão detectado será

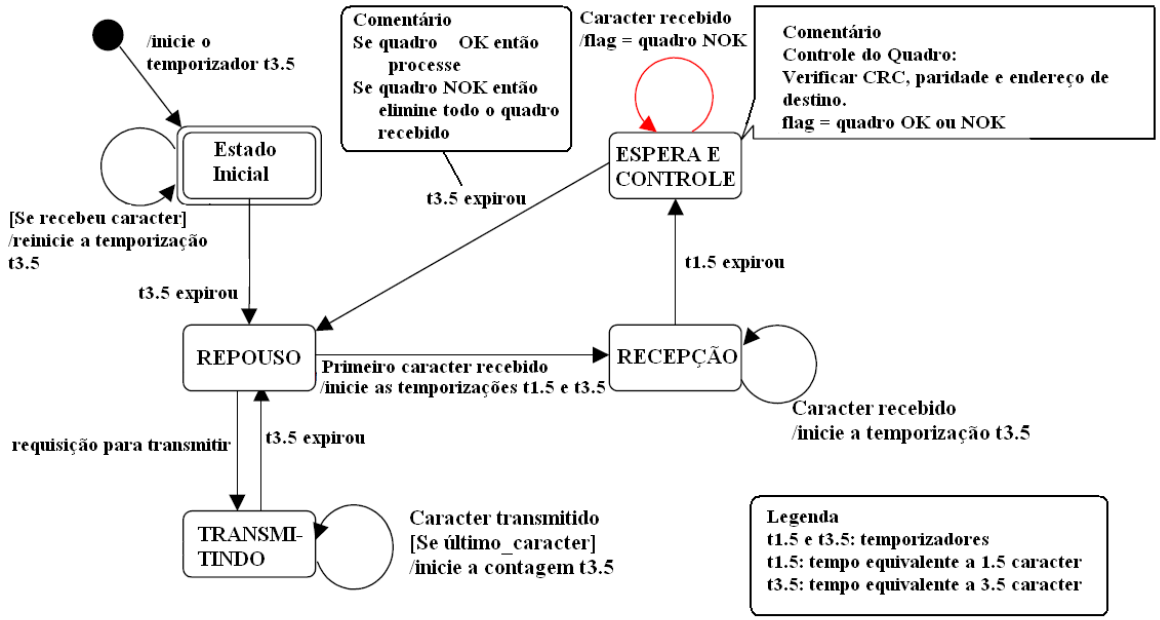

Figura 52: Máquina de estados MODBUS-RTU TX e RX da estação escrava (MODBUS-IRDA, 2007).

identificado como início de um novo quadro. A conexão vai para o estado "recepção". O final do quadro é identificado quando nenhum caractere for recebido tendo passado um intervalo de tempo igual a t3,5.

Após a detecção do final do quadro, um procedimento de cálculo e verificação do valor de CRC (*Cyclical Redundancy Checking*) é realizado. Depois disso, o campo de endereço é verificado para se determinar se o bilhete é endereçado para o dispositivo. Se for, será processado, caso contrário, é descartado. Também serão processados os do tipo *broadcast*, que são aqueles destinados a todas as estações.

5.11.0.8 Listagem da rotina principal da máquina de estados Modbus-RTU implementada

```
for(;;)
                {

                ...

                switch(estate_main)
                {
                case STATE_MAIN_IDLE:
                                if ( INICIANDO  )
                                {
                                        disable_interrupts(GLOBAL);
                                        if (eMBRcvState == STATE_RX_IDLE)
                                        {
                                                INICIANDO=FALSE;
                                                HABILITA_TX
                                                printf("MODBUS RTU - VALTER L.A. CAMARGO\r\n");
                                                HABILITA_RX
                                                Start_ADC();//DÁ INÍCIO AO PROCESSO DE LEITURA ADC
```

```
                                }
                                enable_interrupts(GLOBAL);
                        }
                break;
        case STATE_MAIN_FRAME_RECEIVED:
                if (Processa_bilhete() == MB_ER_NONE) // SE RECEBEU UM BILHETE VÁLIDO
                        {
                                if (Monta_bilhete_resposta())
                                        estate_main = STATE_MAIN_START_TX;
                                else
                                        estate_main = STATE_MAIN_IDLE;
                        }
                        else
                        {
                                estate_main = STATE_MAIN_IDLE;
                        }
                        break;
                case STATE_MAIN_START_TX:
                        Liga_LED(LED_TX);
                        delay_ms(4);
                        HABILITA_TX
                        enable_interrupts(INT_TBE);     //ENABLE FLAG
                        while (!FRAME_TRANSMITTED); // fica aqui até terminar a transmissão do bilhete
                        delay_ms(2);
                        HABILITA_RX
                        Desliga_LED(LED_TX);
                        estate_main = STATE_MAIN_IDLE;
        }
    }
}
```

5.11.1 ROTINA IMPLEMENTAÇÃO MÁQUINA DE ESTADOS TX

Para a transmissão dos bilhetes foi utilizada uma interrupção. Quando se deseja inicar a transmissão, no programa principal é habilitado um bit sinalizador de condição (*flag*) que irá forçar o microcontrolador a gerar uma interrupção. A cada byte transmitido é gerada uma nova interrupção e então o próximo caractere é transmitido. Isto se repete até que o último byte do bilhete, quando então a interrupção é desabilitada. Na listagem a seguir é mostrado como foi feita a implementação da máquina de estados de transmissão do protocolo Modbus-RTU.

```
/*************************/
#int_TBE
void ISR_TBE(void)
/*************************/
{
        while (!txsta.trmt);    //aguarda até que a transmissão esteja completa (buffer empty)
        switch(eMBSndState)
        {
            case STATE_TX_IDLE:
                        eMBSndState = STATE_TX_TRANSMITTING;
                        FRAME_TRANSMITTED = FALSE;
                        gucSndBufferPos=0;
            case STATE_TX_TRANSMITTING:
                        if (gucSndBufferPos < gucSndBufferCount )
                                {
                                putc(gucRcvBuffer[gucSndBufferPos++]);
                                }
                        else
                        {
                                disable_interrupts(INT_TBE);
                                FRAME_TRANSMITTED = TRUE;
                                eMBSndState=STATE_TX_IDLE;
                        }
        }
}
```

5.11.2 MÁQUINA DE ESTADOS RX

Da mesma forma que a máquina de estados TX, também foi utilizada uma interrupção para a rotina de recebimento dos bilhetes. O buffer gucRcvBuffer é utilizado para armazenar os elementos do bilhete. A listagem a seguir ilustra esta rotina.

```
#int_RDA
/*****************/
void ISR_RDA(void)
/*****************/
{
        UCHAR caracter;
        caracter = getc();

    switch ( eMBRcvState )
    {
        case STATE_RX_INIT:
                        /*-------------------------------------------------------------
                                 Se foi recebido um caracter quando estava inicializando,
                                então deve se aguardar até que o quadro termine.
                    -------------------------------------------------------------*/
        case STATE_RX_ERROR:
                        /*-------------------------------------------------------------------
                                Se houve algum erro, deve-se esperar que todos os caracteres
                           do quadro defeituoso seja recebido.
                    -------------------------------------------------------------------*/
                        timeout = 1;
                break;
            case STATE_RX_IDLE:

                        /*-------------------------------------------------------
                         No estado repouso aguarda-se a chegada de um novo caracter.
                                Quando for for recebido o primeiro caracter,
                        grava-o na posição inicial do buffer,
                                habilita os temporizadores e
                                muda o estado para STATE_RX_RCV.
                    -------------------------------------------------------*/
                        expected = 8; //fnc 1 a 6 - numero de caracteres esperados para essas funções
                        timeout = 8; // 4 x 4192 ms = 16384 us
                gucRcvBufferPos = 0;
                        FLAG_LED_RX_ON =TRUE; // FLAG PARA LIGAR O LED RX (EM MAIN.C)
                        eMBRcvState = STATE_RX_RCV;
            case STATE_RX_RCV:
                        /*-----------------------------------------------------------------------------
                                Se foi recebido mais caracteres do que o máximo número possível de bytes
                                do quadro modbus, o quadro inteiro deve ser ignorado.
                    -----------------------------------------------------------------------------*/
                        if( gucRcvBufferPos > 7)
                        {       // já foi recebido o endereço de destino e o código da função
                                if ( (gucRcvBuffer[1] == 15) || (gucRcvBuffer[1] == 16) )
                                {
                                /*-----------------------------------------------------------
                                15 =  write N coil; func 15 for 0x00000 area
                                16 =  write N regs; func 16 for 4x00000 area
                        a contagem dos bytes é
                            SS FF AA AA LL LL BC ?? CR CR
                                -----------------------------------------------------------*/
                                expected = 9 + gucRcvBuffer[6];
                        }
                        }

                        //
                if( gucRcvBufferPos > expected )
                {
                    eMBRcvState = STATE_RX_ERROR;
                        }
                        else
                        {
                    gucRcvBuffer[gucRcvBufferPos++] = caracter;
                }
                        if (gucRcvBufferPos == expected )
                                timeout = 1; //old=gus3_5_timeout;
    }// endswitch
        /*-------------------------------------------------------------------
                Reinicializa o temporizador a cada caracter recebido.
                Quando houver demora maior do que 3,5 vezes o tempo de cada quadro,
                o temporizador colocará a máquina de recepção no estado inicial
                -------------------------------------------------------------------*/
        Reset_timer(timeout);
}
```

5.12 CONSTRUÇÃO DOS PROTÓTIPOS

Foram desenvolvidas duas versões de unidades de rede. A primeira utilizando um microcontrolador AT89S8252 da Atmel e uma segunda utilizando o microcontrolador PIC16F876 da Microchip. Nos códigos inseridos no anexo pode-se verificar que estes possuem regiões exclusivas para cada um dos processadores. A seleção de qual implementação está sendo utilizada é feita no arquivo de cabeçalho mb_slave.h.

O autor inicialmente dispunha de placas de circuito impresso sem a montagem dos componentes de um equipamento desenvolvido para outra finalidade. Para sua implementação o autor já possuía os microcontroladores utilizados nesta placa (AT89S8252), o gravador destes (LP-10 Minipa) e também o ambiente de desenvolvimento de programação (KEIL-C). Para adiantar os trabalhos, antes de se decidir pela aquisição dos microcontroladores definitivos, a idéia era implementar o protocolo nestas placas para se ter uma noção do tamanho do código final. A partir disso seria definido qual o microcontrolador seria utilizado na segunda versão, que já incorporaria a interface analógica de condicionamento de sinal.

A primeira versão foi utilizada durante toda fase de desenvolvimento do protocolo e depois de depurados os programas, o protocolo MODBUS-RTU funcionou como o esperado. Uma fotografia do terminal utilizado na primeira versão pode ser vista na Figura 53.

A segunda fase, consistiu em adaptar o código desenvolvido para a linha de microcontroladores 8051 (da qual o microcontrolador AT89S8252 faz parte) para a linha de microcontroladores PIC, o que demandou algum esforço, já que a forma de endereçamento dos registradores é bem diferente na família 8051 do que é na família PIC.

A seguir são descritos a metodologia e os componentes utilizados para a implementação da segunda versão, que utiliza o microcontrolador PIC16F877.

Para a edição de diagramas eletro-eletrônicos e confecção do *lay-out* da placa de circuito impressa, foi utilizado o software Eagle 4.13 na versão *freeware*. Foram projetadas e confeccionadas placas do equipamento proposto.

Existem várias ferramentas de programação para a família de microcontroladores PIC, muitas delas apoiadas em programas para PC, como o MPLAB, compatíveis com o Windows, que são ambientes para desenvolvimento de programas. Para o presente trabalho, o ambiente de desenvolvimento utilizado foi o MPLAB, da própria Microchip. A Figura 54 mostra uma tela do ambiente de desenvolvimento MPLAB.

As rotinas de programação foram escritas em linguagem 'C' e compiladas utilizando-se o

Figura 53: Fotografia da primeira versão do terminal de rede implementado no microcontrolador AT89S8252 para validação do protocolo MODBUS

compilador CCS Versão 4.038.

Para gravação do programa na memória do PIC foi construído um gravador, comandado pela porta serial baseado no modelo JDM, largamente difundido pela internet por robistas. A Figura 55 mostra o diagrama esquemático do gravador JDM. O gravador foi construído e funcionou corretamente na tarefa de programação.

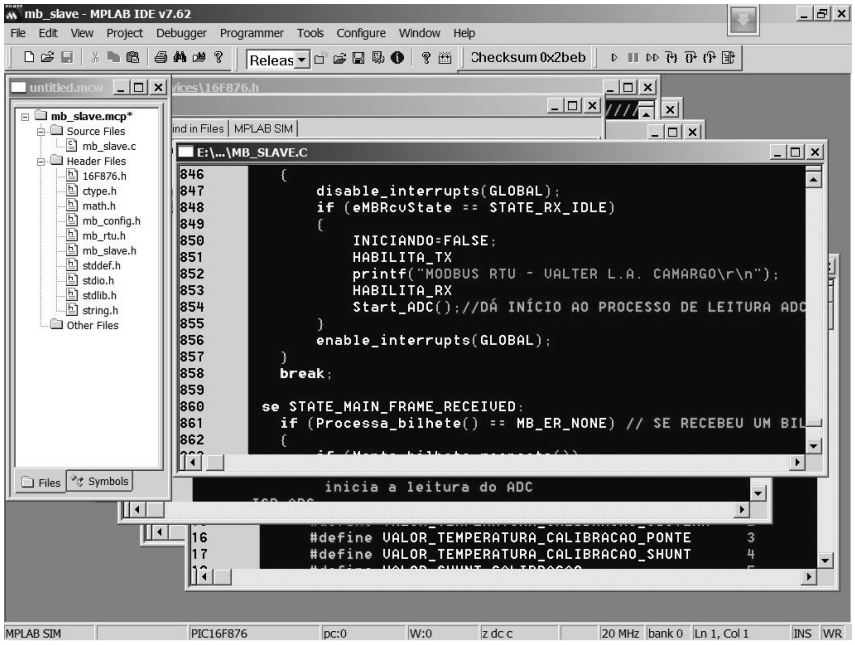

Figura 54: Tela ambiente de desenvolvimento MPLAB (MICROCHIP)

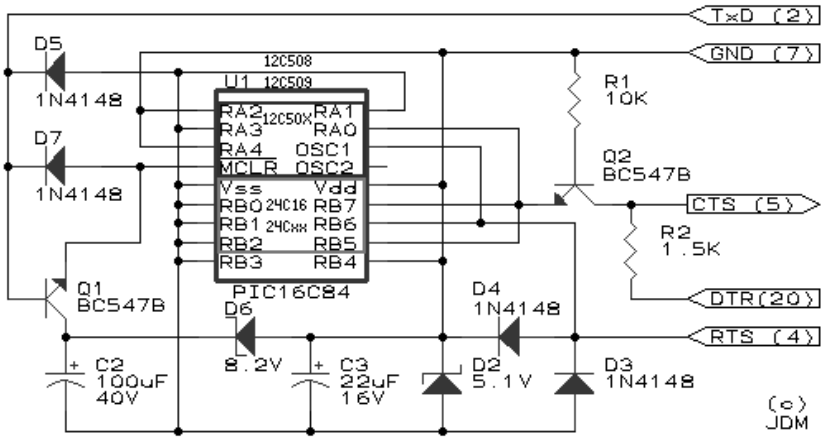

Figura 55: Diagrama esquemático do gravador JDM.

A Figura 56 mostra o diagrama da placa de circuito impresso produzida.

Finalmente, a Figura 57 mostra uma fotografia do protótipo já implementado fisicamente.

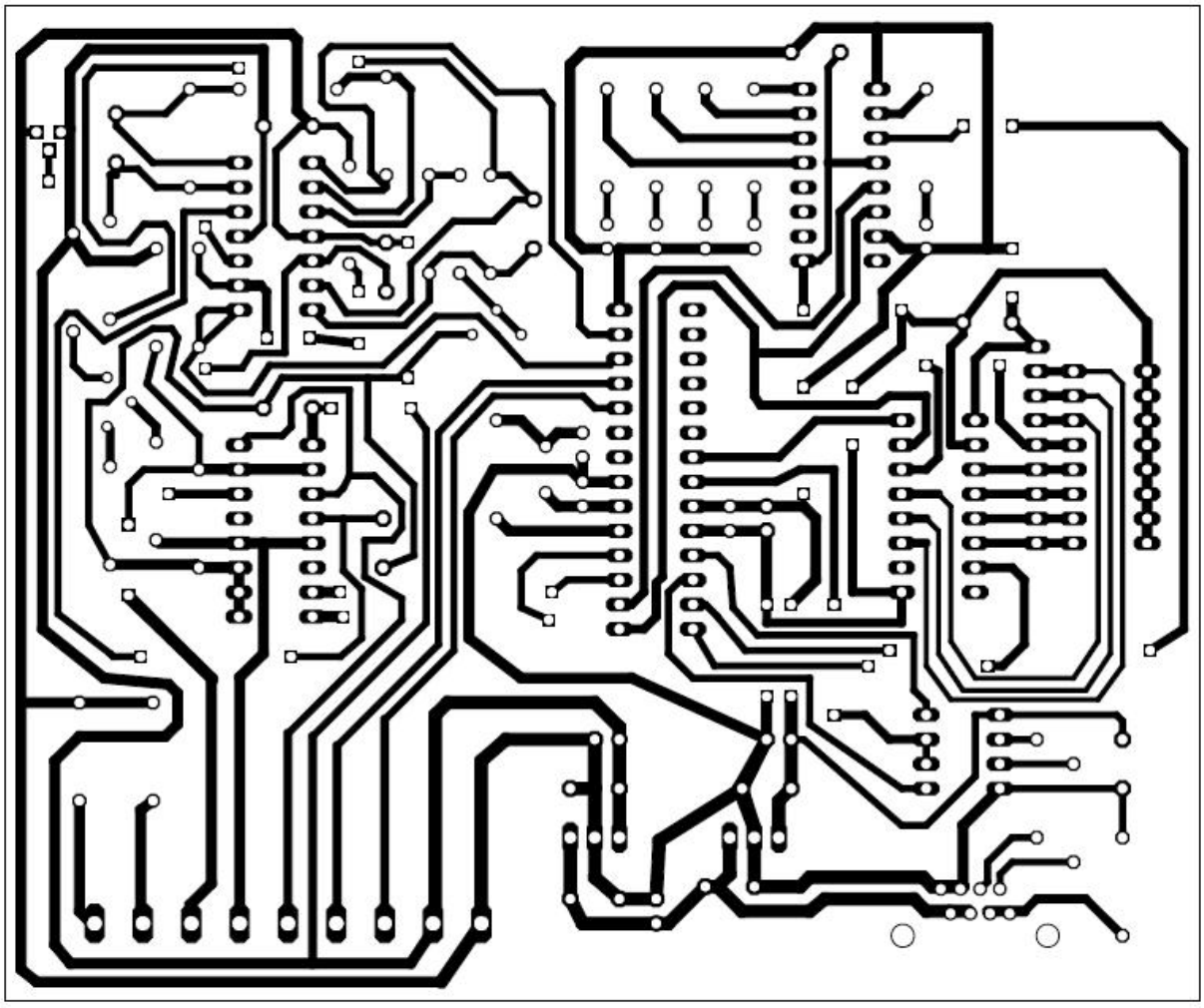

Figura 56: Placa de Circuito Impresso do terminal de aquisição

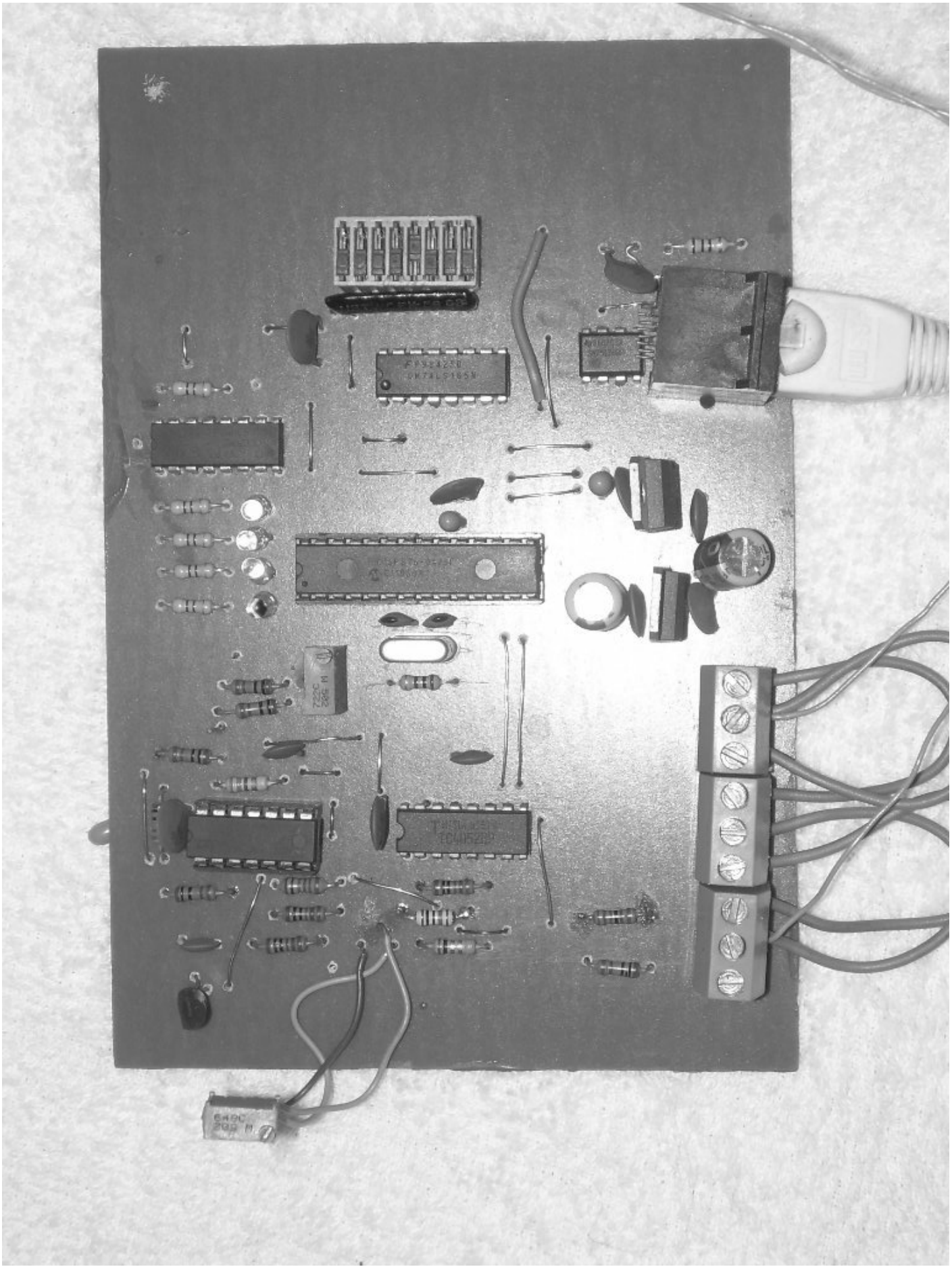

Figura 57: Fotografia da segunda versão do terminal de rede implementado em PIC 16F876 para validação do sistema

6 DESENVOLVIMENTO DO SISTEMA DE AQUISIÇÃO DE DADOS

O objetivo deste capítulo é mostrar a modelagem e a implementação do sistema de aquisição centralizado, responsável pela coleta dos dados e de seu tratamento. Como já mencionado anteriormente, para a comunicação entre a rede de sensores e o microcomputador, fez-se uso no presente trabalho do protocolo Modbus-RTU, utilizando como meio físico de transmissão um barramento serial que utiliza a interface RS485. Lembrando que os microcomputadores normalmente possuem uma interface serial RS232 nativa e que esta deve ser convertida numa outra do tipo RS485, através de um conversor de interface, para poder realizar a comunicação com a rede. Nas seções que se seguem serão apresentados os fundamentos de cada uma dessas interfaces e, na seqüência, será apresentada a solução aqui desenvolvida para fazer esta conversão.

Este capítulo também trata do software de aquisição, o qual foi desenvolvido utilizando um sistema supervisório - elemento bastante comum em ambientes industriais - principalmente pela sua facilidade e forma ágil de lidar com informações de tempo real e de tratá-las graficamente. Também foi utilizado um gerenciador de protocolo que funciona como mestre MODBUS. Para que o supervisório acesse os dados do mestre MODBUS é necessário estabelecer um método de troca de informações entre eles. Para essa tarefa, foi utilizada a tecnologia OPC. Desta maneira, a comunicação entre o mestre MODBUS e o sistema supervisório é feita de forma bidirecional, ou seja, os dados adquiridos do sistema conectado em rede são disponibilizados para o sistema supervisório, tratados pela aplicação em níveis mais elevados, e devolvidos ao sistema via OPC. Uma breve descrição dos sistemas supervisórios e da tecnologia OPC é apresentada para demonstrar os benefícios advindos do uso de ambos. Por fim, são apresentados os passos necessários para a configuração dos dois softwares, visando o estabelecimento da troca de dados entre estes.

6.1 INTERFACE DE COMUNICAÇÃO

6.1.1 TRANSMISSÃO SERIAL

Segundo Ferreira (2007), os padrões RS232 e RS485, foram desenvolvidos pela *Electronic Industry Association* (EIA) para permitir a comunicação entre os periféricos e o computador independentemente do fabricante. Estes dois padrões são destinados à comunicação serial assíncrona. Isto significa que cada bit é transmitido usando um bit de início, sete ou oito bits de dados, um bit de parada, e opcionalmente, um bit de paridade. O dado é auto-sincronizado, isto quer dizer que uma vez que o transmissor e o receptor concordam com taxa de transmissão (*baud rate*), o tempo dos bits individuais é baseado somente nos bit de início (*start bit*) e de parada (*stop bit*) .

Existem outros padrões alternativos, tais como a comunicação síncrona e a comunicação paralela. A comunicação síncrona requer uma linha de sincronismo entre o transmissor e o receptor. A comunicação paralela também requer uma linha de sincronismo de sinais de controle e muitas linhas de dados. Utilizando-se a comunicação síncrona ou a paralela podem ser alcançadas maiores velocidades de transferências de dados do que com a assíncrona, mas são requeridos mais fios de comunicação .

Os padrões RS232 e RS485 especificam as características do hardware do sistema de comunicação tais como níveis de tensão elétrica, resistências terminais, comprimento do fio, entre outros. A comunicação serial pode ser simplex (*half-duplex)* ou duplex *(full-duplex*). Na simplex, em um determinado instante, somente a transmissão ou a recepção pode ser feita. No modo duplex tanto a transmissão quanto a recepção são feitas ao mesmo tempo .

6.1.2 DESCRIÇÃO DO BARRAMENTO RS232

Este é um dos tipos mais populares de padrão de interface serial. Seu verdadeiro nome é EIA-TIA-232-E. Foi desenvolvida pela *Electronic Industry Association* e a *Telecomunication Industry Association* (EIA-TIA) em 1962 e, popularmente é conhecido como RS232 (o termo RS vem de *Recomended Standard*). Além disso, este padrão já foi atualizado 5 vezes com o objetivo de elevar o seu desempenho (FERREIRA, 2007).

O nome oficial da interface RS232 é *Interface Between Data Terminal Equipment (DTE) and Data Circuit-Termination Equipment (DCE) Employing Serial Binary Data Interchange*.

Este tipo de interface garante a compatibilidade entre o computador servidor e os sistemas periféricos quanto a:

1. Tensão elétrica e níveis de sinais.
2. Configuração dos pinos dos conectores de interligação entre ambos os lados.
3. Mínima quantidade de informação de controle entre o servidor e os sistemas periféricos.

Com relação às tensões e níveis de sinais, estes são:

- Nível lógico alto : -3V a -15V (com carga) e -25V (sem carga, no máximo);
- Nível lógico baixo : +3V a +15V (com carga) e +25V (sem carga, no máximo);
- O nível lógico da região entre -3V e +3VC é indefinido.

As especificações elétricas incluem especificações de nível de tensão, taxa de troca de sinais e impedância da linha de comunicação. Como a interface RS232 foi definida em 1962, antes da lógica TTL, não é nenhuma surpresa que este padrão não utilize +5 Volt e terra (gnd) como níveis lógicos. Os 8 bits de dados transmitidos através da interface RS232 são acompanhados por mais dois bits: Um de início (*start bit*) e outro de parada (*stop bit*). Cada caractere começa com um bit de início (neste caso a a linha vai para o sinal lógico "1" exatamente por um período de um bit. O bit menos significativo é enviado em seguida. Para se delimitar o final da transmissão é utilizado o bit de parada logo após o último bit do caractere que está sendo enviado. Também, opcionalmente, a transmissão pode ser feita considerando a paridade (uma das formas para verificação de erro, relacionada com a probabilidade de troca de bits indevidamente durante a transmissão), neste caso o bit de paridade é inserido na continuação do último bit do caractere transmitido e antes do bit de parada.

A interface RS232 também limita a taxa de subida máxima (*slew rate*) no *drive* de saída. Esta limitação foi incluída para ajudar a reduzir a possibilidade de acoplamento capacitivo (*cross-talk*) entre sinais adjacentes. Se os tempos de subida e de descida são lentos, menor é a possibilidade de se ter *cross-talk*, mas isto implica num comprometimento da velocidade de transmissão que pode ser alcançada. O máximo *slew rate* permitido é $30V/\mu s$, o que limita a taxa de transmissão de dados. A impedância da interface entre o *drive* e o receptor está bem definida. A carga vista pelo *drive* deve estar entre 3 $k\Omega$ e 7 $k\Omega$. Também está muito bem definido o comprimento do fio de comunicação, parâmetros estreitamente relacionado com a máxima carga capacitiva, que é da ordem de 2500 pF (FERREIRA, 2007).

6.1.3 DESCRIÇÃO DOS SINAIS DO BARRAMENTO RS232

Para a descrição a seguir serão utilizados os termos DTE (*Data Terminal Equipment*) que significa o lado do servidor de comunicação, normalmente um computador e o DCE (*Data Circuit-Termination Equipment*) que normalmente é um equipamento dispositivo periférico, como por exemplo, um modem.

O padrão RS232, possui as seguintes linhas de sinais:

a) *Transmitted Data* (TD): É a linha por onde os dados são transmitidos do DTE e recebidos pelo DCE.

b) *Received Data* (RD): É a linha por onde os dados são transmitidos do DCE e recebidos pelo DTE.

c) *Request to Send* (RTS): Quando o DTE está pronto para transmitir para o DCE o sinal RTS é levado para o nível lógico um. Nos sistemas simplex ou duplex esta condição mantém o DCE em modo de recepção e inabilita o modo de transmissão. A condição de zero lógico do RTS mantém o DCE em modo de transmissão. Após o sinal RTS ser ativado, o DCE deve ativar seu sinal CTS antes do começo da comunicação.

d) *Clear to Send* (CTS): CTS é usado conjuntamente com RTS para oferecer o *handshake* entre o DTE e DCE. Após o DCE ter detectado a ativação da linha RTS, ele vai ativar a linha CTS quando estiver pronto para a comunicação.

e) *Data Set Ready* (DSR): Este sinal é ativado pelo DCE para indicar que está conectado à linha de telecomunicações.

f) *Data Carrier Detect* (DCD): Este sinal é ativado pelo DCE quando está recebendo o sinal de um DCE remoto. Este sinal permanece ativado durante o tempo necessário para a detecção do carrier do sinal.

g) *Data Terminal Ready* (DTR): DTR indica o estado do DTE. Este sinal é ativado quando o DTE está pronto para transmitir ou receber dados. O sinal DTR deve ser ativado antes que o DCE ative a linha DSR.

h) *Ring Indicator* (RI): Quando o sinal RI é ativado indica que uma sinal de chamada (*ring*) está sendo recebido pelo canal de comunicação.

A Figura 58 mostra um resumo do que foi comentado nos parágrafos anteriores.

Função RS232C	Pinos DB9	Pinos DB25	Direção de fluxo	
			Computador (DTE)	Modem (DCE)
CD (Carrier detect)	1	8	entrada	saída
RXD	2	3	entrada	saída
TXD	3	2	saída	entrada
DTR (Data Terminal Ready)	4	20	saída	entrada
GND	5	7	----	----
DSR (Data Set Ready)	6	6	entrada	saída
RTS (Request to send)	7	4	saída	entrada
CTS (Clear to Send)	8	5	entrada	saída
RI (Ring Indicator)	9	22	entrada	saída

Figura 58: Pinagem dos conectores DB9 e DB25 na norma RS232

6.1.4 LIMITAÇÕES DO RS232

De acordo com Ferreira (2007), o barramento RS232 apresenta algumas limitações, tais como:

a) A interface RS232 não usa nível convencional de tensão elétrica (TTL/CMOS). Isto implica na necessidade do uso de fontes de alimentação adicionais para atingir os níveis de tensão da interface.

b) A máxima taxa de dados (velocidade de transmissão) é de 115 kbps, a qual é muito baixa para as aplicações atuais.

c) Máximo comprimento do fio de comunicação: Este parâmetro está ligado à máxima especificação da capacitância de carga. Tipicamente a máxima distância entre DTE e DCE não podem ser superiores a 20 metros, mesmo para as velocidades mais baixas. Para velocidades maiores esta distância fica limitada a dois metros ou menos, dependendo do cabo utilizado e das condições ambientais relativas à interferências eletromagnéticas.

6.1.5 DESCRIÇÃO DO BARRAMENTO RS485

Segundo Tristão (2004), o padrão RS-485 é a tecnologia de transmissão mais freqüentemente encontrada nas redes industriais. Sua aplicação está presente em todas as áreas nas quais uma alta taxa de transmissão aliada a uma instalação simples e barata são necessárias. Um par trançado de cobre blindado é o suficiente neste caso.

A tecnologia de transmissão RS-485 é muito fácil de manusear. O uso de par trançado não requer nenhum conhecimento ou habilidade especial. Taxas de transmissão entre 9,6 kbps e 12

Mbps podem ser selecionadas, porém uma única taxa de transmissão é selecionada para todos dispositivos no barramento, quando o sistema é inicializado.

A norma RS485 define um esquema de transmissão de dados, utilizando circuitos balanceados, que oferecem soluções robustas para transmitir dados em longas distâncias em ambientes ruidosos. É adotada como camada física para a comunicação dos dados de diversos protocolos, como, por exemplo, PROFIBUS-DP, MODBUS, INTERBUS e muitos outros. Por isso, existe uma grande disseminação do padrão devido às vantagens que o mesmo apresenta em ambientes industriais.

6.1.6 TIA/EIA-485

A norma TIA/EIA-485, conhecida popularmente como RS485, descreve uma interface de comunicação operando em linhas diferenciais capaz de se comunicar com 32 “unidades de carga”. Normalmente, um dispositivo transmissor/receptor corresponde a uma “unidade de carga”, o que faz com que seja possível que até 32 dispositivos possam se comunicar entre si. Entretanto, existem dispositivos que consomem frações de unidade de carga, o que aumenta o máximo número de dispositivos a serem interligados. O meio físico mais utilizado é um par trançado. Através deste único par de fios, cada dispositivo transmite e recebe dados. Cada dispositivo aciona o seu transmissor apenas no instante que necessita transmitir, mantendo-o desligado no resto do tempo de modo a permitir que outros dispositivos transmitam dados, o que caracteriza esta rede como simplex.

6.1.7 LINHAS DE COMUNICAÇÃO BALANCEADAS

O padrão RS485 se caracteriza pela utilização de um meio de comunicação diferencial (também conhecida como transmissão balanceada), utilizando um par de fios trançados. Os circuitos transmissores e receptores adotados nestas interfaces utilizam como informação a diferença de potencial entre os condutores do par trançado. Os códigos binários são identificados pela polaridade (+ ou -) da diferença de tensão entre os condutores do par, ou seja, quando a tensão no condutor “+” for maior que no condutor “-”, é caracterizado um nível lógico “1”; quando, ao contrário, a tensão no condutor “-” for maior que no condutor “+”, é caracterizado um nível lógico “0”. Uma margem de ruído de ± 200 mV é definida para aumentar a tolerância a interferências. Esta técnica resulta no cancelamento de ruídos induzidos no meio de transmissão, pois se o mesmo ruído é induzido nos 2 condutores, a diferença de tensão entre eles não se altera e a informação é preservada. A interferência eletromagnética emitida por um barramento de comunicação diferencial é também menor que a emitida por barramentos de comunicação

não-diferenciais.

6.1.8 RESISTORES DE TERMINAÇÃO

A teoria das linhas de transmissão recomenda a necessidade de terminação de linhas de comunicação com um valor de impedância correspondente à impedância característica da linha. A correta terminação atenua reflexões que distorcem os dados transmitidos, aumentando os limites de velocidade e/ou comprimento da rede.

A impedância característica de um par trançado é de aproximadamente 120Ω, sendo este um valor adequado para o resistor de terminação a ser instalado.

Outro assunto relacionado à terminação é o que fazer com os condutores não usados em um cabo de dados. Condutores não usados poderão auto-ressonar e acoplar ruído aos condutores de dados. Se eles forem deixados abertos, eles irão ressonar em todos os tipos de freqüências; se forem aterrados em uma extremidade, irão ressonar em L/2 (“L” é o comprimento do cabo); se forem aterrados nas duas extremidades, irão ressonar em L/4. A melhor maneira de minimizar a energia de um condutor não utilizado é dissipá-la em forma de calor. Para tanto, deve-se colocar resistores de terminação em ambas as extremidades do condutor para o terra. Os resistores devem possuir um valor igual à impedância característica da linha, ou seja, em torno de 120 Ohms. Uma melhor alternativa é utilizar cabos em que não sobrem condutores (NOVUS, 2007).

6.1.9 LIMITES DE DISTÂNCIA E VELOCIDADE

A norma RS485 especifica um comprimento máximo de 1200 metros para os cabos. A velocidade máxima de comunicação (em bits por segundo – bps) depende de características dos equipamentos instalados, da capacitância dos cabos e dos resistores de terminação instalados. Como regra geral, quanto mais longos os cabos, menor deve ser a velocidade de comunicação.

O comprimento máximo do cabo depende da velocidade de transmissão, conforme mostra a Tabela 6.

6.1.10 NÚMERO MÁXIMO DE DISPOSITIVOS NA REDE RS485

Todos os dispositivos são ligados à uma estrutura de tipo barramento linear. A RS485 não define o número máximo de dispositivos interligados em uma rede, e sim uma série de parâmetros que podem ser utilizados para o cálculo deste limite. Alguns destes parâmetros são os seguintes:

Tabela 6: Distâncias máximas de acordo com a velocidade utilizada

Velocidade (kbps)	Distância segmento (m)
9,6	1200
19,2	1200
93,75	1200
187,5	1000
500	400
1.500	200
12.000	100

Fonte: TRISTÃO (2004)

- Limite inferior para a resistência de carga resultante no barramento.
- Valor de resistência que cada dispositivo da rede representa no barramento, denominada "Carga Unitária" ($15k\Omega$).
- Valor mínimo de corrente que o *driver* (transmissor) de um dispositivo RS485 deve ser capaz de fornecer.

A partir destes dados e considerando a necessidade de resistores de terminação nos dois extremos do barramento (correspondentes a 60Ω), pode ser calculado o limite de 32 dispositivos com carga unitária para um barramento de comunicação RS485. Assim, um total máximo de 32 estações podem ser conectadas a um único segmento. Para assegurar uma operação livre de erros, o barramento deve ter um terminador (elemento responsável pelo casamento de impedância do barramento) no início e fim de cada segmento. No caso em que se necessite conectar de mais de 32 estações ou no caso que a distância total entre as estações ultrapasse um determinado limite, devem ser utilizados repetidores para se interconectar diferentes segmentos do barramento. A Figura 59 ilustra como devem ser conectados os repetidores.

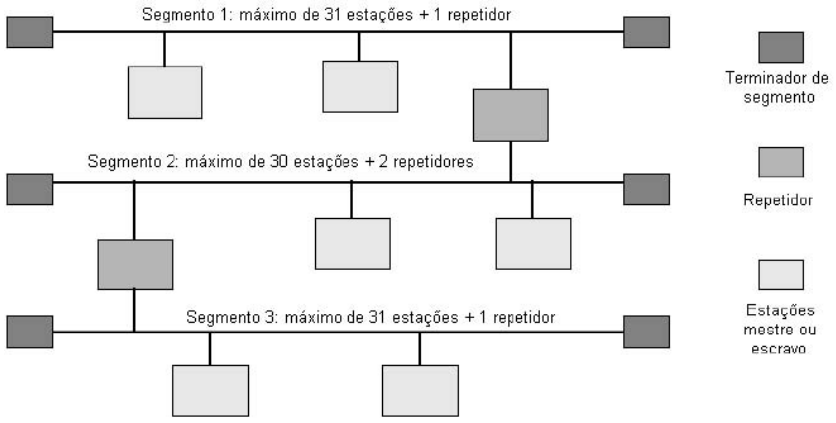

Figura 59: Utilização de repetidores (TRISTÃO, 2004).

Atualmente são comercialmente disponíveis equipamentos RS485 com carga inferior à unitária, sendo usuais os valores de 1/2, 1/4 e 1/8 da carga unitária. Para ampliar o número de dispositivos de uma rede RS485 para 256, uma solução possível é utilizar apenas dispositivos com 1/8 da carga unitária.

Em aplicações menores, onde o comprimento dos cabos da rede é pequeno e/ou a velocidade de comunicação é baixa, pode ser possível eliminar os resistores de terminação. Isto permite aumentar a capacidade de dispositivos da rede de 32 para 282. A desvantagem óbvia é que a operação confiável, nesta condição, não pode ser garantida.

6.1.11 ATERRAMENTO / INTERLIGAÇÃO DO COMUM

De acordo com Novus (2007) este é talvez o tópico menos compreendido e que causa maiores problemas na instalação de redes RS485. Linhas de transmissão diferenciais utilizam como informação apenas a diferença de potencial existente entre os 2 condutores do par trançado, independente da diferença de potencial que eles apresentam em relação ao referencial de tensão (comum ou terra). Isto permite que múltiplos sistemas se comuniquem mesmo que uma referência de potencial comum entre eles não seja estabelecida.

No entanto, os circuitos eletrônicos de transmissão e recepção podem ser danificados se o par trançado apresentar um potencial excessivamente elevado em relação ao referencial (comum ou terra). A norma TIA/EIA-485 especifica que a máxima diferença de potencial entre os equipamentos da rede deve estar entre –7V e +12V. Diferenças de potencial acima destes limites são usuais quando múltiplos dispositivos isolados eletricamente entre si são interligados apenas pelos pares diferenciais de comunicação.

A utilização de aterramento nos dispositivos, apesar de ajudar, não soluciona o problema em todas as situações, pois em uma instalação industrial típica a diferença de potencial entre aterramentos de locais afastados pode ser de muitos volts, podendo chegar a centenas de volts na ocorrência de descargas atmosféricas. A melhor solução para evitar a queima dos circuitos de comunicação é adotar um condutor adicional que interligue o comum (ou terra) de todos os dispositivos da rede.

6.1.12 CABOS DE LIGAÇÃO

Os cabos de ligação representam a ligação física entre os transdutores e os sensores até aos condicionadores de sinais e/ou equipamentos de aquisição de dados. Os cabos de ligação, quando o condicionador de sinal e/ou o sistema de aquisição é fisicamente afastado do com-

putador, também fornecem a ligação física entre estes equipamentos e o computador. Nestas situações estes cabos são comumente designados como cabos de comunicação tal como acontece na comunicação RS-232, RS-485, USB, etc.

Em muitos dos sistemas de aquisição de dados, os cabos de ligação e comunicação representam o maior componente de todo o sistema, podendo tornar o sistema sensível a ruído externo. Este componente passivo dos sistemas de aquisição é muitas vezes negligenciado durante o desenvolvimento dos sistemas, tornando-se uma importante fonte de erro e incerteza.

A utilização de cabo blindado é recomendada sempre que o custo mais elevado deste tipo de cabo não for um problema. A utilização de cabo blindado com a malha adequadamente aterrada torna a rede mais imune a interferências externas mesmo quando o cabo é instalado próximo a fontes de ruído elétrico, como inversores de freqüência, máquinas de solda, chaves eletromagnéticas e condutores de alimentação CA.

Para reduzir custos, pode ser utilizado cabo trançado sem malha de blindagem, mas este deve ser instalado separado de condutores de alimentação CA e distante de fontes de ruído elétrico.

Para a ligação dos barramentos de comunicação entre os dispositivos da rede, deve-se utilizar cabo tipo par trançado, tendo o cuidado de interconectar os terminais 'Comum' de todos os dispositivos da rede. A bitola mínima recomendada para os condutores de comunicação é 24 AWG $(0,2mm^2)$.

A adoção de um condutor adicional para interligação do comum de todos os dispositivos da rede é altamente recomendada para se evitar a queima dos circuitos de comunicação de um ou mais equipamentos da rede.

6.2 CONVERSOR RS232/RS485

Sendo a norma RS232 a implementação padrão para comunicação serial nos PC's, tem-se que fazer a conversão dos sinais do padrão RS232 para RS485 e vice-versa para que haja a comunicação correta entre os equipamentos que trabalham em uma rede industrial RS485 e o PC. Existem hoje no mercado vários conversores para tal finalidade, que praticamente compatibilizam a comunicação serial entre os vários padrões. O padrão serial RS232, sendo desenvolvido para rede ponto-a-ponto, é orientado a conexão, ou seja, deve haver um dispositivo no outro lado da linha, pronto para receber e enviar os dados segundo este padrão. É nesse ponto que o conversor de padrão opera. Tratando-se de um conversor de RS485 para RS232, este deve ter a capacidade da simulação de um dispositivo DCE de modo que responda às solicitações do lado

RS232 para efetuar a conexão e a passagem dos dados para a RS485. Como a RS485 não é um padrão orientado à conexão, a qualquer hora, qualquer equipamento poderá transmitir seus dados na rede. É neste ponto que entra a figura do mestre da rede para coordenar esse tráfego (CUNHA, 2000).

A comunicação serial do padrão RS485 funciona de modo análogo ao de uma porta serial de microcomputador (RS-232). A diferença entre os dois é a forma como o sinal é transmitido: de modo diferencial na RS-485 e como nível lógico na RS-232. Como o modo diferencial não utiliza o sinal de terra (GND) para produzir os níveis lógicos, diminuem-se o efeito de ruídos externos. Como vantagem, pode-se ter um comprimento maior de cabos.

Outra diferença está na capacidade da RS-485 compor barramentos, permitindo a interligação de dispositivos em rede, enquanto que a RS-232 foi projetado para ligar somente dois terminais no modo ponto a ponto. Protocolos do tipo RS-485 e RS-232 transmitem até 8 bits de dados por quadro e têm o mesmo formato de quadro usado na porta serial via UART dos microcomputadores. O sincronismo é feito quando se detecta o início do quadro, sem a necessidade de se transmitir um sinal de *clock* para sincronizar a transmissão dos bits. Os protocolos PROFIBUS e MODBUS utilizam-se da camada física RS-485 e acrescentam outras informações em cada quadro a ser transmitido, tal como verificação de erro, e utilizam mecanismos mais complexos para manter a integridade dos dados.

6.2.1 CONVERSOR RS232/RS485 DESENVOLVIDO

Primeiramente para o projeto deste conversor foi considerado uma questão de segurança. É sabido que a diferença de potencial entre os terminais da rede e a fonte do computador podem causar sérios problemas, podendo chegar até mesmo a danificar a porta serial. Um outro problema que pode surgir são os *loops* de corrente de terra, que provocam ruídos indesejáveis na comunicação. Portanto, foi definido que um dos requisitos para a interface de comunicação é que esta deveria ser isolada galvanicamente. Para tanto, foram utilizados foto-acopladores nos circuitos de transmissão e recepção.

Na fase de testes da primeira versão foram detectados alguns problemas. O primeiro deles é que o *handshake* RTS/CTS, sob o sistema operacional Windows, não obedece rigidamente aos tempos especificados pela norma RS232. Assim, sob certas circunstâncias - vários programas em execução simultânea, por exemplo - pode não funcionar adequadamente.

Como não é desejável utilizar o protocolo RTS/CTS, pelos motivos apresentados anteriormente, e considerando que o meio de transmissão é simplex, ou seja, somente um dos lados

pode transmitir a cada instante, foi necessário a inclusão de um elemento gerenciador do sentido de comunicação, ou seja, um elemento responsável para determinar qual lado pode transmitir em um determinado instante. Para essa tarefa foi utilizado um microcontrolador da Atmel, AT89C1051 escolhido por seu baixo custo e alto desempenho.

Outra questão que surgiu, é que era necessário então, prover uma tensão positiva de aproximadamente 9 Volts para os circuitos de recepção foto-acoplados no lado do computador já que esta não proviria da linha RTS. Para tanto, foi projetado um inversor utilizando o CI 40106 de maneira a obter uma tensão positiva de aproximadamente +9 Volts a partir de uma negativa de -9 Volts retirada das linhas TX e DSR da comunicação RS232. A freqüência de oscilação do sistema inversor foi ajustada para aproximadamente 1,5 kHz.

Para a descrição que se segue, reporte-se ao diagrama elétrico da Figura 60. A lógica de funcionamento do microcontrolador gerenciador de comunicações é a seguinte: A transmissão é simplex, ou seja, somente um dos lados pode transmitir em um dado instante. Para este caso, significa que o barramento RS485 está transmitindo para a porta serial RS232 ou o contrário.

No CI que faz a interface de comunicação (IC2 - MAX481) existem dois pinos para determinar o sentido da transmissão. Estes estão ligados ao pino 19 do microcontrolador, o qual é responsável por fazer o controle. O sentido de transmissão padrão, ou seja, aquele que é selecionado ao se ligar o sistema, é no sentido do barramento RS485 para a interface RS232.

Estando ambos os lados em repouso, quando o lado do computador (RS232) começa a transmitir, o microcontrolador detecta o bit de início do bilhete de transmissão e muda o sentido de transmissão da RS232 para a RS485. A cada byte recebido da interface RS232 é ligado um temporizador de 4 milisegundos, o que significa que após a transmissão do último byte, depois de decorrido mais 4 milisegundos, o microcontrolador volta a habilitar o sentido de transmissão no sentido do barramento RS485 para RS232, onde fica aguardando pela resposta da unidade escrava.

Para o projeto todos os diodos retificadores utilizados são do tipo *schotky* (código BAT85), os quais são recomendados para este tipo de aplicação, graças a seu baixo tempo de recuperação reversa e baixa barreira de potencial (em torno de 200 mV).

Também foram inseridos quatro LEDs para indicação da velocidade de transmissão atual e também da atividade de transmissão e recepção. Quando o microcontrolador detecta uma borda de descida, indicando o início de uma nova transmissão, é disparado um programa que se comporta como um mono-estável que vai ligar o LED por 50 mili-segundos, tempo suficiente para a percepção visual do evento.

Os foto-acopladores são do tipo 4N35, escolhidos pela relativa alta velocidade de comutação e alto fator de transferência, em torno de 0,5 vezes o valor da corrente do LED. Isto é necessário já que a corrente disponível pela interface RS232 é limitada e deve ser suficiente para o acionamento dos LEDs dos foto-acopladores.

O programa foi desenvolvido em linguagem C e compilado com o software KEIL Versão 7.2.

A gravação do microcontrolador foi feita utilizando-se um gravador universal LP-10 da Minipa.

Após a montagem da placa de circuito impresso definitiva (segunda versão - a primeira precisou ser modificada pelos motivos apresentados anteriormente) o sistema foi testado em velocidades de 4800 BPS, 9600 BPS e 19200 BPS, funcionando corretamente.

Na Figura 61 pode ser vista uma foto do conversor já construído na sua versão final.

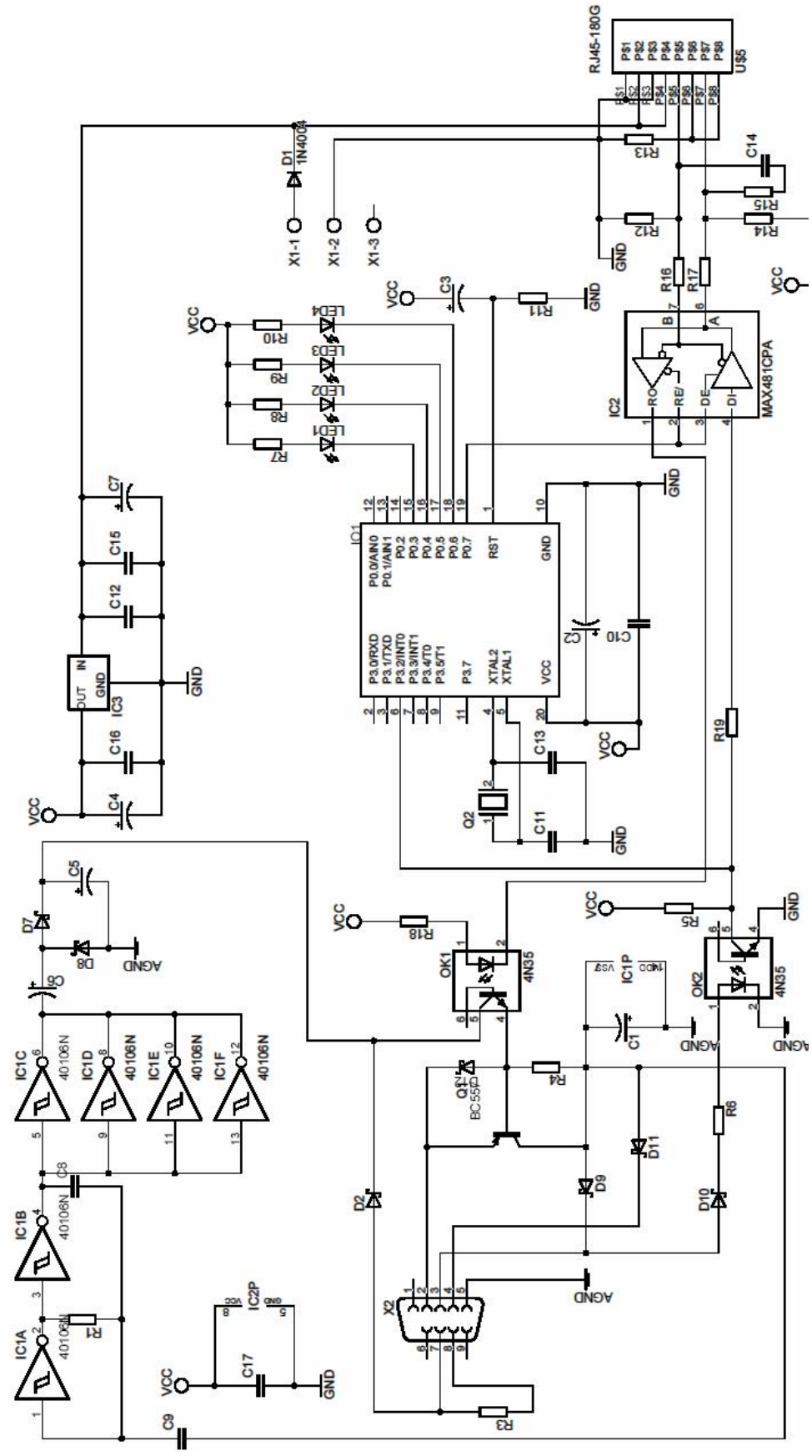

Figura 60: Diagrama elétrico do conversor RS232/RS485 implementado

Figura 61: Fotografia do Conversor RS232/RS485 Implementado

6.3 SOFTWARE SERVIDOR DE COMUNICAÇÕES (MESTRE MODBUS)

O sistema de supervisão tem acesso às variáveis da rede Modbus através de uma base de dados OPC. Mais especificamente, no sistema em questão, o supervisório tem acesso às variáveis dos elementos da rede por meio de um servidor de comunicações mestre MODBUS-RTU, onde estas variáveis podem ser: entradas analógicas ou digitais, saídas analógicas ou ainda saídas digitais.

No sistema em questão, associam-se *tags* a valores de registradores dos terminais de rede. Assim, através do servidor de comunicação serial, os valores dos registradores são passados, em tempo real, às respectivas *tags* e vice-versa. Tais *tags* podem ser ainda processadas internamente pela aplicação supervisória através da execução de scripts.

Neste projeto será utilizado como servidor OPC o produto KEPServerEx Versão 4.105 produzido pela KEPWARE Inc. e o cliente será um sistema supervisório Elipse Scada, Versão 2.29.

6.4 CRIAÇÃO DE UM NOVO CANAL NO SOFTWARE KEPServerEX

Primeiramente, é necessária a criação de um canal de comunicação serial no servidor. Para tanto, deve-se acessar o menu principal e escolher a opção `Edit→ New Channel` após o que deverá aparecer a tela mostrada na Figura 62. Para este projeto o nome escolhido foi PORTASERIAL1.

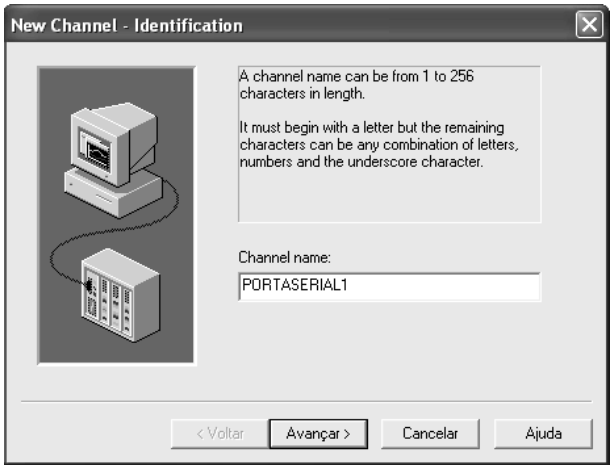

Figura 62: Criação de um novo canal de comunicação no KEPServerEx - etapa1 - Fornecimento do nome do canal

Ao se "clicar" no botão Avançar será apresentada a tela mostrada na Figura 63, onde deve-se escolher o protocolo a ser utilizado. Neste projeto o protocolo é MODBUS SERIAL.

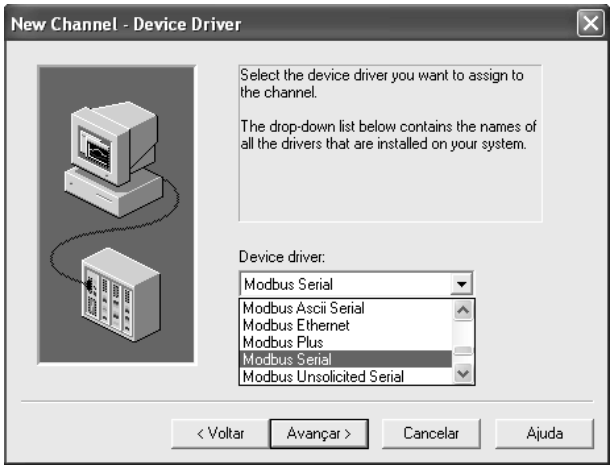

Figura 63: Criação de um novo canal de comunicação no KEPServerEx - Etapa 2 - Escolha do protocolo a ser utilizado

Ao se "clicar" no botão Avançar será apresentada a tela mostrada na Figura 64, onde devem ser informados os parâmetros de comunicação da porta serial. Para este projeto os valores são: COM1, 9600 bps, 8 bits, Nenhuma Paridade, 1 Bit de Parada e linha RTS sempre ativada.

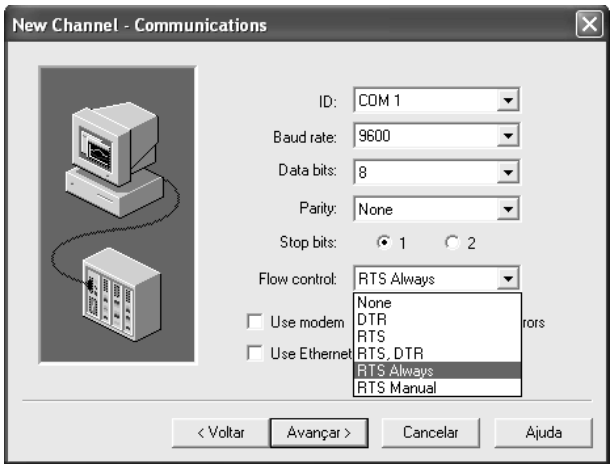

Figura 64: Criação de um novo canal de comunicação no KEPServerEx - Etapa 3 - Configuração dos parâmetros de comunicação serial

6.5 CRIAÇÃO DE UM NOVO DISPOSITIVO NO SOFTWARE KEPServerEX

Os dispositivos existentes na rede de comunicação devem ser primeiramente configurados no servidor de comunicação Modbus.

Inicialmente deve ser configurada a porta serial que irá ser utilizada para a comunicação. Neste caso, foi utilizado a porta COM1 do computador. Para se incluir um novo dispositivo, deve-se fazer um "clique" com o botão direito do mouse sobre o nome dado à porta serial, neste caso, PORTASERIAL1, quando então deve aparecer um menu sensível ao contexto, conforme é ilustrado na Figura 65. Ao ser selecionada a opção do menu `New Device,` é aberta uma seqüência de telas de auxílio de configuração. Este estilo de configuração é também conhecido pelo seu nome original em inglês: *Wizards*.

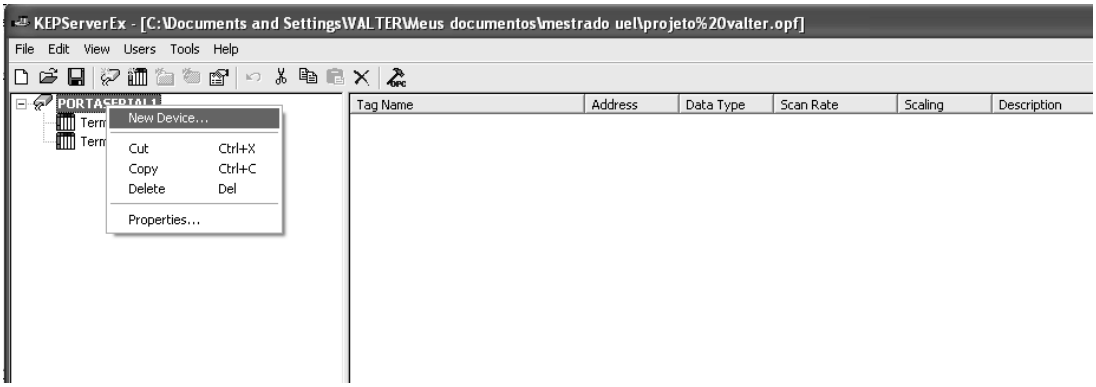

Figura 65: Criação de um novo nó na rede no software KepwareServer Modbus.

A primeira tela tem a finalidade de configurar um nome para o novo dispositivo criado. Neste caso, deseja-se configurar o terminal de número 1, cujo nome será Terminal01. Este procedimento é ilustrado na Figura 66. Após fornecido o nome deve-se "clicar" no botão avançar que chamará a próxima tela do tutor de configuração.

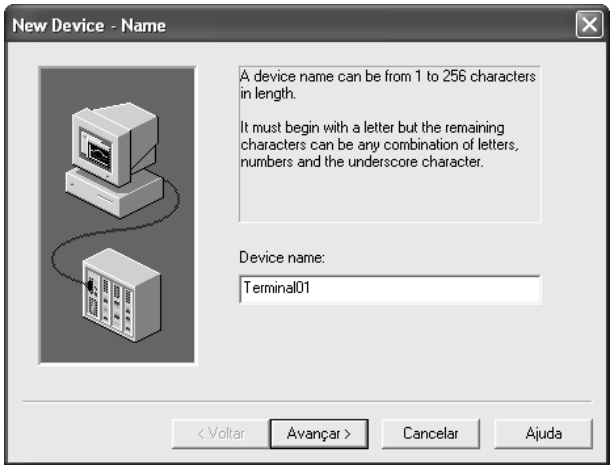

Figura 66: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 1 - fornecer o nome do dispositivo.

Na tela posterior deve ser configurado o protocolo desejado, neste caso Modbus. Isto porque este software também pode ser utilizado como servidor de comunicações para outros protocolos. A Figura 67 ilustra essa etapa.

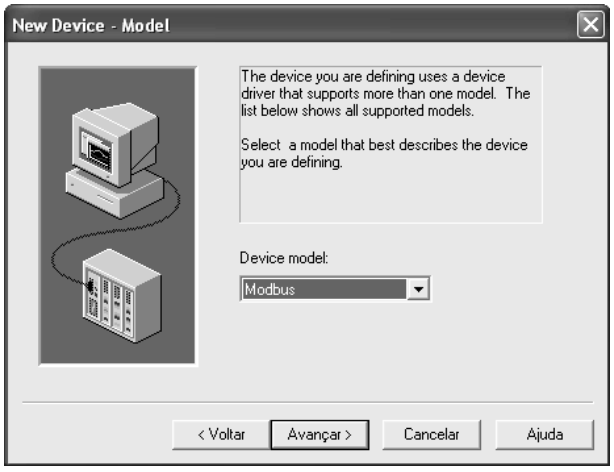

Figura 67: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 2 - fornecer o protocolo desejado.

Na seqüência, deve ser informado o endereço do dispositivo. Este pode ser fornecido no formato decimal, octal ou hexadecimal. No caso do terminal número 1 o formato é irrelevante já que em qualquer formato seu valor será o mesmo. Veja a Figura 68.

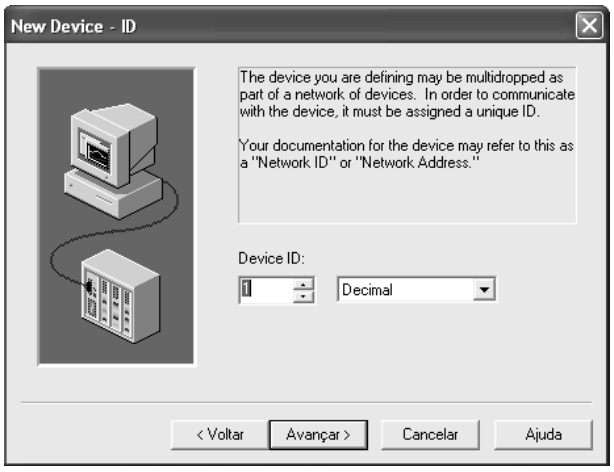

Figura 68: Criação de um novo nó na rede no software Kepware Server Modbus - etapa3 - fornecer o endereço do novo terminal a ser criado.

Na próxima tela chamada devem ser configurados os parâmetros de comunicação. Reporte-se à Figura 69 para maiores detalhes. O primeiro parâmetro (*Connect timeout*) diz respeito ao tempo de espera de resposta do escravo antes de relatar uma falha. Neste caso, o valor definido é de 3 segundos. O segundo parâmetro (*Request timeout*), é o tempo (em milisegundos) em que será solicitado uma nova resposta da estação escrava. No presente caso, este tempo foi fixado em 1 segundo. O terceiro parâmetro (*Fail after*) é o número de vezes sucessivas que o programa vai aguardar antes de considerar o dispositivo em falha.

Na seqüência, é pedido para que se defina a ação requerida em relação à criação do banco

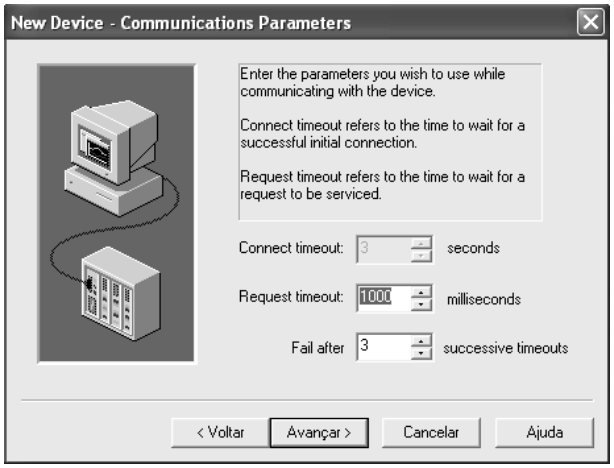

Figura 69: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 4 - parâmetros da comunicação.

de dados das *tags*, se este deve continuar a coletar informações do ponto onde parou ou se exclui os registros anteriores e recomeça novamente ao ser reiniciado o programa servidor. Isto pode ser verificado na Figura 70.

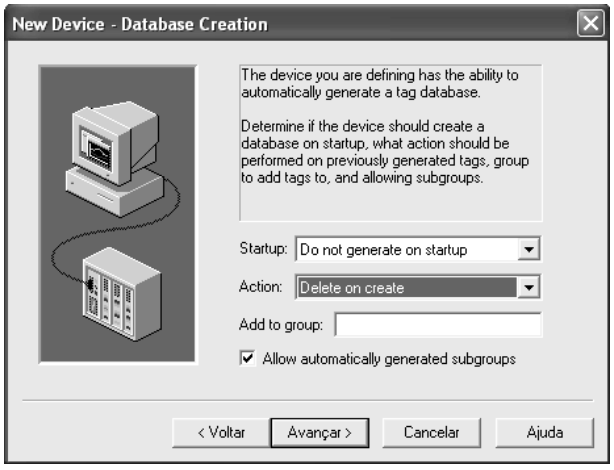

Figura 70: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 5 - banco de dados.

Na tela chamada na seqüência, devem ser configurados os parâmetros do protocolo Modbus. O primeiro item (*Use zero based addressing*) diz se o endereçamento dos dispositivos começa em zero ou em um. Isto porque podem existir dispositivos que obedecem a uma ou outra forma de numeração.

O segundo item (*First word low in 32 bit data types*) informa qual a ordem dos bytes de dados, no caso de informações de 32 bits. Lembrando que os registradores do padrão Modbus são de 16 bits, assim se for desejado obter valor de 32 bits, dois registradores devem ser utilizados. Neste caso informa-se ao sistema qual a ordem do byte menos significativo. Veja a Figura 71 para maiores detalhes.

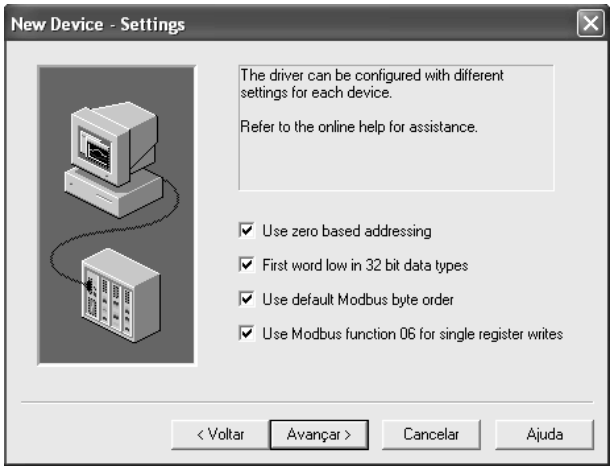

Figura 71: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 6 - configurar os parâmetros do protocolo Modbus.

A última tela diz qual o número de registradores de cada tipo existentes na unidade escrava. Para o caso das entradas e saídas digitais, o número de registradores a ser fornecido deve ser em múltiplos de 8 bits. O mesmo vale para os registradores de 16 bits (comandos 3xxxx e 4xxxxx). A Figura 72 mostra essa configuração.

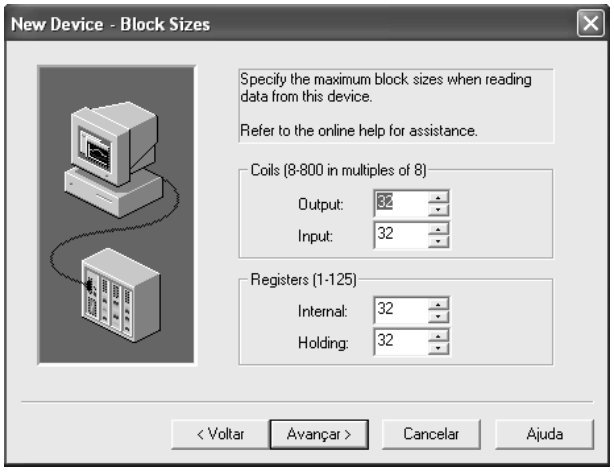

Figura 72: Criação de um novo nó na rede no software Kepware Server Modbus - etapa 7 - definição da quantidade de registradores existentes no dispositivo escravo.

Ao finalizar o tutor de configuração (*wizard*) deve aparecer o novo terminal criado na lista de dispositivos a serem monitorados pelo servidor de comunicações, conforme pode ser visto na Figura 73.

O próximo passo é criar as *tags* desejadas para o nó de rede criado. Isto consiste em se mapear os registradores internos dos escravos, estabelecendo uma relação entre os endereços físicos de seus registradores e mnemônicos responsáveis por identificar com mais facilidade a sua natureza ou função. Para criar ou alterar uma *tag* já existente, basta "clicar" sobre o texto

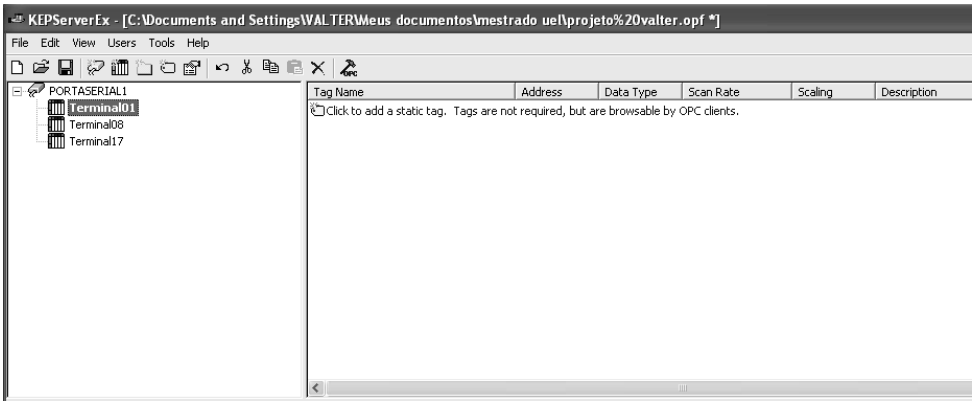

Figura 73: Criação de um novo nó na rede no software Kepware Server Modbus - etapa final - dispositivo criado com sucesso.

que aparece no painel à direita, quando então será chamada a tela mostrada na Figura 74. No caso da aplicação aqui descrita, o endereço do registrador 300002 é responsável por armazenar o fator de deformação atual do(s) extensômetro(s) conectado(s) à unidade escrava em questão, enquanto que o endereço 300001 contém a temperatura da unidade.

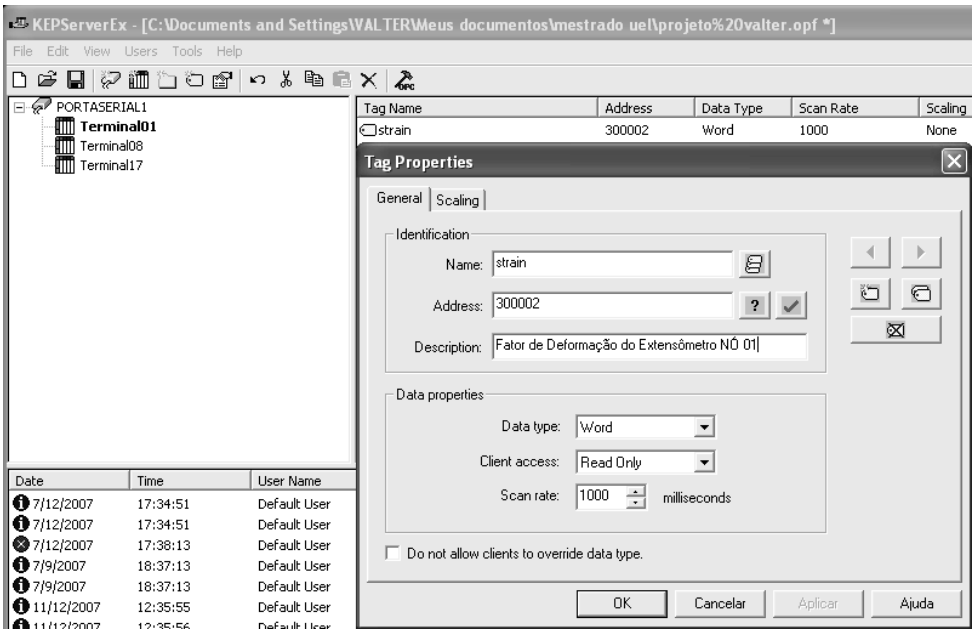

Figura 74: Criação de uma nova *tag* no software Kepware Server Modbus

6.6 CONEXÃO DO SUPERVISÓRIO COM O SERVIDOR MODBUS

Na seção anterior foi mostrado como configurar o servidor de comunicação Modbus. Nesta, serão mostrados os passos para estabelecer uma comunicação entre os *softwares* do servidor

Modbus e o do supervisório. Como já mencionado anteriormente, o software supervisório utilizado nesta aplicação é o Elipse Scada Versão 2.29. No Elipse Scada existe um módulo responsável pelo gerenciamento de todos os componentes do supervisório, chamado de *organizer*. A partir dele, deve-se selecionar o item `OPCServers` e localizar o servidor OPC, conforme mostrado na Figura 75, para fazer a ligação das variáveis do supervisório (ELIPSE SCADA) com as do servidor de comunicações OPC (KEPWARE SERVER).

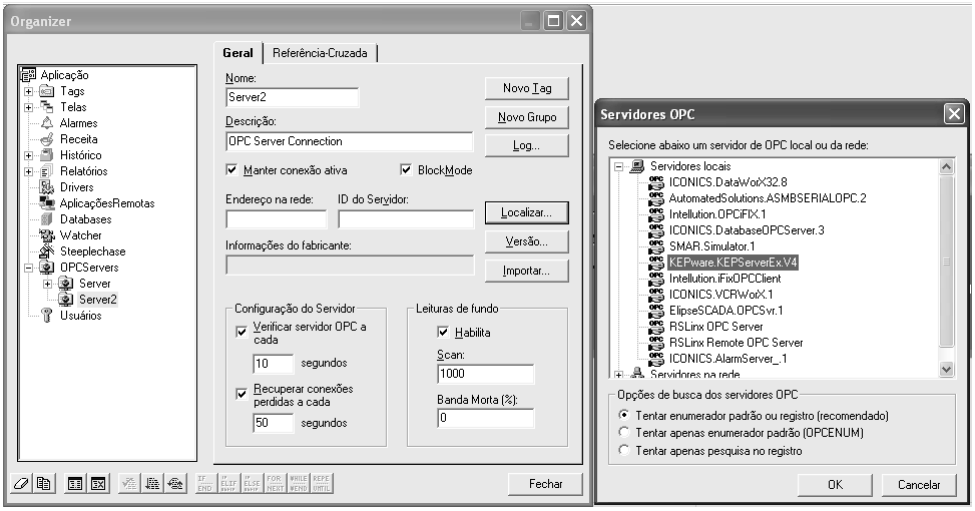

Figura 75: Seleção no Elipse Scada do servidor OPC a ser utilizado

Uma vez selecionado o servidor OPC a ser utilizado, o passo seguinte é importar as *tags* para o supervisório. Isto é feito através da tecnologia selecionar, arrastar e soltar (*select, drag and drop*) do sistema operacional Windows. O assistente de importação pode ser visualizado na Figura 76.

Na Figura 77 é mostrada a situação final com as *tags* já importadas do servidor OPC. A partir desse ponto, pode-se realizar diversas operações de leitura e/ou escrita destas variáveis, tais como associá-las à objetos gráficos de animação do supervisório, armazenar em banco de dados, gerar gráfico de tendências e/ou históricos, entre outras.

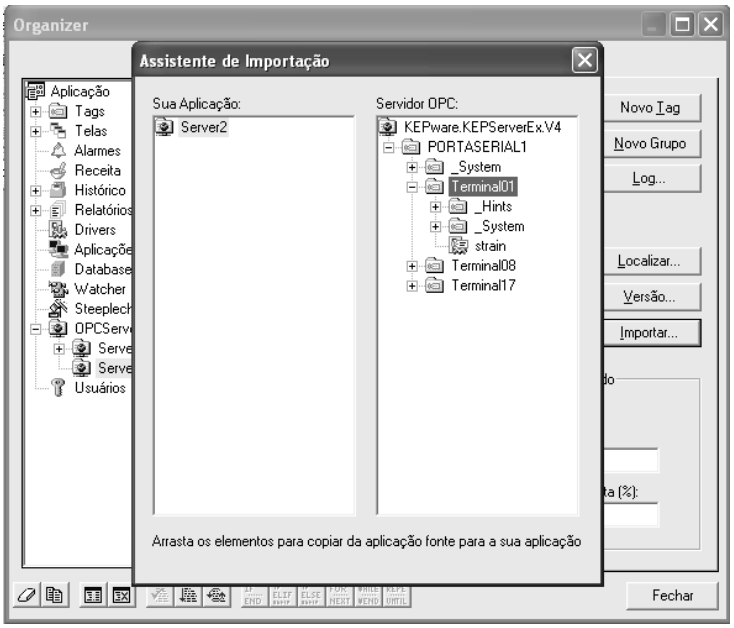

Figura 76: Assistente do Elipse Scada para importação de *tags* do servidor OPC

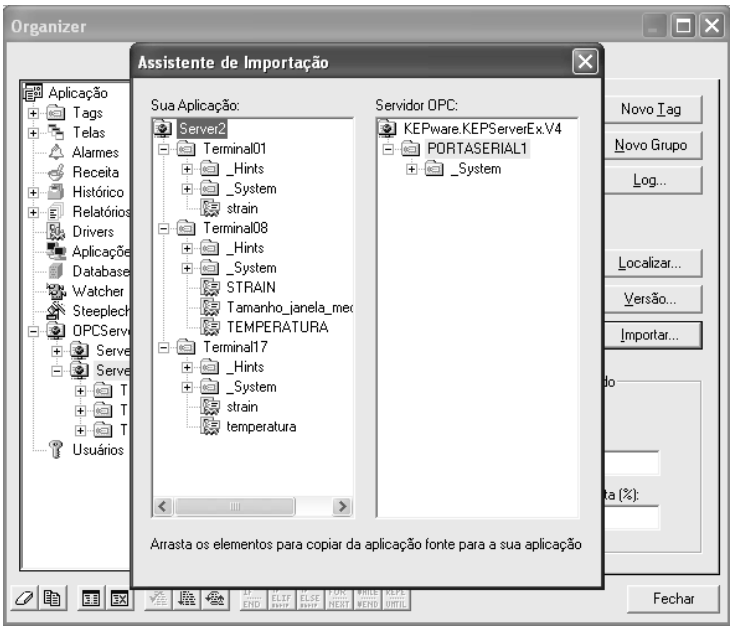

Figura 77: Tela do assistente de importação de tags com a operação já concluída.

6.7 SISTEMAS DE SUPERVISÃO E CONTROLE

Os sistemas de supervisão e controle, também conhecidos por sistemas supervisórios ou pela sua abreviação da língua inglesa SCADA (*Supervisory Control And Data Acquisition system*), são aplicativos usados em grande escala pela indústria que permitem que sejam monitoradas e rastreadas informações do processo produtivo. Estes sistemas fornecem uma interface para a tarefa de aquisição de dados em tempo real. Por meio dele, as informações podem ser visualizadas utilizando o recurso de quadros sinóticos animados com indicações instantâneas das variáveis de processo (vazão, temperatura, pressão, volume, etc.). Também é possível estabelecer ponto de operação desejado do processo (*set-point*) ou controlar elementos distantes, abrir ou fechar válvulas ou chaves, monitorar alarmes e armazenar informações de processo.

Os sistemas supervisórios podem gerenciar processos de qualquer tamanho ou natureza, sendo comum existir algumas centenas (e até mesmo alguns milhares) de pontos de entrada e saída. Desta forma, a escolha do software de supervisão é muito importante na estratégia de automação de uma empresa (ZEILMANN, 2001 - apud GONÇALVES, 2004).

Suas unidades básicas de dados são os "tags", nomes que associam um endereço ou registrador de um dispositivo ao sistema supervisório. Os dados do processo são tratados como tags, sejam eles vindos de um CLP, de um servidor de protocolos de comunicação (via OPC) ou mesmo de um banco de dados.

De acordo com Costa (1995), os sistemas supervisórios dos mais diversos fabricantes, oferecem em comum as seguintes funcionalidades:

- Um ambiente gráfico para a edição das telas de monitoração (que permite a criação de *displays*, botões, animações, gráficos com tendências, etc.);
- Gráficos de Tendências e de Históricos;
- Relatórios;
- Cálculos envolvendo funções matemáticas diversas;
- Tratamento de alarmes;
- Arquivo de dados e interação com outros bancos de dados;
- Criação de estratégias de controle;
- Manipulação de Receitas pré-definidas;

- Mecanismos de suporte à configuração das aplicações;
- Interface de comunicação com diversos tipos de redes, interfaces e protocolos;
- Importação e exportação de dados para planilhas de cálculo;
- Editor de *scripts* (pequenos trechos de programas escritos em linguagem de programação).

Normalmente, nas estações remotas existem sensores conectados aos equipamentos a serem monitorizados que convertem parâmetros físicos, tais como velocidade, nível de coluna de líquido e temperatura, para sinais digitais. O processo de aquisição de dados inicia-se nas estações remotas com a leitura dos valores atuais dos dispositivos a que estão associados. Um servidor de protocolo de comunicações é responsável por recolher estas informações e enviá-las para um sistema supervisório num único computador, ou distribuídos por uma rede de computadores de modo a permitir o compartilhamento da informação proveniente dos sensores.

Pode-se citar como exemplo de sistemas supervisórios que seguem o padrão OPC: Elipse Scada, Elipse E3, Win-cc, I-Fix, Wonderware Intouch, Process View, Iconics e Cimplicity, entre outros.

6.7.1 INTERFACEAMENTO DOS SISTEMAS SUPERVISÓRIOS

Os sistemas supervisórios precisam trocar dados com outras aplicações. Para tal tarefa podem ser utilizadas muitas tecnologias abertas, entre elas pode-se citar o ODBC (*Open DataBase Connectivity*), API (*Application Program Interface*), DDE (*Dynamic Data Exchange*) e OPC (*OLE for Process Control*). Através do uso destas tecnologias, o interfaceamento dos sistemas SCADA com outras aplicações torna-se possível.

O ODBC é uma interface de programação de aplicativos que são programados utilizando-se a linguagem SQL (*Structured Query Language*). A conexão lógica de uma base de dados ao sistema de supervisão permite uma leitura e atualização dos parâmetros de uma rede ou de um CLP, o controle de um processo ou a alteração de um relatório. Esta conexão com a base de dados é feita através de uma interface normatizada ODBC.

O desenvolvimento de API, que é um modo de implementar um software, que pode ser, por exemplo, um *gateway* responsável pelo gerenciamento da comunicação e que proporcione serviços de aplicação entre um servidor Web e um servidor SGDB (Sistema Gerenciador de Banco de Dados). As APIS são pequenos trechos de programas (*scripts*) escritos em uma linguagem de programação, como por exemplo, o Visual Basic.

Através dos DDE, os supervisórios podem exportar e importar variáveis de outro *software*, como por exemplo, um aplicativo do pacote Microsoft Office ou do Matlab, sendo que este último conta com pacotes de software específicos para o controle de processos.

O OPC é uma tecnologia disponibilizada por alguma aplicação servidora de protocolo de comunicações, como por exemplo, um servidor mestre MODBUS, que neste caso funciona como o servidor da aplicação. A interface OPC torna possível a interoperabilidade entre aplicações de automação e controle, sistemas de dispositivos de campo e aplicações situadas em níveis mais altos na hierarquia de uma planta industrial.

6.7.2 COMPUTADOR

O computador utilizado pode influenciar de modo preponderante a velocidade à qual se pretende adquirir os dados e como tal a precisão, processamento e armazenamento dos dados. Existem diversos fatores na arquitetura do computador que afetam os parâmetros referidos anteriormente, tais como o tipo de processador, as placas de expansão disponíveis (ISA ou PCI), tempo de acesso ao disco rígido, utilização de Acesso Direto à Memória (DMA), etc. Estes se tornam extremamente relevantes quando se pretendem efetuar leituras com elevadas transferência de dados, não sendo particularmente decisivos para a tarefa de medições de estruturas estáticas, como é o caso deste projeto. Assim os requisitos de capacidade de processamento do computador não é crítico, podendo ser utilizado qualquer um que obedeça as recomendações mínimas do fabricante do sistema supervisório e do gerenciador mestre Modbus, ou seja, processador Pentium III, 1 GHz velocidade de barramento e 512 MB de memória RAM.

As aplicações desenvolvidas são executadas no computador sobre o sistema operacional Windows®.

6.8 SOFTWARE DE AQUISIÇÃO DE DADOS

Com a finalidade de validar os nós de rede e também da factibilidade do conceito, foi desenvolvido um sistema de aquisição de dados baseado em computadores pessoais que fornece interação e comunicação com os nós.

O equipamento de aquisição de dados utiliza um software para coletar e visualizar os dados. No PC é instalado um software que monitora, em tempo real, os sinais provenientes da rede de sensores.

Existem diversos tipos de softwares disponíveis para efetuar aquisição de dados, desde os específicos para determinadas aplicações até plataformas de desenvolvimento de aplicações de alto nível.

Em particular, as telas para o gerenciamento, monitoramento, controle, ensaios e simulações do protótipo, foram geradas através do software supervisório Elipse SCADA, da empresa Elipse Software Ltda. Escolheu-se este produto por ser um sistema SCADA com alguns recursos de comunicação com outros programas e ainda porque o laboratório de automação industrial do CESUMAR, onde o autor trabalha, já possuía licença de uso para este. Existem muitos sistemas similares a este no mercado, mas a escolha foi baseada nos critérios anteriormente mencionados.

Este tipo de ferramenta apresenta um conjunto de objetos gráficos previamente definido, permitindo desenvolver programas para adquirir os sinais recebidos pelo equipamento de aquisição, transformando-os para as grandezas pretendidas. Também permite uma interação amigável com o usuário e disponibiliza a este diversas possibilidades para tratar os dados provenientes da rede de comunicação. Assim, é possível visualizar os dados na tela do monitor, armazená-los em memória e acionar sinais de saída para ações de controle. O desenvolvimento de programas utilizando um sistema supervisório é orientada por telas (ou janelas) que funcionam sob o formato da possibilidade de abertura destas conforme a necessidade. É possível criar várias formas de apresentação dos valores medidos, quer em termos de valor absoluto, quer em termos de gráficos ou tabelas. Serão descritos a seguir a utilidade e o funcionamento daquelas mais importantes.

6.8.1 TELA PRINCIPAL

Na Figura 78 tem-se a tela principal da aplicação supervisória desenvolvida no Elipse Scada. A partir desta é possível acessar outras com funcionalidades específicas.

A partir da tela principal, há um conjunto de botões que podem ser utilizados para chamar as seguintes telas:

- Configuração de Hardware.
- Configuração do Canal
- Carregar Configuração
- Aquisição e Visualização
- Exportar Dados
- Sair.

6.8.2 TELA HABILITAÇÃO TERMINAIS

Existe uma tela para habilitar os terminais que serão monitorados. Na Figura 79 é mostrado que estão sendo monitorados os valores dos terminais número 1, 8 e 17. Observe que a tela permite o gerenciamento de 32 sensores. O sistema pode, ainda, ser estendido para um número maior de sensores, limitados a 247 na rede MODBUS-RTU, tendo em vista que cada unidade sensora terá seu endereço próprio e diferente das demais.

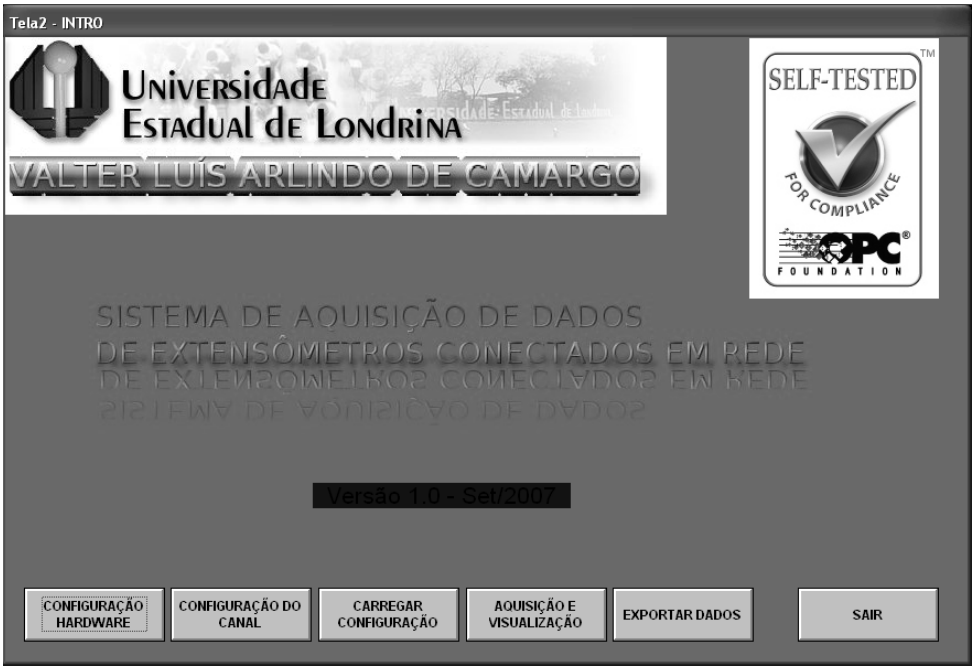

Figura 78: Tela inicial do sistema supervisório (Elipse Scada)

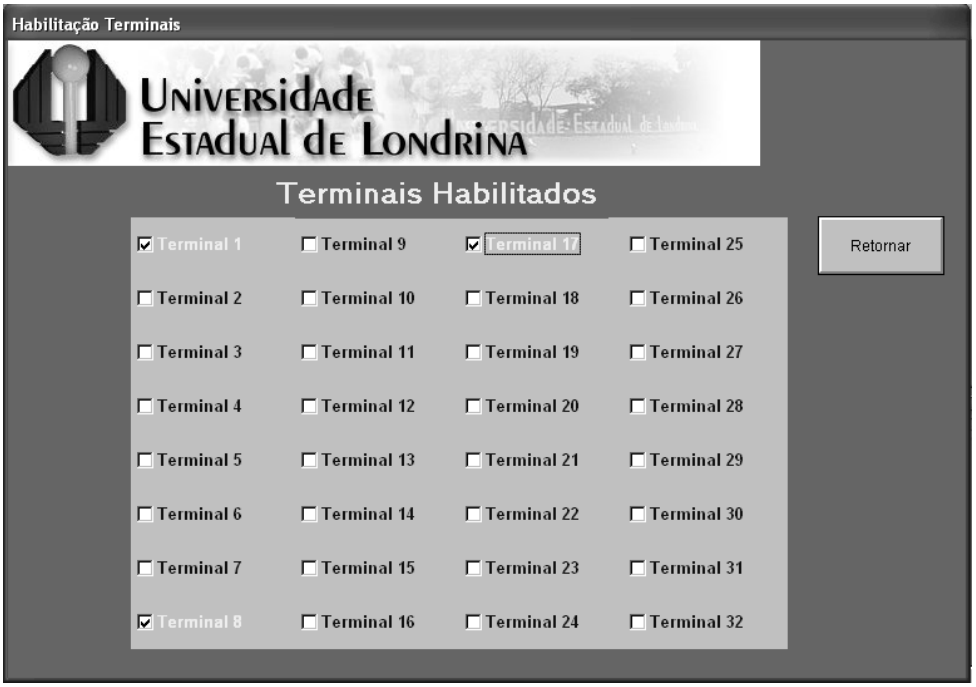

Figura 79: Tela do supervisório (Elipse Scada) para a habilitação dos terminais

6.9 TELA DE AQUISIÇÃO DOS DADOS

Nesta tela, o usuário consegue visualizar os valores da variáveis, tais como o fator de deformação e a temperatura dos sensores extensométricos de um dos nós da rede, através de gráficos

ou por meio de *displays*.

A tela de aquisição dos dados pode ser vista na Figura 80. Nesta, há um objeto do tipo *setpoint* para digitar o valor do terminal que se deseja monitorar. Um gráfico de tendência mostra os valores do fator de deformação e da temperatura a cada instante de tempo, através do qual pode-se acompanhar a realização de ensaios. Dois botões, Iniciar e Finalizar, permitem disparar e finalizar o ensaio em qualquer instante de tempo.

Figura 80: Tela inicial do sistema supervisório (Elipse Scada)

Existe um objeto do tipo tabela para se visualizar os valores numéricos das amostras sendo coletadas a cada instante.

Também é possível mudar o tamanho da janela do filtro de média móvel do nó da rede através desta tela.

6.9.1 TELA DE HISTÓRICOS

Também ao se selecionar o botão "clique aqui para visualizar graficamente os dados do histórico", abre-se a tela de visualização dos gráficos, denominada tela de Históricos. Por meio desta tela é possível gerar arquivos de dados históricos para posterior análise e carregar antigos arquivos históricos.

Na primeira aba é visualizado o histórico desejado, conforme pode ser visto na Figura 81.

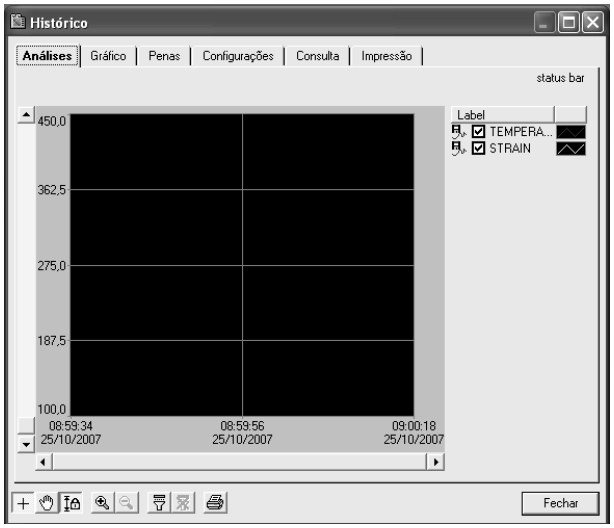

Figura 81: Tela de visualização dos dados do histórico (Elipse Scada)

Na terceira aba é visualizado as opções relativas as "penas" de marcação no gráfico. A palavra "penas" provavelmente seja inspirada nas canetas antigas. Aqui, significa o equivalente a uma caneta marcadora que desliza sobre um papel deixando marcada sua trajetória, ou seja, é utilizada para compor gráficos de evolução de uma variável ao longo do tempo. Isto pode ser visualizado na Figura 82.

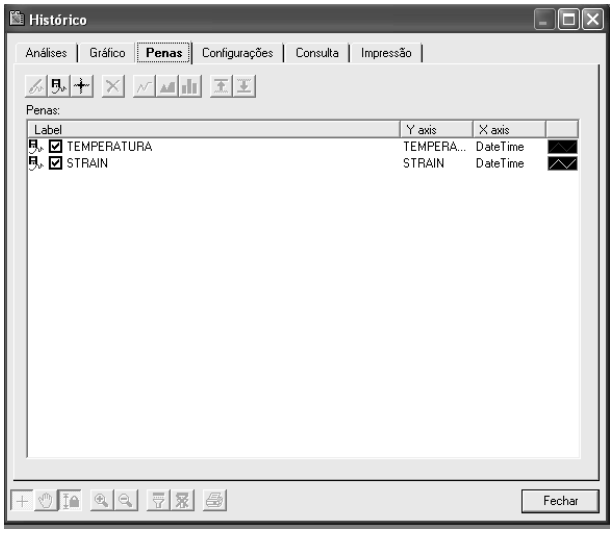

Figura 82: Tela do Elipse Scada para configuração das "penas" do histórico.

Na quarta aba (configurações) são definidos a base de dados desejada e as informações da batelada que se deseja acessar. Na Figura 83 é mostrada a tela para seleção do banco de dados a ser utilizado para armazenar os dados coletados.

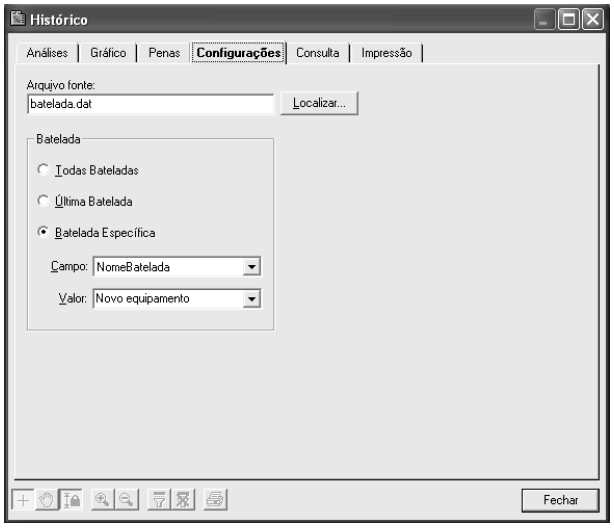

Figura 83: Tela do Elipse Scada para configuração dos dados do histórico.

Na quinta aba (consulta) podem ser configurados os critérios de consulta, por lote, por data, etc. A Figura 84 ilustra essa situação.

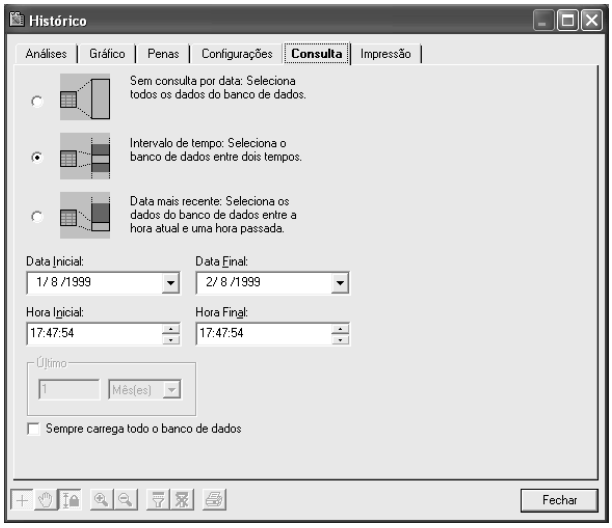

Figura 84: Tela do Elipse Scada para configuração para recuperação de dados por consulta entre datas e horas.

A sexta tela (Figura 85) configura a impressora e imprime os dados do histórico.

Diversas outras funções podem ser utilizadas. No entanto, por limitação de tamanho, só foram mostradas as principais.

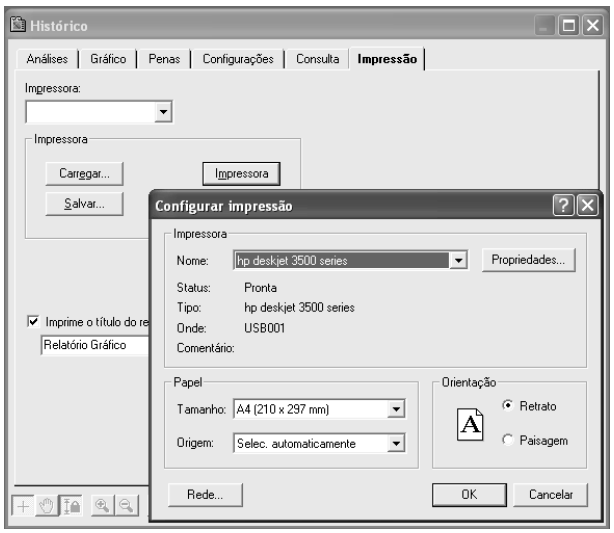

Figura 85: Tela do Elipse Scada para configurar a Impressão dos dados do histórico.

7 RESULTADOS E DISCUSSÕES

Neste capítulo são descritos os resultados obtidos com base em observações colhidas dos protótipos desenvolvidos. Segue-se uma discussão dos mesmos procurando estabelecer uma correlação dos resultados obtidos com os esperados inicialmente.

7.1 CARACTERÍSTICAS TÉCNICAS DO SISTEMA DESENVOLVIDO

- Protocolo: MODBUS-RTU.
- Interface de comunicação: RS485 par diferencial.
- Topologia: Barramento.
- Distância máxima dos nós: 1200 metros (contados a partir da unidade mestre).
- Quantidade máxima de pontos: 32 (31 terminais mais a unidade mestre).
- Velocidade: 9600 BPS;
- Paridade: Nenhuma.
- Número de Bits: 8.
- Número de bits de parada: 1.
- Número máximo de Aquisições de cada nó: 10 por segundo.
- Resolução: 10 bits (2^{10}=1024 divisões).
- Extensômetros elétricos de resistência de 120 Ω.
- Configuração dos extensômetros em 1/4, 1/2 ou Ponte completa.

7.2 VALIDAÇÃO DA QUALIDADE DA COMUNICAÇÃO

Um dos maiores esforços deste trabalho foi no sentido de desenvolver um sistema de comunicação robusto e confiável para a aquisição dos dados distribuídos remotamente. Para tanto foi implementado no *firmware* de cada microcontrolador dos terminais de rede os comandos básicos do protocolo MODBUS-RTU. No capítulo anterior, DESENVOLVIMENTO DO SISTEMA DE AQUISIÇÃO DE DADOS, foi apresentado o conversor de interface RS232 para RS485. A seguir são mostrados os resultados dos testes de comunicações realizados.

7.2.1 TESTES

Os procedimentos de testes para validar a implementação, envolveram a análise do comportamento do protótipo utilizando softwares analisadores de protocolo de comunicação Modbus de dois fabricantes diferentes, cuja finalidade específica é a de realizar a análise da qualidade do bilhete e da qualidade da comunicação.

Para a realização dos experimentos, foram instaladas três unidades de rede, a primeira com o número 1, foi instalada a 10 metros da unidade mestre. A segunda (número 8) foi instalada a 12 metros de distância e a terceira (número 17) foi colocada a 15 metros.

A primeira fase de testes consistiu em verificar a conformidade dos comandos Modbus implementados nas estações escravas. Para tanto, foram configurados os parâmetros de comunicação utilizando o software Modbus Poll, conforme mostrado na Figura 86 e selecionada como unidade de teste a estação número 8, conforme mostrado na Figura 87.

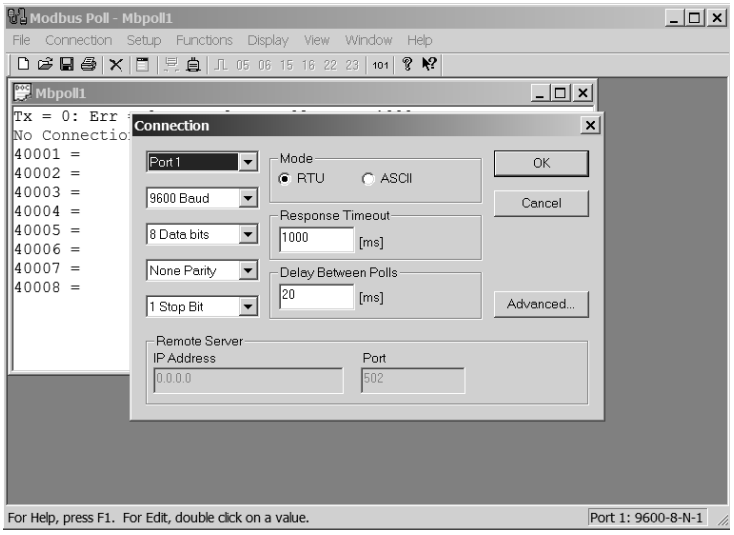

Figura 86: Configuração da Comunicação no software MBPOLL

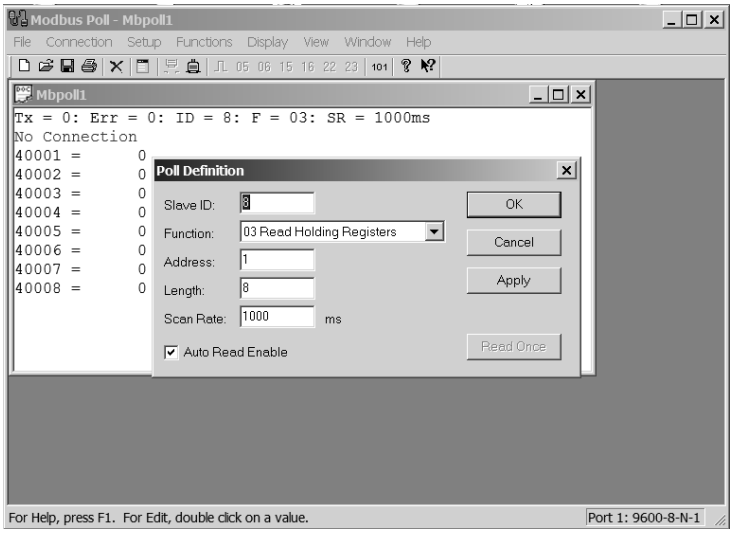

Figura 87: Configuração dos Comandos no MBPOLL

Foram enviados manualmente uma série de bilhetes para a estação número 8, conforme mostrado na Figura 88. Todos os bilhetes foram transmitidos com sucesso pelo servidor Modbus e também foram obtidas com sucesso as respostas esperadas, o que indicou que os comandos implementados nas estações escravas estavam funcionando adequadamente.

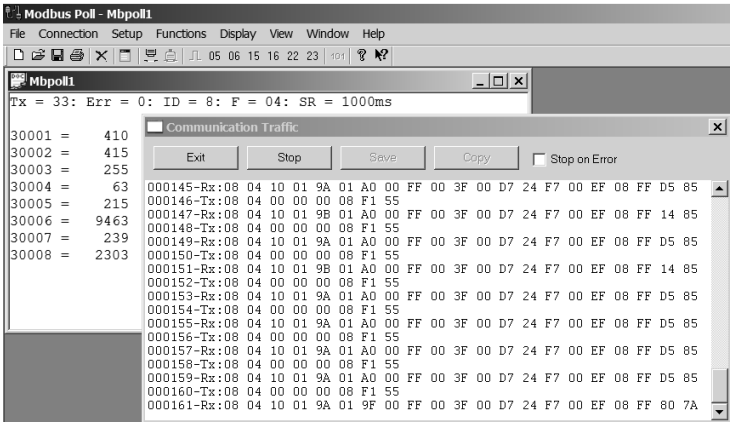

Figura 88: Tráfego de dados no software MBPOLL

O segundo teste foi realizado agora no modo automático, ou seja, o mesmo bilhete é retransmitido periodicamente em taxas de: 10, 5, 4, 2, 1 e 0,5 aquisições por segundo. Para essa tarefa também foi utilizada o software Modbus Poll. Para cada uma das funções Modbus (1xxxxx, 2xxxxx, 3xxxxx e 4xxxxx) foram feitas no mínimo 10.000 transações com resultados excelentes. A Figura 89 mostra o resultado de uma bateria de testes, onde foram enviados 14.812 bilhetes para a estação número 8, a uma taxa de 4 bilhetes por segundo. Todos os bilhetes foram enviados com sucesso pela unidade mestre e a estação escrava número 8 também respondeu com sucesso a todas as solicitações recebidas. Em nenhum caso houve um erro sequer.

Após os resultados animadores verificados anteriormente foi repetida a mesma seqüência

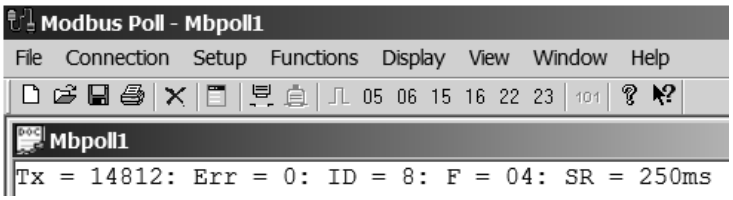

Figura 89: Nenhum erro em 14812 transações utilizando o software Modbus Poll

de testes, agora utilizando outro software analisador de protocolo, o Modbus Tester Versão 0.3 (beta). Foram enviados 12.545 bilhetes para a estação número 8, a uma taxa de 10 bilhetes por segundo. A Figura 90 mostra o software utilizado para a segunda seqüência de testes.

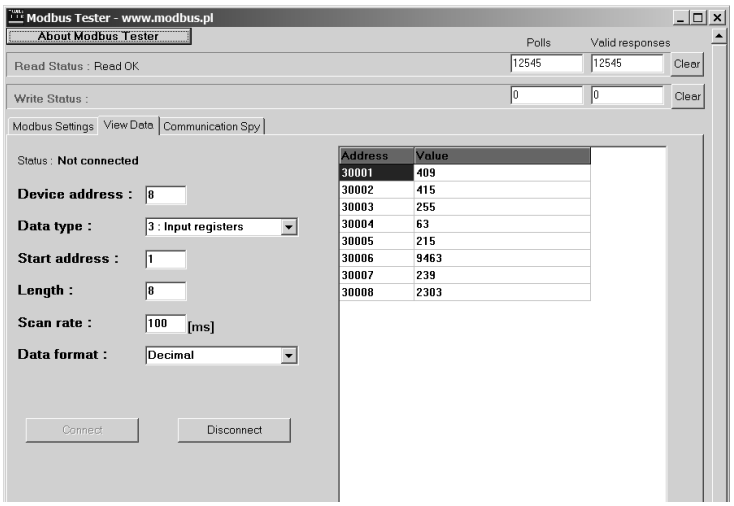

Figura 90: Nenhum erro em 12545 transações utilizando o software Modbus Tester Versão 0.3 (beta)

Os resultados foram similares aos da experiência anterior. Também não foi detectado nenhum erro.

Outros testes foram realizados, utilizando outras estações e outros comandos, com resultados semelhantes. Isto levou a concluir que os códigos dos programas implementados no *firmware* das estações escravas estão em conformidade com o padrão do protocolo Modbus RTU e o meio físico de comunicação utilizado (RS485) é confiável. Quanto a confiabilidade do meio físico, isto só veio a confirmar o resultado positivo verificado em outras milhares de aplicações nas indústrias que o utilizam. Deve-se ressaltar que, pela própria característica do protocolo, mesmo que houvesse alguma transmissão com erro, o protocolo automaticamente descartaria o bilhete defeituoso.

7.3 VALIDAÇÃO DA QUALIDADE DOS DADOS

Nesta seção será analisado a qualidade do sistema condicionador de sinais. Primeiramente foram feitos os ajustes de calibração necessários. Inicialmente foi ajustado o patamar de tensão de meia escala. Este valor foi fixado em 2,100 Volts. O procedimento para calibração foi descrito previamente no capítulo anterior. Em seguida foi feito o ajuste de 'zero' da ponte. Estando o nó da rede calibrado, o próximo passo foi fazer a coleta dos dados a partir do sistema supervisório.

7.3.1 PREPARAÇÃO DO EXPERIMENTO

A especificação do extensômetro a ser aplicado não é trivial. Uma escolha adequada requer conhecimento de vários aspectos relacionados ao objetivo de sua aplicação, ao processo de medição e aos materiais aplicados. Existe uma ampla gama de modelos, disponibilizados pelo mercado por diversos fabricantes. Para atender a finalidade do presente trabalho, foram utilizados extensômetros da marca Kyowa, do tipo KFG-5-120-C1-11, uniaxiais, cujas características passam a ser descritas (PENTEADO NETO, 2005). A denominação KFG diz respeito à série selecionada, e as letras iniciais são oriundas das palavras **K**yowa **F**oil **G**auges, relacionadas a extensômetros de uso geral e que possuem uma forma plana. Os extensômetros da série KFG podem ser de três tipos, denominados uni, bi ou triaxiais, em função das direções nas quais se pretende medir os esforços mecânicos. As séries variam, ainda, em função do material sobre os quais serão aplicados como, por exemplo: metais, concreto, materiais compósitos e papéis. As séries são divididas, também, pela finalidade para as quais os extensômetros são especificados como, por exemplo, para a medição de tração; para o acompanhamento do desenvolvimento de fissuras e respectiva velocidade de propagação, medição de tensão residual, etc. O código 5, diz respeito ao comprimento do extensômetro, que nesse caso é de 5 milímetros. O código 120, diz respeito à resistência ôhmica que é de 120Ω. C1 é o *lay-out* do extensômetro, dentre os aproximadamente 40 modelos oferecidos pelo fabricante. O número 11 se refere ao coeficiente de expansão linear que, neste caso, é de $11,7 \times 10^{-6}/°C$

Para a validação foram utilizados extensômetros modelo KFG-5-120-C1-11 (descrito no parágrafo anterior), compensado termicamente para estruturas de aço, de resistência nominal de 120.2Ω ± 0.2Ω, Fator de Sensibilidade (*Gauge-Factor*) 2.11 ± 1,0 % (@24 °C). Fabricados pela Kyowa. Estes extensômetros foram ligados em configuração de 1/4 de ponte para medição de esforço devido ao momento fletor aplicado a uma chapa de de aço plana de 150 mm de comprimento por 50 mm de largura e 1 mm de espessura. O extensômetro foi fixado na região

central da placa, eqüidistante das extremidades. A Figura 91 mostra o extensômetro fixado no elemento de teste.

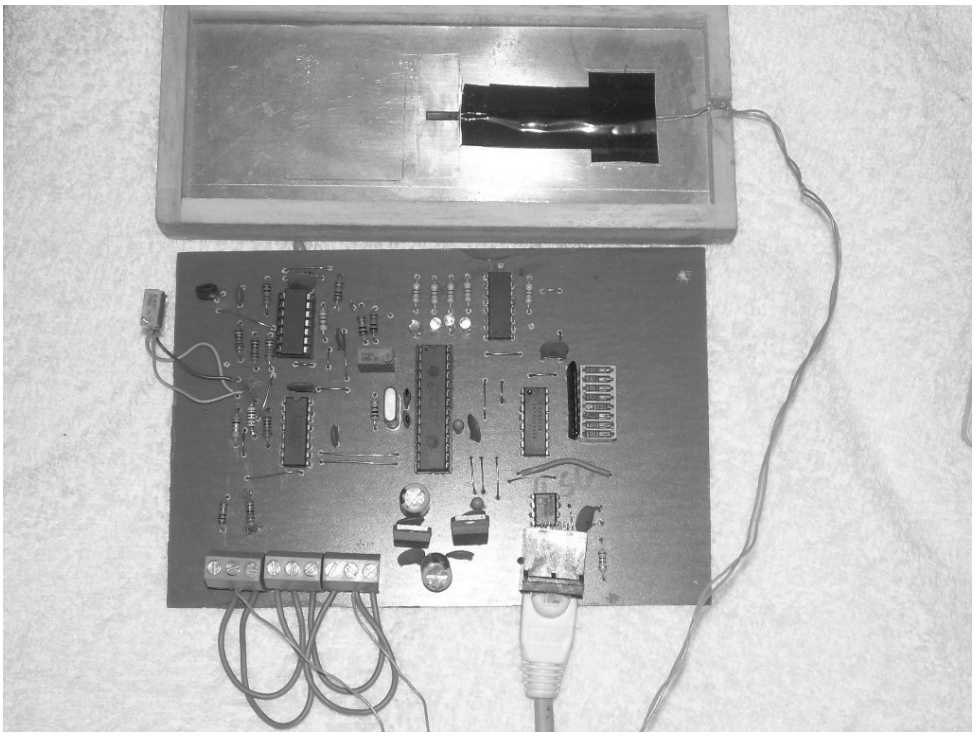

Figura 91: Extensômetro elétrico ligado em configuração de 1/4 de ponte utilizado para o experimento

Como já mencionado, o extensômetro utilizado é do tipo laminar (*foil gauge*) cuja representação esquemática pode ser vista na Figura 92.

Figura 92: Extensômetro CFG-5-120-C1-11 utilizado

Os softwares mencionados anteriormente são só analisadores de protocolo e foram utilizados para validar a implementação do protocolo nos terminais de rede. Para a aquisição dos dados dos experimentos foi utilizado o servidor Modbus Kepware Server Versão 4.0. Como já explicado anteriormente, este funciona como um mestre Modbus e também como um servidor OPC. A configuração dos parâmetros de comunicação pode ser vista na Figura 93.

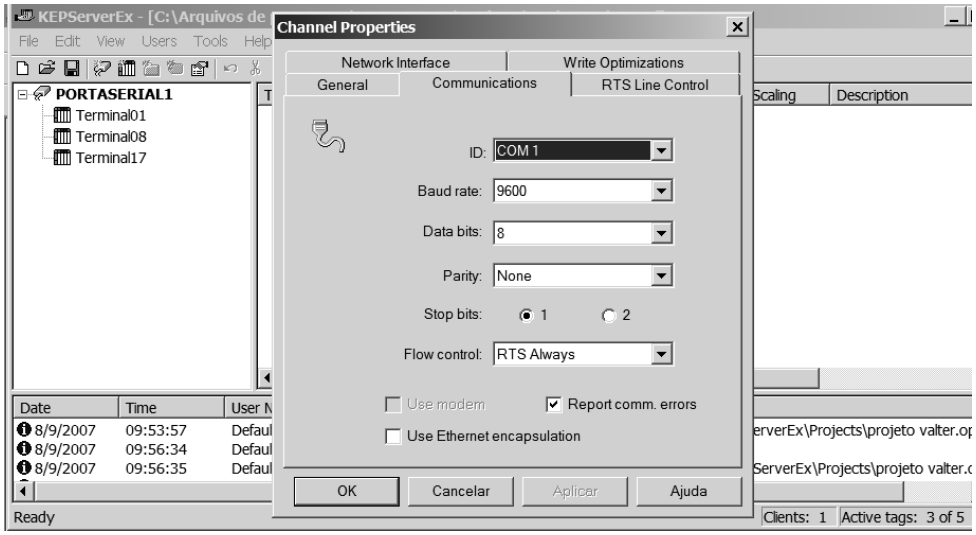

Figura 93: Configuração dos parâmetros de Comunicação do Servidor OPC MODBUS

Para o teste inicial, foram configurados três nós de redes, com os números 1, 8 e 17, respectivamente. A configuração dos nós pode ser visualizada na Figura 94.

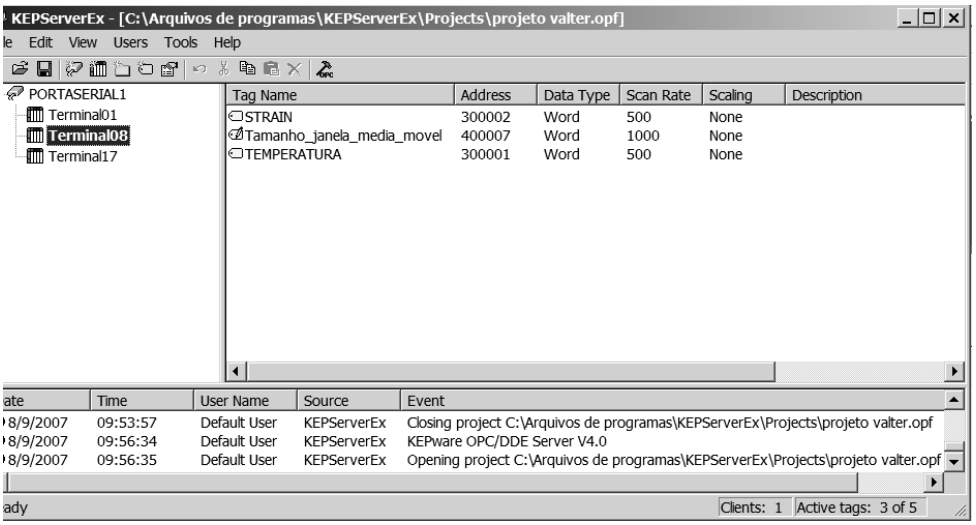

Figura 94: Configuração do Servidor OPC MODBUS

7.3.2 DESVIO COM O PASSAR DO TEMPO

O sistema foi iniciado várias vezes, conforme mostrado na Tabela 7.

Tabela 7: Tempo de Estabilização do sistema

número	Tempo
1	2 min, 32 seg
2	2 min, 28 seg
3	2 min, 24 seg
4	2 min, 22 seg
5	2 min, 23 seg
6	2 min, 25 seg
7	2 min, 25 seg
8	2 min, 19 seg
9	2 min, 22 seg
10	2 min, 25 seg

Foram realizadas 10 seqüências de ligamento do sistema e aguardou-se até que ficasse estável quando então este tempo foi medido. Em todas as experiências o sistema ficou ligado por 5 minutos quando então era desligado. Aguardava-se outros 5 minutos para o resfriamento do sistema e em seguida era ligado novamente. O tempo máximo para estabilização foi de 2 minutos e 32 segundos. O tempo médio ficou em torno de 2 minutos e 24 segundos. A Figura 95 ilustra o instante inicial de uma das experiências quando o sistema ainda estava em regime transiente.

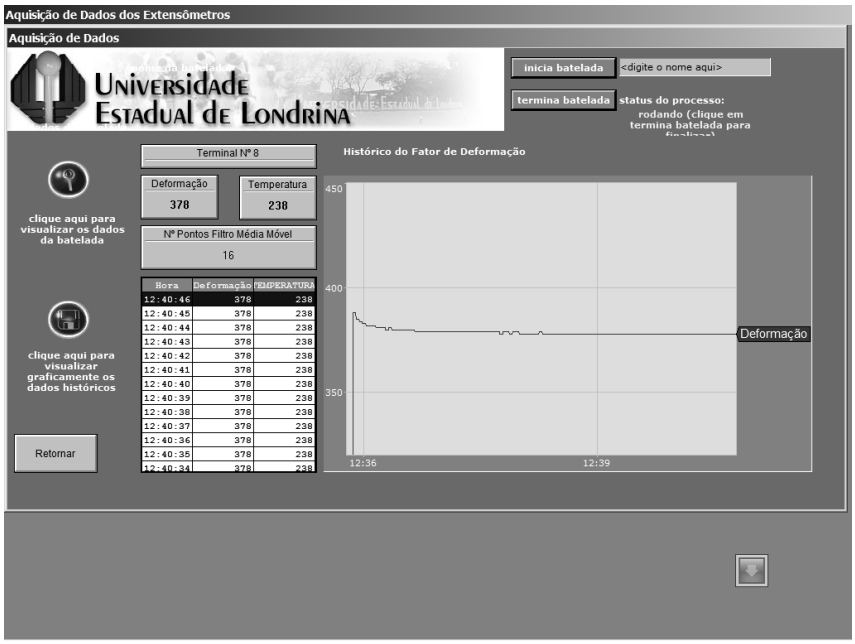

Figura 95: Tela de coleta de dados logo após o sistema ter sido ligado (regime transiente)

Depois de entrar em regime permanente, observou se que o sistema é estável com o passar do tempo. Isto pode ser verificado através da Figura 96.

Figura 96: Tela de coleta de dados mostrando estabilidade com o passar do tempo (regime permanente)

7.4 REPETIBILIDADE

O termo repetibilidade refere-se a capacidade de um instrumento fornecer a mesma leitura de peso para um determinado objeto e de retornar a zero após cada ciclo de pesagem. O teste é feito pesando-se repetidamente um mesmo objeto.

Para a aquisição de dados foi utilizado o software Elipse Scada V. 2.9. A seqüência de procedimentos foi a seguinte: Inicialmente o sistema foi deixado em repouso até que o sistema se estabilizasse.

Depois foi colocado e retirado alternadamente por 13 vezes um corpo de carga de 255 gramas (no caso foi utilizado como carga, a carcaça de um transformador). A Figura 97 mostra o corpo de prova colocado no elemento de medição, enquanto que a Figura 98 mostra a tela com o resultado das medições. Como resultado, quando aplicado a carga o resultado medido foi de 341 unidades ± 1, o que equivale a uma repetibilidade de 1 bit de resolução do conversor AD (10 bits), valor que pode ser considerado excelente se levarmos em conta a limitação dos componentes utilizados.

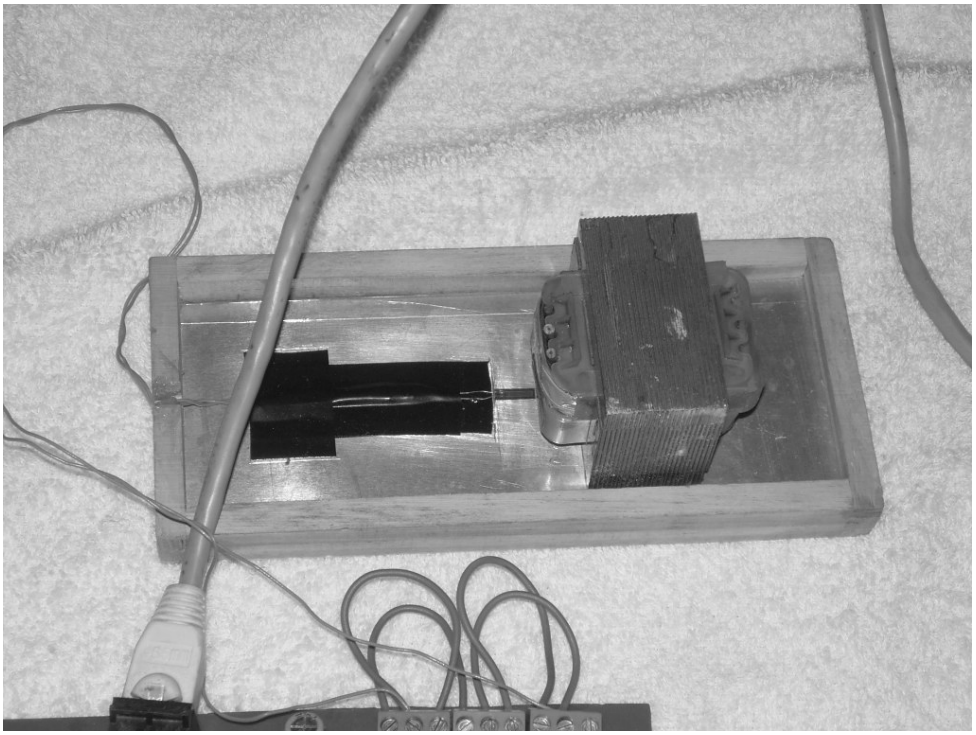

Figura 97: Elemento utilizado como carga para o experimento

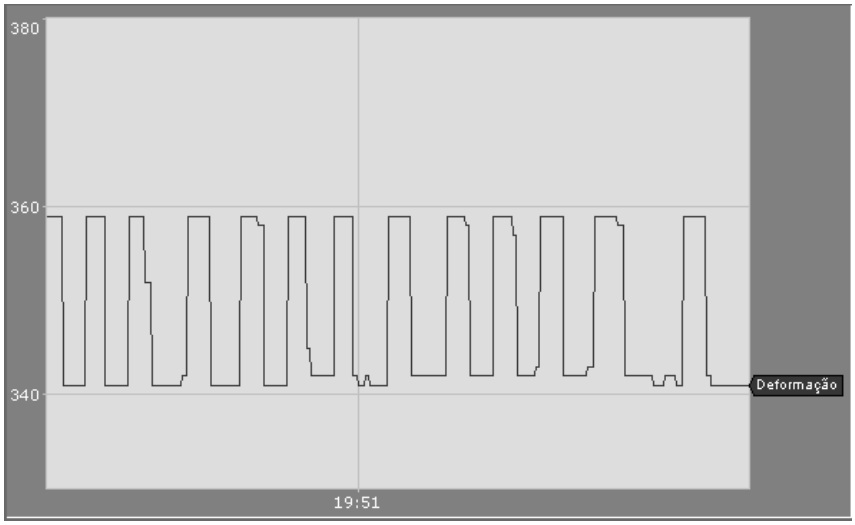

Figura 98: Teste de Repetibilidade

7.4.1 HISTERESE

Histerese é uma propriedade dos materiais elásticos, resultando em diferentes indicações de deformação para uma mesma força aplicada, dependendo da direção da aplicação da força, ou seja, se o carregamento está aumentando ou diminuindo sobre o corpo. É mais pronunciado no ponto de meia escala. A Figura 99 mostra uma curva típica.

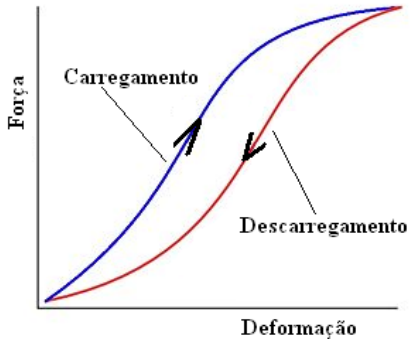

Figura 99: Curva de histerese para materiais elásticos

Dois corpos de pesos diferentes foram adicionados um após o outro sobre a chapa metálica que contém um extensômetro elétrico de resistência. Foi observado que ao se retirar o segundo corpo a medida não retornou ao valor inicial antes da sua inserção, mesmo após um longo período, confirmando que o sistema de medida utilizado possui uma certa histerese associada. Isto pode ser observado na Figura 100.

Figura 100: Tela de Aquisição

7.4.2 DERIVA DEVIDO À TEMPERATURA

A sensibilidade a deriva é como uma variação em temperatura pode afetar a medida de um sistema. Basicamente esse parâmetro diz qual deve ser o comportamento esperado com a variação da temperatura.

Idealmente, o valor do extensômetro deveria mudar somente devido a deformações da superfície a que está fixado. No entanto, tanto o material de que é feito o extensômetro quanto o corpo de prova em que ele está fixado também respondem a variações de temperatura. Os fabricantes dos extensômetros tentam minimizar estes efeitos através da utilização de materiais

que compensem a expansão térmica do material para o qual é projetado. Enquanto que essa compensação reduz a sensibilidade térmica, esta não é totalmente eliminada.

O procedimento consistiu em colocar o conjunto do corpo de prova e o sistema de aquisição confinados em uma caixa fechada, simulando uma estufa, de maneira que a temperatura fosse igual para ambos os elementos. Para tanto aguardou-se um tempo suficiente para que o sistema entrasse em equilíbrio térmico, caracterizado através da estabilidade dos valores de ambos os elementos: extensômetro e sensor de temperatura. Veja a Figura 101.

Figura 101: Gráfico mostrando a situação inicial do experimento

A relação entre número de pontos da escala e temperatura pode ser obtida da seguinte forma:

O LM35 fornece uma tensão proporcional de 10 mV/°C a partir de 0°C. Assim, se sua saída em um dado instante for de 200 mV, pode-se inferir que a temperatura é de 20 *°C*. Para aumentar a resolução, foi acrescentado um amplificador de ganho igual a $1 + \frac{10.2}{3.3} = 4.09$. Assim, para cada grau a partir de zero, tem-se uma tensão de 40.9 mV na saída do amplificador. Esta tensão é capturada pelo conversor AD. Como a resolução é de 10 bits (1024 pontos) e a alimentação é de 5 Volts, pode-se obter a relação entre o número de pontos e a temperatura, na seguinte equação:

$$T = n \times 0.1194$$

onde T é a temperatura em graus Celsius e n é o número de pontos medido.

O procedimento seguinte foi ligar um soprador térmico com jato direcionado para o interior do espaço. A temperatura inicial era de 238 pontos $\cong$ 28.5 *°C,* após o aquecimento registrou-se

um pico de 325 pontos ≅ 38.8 $°C$. Os resultados podem ser vistos na Figura 102. Como pode-se observar, o sistema manteve-se relativamente estável, apesar da grande variação de temperatura a que o conjunto foi submetido, o que indica que o algoritmo implementado para correção dos desvios térmicos conforme descrito no Capítulo PROJETO, MONTAGEM E TESTES DO TERMINAL DE AQUISIÇÃO, Seções 5.7 a 5.9, funcionou como esperado.

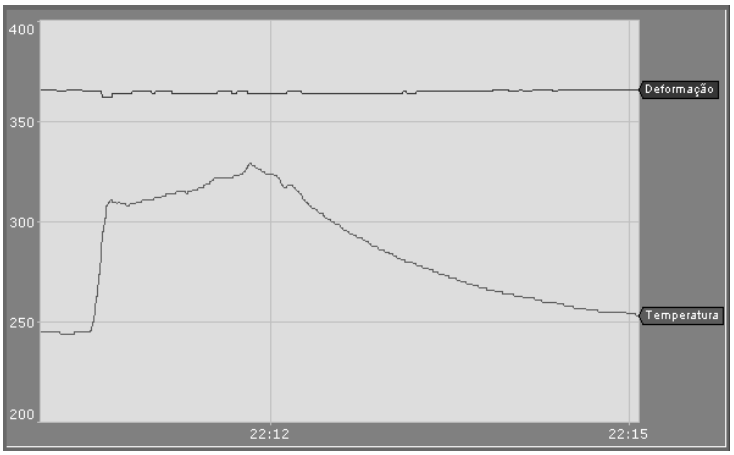

Figura 102: Gráfico mostrando a evolução da temperatura e da sensibilidade térmica do extensômetro durante a fase do aquecimento e desaquecimento.

Foi observado durante a fase de testes que, devido a inércia, histerese e capacidade térmica dos materiais envolvidos, o sensor de temperatura detecta primeiro a variação do que o corpo de prova. Observou-se que após o desligamento do soprador térmico, o valor da temperatura do sensor LM35 estabilizou muito mais rapidamente do que a do extensômetro, que ainda continuava a subir, demonstrando uma inércia térmica maior do que a do sensor. Considerando que os materiais são diferentes na sua composição, massa, volume e geometria e também estão separados espacialmente, era esperado que os coeficientes de transferência de calor de ambos fossem diferentes, como de fato se comprovou.

O próximo passo foi monitorar a curva de resposta do resfriamento de ambos por um tempo de cinco minutos, que também mostrou diferença significativa entre a temperatura medida pelo sensor e a resposta do extensômetro, como já esperado.

Como conclusão, pode-se afirmar que a compensação de extensômetros ligados em configuração de ¼ de ponte só pode ser feita após transcorrido um tempo suficiente para que ambos, sensor e corpo de prova, atinjam a mesma temperatura. No entanto, isto é dependente do material e das condições do local, sendo muito difícil a realização deste procedimento. Recomenda-se, portanto, que sempre que possível, se utilize a configuração de ½ ponte utilizando um extensômetro inativo (*dummy gauge*) para compensação. Esta técnica passa a ser descrita a seguir.

Através da utilização de dois extensômetros na ponte, o efeito da temperatura pode ser

praticamente eliminado. Por exemplo, na Figura 103 é ilustrado uma configuração onde um extensômetros é ativo ($R + \Delta R$), e um segundo extensômetro inativo (conhecido como *dummy gauge)* é fixado em uma peça do mesmo tipo de material que o extensômetro ativo. Assim, qualquer alteração da temperatura vai afetar ambos os extensômetros da mesma maneira. Então, se a variação da temperatura for idêntica para ambos, a razão entre suas resistências não se altera e por conseqüência, também a tensão de saída Vo permanece inalterada.

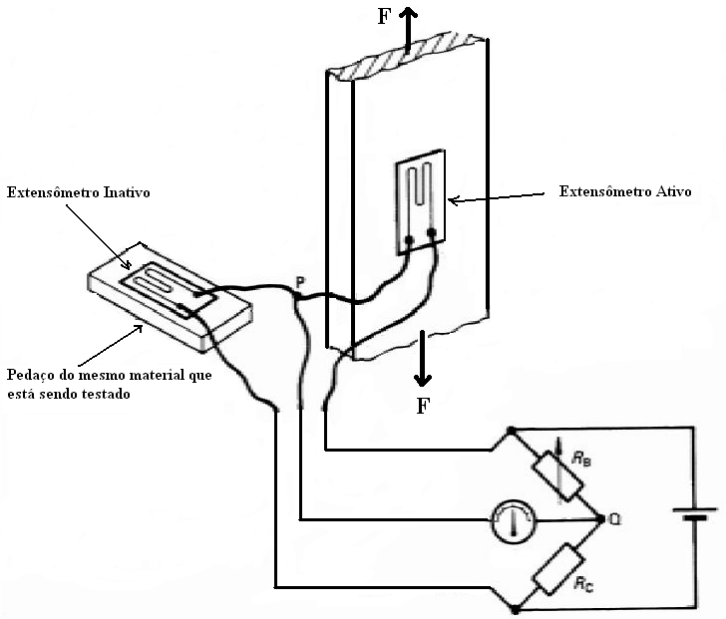

Figura 103: Utilização de um extensômetro inativo para eliminar os efeitos da temperatura

No entanto, como já é prática comum dos laboratórios de ensaios de esforços, deve se utilizar preferencialmente, a configuração em ponte completa utilizando 4 extensômetros ativos, onde os efeitos térmicos são auto-compensados.

8 *CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS*

Este trabalho descreve o projeto, desenvolvimento, implementação e validação de um sistema conectado em rede para aquisição de dados de esforços em estruturas físicas utilizando como elementos sensores os extensômetros elétricos de resistência (*strain gauges).*

O objetivo deste trabalho consistiu em se produzir uma interface de aquisição de dados distribuída em rede, a implementação de um protocolo de comunicação entre estas unidades remotas e um microcomputador, no qual desenvolveu-se um programa para a exibição dos resultados coletados nas unidades de rede.

Neste capítulo são elencados os pontos fortes e fracos identificados durante a produção deste trabalho. Algumas conclusões são fornecidas e são apontadas possíveis futuras direções para complementá-lo.

8.1 VIABILIDADE DO CONCEITO

Foram construídos e testadas quatro unidades de rede no laboratório experimental e em bancada, com sucesso.

No que diz respeito à rede de comunicação, tanto a recepção quanto a transmissão do sinal se mostraram confiáveis e livres de erros. Os resultados foram motivadores para que se persista confiando nessa solução.

O sistema supervisório mostrou-se eficiente e robusto na tarefa de coleta, armazenamento e visualização dos dados vindos da rede de sensores. Por meio de uma interface amigável foi possível a realizacão de ensaios. A leitura de outras variáveis, tal como a temperatura também foi satisfatória. Os bons resultados apresentados pelo supervisório demonstraram a potencialidade deste sistema e encoraja o prosseguimento desta solução para a implementação de novas funcionalidades eventualmente necessárias.

A filosofia empregada para a leitura dos valores dos extensômetros, o tratamento eletrônico de sinal, tanto em nível de *hardware* quanto em nível de *software*, se mostrou consistente. Em relação ao tempo de resposta do conjunto, este se mostrou adequado ao objetivo proposto incialmente de monitoração de estruturas estáticas.

É ainda importante ressaltar que, com o desenvolvimento desse sistema, o projeto e a construção da plataforma de aquisição e seus elementos de apoio, como o aplicativo de software, foi possível agregar *know-how* nacional e local nesta área de desenvolvimento.

Os resultados obtidos com a utilização do protótipo em experimentos pilotos são encorajadores, pois indicam que os conceitos apresentados aqui, tanto os sistemas de *hardware* e *software* desenvolvidos, são viáveis.

8.2 LIMITAÇÕES

O protocolo MODBUS-RTU foi o protocolo escolhido por apresentar diversas vantagens, tais como: a interoperabilidade, o determinismo, a confiabilidade dos dados, entre outras. O propósito não foi resolver um problema isolado, e sim estender a solução encontrada para outras aplicações. A aplicação desenvolvida, da coleta de dados de sensores distribuídos em rede, forneceu subsídios importantes para o desenvolvimento proposto. Apesar de ter atingido os objetivos propostos inicialmente, esta implementação merece aperfeiçoamentos em alguns aspectos.

O consumo de energia é relativamente alto, principalmente pela utilização de sensores extensométricos de baixa resistencia ôhmica, que ficam constantemente drenando energia. Para reduzir o consumo de corrente pela ponte de Wheatstone, poderia-se considerar a utilização de extensômetros de maior resistência, ou ainda, implementar circuitos de controle de alimentação, que somente seria ligada no instante de se obterem as amostras. Também poderia-se considerar o uso de circuitos integrados dedicados à operação em ponte, mas com baixo consumo de potência.

Outra limitação está associada à resolução obtida de 10 bits. Embora satisfatória para aplicações que envolvam grandes variações relativas do valor de carga, como foi o caso da aplicação deste projeto, pode não ser adequada para outras, que necessitem detectar pequenas variações.

8.3 SUGESTÕES PARA TRABALHOS FUTUROS

Os resultados obtidos com o protótipo mostram que o mesmo atende às expectativas inicialmente definidas e possibilitam a abertura de oportunidades para a continuidade do trabalho visando otimizá-lo ainda mais. Podemos sugerir, por exemplo:

- Os sistemas comerciais profissionais utilizam normalmente resolução de 12 bits. Este índice de desempenho pode ser facilmente atingido se forem utilizados conversores analógico/digital tipo delta-sigma de 21 ou 24 bits, relativamente fáceis de se encontrar atualmente no mercado e mais baratos que de outras tecnologias.

- Determinar a viabilidade de uso de uma fonte de energia alternativa, que atenda a demanda operacional do sistema, tal como, por exemplo, uso da energia solar.

- O sistema desenvolvido incorpora algumas das vantagens dos sistemas sem-fios, tal qual a aquisição e condicionamento local dos sinais e facilidade de instalação. Embora originalmente o sistema proposto não tenha sido desenvolvido para ser *wireless*, também há a possibilidade de vir a ser. O protocolo de comunicação utilizado para este projeto é o MODBUS-RTU que define apenas a camada 7 (camada de aplicação) do modelo ISO/OSI (MODBUS-IDA, 2007a) e portanto não especifica a camada física de transmissão, que pode ser serial cabeada, serial fibra ótica, Ethernet TCP/IP ou qualquer outra compatível, inclusive sem-fios. Pode-se implementar ainda um sistema misto contendo elementos cabeados e sem-fios, pois existem no mercado equipamentos que fazem a conversão de protocolos (*gateways*), como por exemplo de MODBUS-RTU para ETHERNET sem fio (802.11b/g), MODBUS-RTU/ZIGBEE, MODBUS-RTU/BLUETOOTH e mais recentemente surgiram alguns fabricantes oferecendo conversores GPRS/MODBUS, que podem permitir o acesso à rede de sensores através de um aparelho celular. Assim se cada unidade for acoplada a um desses conversores o sistema passa a ser também sem-fio, necessitando apenas a adição dos equipamentos de conversão e de uma fonte de alimentação externa. Neste caso, os eventos seriam hierarquizados, podendo ser comunicados por e-mail, sms, utilizados para acionamento de processos de segurança e controle da concessionária de pedágio (como por exemplo, em monitoramento de pontes), e armazenados em banco eletrônico de dados.

- Futuramente o *front-end* analógico do sistema de aquisição poderia ser implementado por um único chip com funções de medição, calibragem, processamento e comunicação. Pode-se integrar os módulos analógicos através do programa MOSIS (MOSIS, s.d.) de fa-

bricação de circuitos integrados, de forma a reduzir custos em caso de eventual produção em escala do equipamento.

- Nas unidades de rede poderiam ser incluídos outros recursos adicionais, tais como monitoramento de outras grandezas, tais como corrosão, umidade, etc.
- Aprimoramento de estruturas do software, como por exemplo:
  - aperfeiçoamento da interface usuário sistema,
  - melhoria das rotinas de armazenamento busca, e apresentação de dados na tela;
  - geração de relatórios de forma que possam estar disponíveis por acesso remoto, via Pager, Internet e outros;
- Estudar a possibilidade de medir os esforços estruturais pelo sistema piezoelétrico com utilização, por exemplo, de cristais de quartzo, como alternativa à extensometria.
- Incorporação de unidades de medidas de efeitos dinâmicos das estruturas, como, por exemplo, vibrações.

8.4 CONSIDERAÇÕES FINAIS

Como consideração final, fica a sugestão de que o modelo desenvolvido possa, numa etapa posterior, ser transferida para uma empresa do setor produtivo e a transformação da unidade protótipo em produto. Com isto, haveria um envolvimento contínuo entre o meio acadêmico e o setor empresarial, possibilitando futuras cooperações na continuidade deste projeto contribuindo para o desenvolvimento da tecnologia nacional.

Referências Bibliográficas

AGILENT TECHNOLOGIES, **application Note 290-1—Practical Strain Gage Measurements**, 1999.

ALLEN, P., E.; GEIGER, R.L.; STRADER, N.R., **Design Techniques for Analog and Digital Circuits**. McGraw Hill. 1990.

ANDOLFATO, Rodrigo P.; CAMACHO, Jefferson S. e BRITO, Gilberto A. de, **Extensometria Básica**, Unesp Ilha Solteira, Núcleo de Ensino e Pesquisa da Alvenaria Estrutural, endereço: http://www.nepae.feis.unesp.br/Apostilas/Extensometria basica.pdf, acessado em 30/08/2007.

ARMS, S.W., GALBREATH, J.H., NEWHARD, A.T., TOWNSEND, C.P., "Remotely Reprogrammable Sensors for Structural Health Monitoring", **Structural Materials Technology (SMT): NDE/NDT for Highways and Bridges,** 16 September 2004, Buffalo, NY, USA, 2004.

ARMS, S.W. e TOWNSEND, C.P., "Wireless Strain Measurement Systems - Applications & Solutions", **NSF-ESF Joint Conference on Structural Health Monitoring,** Strasbourg, France, Oct 3-5, 2003.

BAGESTEIRO, L. B. **Desenvolvimento de uma plataforma para análise de forças produzidas por pessoas**. Dissertação (Mestrado em Engenharia Mecânica), UFRGS, Porto Alegre,1996

BIELEN, P.; LOSSIE, M., VANDEPITTE, D., "A low cost wireless multi-channel measurement system for strain gauges" **PROCEEDINGS OF ISMA2002** - VOLUME II, 2002, http://www.isma-isaac.be/publications/PMA_MOD_publications/ISMA2002/663_670.pdf

BLUETOOTH-ORG, Bluetooth Alliance, endereço http://www.bluetooth.org acessado em novembro de 2007.

BOARETTO, Neury, **TECNOLOGIA DE COMUNICAÇÃO EM SISTEMA SCADA – ENFOQUE EM COMUNICAÇÃO WIRELESS COM ESPALHAMENTO ESPECTRAL**, Dissertação (Mestrado em Engenharia de Produção), CEFET, PONTA GROSSA, 2005.

BROWNJOHN, John; TJIN, Swee-Chuan; TAN, Guan-Hong, TAN, Boon-Leong e CHAKRABOORTY, Sushanta, "A Structural Health Monitoring Paradigm for Civil Infrastructure", **1st FIG International Symposium on Engineering Surveys for Construction Works and Structural Engineering**, Nottingham, United Kingdom, 28 June – 1 July 2004.

BU, Shengrong, **Wireless Ad-Hoc Control Networks**, Dissertação de Mestrado em Engenharia Elétrica, Escola de Engenharia Elétrica, Computação e Engenharia de Telecomunicações da Universidade de WOLLONGONG, CHINA, 2005.

CASTALDO, F.C., **Investigação de Ruído e Sensibilidade em MAGFETs e Avaliação de seu Emprego no Controle de Emissão Eletromagnética em Circuitos Integrados de Potência**. Tese (Doutorado em Engenharia Elétrica), UNICAMP, 2005.

CÉLULA. **Células de carga**. Endereço: www.celuladecarga.com.br, acessado em, agosto de 2007.

CHAVES, João Carlos, **Uso da Tecnologia GPS na Monitoração de Deformação: Sistemas, Etapas e Experimentos**. Tese (Doutorado em Engenharia Civil), USP - São Carlos, 2001.

COSTA, António Casimiro Ferreira da, **NavCim Uma Arquitectura Distribuída de Suporte ao Controlo e Supervisão em Tempo-real de Processos Industriais**. Dissertação (Mestrado em Engenharia Electrotécnica e de Computadores), Universidade Técnica de Lisboa, Setembro de 1995.

CUNHA, Judson Michel. Protótipo de rede industrial utilizando o padrão serial RS485 e protocolo modbus. 2000. 108 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. DATAFORTH, **Dataforth Corporation** - AN118, endereço: www.dataforth.com, acessado em agosto de 2007.

DEMOLDER, S. e CARLSTER, A. V., "The measuring of 1/f noise of thick and thin film resistors", Journal of Physics E: Scientific Instrumentation, Vo. 13, 1980.

DOEBELIN, Ernest O., **Measurement Systems - Application and Design**, 5 Ed., McGrawHill, 2004.

DOEBLIN, S. W.; FARRAR, C. R.; PRIME, M. B.; SHEVITZ, D. W., **Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in Their Vibration Characteristics: A Literature Review**, Los Alamos National Laboratory, LA, USA, 1996.

DYDENSBORG, Mads Bondo, **Connection Oriented Sensor Networks**, Tese (Doutorado em Ciência da computação), Universidade de Copenhagen, Dinamarca, 2004.

ELGAMAL, Ahmed; CONTE, Joel P.; YAN, Linjun; FRASER, Michael, **A Framework for Monitoring Bridges and Civil Infrastructure,** Department of Structural Engineering University of California, San Diego La Jolla, CA, 2005.

FERREIRA, Elnatan Chagas, DEMIC – DEPARTAMENTO DE ELETRÔNICA E MICROELETRÔNICA. São Paulo, Disponível em:http://www.demic.fee.unicamp.br/~elnatan/ie321/ie321.htm. Acesso em: 05 out. 2007.

FRADEN, Jacob, **Handbook of modern sensors: physics, designs, and applications**, Editora Springer, 3ª Edição,589 p., 2004.

FURMAN, B.J., **ME 120 Experimental Methods Force, Torque, Stress, and Strain Measurement**, 2003.

GIACOLETTO, L. J., **Electronics Designers Handbook**, second edition, McGraw-Hill Book Company, New York, 1977.

GALBREATH, J. H., TOWSEND, C.P.; MUNDELL, S. W.; HAMEL, M. J.; ESSER, B.; HUSTON, D.; ARMS, S. W., "Civil Structure Strain Monitoring with Power-Efficient High-Speed Wireless Sensor Networks", **International Workshop for Structural Health Monitoring**, Stanford, CA, September, 20, 2003.

GOMES, Francisco Carlos; CALIL JÚNIOR, Carlito, "Estudo teórico e Experimental das Ações em Silos Horizontais", **Cadernos de Engenharia de Estruturas**, USP - São Carlos, v. 7, n. 24, p. 35-63, 2005.

GONÇALVES, Luiz Fernando, **Contribuições para o Estudo Teórico e Experimental de Sistema de Geração Distribuída**, Dissertação (mestrado em Engenharia Elétrica), Universidade Federal do Rio Grande do Sul, Porto Alegre, março de 2004.

HEJLL, Arvid, **Civil Structural Health Monitoring - Strategies, Methods and Applications**, Tese (Doutorado em Engenharia Civil), Luleå University of Technology, 2007.

HILL, Jason Lester, **System Architecture for Wireless Sensor Networks**, Tese (Doutorado em Ciência da Computação), Universidade da Califórnia, Berkeley, 2003.

HILL, JASON; SZEWCZYK, ROBERT; WOO, ALEC; HOLLAR, SETH; CULLER, DAVID; PISTER, KRISTOFER, **"**System Architecture Directions for Networked Sensors"**,Proceedings of the Ninth International Conference on Achitetural Support for Programming Languagens and Operating Systems**, pp. 93-104, Cambridge,MA, USA, 2000. endereço: http://www.tinyos.net/papers/tos.pdf., acessado em 30/08/2007.

ILARIONOV, Rajcho; SIMEONOV, Ivan; KILIFAREV, Hristo, "Information system with light display for visualization of temperature, pressure and time". International Conference on Computer Systems and Technologies - CompSysTech' 2005, 2005.

JDM PROGRAMMER, endereço: http://www.jdm.homepage.dk/newpics.htm, acessado em 15 de novembro de 2007.

KESTER, Walt, Analog Devices Inc., **Practical Design Techniques for sensor signal conditioning**. Published by Analog Devices.

KIJEWSKI-CORREA, T., HAENGGI, M. e ANTSAKLIS, P., **Multi-scale wireless sensor networks for structural health monitoring**, http://www.nd.edu/~pantsakl/publications/345-ISHMII05.pdf, 2005.

LIN, Mark W.; THADURI, Jagan; GOPU, Vijaya, **A Distributed Strain Sensor for Bridge Monitoring**, Department of Civil and Environmental Engineering, UTCA - University Transportation Center for Alabama, 2003.

LEONG, Wai Yie, Smart Battery Monitoring System, Monografia (Bachelor of Engineering (Honours)), Department of Information Technology and Electrical Engineering, University of Queensland, USA, October, 2001.

LENG, J. S. Structural Health Monitoring of Smart Civil Structures Using Fibre Optic Sensors", **Electronic Publication www.NDT.net**, Vol. 10, No. 5, Maio 2005.

LUI, S.C.; TOMIZUKA, M., "Strategic Research for Sensors and Smart Structures Technology" **First International Conference on Structural Health Monitoring and Intelligent Infrastructure (SHMII-1'2003)**, Tokyo, Japan, November 13–15, 2003.

LYNCH, J. P.; LAW, K. H.; KIREMIDYIAN, A. S.; KENNY, T. W.; CARRYER, E.; PARTRIDGE, A., The Design of a Wireless Sensing Unit for Structural Health Monitoring. Proceedings os the 3rd International Workshop on Structural Health Monitoring. Stanford, CA, USA, September 12-14, 2001.

LYNCH, J. P.; KINCHO, H. L.; KIREMIDJIAN, A., "A wireless modular monitoring system for civil structures", **Proceedings of the 20th International Modal Analysis Conference (IMAC XX)**, Los Angeles, CA, USA, February 4-7, 2002.

MADGETECH, STRAIN GAGES Use with BRIDGE110 datalogger - Application Note, endereço: http://www.madgetech.com/pdf_files/app_notes/bridge110_app _note.pdf, acessado em agosto de 2007.

MALLADI, R.; AGRAWAL, D. P., "Current and future applications of mobile and wireless networks", **Communications of the ACM**, ACM Press, ISSN 0001- 0782,vol. 45, no. 10, pp 144-146, 2002.

MAXIM-DALLAS, APPLICATION NOTE 3426 **Resistive Bridge Basics: Part One**, acessado em 30/08/2007 no emdereço eletrônico http://www.maxim-ic.com/an3426, 2004.

MAXIM-DALLAS, APPLICATION NOTE 3545 **Resistive Bridge Basics: Part Two**, acessado em 30/08/2007 no emdereço eletrônico http://www.maxim-ic.com/an3545, 2004.

Measurements Group, **Strain Gage Measurement System**, http://www.measurementsgroup.com/guide/ta/sgms/sgmsndex.htm, acessado em janeiro de 2007.

MELLO, Alexandre José Tuoto Silveira, **Um Estudo da Aplicação de Redes de Sensores para Monitoração da Proteção Catódica em Dutos**, Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial), Universidade Tecnológica Federal do Paraná - Campus Curitiba, ABRIL-2007.

MODBUS-IDA.ORG, **MODBUS over Serial Line Specification and Implementation Guide V1.01** - 2006, endereço: http://www.modbus.org/, acessado em agosto 2007.

MODBUS-IDA.ORG, **MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1a - 2006**, endereço: http://www.modbus.org/, acessado em agosto 2007.

MODICON, **MODBUS PROTOCOL**, endereço: http://www.http://www.modicon.com/techpubs/toc7.html, acessado em agosto, 2007

MORAIS, P. Gil; SANTOS, C. Almeida; CARVALHO, Mariana R., **Sistema Automático de Medição para Ensaios de Ancoragens**, Centro de Instrumentação Científica (CIC) Departamento de Geotecnia (DG) Laboratório Nacional de Engenharia Civil, Lisboa, Portugal, 2005.

endereço: http://www.mosis.org.

NAGAYAMA, T.; RUIZ-SANDOVAL, M.; SPENCER JR., B. F.; MECHITOV, K. A.; AGHA, G., **Wireless Strain Sensor Development for Civil Infrastructure**, 2004.

NAPOLINANO, Flávio, **Projeto de uma célula de carga para aplicação em suspensões veiculares**, Dissertação (Mestrado em Engenharia Mecânica), UNICAMP, CAMPINAS - SP, 2000.

NAVES, E. L. M. Desenvolvimento de uma plataforma de força para análise da performance biomecânica. Dissertação (Mestrado em Engenharia Elétrica), UFU, 2001.

NOVUS PRODUTOS ELETRÔNICOS LTDA, **Conceitos básicos de RS485 E RS422**, endereço: www.novus.com.br, acessado em novembro de 2007

PANHAN, André Marcelo, **Sistema de Aquisição de Dados e Monitoramento Remoto para Câmaras Frias e Sistemas de Refrigeração**, Dissertação (Mestrado em Engenharia Elétrica), Faculdade de Engenharia Elétrica e de Computação - UNICAMP, CAMPINAS, 2002.

PENTEADO NETO, Renato de Arruda, **Sistemas para Detecção de Falta de Alta Impedância e de Rompimento de Condutores em Redes de Distribuição de Energia Elétrica**, Tese (Doutorado em Engenharia dos Materiais), Universidade Federal do Paraná, CURITIBA, 2005.

PEREIRA, Marluce R., AMORIM, Cláudio L., CASTRO, Maria Clicia Stelling, **Tutorial sobre Redes de Sensores**, Departamento de Informática e Ciência da Computação Instituto de Matemática e Estatística-CTC Universidade do Estado do Rio de Janeiro, Brasil, 2003.

RAGHUVANSHI, Manish, **Implementation of Wireless Sensor Mote** , Dissertação de Mestrado em Tecnologia, DEPARTMENT OF NUCLEAR ENGINEERING AND TECHNOLOGY INDIAN INSTITUTE OF TECHNOLOGY, KANPUR, INDIA, 2006.

ROSSI, Silvano Renato, **Implementação de um Nó IEEE 1451, baseado em ferramentas abertas e padronizadas, para aplicações em ambientes de instrumentação distribuída,** Tese (Doutorado em Engenharia Elétrica), UNESP, Ilha Solteira, 2005.

RUBIO, MÁRIO GONGORA, **Curso de Introdução à Instrumentação em Engenharia - Módulo Básico**, Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT Divisão de engenharia Mecânica, São Paulo, 2000.

SANDOVAL, Manuel E. Ruiz, **"Smart" Sensors for Civil Infrastructure Systems**, Tese (doutorado em Engenharia Civil e Ciências Geológicas), Universidade de Notre Dame, Indiana, EUA, 2004.

SATO, Flávio Hiochio, **Automação do Sistema de Carregamento em Ensaios Estruturais**, Dissertação (Mestrado em Engenharia Civil), UNESP - ILHA SOLTEIRA, 2002.

SEDRA e SMITH, **Microeletrônica**, 3 ed. , Prentice Hall, 2004.

SEIXAS FILHO, Constantino, **Capítulo 3 -Protocolos Orientados a caracter**, UFMG – Departamento de Engenharia Eletrônica, 2006.

SCHULZ, M.J.; SUNDARESAN, M.J., **Smart Sensor System for Structural Condition Monitoring of Wind Turbines**, National Renewable Energy Laboratory, May 30, 2002 — April 30, 2006.

SEIXAS, C., **Arquiteturas de sistemas de automação - Uma introdução.** endereço: http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/IIDownload.htm. Acesso em: 16/03/2007

SERRANO, L. M. V.; ALCOBIA, C. J. O. P. J.; MATEUS, M. L. O. S.; SILVA, M. C. G., **Sistemas de Aquisição, Processamento e Armazenamento de Dados**, Universidade de Coimbra, Portugal, 2003.

SIMUNIC, Zelimir; GASPARAC, Ivan; PAVLOVIC, Bozidar, “Implementation Of Digital Measurement System In Monitoring Of Structures”. **Construction Informatics Digital Library** , 2001, endereço: http://itc.scix.net/paper ecce-2001-33.content, acessado em agosto de 2007.

SMITH, Steven W., **The Scientist and Engineer’s Guide to Digital Signal Processing**, disponível no endereço: http://www.dspguide.com, acessado em janeiro de 2007.

SNOWDON, David, **Hardware and Software Infrastructure for the Optimisation of Sunswift II**, Monografia (Bacharelado em Engenharia), UNIVERSITY OF NEW SOUTH WALES, SCHOOL OF ELECTRICAL ENGINEERING & TELECOMMUNICATIONS SCHOOL OF COMPUTER SCIENCE & ENGINEERING, USA, 2002.

SPENCER, B. F., “Opportunities and Challenges for Smart Sensing Technology”, **Proceedings of the First International Conference on Structural Health Monitoring and Intelligent Infrastructure**, Tokyo, Japan, 2003.

SYMANS, Michael D. e KELLY, Steven W., **Fuzzy Logic Control of Bridge Structures Using Intelligent Semi-Active Seismic Isolation Systems**, Earthquake Engineering and Structural Dynamics, v. 28, 1999.

TANEMBAUM, A. S., **Redes de Computadores**. Editora Campus, Rio de Janeiro, 1997

THOMÉ, V., **Galerkin Finite Element Methods for Parabolic Problems**, Springer-Verlag, Berlin-Heidelber, 1997.

TRISTÃO, Ivanio Machado, **IMPLEMENTAÇÃO DO PROTOCOLO PROFIBUS PARA APLICAÇÕES INDUSTRIAIS BASEADAS EM MICROCONTROLADORES,** Monografia (Trabalho de Conclusão de Curso de Ciência da Computação) da Universidade Luterana do Brasil, câmpus Gravataí, 2004.

UCSD, endereço http://healthmonitoring.ucsd.edu, acessado em agosto de 2007

Projetos de Circuitos Integrados na Universidade Estadual de Londrina: Implementação Monolítica de Sensores de Corrente e Potência Elétrica. Projeto de Pesquisa cadastro no. 04379.

XU, Ning; RANGWALE, Sumit; CHINTALAPUDI, Krishn Kant; GANESAN, Deepak; BROAD, Alan; GOVINDAN, Ramesh; ESTRIN, Deborah, “A Wireless Sensor Network For Structural Monitoring”, **SenSys’04**, Baltimore, Maryland, USA. ACM, November 3–5, 2004.

WEBSTER, J.G., **Measurement, Instrumentation and Sensors Handbook**, CRC Press, 2617p, 1999.

WILLIAMS, C.B.; PAVIC, A.; CROUCH, R.S. e WOODS, R.C., “Feasibility study of vibration-electric generator for bridge vibration sensors.” **Proceedings of the 16th International Modal Analysis Conference IMAC**, pp. 1111– 1117, 1998.

WI-FI-ORG, **Wi-Fi Alliance**, endereço: http://www.wi-fi.org acessado em novembro de 2007.

ZEILMANN, R. P. **Uma estratégia Para Controle e Supervisão de Processos Industriais Via Internet**. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

ZIGBEE-ORG, **Zigbee Specification**, endereço : http://www.zigbee.org acessado em agosto de 2007.

ANEXO I - ESQUEMA ELÉTRICO COMPLETO DO TERMINAL DE REDE

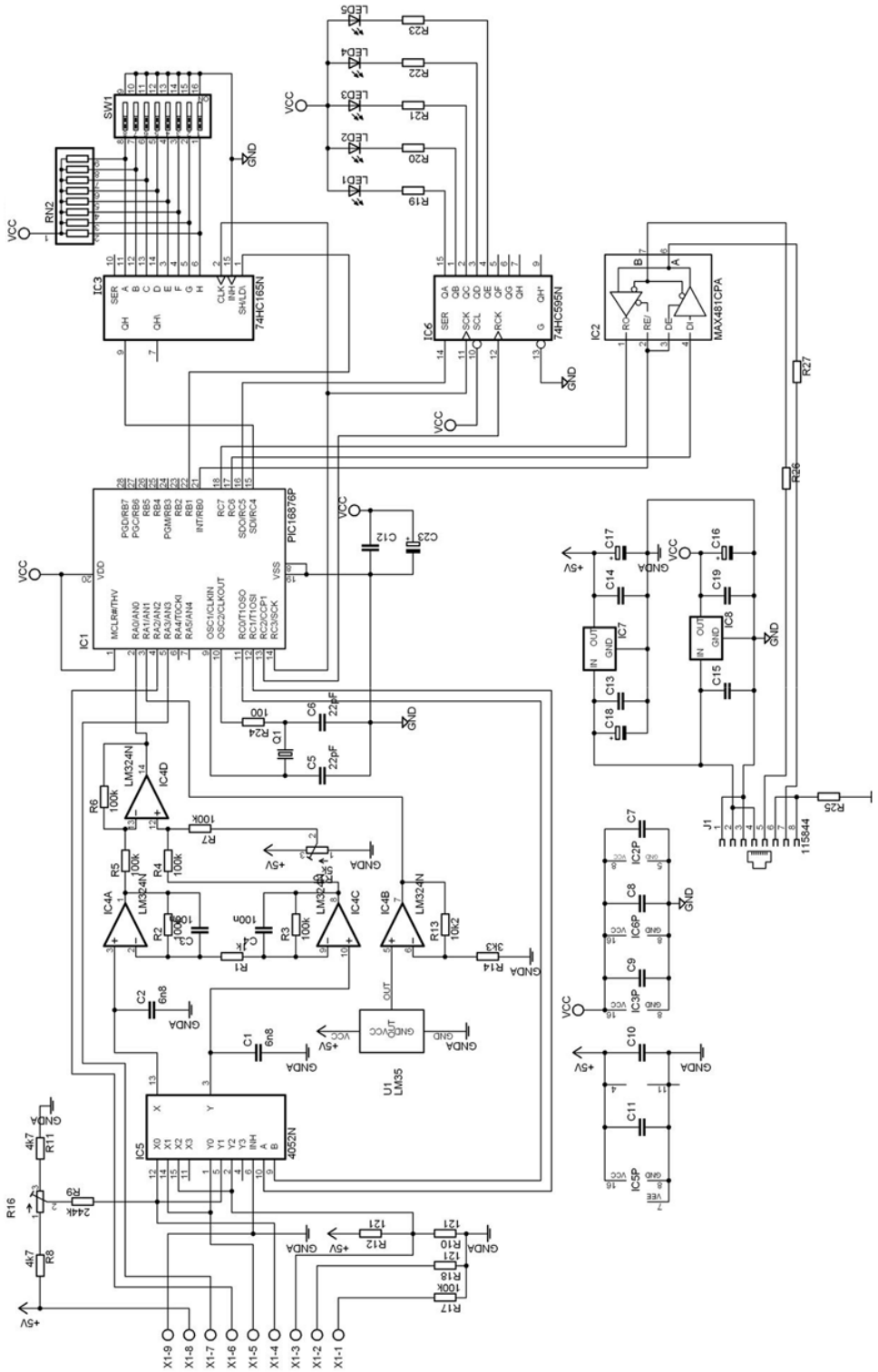

Figura 104: DIAGRAMA ESQUEMÁTICO COMPLETO DO NÓ DE REDE

ANEXO II - LISTAGEM DOS PROGRAMAS DO TERMINAL DE REDE

MB_SLAVE.H

```
/* ARQUIVO CABEÇALHO COM AS DEFINIÇÕES DAS VARIÁVEIS,
ENDEREÇOS DE DISPOSITIVOS DE ENTRADA E SAÍDA E
DEMAIS CONSTANTES UTILIZADAS POR TODOS OS PROGRAMAS DO SISTEMA */
#ifndef _MB_SLAVE
        #define _MB_SLAVE TRUE
#define _PIC
#define WDT_EN TRUE  // WDT HABILITADO
#ifdef _PIC
        #include <16F876.h>
        #device = *=16
        #device adc=10
        #use fast_io(A)
        #use fast_io(B)
        #use fast_io(C)
        #FUSES XT                       //Crystal osc <= 4mhz
        #FUSES PUT                      //Power Up Timer
        #FUSES NOPROTECT                //Code not protected from reading
        #FUSES BROWNOUT                 //Reset when brownout detected
        #FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
        #FUSES NOCPD                    //No EE protection
        #FUSES NOWRT                    //Program memory not write protected
        #FUSES NODEBUG                  //No Debug mode for ICD
#if WDT_EN
        #FUSES WDT                              // Watch Dog Timer ENABLED
        #use delay(clock=4000000,RESTART_WDT)
        #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK, ERRORS)
#else
        #FUSES NOWDT                            // Watch Dog Timer DISABLED
        #use delay(clock=4000000)
        #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK, ERRORS)
#endif
typedef int1                            BOOL; // 0 ou 1
typedef unsigned char           UCHAR;  // 0 - 255
typedef unsigned int16          USHORT;
typedef unsigned int32          ULONG;
#define LED_TX          1               //LED1 - QA
#define LED_RX          2               //LED2 - QB
#define LED_LEFT        16              //LED5 - QE
#define LED_CENTER      8               //LED4 - QD
#define LED_RIGHT       4               //LED3 - QC
#endif
#ifdef _8051
        sbit LED_RX                             = P2^0; // pino 21->R7->Q1->R9->D3
        sbit LED_TX             = P2^2; // pino 23
        sbit AF                 = P2^4;
        #define ENABLE_WDT              WMCON |= 1;
        #define RESET_WDT               WMCON |= 2;
        #define SET_TIME_WDT_128MS      WMCON &= (255-128-64-32); WMCON |= 128;
#endif
#define LIGADO                  0
#define DESLIGADO               1
#define BITS_UCHAR              8U
#ifndef TRUE
        #define TRUE            1
#endif
#ifndef FALSE
        #define FALSE           0
#endif
#define MB_REG_WRITE    0
#define MB_REG_READ             1
#define MB_ADDRESS_BROADCAST    0   /* ENDEREÇO PARA 'BROADCAST' DO MODBUS. */
#define MB_ADDRESS_MIN          1   /* MENOR ENDEREÇO POSSÍVEL  */
#define MB_ADDRESS_MAX          247 /* MAIOR ENDERECO POSSÍVEL */
#define MB_FUNC_NONE                            0
#define MB_FUNC_READ_COILS                      1
#define MB_FUNC_READ_DISCRETE_INPUTS            2
#define MB_FUNC_READ_HOLDING_REGISTER           3
#define MB_FUNC_READ_INPUT_REGISTER             4
```

```
#define MB_FUNC_WRITE_SINGLE_COIL                   5
#define MB_FUNC_WRITE_SINGLE_REGISTER               6
#define MB_FUNC_DIAG_READ_EXCEPTION                 7
#define MB_FUNC_DIAG_DIAGNOSTIC                     8
#define MB_FUNC_DIAG_GET_COM_EVENT_CNT              11
#define MB_FUNC_DIAG_GET_COM_EVENT_LOG              12
#define MB_FUNC_WRITE_MULTIPLE_COILS                15
#define MB_FUNC_WRITE_MULTIPLE_REGISTERS            16
#define MB_FUNC_OTHER_REPORT_SLAVEID                17
#define MB_FUNC_READWRITE_MULTIPLE_REGISTERS        23
#define MB_FUNC_READ_DEVICE_IDENTIFICATION                  43
#define MB_FUNC_ERROR                               128
//Modbus Coils  Bits, binary values, flags                  00001
//Digital Inputs        Binary inputs                               10001
//Analog Inputs         Analog inputs                               30001
//Modbus Registers      Analog values, variables            40001
#define MB_SER_PDU_ADDR_OFF     1       /* OFFSET DO ENDEREÇO DA ESTAÇÃO ESCRAVA NO QUADRO SER-PDU */
#define MB_SER_PDU_PDU_OFF      1       /* OFFSET DO MODBUS-PDU NO QUADRO SER-PDU. */
#define MB_PDU_SIZE_MAX     64  /* TAMANHO MÁXIMO DA PDU. */
#define MB_PDU_SIZE_MIN     1   /* CÓDIGO DA FUNÇÃO */
#define MB_PDU_FUNC_OFF     0   /* OFFSET DO CÓDIGO DA FUNÇÃO NA PDU. */
#define MB_PDU_DATA_OFF     1   /* OFFSET PARA O DADO DE RESPOSTA NA PDU */
/*------------------------------ READ ------------------------------*/
#define MB_PDU_FUNC_READ_SIZE                   ( 4 )
#define MB_PDU_FUNC_READ_REGCNT_MAX             ( 0x007D )
#define MB_PDU_FUNC_READ_DISCCNT_MAX            ( 0x07D0 )
#define MB_PDU_FUNC_WRITE_MUL_COILS_REGCNT_MAX      ( 0x0078 )
#define MB_PDU_FUNC_READ_RSP_BYTECNT_OFF        ( MB_PDU_DATA_OFF )
#define MB_PDU_FUNC_READ_ADDR_OFF               ( MB_PDU_DATA_OFF )
#define MB_PDU_FUNC_READ_REG_START_OFF          ( MB_PDU_DATA_OFF )
#define MB_PDU_FUNC_READ_REGCNT_OFF             ( MB_PDU_DATA_OFF + 2 )
#define MB_PDU_FUNC_READ_COILCNT_OFF            ( MB_PDU_DATA_OFF + 2 )
#define MB_PDU_FUNC_READ_DISCCNT_OFF            ( MB_PDU_DATA_OFF + 2 )
/*------------------------------ WRITE ------------------------------*/
#define MB_PDU_FUNC_WRITE_SIZE                  ( 4 )
#define MB_PDU_FUNC_WRITE_ADDR_OFF              ( MB_PDU_DATA_OFF )
#define MB_PDU_FUNC_WRITE_VALUE_OFF             ( MB_PDU_DATA_OFF + 2 )
/*------------------------------ WRITE MULT ------------------------------*/
/*------------------------------ READWRITE ------------------------------*/
#define MB_PDU_FUNC_READWRITE_SIZE_MIN          ( 9 )
#define MB_PDU_FUNC_READWRITE_READ_ADDR_OFF     ( MB_PDU_DATA_OFF )
#define MB_PDU_FUNC_READWRITE_WRITE_ADDR_OFF    ( MB_PDU_DATA_OFF + 4)
#define MB_PDU_FUNC_READWRITE_BYTECNT_OFF       ( MB_PDU_DATA_OFF + 8)
#define MB_PDU_FUNC_READWRITE_READ_REGCNT_OFF   ( MB_PDU_DATA_OFF + 2)
#define MB_PDU_FUNC_READWRITE_WRITE_REGCNT_OFF  ( MB_PDU_DATA_OFF + 6)
#define MB_PDU_FUNC_READWRITE_WRITE_VALUES_OFF  ( MB_PDU_DATA_OFF + 9)
#define MB_POS_START_PDU_RTU    1
#define MB_POS_START_PDU_ASCII  3       // ':' 'A(h) A(l) fnc
#ifndef MB_PORT_HAS_CLOSE
    #define MB_PORT_HAS_CLOSE 0
#endif
#define MB_FUNC_OTHER_REP_SLAVEID_BUF           ( 32 )
#ifndef _MBERRORCODE
    #define _MBERRORCODE TRUE
    typedef enum
    {
        MB_ER_NONE,                 /* SEM ERRO */
        MB_ER_INV_REG_ADDRESS,      /* ENDEREÇO IEGAL DE REGISTRADOR */
        MB_ER_INV_ARGUMENT,         /* ARGUMENTO ILEGAL */
        MB_ER_INVALID_PROTOCOL,             // PROTOCOLO_INVALIDO
        MB_ER_PORT_SERIAL,              // erro na porta serial
        MB_ER_PORT_TIMER,               // erro no timer
        MB_ER_PORT_EVENT,                   // ERRO EM EVENTOS
        MB_ER_TOO_LATE,    // Recebeu outra transmissão antes de enviar a resposta anterior
        MB_ER_NO_RESOURCES,         /* RECURSOS INSUFICIENTES */
        MB_ER_IO,                   /* ERRO DE ENTRADA OU SAIDA */
        MB_ER_ILLSTATE,             /* A CAMADA DO PROTOCOLO ESTÁ EM UM ESTADO ILEGAL */
        MB_ER_TIMEDOUT,             /* OCORREU UM ERRO POR ESGOTAMENTO DE TEMPO */
        MB_ER_CRC,
        MB_ER_PDU_LENGTH
    } MBErrorCode;
#endif
#ifndef _MBEXCEPTION
    #define _MBEXCEPTION            TRUE
```

```
        typedef enum
        {
            MB_EX_NONE = 0x00,
            MB_EX_ILLEGAL_FUNCTION = 0x01,
            MB_EX_ILLEGAL_DATA_ADDRESS = 0x02,
            MB_EX_ILLEGAL_DATA_VALUE = 0x03,
            MB_EX_SLAVE_DEVICE_FAILURE = 0x04,
            MB_EX_ACKNOWLEDGE = 0x05,
            MB_EX_SLAVE_BUSY = 0x06,
            MB_EX_MEMORY_PARITY_ERROR = 0x08,
            MB_EX_GATEWAY_PATH_FAILED = 0x0A,
            MB_EX_GATEWAY_TGT_FAILED = 0x0B
        } MBException;
#endif
#ifndef _MBMODE
        #define         _MBMODE                 TRUE
        typedef enum
        {
            MB_RTU,                     /* MODO RTU */
            MB_ASCII,                   /* MODO ASCII */
            MB_TCP                      /* MODO TCP */
        } MBMode;
#endif
#ifndef _MBPARITY
        #define         _MBPARITY               TRUE
        typedef enum
        {
            MB_PAR_NONE,                /* SEM PARIDADE */
            MB_PAR_ODD,                 /* PARIDADE IMPAR */
            MB_PAR_EVEN                 /* PARIDADE PAR */
        } MBParity;
#endif
#ifndef _MBRCVSTATE
        #define         _MBRCVSTATE             TRUE
        typedef enum
        {
                STATE_RX_INIT=0,
            STATE_RX_IDLE,              /* O RECEPTOR ESTÁ NO ESTADO DE REPOUSO */
            STATE_RX_RCV,               /* UM QUADRO ESTÁ SENDO RECEBIDO */
                STATE_RX_ERROR
        } MBRcvState;
#endif
#ifndef _MSSNDSTATE
        #define         _MSSNDSTATE             TRUE
        typedef enum
        {
            STATE_TX_IDLE,              /* O TRANSMISSOR ESTÁ NO ESTADO DE REPOUSO */
            STATE_TX_TRANSMITTING       /* O TRANSMISSOR ESTÁ NO ESTADO TRANSMITINDO */
        } MBSndState;
#endif
#ifndef _STATE_MAIN
        #define         _STATE_MAIN     TRUE
        typedef enum {
                STATE_MAIN_IDLE=0,
                STATE_MAIN_FRAME_RECEIVED,
                STATE_MAIN_START_TX,
                STATE_MAIN_TRANSMITTING
        } state_main;
#endif
/*------------------ variáveis externas ------------------------------*/
#ifdef _PIC
static          MBErrorCode eMBErrorCode;
static          MBException eMBException;
static          MBMode          eMBCurrentMode;
static          state_main      estate_main;
static          MBSndState      eMBSndState;
static          MBRcvState      eMBRcvState;
static          UCHAR           var_output;
static          UCHAR           gucRcvAddress;          // ADDRESS = endereço recebido
static          UCHAR           gucRcvFunction;         //       FUNCTION = FUNÇÃO RECEBIDA
static          UCHAR           gucThisStationAddress; // endereço desta estação
static          UCHAR           gucSndBufferCount;
static          UCHAR           gucSndBufferPos;        // posição de transmissão do caracter atual
static          UCHAR           gucRcvBufferPos;        // posição de recebimento do caracter atual
```

```
static          UCHAR           gucPUCFrameLength;
static          UCHAR           gucRcvBuffer[MB_PDU_SIZE_MAX]; // buffer de recepção (também de TRANSMISSÃO)
static          BOOL            FLAG_FRAME_RECEIVED;
// static       BOOL            FRAME_RECEIVED;
static          BOOL            FRAME_TRANSMITTED;
static          BOOL            FLAG_STATE_RX_IDLE;
static          BOOL            FLAG_LED_RX_ON;
static          BOOL            LED_RX_ON;
static          BOOL            FLAG_LED_RX_OFF;
static          BOOL            LED_RX_OFF;
static          BOOL            INICIANDO = TRUE;
static          BOOL            CALIBRANDO_PONTE=FALSE;
static          BOOL            CALIBRANDO_SISTEMA=FALSE;
static          BOOL            CALIBRANDO_SHUNT=FALSE;
static          BOOL            RECALIBRANDO=FALSE;
static          BOOL            LENDO_TEMPERATURA=FALSE;
static          BOOL            LENDO_RECALIBRACAO;
static          BOOL            FIRST_READING;
static          BOOL            SISTEMA_CALIBRADO;
// static       BOOL            cRxErrorFlag = FALSE;
static          UCHAR           expected;
static          UCHAR           timeout;
static          USHORT          offset_atual;
#endif
/*------------------ funções externas ------------------------------------*/
void Start_TX(void);                                // serial.c
void Start_RX(void);                                // SERIAL.C
void Init_SERIAL(USHORT usBaudRate, MBParity eParity); // serial.c
void Liga_LED(UCHAR mask);
void Desliga_LED(UCHAR mask);
void spi_escreve_byte (UCHAR dado);
void Le_ADC(void);
#ifdef _8051
void enable_interrupts(UCHAR interruption); // main.c
void disable_interrupts(UCHAR interruption); // main.c
void set_timer2(USHORT timeout);          // portimer2.c
#endif
void Init_TIMER(void);  //porttimer2.c
void Start_timer(UCHAR timeout);          // porttimer2.c
void Start_timer2(UCHAR timeout);
void eMBRTUInit ( USHORT usBaudRate, MBParity eParity ); //mb_rtu.c
void eMBRTUStart( void );
MBErrorCode eMBRTUReceive(void);
MBErrorCode     eMBRTUSend(void);
#endif
```

MB_SLAVE.C

```
/*
Finalidade: programa principal
parâmetros: nenhum
retorno: nenhum
*/
#include <mb_slave.h>
#include <mb_config.h>
// algumas funções podem ser implementadas ou não
// o arquivo de cabeçalho MB_CONFIG.H é utilizado para configurar
// as linhas abaixo
#if FUNC_READ_COILS_ENABLED
        #include "Func_read_coils.c"
#endif
#if FUNC_READ_DISCRETE_INPUTS_ENABLED
        #include "Func_read_discrete_inputs.c"
#endif
#if FUNC_READ_HOLDING_REGISTER_ENABLED
        #include "Func_read_holding_register.c"
#endif
#if FUNC_READ_INPUT_REGISTER_ENABLED // 0X04
        #include "Func_read_input_register.c"
#endif
#if FUNC_WRITE_SINGLE_COIL_ENABLED
        #include "Func_write_single_coil.c"
#endif
#if FUNC_WRITE_SINGLE_REGISTER_ENABLED // 0X06
        #include "Func_write_single_register.c"
#endif
#if FUNC_DIAG_READ_EXCEPTION_ENABLED
        #include "Func_diag_read_exception.c"
#endif
#if FUNC_DIAG_DIAGNOSTIC_ENABLED
        #include "Func_diag_diagnostic.c"
#endif
#if FUNC_DIAG_GET_COM_EVENT_CNT_ENABLED
        #include "Func_diag_get_com_event_cnt.c"
#endif
#if FUNC_DIAG_GET_COM_EVENT_LOG_ENABLED
        #include "Func_diag_get_com_event_log.c"
#endif
#if FUNC_WRITE_MULTIPLE_COILS_ENABLED
        #include "Func_write_multiple_coils.c"
#endif
#if FUNC_WRITE_MULTIPLE_REGISTERS_ENABLED
        #include "Func_write_multiple_registers.c"
#endif
#if FUNC_OTHER_REPORT_SLAVEID_ENABLED
        #include "Func_other_report_slaveid.c"
#endif
#if FUNC_READWRITE_MULTIPLE_REGISTERS_ENABLED
        #include "Func_readwrite_multiple_registers.c"
#endif
#if FUNC_READ_DEVICE_IDENTIFICATION_ENABLED
        #include "Func_read_device_identification.c"
#endif
#include "mb_rtu.c"
#ifdef _PIC
        #include "porttimer2_PIC.c"
        #include "portserial_PIC.c"
        #include "spi.c"
        #include "portADC.c"
#endif
#ifdef _8051
        #include "porttimer2_8051.c"
        #include "portserial_8051.c"
        USHORT          gus3_5_timeout;
        MBErrorCode eMBErrorCode;
        MBException eMBException;
        MBMode          eMBCurrentMode;
        MBParity        eMBParity;
        state_main      estate_main;
        MBSndState      eMBSndState;
        MBRcvState      eMBRcvState;
        UCHAR           gucRcvAddress;          // ADDRESS = endereço recebido
        UCHAR           gucRcvFunction;         //       FUNCTION = FUNÇÃO RECEBIDA
        UCHAR           gucThisStationAddress; // endereço desta estação
idata   UCHAR           gucRcvBuffer[MB_PDU_SIZE_MAX]; // buffer de recepção
idata   UCHAR           *pucPUCFrame;
        UCHAR           gucPUCFrameLength;
        UCHAR           gucSndBufferCount;
        UCHAR           gucSndBufferPos;        // posição de transmissão do caracter atual
        UCHAR           gucRcvBufferPos;        // posição de recebimento do caracter atual
        BOOL            FRAME_RECEIVED=FALSE;
        BOOL            FRAME_TRANSMITTED=FALSE;
#endif
/*********************************/
BOOL Executa_Comando(void)
/*********************************/
{
        switch(gucRcvFunction)
      {
                #if FUNC_READ_COILS_ENABLED
          case  MB_FUNC_READ_COILS: //                   1
                                Func_read_coils();
                                break;
                #endif
                #if FUNC_READ_DISCRETE_INPUTS_ENABLED
          case  MB_FUNC_READ_DISCRETE_INPUTS://         2
```

```
                                Func_read_discrete_inputs();
                                break;
                #endif
                #if FUNC_READ_HOLDING_REGISTER_ENABLED
        case    MB_FUNC_READ_HOLDING_REGISTER://             3
                                Func_read_holding_register();
                                break;
                #endif
                #if FUNC_READ_INPUT_REGISTER_ENABLED
        case    MB_FUNC_READ_INPUT_REGISTER://               4
                                Func_read_input_register();
                                break;
                #endif
                #if FUNC_WRITE_SINGLE_COIL_ENABLED
        case    MB_FUNC_WRITE_SINGLE_COIL://                 5
                                Func_write_single_coil();
                                break;
                #endif
                #if FUNC_WRITE_SINGLE_REGISTER_ENABLED
        case    MB_FUNC_WRITE_SINGLE_REGISTER://             6
                                Func_write_single_register();
                                break;
                #endif
                #if FUNC_DIAG_READ_EXCEPTION_ENABLED
        case    MB_FUNC_DIAG_READ_EXCEPTION://               7
                                Func_diag_read_exception();
                                break;
                #endif
                #if FUNC_DIAG_DIAGNOSTIC_ENABLED
        case    MB_FUNC_DIAG_DIAGNOSTIC://                   8
                                Func_diag_diagnostic();
                                break;
                #endif
                #if FUNC_DIAG_GET_COM_EVENT_CNT_ENABLED
        case    MB_FUNC_DIAG_GET_COM_EVENT_CNT://                    11
                                Func_diag_get_com_event_cnt();
                                break;
                #endif
                #if FUNC_DIAG_GET_COM_EVENT_LOG_ENABLED
        case    MB_FUNC_DIAG_GET_COM_EVENT_LOG://                    12
                                Func_diag_get_com_event_log();
                                break;
                #endif
                #if FUNC_WRITE_MULTIPLE_COILS_ENABLED
        case    MB_FUNC_WRITE_MULTIPLE_COILS://                      15
                                Func_write_multiple_coils();
                                break;
                #endif
                #if FUNC_WRITE_MULTIPLE_REGISTERS_ENABLED
        case    MB_FUNC_WRITE_MULTIPLE_REGISTERS://                  16
                                Func_write_multiple_registers();
                                break;
                #endif
                #if FUNC_OTHER_REPORT_SLAVEID_ENABLED
                  case  MB_FUNC_OTHER_REPORT_SLAVEID://                      17
                                Func_other_report_slaveid();
                                break;
                #endif
                #if FUNC_READWRITE_MULTIPLE_REGISTERS_ENABLED
        case    MB_FUNC_READWRITE_MULTIPLE_REGISTERS://              23
                                Func_readwrite_multiple_registers();
                                break;
                #endif
                #if FUNC_READ_DEVICE_IDENTIFICATION_ENABLED
                 case MB_FUNC_READ_DEVICE_IDENTIFICATION:
                        Func_read_device_identification();    // 43 - sub codigo 14
                        break;
                #endif
        default:
                                eMBException=MB_EX_ILLEGAL_FUNCTION;
   }
        return (eMBException==MB_EX_NONE);
}
/*********************************/
MBErrorCode Decodifica_frame(void)
/*********************************/
{
        #if MB_ASCII_ENABLED
        if (gucRcvBuffer[0]==':') // MODBUS_ASCII
        {
                eMBCurrentMode = MB_ASCII;
                return eMBASCIIReceive();
        }
        else    // MODBUS_RTU
        #endif
        {
        #if MB_RTU_ENABLED
                eMBCurrentMode = MB_RTU;
                return eMBRTUReceive();
         /* esta função retorna:
            *pucRcvAddress =  o endereço de destino
                gusRcvFunction =  o código da função recebida
            *pusLength = tamanho total do PDU a ser processado, ou seja,
            tamanho do pdu -  byte total de bytes -  byte CRC
         */
        #endif
        }
}
/************************************/
```

```
BOOL Executa_Comando_Broadcasting(void)
/************************************/
{
//#define PIN_ENABLE_ADDRESS_CHANGING PIN_AX
//      if (CMD_SET_SLAVE_ID)
//      {
//              if (PIN_ENABLE_ADDRESS_CHANGING)
//                      eStatus = eMBSetSlaveID( 0x34, TRUE, gucMBSlaveID, 3 );
//      }
return TRUE;
}
/*********************************/
MBErrorCode  Processa_bilhete(void)
/*********************************/
{
        eMBErrorCode = MB_ER_NONE;
        eMBException = MB_EX_NONE;

        if (Decodifica_frame() == MB_ER_NONE) // se bilhete válido
        {
                if( gucRcvAddress == MB_ADDRESS_BROADCAST )
                {
                        Executa_Comando_Broadcasting();
                }

                if( gucRcvAddress == gucThisStationAddress )
                // se o bilhete é pra esta estação - processa. Caso contrário, ignora!
                {
                        Executa_Comando();
                }
        }
        return eMBErrorCode;
}
/************************************/
UCHAR Monta_bilhete_resposta(void)
/************************************/
{
        UCHAR eStatus;
        if      (
                ( gucRcvAddress == MB_ADDRESS_BROADCAST )    // SE BROADCASTING
                ||
                ( gucRcvAddress != gucThisStationAddress )  // OU NÃO É PRÁ ESTA ESTAÇÃO
                )
        {
                return FALSE;
        }
        #if MB_ASCII_ENABLED
        if (eMBCurrentMode == MB_ASCII)
        {
        eStatus = eMBASCIISend();
        }
        else // MB_RTU
        #endif
        {
        #if MB_RTU_ENABLED
                eStatus = eMBRTUSend();
         /* esta função retorna:
            *pucRcvAddress =  o endereço de destino
            gusRcvFunction =  o código da função recebida
            *pusLength = tamanho total do PDU a ser processado, ou seja,
            tamanho do pdu -  byte total de bytes -  byte CRC
         */
        #endif
        }
        return (eStatus == MB_ER_NONE);
}
/************************/
MBErrorCode eMBStart(void)
/************************/
{
#if MB_RTU_ENABLED
        if (eMBCurrentMode == MB_RTU)
        eMBRTUStart();
#endif
#if MB_ASCII_ENABLED
        if (eMBCurrentMode == MB_ASCII)
        eMBASCIIStart();
#endif
return MB_ER_NONE;
}
/************************************/
MBErrorCode      eMBInit(
                                MBMode eMode,
                                UCHAR ucSlaveAddress,
                                USHORT usBaudRate,
                                MBParity eParity
                                )
/************************************/
{
    MBErrorCode     eStatus = MB_ER_NONE;
    /* check preconditions */
    if( ( ucSlaveAddress == MB_ADDRESS_BROADCAST ) ||
        ( ucSlaveAddress < MB_ADDRESS_MIN ) || ( ucSlaveAddress > MB_ADDRESS_MAX ) )
    {
        eStatus = MB_ER_INV_ARGUMENT;
    }
        gucThisStationAddress = ucSlaveAddress;
    switch ( eMode )
    {
```

```
#if MB_RTU_ENABLED
    case MB_RTU:
        eMBRTUInit(usBaudRate, eParity );
        break;
#endif
#if MB_ASCII_ENABLED
    case MB_ASCII:
        eMBASCIIInit(usBaudRate, eParity );
        break;
#endif
    default:
        eStatus = MB_ER_INVALID_PROTOCOL;        //PROTOCOLO_INVALIDO;
    }
    if( eStatus == MB_ER_NONE )
    {
      eMBCurrentMode = eMode;
    }
    return eStatus;
}

/******************/
void Init_sys(void)
/******************/
{
/*
Finalidade: inicialização das variáveis do sistema
parâmetros: nenhum
retorno: nenhum
*/
#ifdef _PIC
        Init_ADC(); // INICIALIZA PORTA
        set_tris_b(0b01111100);
// 0111 1100 = 0x7C      B0–B1=SAÍDAS; B2–B6=ENTRADAS;  B7=SAÍDA;
        output_high(PIN_B7);
        output_high(PIN_B1);
        output_high(PIN_B0);
        set_tris_c(0b11010000);
// 1101 0000 = 0xD0      C0–C3=SAÍDAS;  C4(SDI)=ENTRADA; C5 (SDO)=SAÍDA; C6,C7 =TX/RX (ENTRADAS)
        output_high(PIN_C5);
        output_high(PIN_C3);
        output_high(PIN_C2);
        output_high(PIN_C1);
        output_high(PIN_C0);
        setup_spi(FALSE);
        setup_timer_0(RTCC_INTERNAL);
        #if WDT_EN
                setup_wdt(WDT_144MS);
        #endif
//      setup_timer_0(RTCC_INTERNAL); // o timer 0 é utilizado pelo WDT
        setup_timer_1(T1_DISABLED);
        setup_timer_2(T2_DISABLED,0,1);
#endif
#ifdef _8051
/***
REGISTRO IE = Interrupt Enable
        7       6       5       4       3       2       1       0
        EA      X       X       ES      ET1     EX1     ET0     EX0
        1       0       0       0       0       0       0       0
***/
IE=0;
/***
REGISTRO IP = Interrupt Priority
        7       6       5       4       3       2       1       0
        X       X       X       PS      PT1     PX1     PT0     PX0
        0       0       0       0       0       0       1       0
****/
IP=0;
/***
REGISTRO TMOD = Timer Mode
        7       6       5       4       3       2       1       0
        GATE1   C/~T–1  M1–1    M0–1    GATE–0  C/~T–0  M1–0    M0–0
        0       0       1       0       0       0       0       1
                        32                                      1
timer0     MODO 1 M1.0=0 E M0.0=1 – 16 BITS C/ RECARGA P/ SOFTWARE
timer1     MODO 2 M1.1=1 E M0.1=0 – 8 BITS C/ RECARGA AUTOMATICA
***/
//#define T0_CT_   0x04   /* Timer 0 Counter/Timer Select: 0=Counter, 1=Timer */
TMOD=33;
/***
REGISTRO TCON = Timer Control
        7       6       5       4       3       2       1       0
        TF1     TR1     TF0     TR0     IE1     IT1     IE0     IT0
        0        0      0       0       0       0       0       0
***/
TCON=0;
//EN_LIGHT = RS = 1; // TO and T1 PINS
SET_TIME_WDT_128MS;
ENABLE_WDT;
#endif
}
#ifdef _8051
/************************/
void delay_ms(UCHAR tempo)
/************************/
{
unsigned int i;
for(i=0;i<5000;i++);
}
/********************/
```

```
void restart_wdt(void)
/*********************/
{
        RESET_WDT;
}
#endif
/*****************************************/
void Salva_parametro_calibracao_sistema(void)
/*****************************************/
{
/*
Finalidade: salvar os parâmetros de calibração do sistema
*/
        // salva temperatura em que foi calibrado
        gusRegHolding[VALOR_TEMPERATURA_CALIBRACAO_SISTEMA]=
                        gusRegInputRegister[REG_INPUT_TEMPERATURE_VALUE];
        gusRegHolding[VALOR_REFERENCIA_CALIBRACAO]=offset_atual;
        SISTEMA_CALIBRADO=TRUE;
}
/*****************************************/
void Salva_parametro_calibracao_ponte(void)
/*****************************************/
{
/*
Finalidade:  salvar a temperatura em que o sistema foi calibrado
*/
        gusRegHolding[VALOR_TEMPERATURA_CALIBRACAO_PONTE]=
                        gusRegInputRegister[REG_INPUT_TEMPERATURE_VALUE];
}
/*****************************************/
void Salva_parametro_calibracao_shunt(void)
/*****************************************/
{
/*
Finalidade: salva o valor e a temperatura em que foi calibrado
*/
       gusRegHolding[VALOR_TEMPERATURA_CALIBRACAO_SHUNT]=
                        gusRegInputRegister[REG_INPUT_TEMPERATURE_VALUE];
        gusRegHolding[VALOR_SHUNT_CALIBRACAO]=
                        gusRegInputRegister[REG_INPUT_STRAIN_VALUE];
}
/***********************************/
void Processa_chaves(UCHAR var_input)
/***********************************/
/*
Finalidade: detectar qual chave foi selecionada e
                        tomar as decisões correspondentes
00 = operação normal
01 = calibração do sistema
        equivale a fazer com que as duas entradas fiquem curto-circuitadas
        o que é feito com o sinal 02
02 = calibração da ponte
        o objetivo é zerar a tensão da ponte
03 = (chave 1 + chave 2)
        =resistor shunt
        = tomar o valor de calibração

*/
{
        BOOL DIP1, DIP2, DIP3, DESLIGA_LEDS;
        USHORT  valor_atual, valor_ref_minimo, valor_ref_maximo;
        DIP1 =  (var_input & 128)?1:0;
        DIP2 =  (var_input & 64)?1:0;
        DIP3 =  (var_input & 32)?1:0;
        DESLIGA_LEDS=FALSE;

        if (CALIBRANDO_SISTEMA & !DIP1 ) // ESTAVA SOMENTE COM A CHAVE 1 ATIVADA
        {
                // verifica se desligou a chave 1
                Salva_parametro_calibracao_sistema();
                DESLIGA_LEDS=TRUE;
        }

        if (CALIBRANDO_PONTE & !DIP2) // ESTAVA SOMENTE COM A CHAVE 2 ATIVADA
        {
                // verifica se desligou a chave 2
                Salva_parametro_calibracao_ponte();
                DESLIGA_LEDS=TRUE;
        }

        if (CALIBRANDO_SHUNT & !DIP3) // ESTAVA SOMENTE COM A CHAVE 3 ATIVADA
        {
                // verifica se desligou a chave 3
                Salva_parametro_calibracao_shunt();
                DESLIGA_LEDS=TRUE;
        }

        CALIBRANDO_SISTEMA =(   DIP1 & !DIP2 & !DIP3); // SÓ PODE ESTAR LIGADA A CHAVE 1
        CALIBRANDO_PONTE   =( !DIP1 &  DIP2 & !DIP3); // SÓ PODE ESTAR LIGADA A CHAVE 2
        CALIBRANDO_SHUNT   =( !DIP1 & !DIP2 &  DIP3);  // só CHAVES 3 LIGADAS

        if (CALIBRANDO_SISTEMA | CALIBRANDO_PONTE)
        {
            valor_ref_minimo=
              gusRegHolding[VALOR_REFERENCIA_CALIBRACAO]-gusRegHolding[VALOR_OFFSET_REFERENCIA_MAXIMO];
            valor_ref_maximo=
              gusRegHolding[VALOR_REFERENCIA_CALIBRACAO]+gusRegHolding[VALOR_OFFSET_REFERENCIA_MAXIMO];
            valor_atual = gusRegInputRegister[REG_INPUT_STRAIN_VALUE];
                if (valor_atual < valor_ref_minimo)
```

```
                {
                        Liga_LED(LED_LEFT);
                        Desliga_LED(LED_CENTER);
                        Desliga_LED(LED_RIGHT);
                }
                else if (valor_atual > valor_ref_maximo)
                {
                        Desliga_LED(LED_LEFT);
                        Desliga_LED(LED_CENTER);
                        Liga_LED(LED_RIGHT);
                }
                else
                {
                        Desliga_LED(LED_LEFT);
                        Liga_LED(LED_CENTER);
                        Desliga_LED(LED_RIGHT);
                }
        }

        if (DESLIGA_LEDS)
        {
                Desliga_LED(LED_LEFT);
                Desliga_LED(LED_CENTER);
                Desliga_LED(LED_RIGHT);
        }
}
/*************/
void main(void)
/*************/
{
/*
programa principal
*/
        USHORT usbaudrate;
        UCHAR var_input, old_var_input=0;
        BOOL CHANGED=0;
        delay_ms(10); // aguarda estabilização
        Init_sys(); // inicializa o sistema
        var_output = 255;               // todos os leds apagados
        spi_escreve_byte (var_output);  // envia para a saída
        var_input = spi_le_byte();       // lê DIP SWITCH

        gucThisStationAddress = (var_input & 31);
        if (gucThisStationAddress == 0)
                gucThisStationAddress = 31; //começa com 31
        usbaudrate=9600;  // velocidade padrão MODBUS RTU
        eMBInit(MB_RTU, gucThisStationAddress, usbaudrate, MB_PAR_NONE);
        eMBStart();
        for(;;)
        {
                #if WDT_EN
                        restart_wdt();      // LIMPA O WDT
                #endif
                if (!INICIANDO)
                        var_input = spi_le_byte();

                if (old_var_input != var_input)
                {
                        old_var_input = var_input;
                        CHANGED=TRUE;
                }
                else
                        CHANGED = FALSE;
                if (CHANGED | CALIBRANDO_PONTE | CALIBRANDO_SISTEMA | CALIBRANDO_SHUNT)
                        Processa_chaves(var_input);

                /*-----------------------------------------------*/
                /* TRATA AS VARIÁVEIS QUE PODEM SER CONFLITADAS NAS INTERRUPÇÕES */
                /*-------------------------------*/
                disable_interrupts(GLOBAL);
                /*-------------------------------*/
                /* VARIÁVEIS UTILIZADAS EM ISR_INT1
                        *************
                        FLAG_STATE_RX_IDLE=TRUE; (eMBRcvState)
                        FLAG_FRAME_RECEIVED = TRUE;
                        FLAG_LED_RX_OFF=TRUE;
                        *************
                */
                if (FLAG_STATE_RX_IDLE)
                {
                        FLAG_STATE_RX_IDLE=FALSE;
                        eMBRcvState = STATE_RX_IDLE;
                }

                if (FLAG_FRAME_RECEIVED)
                {
                        FLAG_FRAME_RECEIVED=FALSE;
                        estate_main=STATE_MAIN_FRAME_RECEIVED;
                }

                if (FLAG_LED_RX_ON)
                {
                        FLAG_LED_RX_ON=FALSE;
                        LED_RX_ON=TRUE;
                }

                if (FLAG_LED_RX_OFF)
                {
                        FLAG_LED_RX_OFF=FALSE;
```

```
                LED_RX_OFF=TRUE;
        }

        /*-------------------------------*/
        enable_interrupts(GLOBAL);
        /*-------------------------------*/

        if (LED_RX_ON)
        {
                LED_RX_ON=FALSE;
                Liga_LED(LED_RX);
        }

        if (LED_RX_OFF)
        {
                LED_RX_OFF=FALSE;
                Desliga_LED(LED_RX);
        }

        switch(estate_main)
        {
        case STATE_MAIN_IDLE:
                        if ( INICIANDO  )
                        {
                                disable_interrupts(GLOBAL);
                                if (eMBRcvState == STATE_RX_IDLE)
                                {
                                        INICIANDO=FALSE;
                                        HABILITA_TX
                                        printf("MODBUS RTU - VALTER L.A. CAMARGO\r\n");
                                        HABILITA_RX
                                        Start_ADC();//DÁ INÍCIO AO PROCESSO DE LEITURA ADC
                                }
                                enable_interrupts(GLOBAL);
                        }
                break;
        case STATE_MAIN_FRAME_RECEIVED:
                if (Processa_bilhete() == MB_ER_NONE) // SE RECEBEU UM BILHETE VÁLIDO
                        {
                                if (Monta_bilhete_resposta())
                                        estate_main = STATE_MAIN_START_TX;
                                else
                                        estate_main = STATE_MAIN_IDLE;
                        }
                        else
                        {
                                estate_main = STATE_MAIN_IDLE;
                        }
                        break;
                case STATE_MAIN_START_TX:
                        Liga_LED(LED_TX);
                        delay_ms(4);
                        HABILITA_TX
                        enable_interrupts(INT_TBE);       //HABILITA FLAG
                        while (!FRAME_TRANSMITTED);
                        // fica aqui até terminar a transmissão do bilhete
                        delay_ms(2);
                        HABILITA_RX
                        Desliga_LED(LED_TX);
                        estate_main = STATE_MAIN_IDLE;
        }
    }
}
```

PORTADC.C

```
#ifndef _ADC
        #define _ADC
#define CHANNEL_TEMP    0
#define CHANNEL_STRAIN  1
/**********
funcionamento geral do módulo ADC
        – É INICIALIZADO (Start_ADC dentro de main, depois de enviar a mensagem inicial)
           HABILITANDO O TIMER2 PARA INTERROMPER A CADA 10 MS
        –ISR_TIMER2()
                        seta o canal
                        e liga o timer0 para aguardar estabilização (20 us)
        –ISR_TIMER0()
                         inicia a leitura do ADC
        –ISR_ADC
                         coleta o valor lido
*******/

/**********************/
void Init_ADC(void)
/**********************/
{
/* FINALIDADE: Inicializar o ADC */
        set_tris_a(0b11111111);
        setup_adc_ports(AN0_AN1_AN4_VREF_VREF);
        setup_adc(ADC_CLOCK_INTERNAL);
}
/************************/
void Start_ADC()
/************************/
{
        /*  a leitura do adc é comandada pelo timer2 que é interrompido a cada 10 ms  */
        setup_timer_2(T2_DIV_BY_4,250,10); // 10000 cycles
        clear_interrupt(INT_TIMER2);
        enable_interrupts(INT_TIMER2);
}
/******************/
void Le_ADC(void)
/******************/
{
        if (CALIBRANDO_SISTEMA | LENDO_RECALIBRACAO)
        {
                output_high(PIN_C0);
                output_low(PIN_C1);
        }
        else
        {
                output_low(PIN_C0);
                if (gusRegHolding[REVERSE_READING_ENABLED])
                {
                        if (FIRST_READING)
                        {
                                output_low(PIN_C1);
                        }
                        else
                        {
                                output_high(PIN_C1);
                        }
                }
                else
                        output_low(PIN_C1);
        }
        if (LENDO_TEMPERATURA)
                set_adc_channel(CHANNEL_TEMP);
        else
                set_adc_channel(CHANNEL_STRAIN);
        /*
        É PRECISO AGUARDAR PELO MENOS 20US até que o sinal se estabilize,
        antes de fazer a leitura.
        */
        delay_us(20);
        clear_interrupt(INT_AD);
        enable_interrupts(INT_AD);
        read_adc(ADC_START_ONLY);  // SOMENTE inicia a conversão
}
/****************/
#int_AD
void AD_isr(void)
/***************/
{
        static UCHAR
                        num_amostras=0,
                        pos_amostra_atual=0,
                        cont_aux_temp=0,
                        ucWindow_size;
        static signed int16
                        diferenca_offset;
        static USHORT
                ACC=0,
                acc_cal=0,
                acc_temp=0,
                temperatura_atual,
                offset_atual,
                val1,
                ustemp;

        clear_interrupt(INT_AD);
        if (FIRST_READING)
        {
```

```
        val1=read_adc(ADC_READ_ONLY);
        FIRST_READING=FALSE;
        Le_ADC();
        return;
}
ustemp=read_adc(ADC_READ_ONLY) + val1;
ustemp = (ustemp >> 1); // divide por dois - duas leituras - uma única amostra
/*********************************/
if (ucWindow_Size != gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL])
{
        ucWindow_Size = (UCHAR) gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL];
        ACC = num_amostras = pos_amostra_atual = 0;
}

if ( (ucWindow_Size < 1) || (ucWindow_Size > 32) )
{
        gusRegHolding[NUMERO_PONTOS_JANELA_MEDIA_MOVEL] = ucWindow_Size = 8;
}
/*********************************************/
if (LENDO_RECALIBRACAO)
{
        acc_cal += ustemp;
        LENDO_RECALIBRACAO=FALSE;
        LENDO_TEMPERATURA=TRUE;
        FIRST_READING=TRUE;
        Le_ADC();
}
else if (LENDO_TEMPERATURA) // trata o canal 'temperature'
{
        acc_temp += ustemp;
        LENDO_TEMPERATURA=FALSE;
}
else  // trata o canal 'strain'
{
        if ( ucWindow_Size == 1) // não está utilizando filtro média móvel
        {
                gusRegInputRegister[REG_INPUT_STRAIN_VALUE] = ustemp;
        }
        else  // está utilizando filtro média móvel
        {
                if (num_amostras < ucWindow_Size)
                {
                        num_amostras++;
                }
                else
                {
                   ACC -=  gusRegInputRegister[pos_amostra_atual+OFFSET_SAMPLE];
                   // elimina amostra mais antiga
                }
                gusRegInputRegister[pos_amostra_atual+OFFSET_SAMPLE] = ustemp ;
                // duas leituras, uma só amostra

                ACC += ustemp; // soma a amostra atual

                gusRegInputRegister[REG_INPUT_STRAIN_VALUE] = ACC/num_amostras;
                if (SISTEMA_CALIBRADO && gusRegHolding[RECALIBRATION_ENABLED])
                {
                   gusRegInputRegister[REG_INPUT_STRAIN_VALUE] += diferenca_offset;
                }
                if (++pos_amostra_atual == ucWindow_Size)
                        pos_amostra_atual = 0;
        }
        if (gusRegHolding[RECALIBRATION_ENABLED])
        {
                if (RECALIBRANDO)
                {
                        LENDO_RECALIBRACAO=TRUE; // AGORA VAI LER A recalibração
                        FIRST_READING=TRUE;
                        Le_ADC();
                }
                cont_aux_temp++;
                if (cont_aux_temp == 250)
                {
                        cont_aux_temp = 0;
                        acc_cal=0;
                        acc_temp=0;
                        RECALIBRANDO=TRUE;
                }
                else if (cont_aux_temp == 8) // 8 AMOSTRAS
                {
                        if (SISTEMA_CALIBRADO)
                        {
                          offset_atual = (acc_cal >> 3);
                          diferenca_offset = (signed int16)
                             (gusRegHolding[VALOR_REFERENCIA_CALIBRACAO] - offset_atual);
                          gusRegInputRegister[REG_OFFSET_DIF] = diferenca_offset;
                        }
                        temperatura_atual= (acc_temp >> 3);
                        gusRegInputRegister[REG_INPUT_TEMPERATURE_VALUE] = temperatura_atual ;
                        RECALIBRANDO=FALSE;
                }
        }
        else
                cont_aux_temp = 0;
}
/* em algum outro lugar deverá ter sido emitido o comando:
        Le_ADC()
  modos:
  ADC_START_AND_READ (this is the default)
```

```
            ADC_START_ONLY (starts the conversion and returns)
            ADC_READ_ONLY (reads last conversion result)
          */
}
#endif
```

```
MB_RTU.H
#ifndef _MB_RTU_H
        #define _MB_RTU_H TRUE
        /* ----------------------------- Defines ----------------------------------------------------*/
        #define MB_SER_PDU_SIZE_MIN     4       /* TAMANHO MÍNIMO DO QUADRO Modbus RTU */
        #define MB_SER_PDU_SIZE_MAX     256     /*!TAMANHO MÁXIMO DO QUADRO Modbus RTU */
        #define MB_SER_PDU_SIZE_CRC     2       /* TAMANHO DO CAMPO CRC */
        #define MB_RTU_POS_ADDRESS              0
        #define MB_RTU_POS_FUNCTION             1
        #define MB_RTU_POS_REG_START    2
        #define MB_RTU_POS_REG_COUNT    3
        #define MB_RTU_POS_CRC                  4
//OS vetores a seguir fazem o look-up table para o cálculo do CRC
const UCHAR  aucCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
/* CRC16 Table Low byte */
const  UCHAR aucCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2,
0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6,
0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE,
0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA,
0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62,
0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE,
0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76,
0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A,
0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86,
0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

#endif
```

MB_RTU.C

```
#include <mb_slave.h>
#include <mb_rtu.h>
/*********************************************/
USHORT usMBCRC16( char *pucFrame, USHORT usLen )
/*********************************************/
{
    UCHAR             ucCRCHi = 0xFF;
    UCHAR             ucCRCLo = 0xFF;
    UCHAR                 iIndex=0;
        USHORT                  usret;
    while( usLen-- )
    {
        iIndex =  ucCRCLo ^ *pucFrame++;
        ucCRCLo = ucCRCHi ^ aucCRCHi[iIndex];
        ucCRCHi = aucCRCLo[iIndex];
    }
        usret = (ucCRCHi << 8);
        usret += ucCRCLo;
// return (ucCRCHi << 8 | ucCRCLo); // desta maneira não funciona = por quê???????

        return (usret);
}
/*********************************************************/
void eMBRTUInit ( USHORT usBaudRate, MBParity eParity )
/*********************************************************/
{
    Init_SERIAL( usBaudRate, eParity );
        Init_TIMER();
}
/**********************/
void eMBRTUStart( void )
/**********************/
{
        estate_main = STATE_MAIN_IDLE;
        eMBSndState = STATE_TX_IDLE;
        eMBRcvState = STATE_RX_INIT;

     /* INICIALMENTE O RECEPTOR ESTÁ NO ESTADO STATE_RX_INIT.
      liga-se o temporizador e aguarda-se até que transcorra
      o tempo de 3.5 caracteres. Caso não haja recebimento de nenhum caracter
      dentro desse tempo, muda-se o estado para STATE_RX_IDLE.
      Isto garante que as camadas do protocolo só irão estar ativas
      se o barramento estiver livre.
     */
        Start_timer(1);
}
/*****************************/
MBErrorCode eMBRTUReceive(void)
/*****************************/
{
        // CHECH LENGTH
        if (gucRcvBufferPos < MB_SER_PDU_SIZE_MIN)
        {
                eMBErrorCode = MB_ER_PDU_LENGTH;
        }
        else
        {
                gucRcvAddress =  gucRcvBuffer[0];       // endereço
                gucRcvFunction = gucRcvBuffer[1];       // função
                gucPUCFrameLength = gucRcvBufferPos - 3;
        }
    /* Verificação do CRC */
        if  (usMBCRC16(gucRcvBuffer, gucRcvBufferPos) )
                {
                        eMBErrorCode = MB_ER_CRC;
                }
    return (eMBErrorCode);
}
/*****************************************************/
MBErrorCode     eMBRTUSend(void)
/*****************************************************/
{
    USHORT          usCRC16;
 /*  verifica se o receptor está no estado de repouso (IDLE_STATE).
    Caso não esteja, significa que o processamento do bilhete
    está sendo muito lento e o mestre já está enviando outro bilhete
    mesmo sem ter processado o anterior.
   então deve-se abortar o envio do bilhete de resposta atual
*/
      if( eMBRcvState != STATE_RX_IDLE )
      {
                /*
                        O primeiro byte a ser transmitido antes do Modbus-PDU
                                é o endereço do escravo
                                que já está armazenado em gucRcvBuffer[0]
                */
        if( eMBException != MB_EX_NONE )
                {
                /* Ocorreu uma exceção. Deve ser construído um quadro de erro */
                gucRcvBuffer[1] |= MB_FUNC_ERROR;// começa em 1
                gucRcvBuffer[2] = eMBException;
                        gucPUCFrameLength=2;
                }
          usCRC16 = usMBCRC16(gucRcvBuffer, gucPUCFrameLength + 1 );
                // +1 porque inicia com o número do terminal
          gucRcvBuffer[gucPUCFrameLength + 1]=make8(usCRC16,0);
                // reg. value LSB
          gucRcvBuffer[gucPUCFrameLength + 2]=make8(usCRC16,1);
```

```
                // reg. value MSB
            gucSndBufferCount = gucPUCFrameLength + 3;
                // 2 bytes crc + 1 byte endereço
        }
        else
        {
        eMBErrorCode = MB_ER_TOO_LATE;
        }
//
          return (eMBErrorCode);
}
```

PORTSERIAL_PIC.C

```
#ifndef _PORT_SERIAL
        #define _PORT_SERIAL TRUE
        #include "mb_slave.h"
        #define HABILITA_TX     disable_interrupts(INT_RDA); output_high(PIN_B0);
        #define HABILITA_RX     output_low(PIN_B0); enable_interrupts(INT_RDA);


#int_RDA
/*****************/
void ISR_RDA(void)
/*****************/
{
        UCHAR caracter;
        caracter = getc();

    switch ( eMBRcvState )
    {
        case STATE_RX_INIT:
                    /*-------------------------------------------------------------------
                            Se foi recebido um caracter quando estava inicializando,
                            então deve se aguardar até que o quadro termine.
                    -------------------------------------------------------------------*/
        case STATE_RX_ERROR:
                    /*-------------------------------------------------------------------
                            Se houve algum erro, deve-se esperar que todos os caracteres
                       do quadro defeituoso seja recebido.
                    -------------------------------------------------------------------*/
                        timeout = 1;
                break;
            case STATE_RX_IDLE:

                        /*-------------------------------------------------------------
                         No estado repouso aguarda-se a chegada de um novo caracter.
                            Quando for for recebido o primeiro caracter,
                        grava-o na posição inicial do buffer,
                            habilita os temporizadores e
                            muda o estado para STATE_RX_RCV.
                    -------------------------------------------------------------*/
                        expected = 8; //fnc 1 a 6 - numero de caracteres esperados para essas funções
                        timeout = 8; // 4 x 4192 ms = 16384 us
                gucRcvBufferPos = 0;
                        FLAG_LED_RX_ON =TRUE; // FLAG PARA LIGAR O LED RX (EM MAIN.C)
                        eMBRcvState = STATE_RX_RCV;
            case STATE_RX_RCV:
                        /*-------------------------------------------------------------------------------
                            Se foi recebido mais caracteres do que o máximo número possível de bytes
                            do quadro modbus, o quadro inteiro deve ser ignorado.
                    -------------------------------------------------------------------------------*/
                        if( gucRcvBufferPos > 7)
                        {       // já foi recebido o endereço de destino e o código da função
                            if ( (gucRcvBuffer[1] == 15) || (gucRcvBuffer[1] == 16) )
                            {
                            /*-----------------------------------------------------------
                            15 =  write N coil;
                            16 =  write N regs;
                              a contagem dos bytes é: SS FF AA AA LL LL BC ?? CR CR
                            -----------------------------------------------------------*/
                            expected = 9 + gucRcvBuffer[6];
                        }
                        }
                if( gucRcvBufferPos > expected )
                {
                    eMBRcvState = STATE_RX_ERROR;
                        }
                        else
                        {
                    gucRcvBuffer[gucRcvBufferPos++] = caracter;
                }
                        if (gucRcvBufferPos == expected )
                            timeout = 1; //old=gus3_5_timeout;
    }// endswitch
        /*-------------------------------------------------------------------
                Reinicializa o temporizador a cada caracter recebido.
                Quando houver demora maior do que 3,5 vezes o tempo de cada quadro,
                o temporizador colocará a máquina de recepção no estado inicial
               -------------------------------------------------------------------*/
        Reset_timer(timeout);
}
/*************************/
#int_TBE
void ISR_TBE(void)
/*************************/
{
//      while (!txsta.trmt);    //aguarda até que a transmissão esteja completa (buffer empty)
        switch(eMBSndState)
        {
            case STATE_TX_IDLE:
                        eMBSndState = STATE_TX_TRANSMITTING;
                        FRAME_TRANSMITTED = FALSE;
                        gucSndBufferPos=0;
            case STATE_TX_TRANSMITTING:
                        if (gucSndBufferPos < gucSndBufferCount )
                            {
                            putc(gucRcvBuffer[gucSndBufferPos++]);
                            }
                        else
                        {
                            disable_interrupts(INT_TBE);
```

```
                                FRAME_TRANSMITTED = TRUE;
                                eMBSndState=STATE_TX_IDLE;
                            }
        }
}
#endif
```

PORTTIMER2_PIC.C

```
#ifndef _PORT_TIMER
        #define _PORT_TIMER                TRUE
#include "mb_slave.h"
/****************/
void Init_TIMER()
/****************/
{
        setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
        enable_interrupts(GLOBAL);
}
/*************************************/
void Start_timer(UCHAR timeout)
/*************************************/
{
/*
 timeout:
                1 = 4096 cycles
                2 = 8192 cycles
                4 = 16384 cycles
*/
        USHORT tempo;
        switch(timeout)
        {
                case 1:
                        tempo = 0xF000; //65535-4095;
                        break;
                case 2:
                        tempo = 0xE000; //65535-8191;
                        break;
                case 4:
                        tempo = 0xC000; //65535-16383;
                        break;
                case 8: // 8 TIMES
                        tempo = 0x8000; //65535-32767;
                        break;
                default: // 12 TIMES
                        tempo = 0x3000; // 53247 us;
        }
        set_timer1(tempo);
        clear_interrupt(INT_TIMER1);
        enable_interrupts(INT_TIMER1);
}
/*************************************/
void Reset_timer(UCHAR timeout)
/*************************************/
{
/*
 timeout:
                1 = 4096 cycles
                2 = 8192 cycles
                4 = 16384 cycles
*/
        USHORT tempo;
        switch(timeout)
        {
                case 1:
                        tempo = 0xF000; //65535-4095;
                        break;
                case 2:
                        tempo = 0xE000; //65535-8191;
                        break;
                case 4:
                        tempo = 0xC000; //65535-16383;
                        break;
                case 8: // 8 TIMES
                        tempo = 0x8000; //65535-32767;
                        break;
                default: // 12 TIMES
                        tempo = 0x3000; // 53247 us;
        }
        set_timer1(tempo);
        clear_interrupt(INT_TIMER1);
        enable_interrupts(INT_TIMER1);
}
/****************/
#int_TIMER1
void ISR_TIMER1(void)
/****************/
/* é gerada uma interrupção toda vez que o tempo expira. */
{
        switch(eMBRcvState)
        {
            case STATE_RX_INIT:         /*  a fase de inicialização está terminada? */
                case STATE_RX_ERROR:
                case STATE_RX_IDLE:
                        break;
                case STATE_RX_RCV:    // finaliza
                /* Um quadro foi recebido e o tempo t35 expirou.
                 Muda o flag indicando que um novo quadro foi recebido */
                        FLAG_FRAME_RECEIVED = TRUE;
                        break;
    }
        FLAG_STATE_RX_IDLE=TRUE;
        FLAG_LED_RX_OFF=TRUE;
        disable_interrupts(INT_TIMER1); // O temporizador se auto-desativa. Tem que ser chamado explicitamente de outra rotina
}
/****************/
#int_TIMER2
```

```
void ISR_TIMER2(void)
/***************/
{
/* rotina utilizada para indicar que terminou a primeira leitura do ADC */
        FIRST_READING=TRUE;
        Le_ADC();
}
#endif
```

```
FUNC_READ_COILS.C
#if FUNC_READ_COILS_ENABLED
        #ifdef _FUNC_READ_COILS
                return
        #endif

        #define _FUNC_READ_COILS TRUE

        #define REG_COILS_START          0        // READ_COILS_OUTPUTS
        #define REG_COILS_SIZE       64
        #define REG_COILS_COILCNT_MAX          ( 0x07D0 )
#ifdef _8051
        idata UCHAR gucRegCoil[REG_COILS_SIZE/8];
#endif
#ifdef _PIC
        UCHAR gucRegCoil[REG_COILS_SIZE/8];
#endif
/*************************************************************************/
UCHAR xMBUtilGetBits(UCHAR *pointer, UCHAR ucByteOffset, UCHAR ucBitOffset)
/*************************************************************************/
{
        UCHAR ucMask;

        pointer += ucByteOffset;
        if (ucBitOffset == 8) // retornar o byte inteiro
                return *pointer;
        else
        {
                ucMask=0;
                while (ucBitOffset--)
                {
                        ucMask = (ucMask << 1) + 1;
                }
        return (*pointer & ucMask);
        }
}
/*************************************************************************/
void  xMBUtilSetBits(UCHAR *pointer, UCHAR ucByteOffset, UCHAR ucBitOffset)
/*************************************************************************/
{
        UCHAR ucMask;

        pointer += ucByteOffset;
        ucMask=1;
        while (ucBitOffset--)
        {
                ucMask = (ucMask << 1);
        }
        *pointer |= ucMask;
}
/*************************************************************************************/
void eMBRegCoilsCB( UCHAR eMode, USHORT usRcvRegStart, USHORT  usRcvCoilCount)
/*************************************************************************************/
{
        UCHAR                   ucBitOffset, cont_aux;
        UCHAR                   ucByteOffset;

        /* VERIFICA SE HÁ REGISTRADORES MAPEADOS NESTE BLOCO. */
        if      (
                ( usRcvRegStart >= REG_COILS_START )
                &&
                ( (usRcvRegStart + usRcvCoilCount)  <= (REG_COILS_START + REG_COILS_SIZE) )
                )
        {
                ucByteOffset =  (UCHAR) ( usRcvRegStart - REG_COILS_START );
                switch ( eMode )
                {
                        /* LÊ OS VALORES CORRENTES E PASSA DE VOLTA PARA A CAMADA DO PROTOCOLO */
                        case MB_REG_READ:
                                cont_aux=3;
                                while (usRcvCoilCount)
                                {
                                        ucBitOffset = ( UCHAR ) ( (usRcvCoilCount > 8) ? 8 : usRcvCoilCount);
                                        gucRcvBuffer[cont_aux++] =
                                                xMBUtilGetBits(gucRegCoil, ucByteOffset, ucBitOffset);
                                        ucByteOffset++;
                                        usRcvCoilCount -= ucBitOffset;
                                }
                                break;

                        /* ATUALIZA OS VALORES ATUAIS DOS REGISTRADORES */
                        case MB_REG_WRITE:
                                while (usRcvCoilCount)
                                {
                                        ucBitOffset  = ( UCHAR ) ( (usRcvCoilCount > 8) ? 8 : usRcvCoilCount);
                                        xMBUtilSetBits(gucRegCoil, ucByteOffset, ucBitOffset);
                                        ucByteOffset++;
                                        usRcvCoilCount -= ucBitOffset;
                                }
                                break;
                }// ENDSWITCH
        }
        else
        {
                eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
        }
}
/*************************************************************/
void Func_read_coils(void)
```

```
/*************************************************************/
/*        MB_FUNC_READ_COILS = FUNÇÃO 1
ESTA FUNÇÃO É UTILIZADA PARA LER DE 1 A 2000 ENDEREÇOS CONTÍGUOS DE BOBINAS DE ESTADO.
AS BOBINAS NA MENSAGEM DE RESPOSTA SÃO EMPACOTADAS NA FORMA DE UMA BOBINA PARA CADA BIT DE DADOS.
SE A QUANTIDADE RETORNADA NÃO FOR UM MÚLTIPLO DE OITO,
OS BITS RESTANTES SÃO COMPLETADOS COM ZEROS (EM DIREÇÃO AO BIT MAIS SIGNIFICATIVO).
O CAMPO DE CONTADOR DE BYTES ESPECIFICA A QUANTIDADE DE BYTES COMPLETOS.
PDU-RX
Function code           1 Byte      0x01
Starting Address        2 Bytes     0x0000 to 0xFFFF
Quantity of coils       2 Bytes     1 to 2000 (0x7D0)
PDU-Response
Function code           1 Byte      0x01
Byte count              1 Byte      N*
Coil Status             n Byte n = N or N+1
*N = Quantidade de saidas / 8, se o resto é diferente de zero então N = N+1
Error
Function code         1 Byte        Function code + 0x80
Exception code        1 Byte        01 or 02 or 03 or 04
*/
{
        USHORT                  usRcvRegStart;
    USHORT          usRcvCoilCount;
    UCHAR           ucNBytes;
    if( gucPUCFrameLength == ( MB_PDU_FUNC_READ_SIZE + MB_PDU_SIZE_MIN ) )
    {
        usRcvRegStart = (gucRcvBuffer[2] << 8);
        usRcvRegStart += gucRcvBuffer[3];
        usRcvCoilCount = (gucRcvBuffer[4] << 8);
        usRcvCoilCount += gucRcvBuffer[5] ;
        /* Verifica se o número de registradores a serem lidos é válido.
           Caso não seja, retorna uma exceção de 'illegal data value' para a camada Modbus */
              if      (
                      ( usRcvCoilCount > 0 )
                      &&
                      ( usRcvCoilCount <= REG_COILS_COILCNT_MAX )
                      )
              {
                      /* Testa se a quantidade de bobinas é múltiplo de oito.
                       Caso não seja, o último byte é preenchido com zeros nos bytes mais significativos */
                      ucNBytes = (UCHAR ) (usRcvCoilCount+7) / 8;
                      /* O primeiro bytes da resposta contém o código da função,
                      * Vamos reutilizar este valor, uma vez que continua válido. */
                      // O segundo byte contém a quantidade de registradores.
                      gucRcvBuffer[2] = ucNBytes;
                      eMBRegCoilsCB( MB_REG_READ, usRcvRegStart, usRcvCoilCount );
                      gucPUCFrameLength = ucNBytes + 2;
        }
        else
        {
            eMBException = MB_EX_ILLEGAL_DATA_VALUE;
        }
    }
    else
    {
        /* Não pode ser uma requisição válida para leitura de registrador de bobinas,
           já que o comprimento é incorreto. */
        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
    }
}
#endif
```

```
READ_DISCRETE_INPUTS.C
#if FUNC_READ_DISCRETE_INPUTS_ENABLED
        #ifdef _FUNC_READ_DISCRETE_INPUTS
                return
        #endif

        #define _FUNC_READ_DISCRETE_INPUTS TRUE
        #define REG_INPUT_DISCRETE_START          0   // READ_DISCRETE_INPUTS
        #define REG_INPUT_DISCRETE_SIZE           64
#ifdef _8051
        idata UCHAR      gucRegInputDiscrete[REG_INPUT_DISCRETE_SIZE/8];
#endif
#ifdef _PIC
        UCHAR    gucRegInputDiscrete[REG_INPUT_DISCRETE_SIZE/8];
#endif
/*************************************************************************************/
void eMBRegDiscreteCB( UCHAR eMode, USHORT usRcvRegStart, USHORT  usRcvDiscreteCount)
/*************************************************************************************/
{

        UCHAR                   ucBitOffset, cont_aux;
        UCHAR                   ucByteOffset;


        /* Verifica se existem registradores mapeados neste bloco */
        if (
           ( usRcvRegStart >= REG_INPUT_DISCRETE_START )
           &&
           ( usRcvRegStart + usRcvDiscreteCount  <= REG_INPUT_DISCRETE_START + REG_INPUT_DISCRETE_SIZE )
           )
        {
           ucByteOffset = ( UCHAR ) ( usRcvRegStart - REG_INPUT_DISCRETE_START );
           switch ( eMode )
           {
              /* lê os valores correntes e retorna para a camada correspondente do protocolo */
              case MB_REG_READ:
                 cont_aux=3;
                 while(usRcvDiscreteCount)
                 {
                    ucBitOffset = (UCHAR)( (usRcvDiscreteCount > 8) ? 8 : usRcvDiscreteCount);
                    gucRcvBuffer[cont_aux++]=xMBUtilGetBits(gucRegInputDiscrete, ucByteOffset, ucBitOffset);
                    ucByteOffset++;
                    usRcvDiscreteCount -= ucBitOffset;
                 }
              break;

             /* Atualiza os valores atuais dos registradores */
              case MB_REG_WRITE:
                 while(  usRcvDiscreteCount )
                 {
                    ucBitOffset  = (UCHAR)((usRcvDiscreteCount > 8) ? 8 :  usRcvDiscreteCount);
                    xMBUtilSetBits(gucRegInputDiscrete, ucByteOffset, ucBitOffset);
                    ucByteOffset++;
                    usRcvDiscreteCount -= ucBitOffset;
                  }
                  break;
          }// ENDSWITCH
        }
        else
        {
                eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
        }
}
/*********************************/
void Func_read_discrete_inputs(void)
/*********************************/
/*
02 (0x02) Read Discrete Inputs
Esta função é utilizada pra ler de 1 a 2000 registradores contíguos de entradas discretas no dispositivo remoto.
O bilhete de requisição especifica o endereço inicial e o número de entradas.
O endereço das entradas começam em zero.
Desta forma, as entradas de 1 a 16 são endereçadas como 0-15.
As entradas discretas na mensagem de resposta são empacotadas como uma entrada para cada bit do campo de dados.
O Estado é indicado como 1= ON; 0= OFF.
O bit menos significativo do primeiro byte de dados contém o endereço solicitado pelo mestre.
Se a quantidade de entradas retornada não é um múltiplo de oito,
os bits restantes do último byte de dados serão completados com zeros (nos bits mais significativos).
Bilhete de requisição
        Function code           1 Byte                  0x02
        Starting Address        2 Bytes                 0x0000 to 0xFFFF
        Quantity of Inputs      2 Bytes                 1 to 2000 (0x7D0)
Bilhete de resposta
        Function code           1 Byte                  0x02
        Byte count                      1 Byte N*
        Input Status            N* x 1 Byte
        *N = Quantidade de entradas / 8, se o resto for diferente de zero então N = N+1
Bilhete de erro
        Código do erro          1 Byte                  0x82
        Código de exceção       1 Byte                  01 or 02 or 03 or 04
*/
{
    USHORT          usRcvRegStart;
    USHORT          usRcvDiscreteCount;
    UCHAR           ucNBytes;
    if( gucPUCFrameLength == ( MB_PDU_FUNC_READ_SIZE + MB_PDU_SIZE_MIN ) )
    {
                usRcvRegStart = (gucRcvBuffer[2] << 8);
        usRcvRegStart |= gucRcvBuffer[3];
        usRcvDiscreteCount = (gucRcvBuffer[4] << 8);
```

```
        usRcvDiscreteCount |= gucRcvBuffer[5] ;
        /* verifica se o número de registradores a serem lidos é válido.
          Caso não seja, retorna uma exceção do tipo 'illegal data value'.
         */
        if( ( usRcvDiscreteCount > 0 ) &&
            ( usRcvDiscreteCount <= MB_PDU_FUNC_READ_DISCCNT_MAX ) )
        {
            /* Testa se a quantidade de bobinas é um múltiplo de oito.
                Caso não seja, os bits não utilizados do último byte são preenchidos com zero.*/
                    ucNBytes = (UCHAR ) (usRcvDiscreteCount+7) / 8;
                    /* O primeiro byte do bilhete de resposta deve conter o código da função,
                     vamos reutilizar esta valor uma vez que este continua o mesmo */
                    // O segundo contém a quantidade de registradores.
                    gucRcvBuffer[2]  = ucNBytes;
                    eMBRegDiscreteCB( MB_REG_READ, usRcvRegStart, usRcvDiscreteCount);
                    gucPUCFrameLength = ucNBytes + 2;
            }
        else
        {
            eMBException = MB_EX_ILLEGAL_DATA_VALUE;
        }
    }
    else
    {
        /* Não pode ser uma requisição legal porque o comprimento é incorreto. */
        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
    }
}
#endif
```

FUNC_READ_HOLDING_REGISTER.C

```
#if FUNC_READ_HOLDING_REGISTER_ENABLED
        #ifdef _FUNC_READ_HOLDING_REGISTER
                return
        #endif

        #define _FUNC_READ_HOLDING_REGISTER TRUE
        #define REG_HOLDING_START 0
        #define REG_HOLDING_NREGS 16
        #define VALOR_REFERENCIA_CALIBRACAO                             0
        #define VALOR_OFFSET_REFERENCIA_MAXIMO                  1
        #define VALOR_TEMPERATURA_CALIBRACAO_SISTEMA    2
        #define VALOR_TEMPERATURA_CALIBRACAO_PONTE      3
        #define VALOR_TEMPERATURA_CALIBRACAO_SHUNT              4
        #define VALOR_SHUNT_CALIBRACAO                                  5
        #define NUMERO_PONTOS_JANELA_MEDIA_MOVEL                6
        #define REVERSE_READING_ENABLED                                 7
        #define RECALIBRATION_ENABLED                                   8
#ifdef _8051
        idata USHORT gusRegHolding[REG_HOLDING_NREGS];
#endif
#ifdef _PIC
        USHORT gusRegHolding[REG_HOLDING_NREGS];
#endif
/*****************************************************************************/
void    eMBRegHoldingCB(
                                                USHORT usAddress,
                                                USHORT usNRegs,
                                                UCHAR eMode
                                                )
/*****************************************************************************/
{
        USHORT              iRegIndex;
        UCHAR                               cont_aux;
        if      (
                ( usAddress >= REG_HOLDING_START )
                &&
                ( (usAddress + usNRegs) <= (REG_HOLDING_START + REG_HOLDING_NREGS) )
                )
        {
                iRegIndex =  usAddress - REG_HOLDING_START ;
                cont_aux=3;
                while(usNRegs)
                {
                        gucRcvBuffer[cont_aux++] = ( UCHAR )( gusRegHolding[iRegIndex] >> 8 );
                        gucRcvBuffer[cont_aux++] = ( UCHAR )( gusRegHolding[iRegIndex] & 0x00FF );
                        iRegIndex++;
                        usNRegs--;
                }
        }
        else
        {
                eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
        }
}
/*
Slave Address                           01
Function                                        03
Starting Address High           00
Starting Address Low            00
Number of Points High           00
Number of Points Low            02
Error Check Low                         C4
Error Check High                        0B
Slave Address                           01
Function                                        03
Byte Count                                      04
Data, High Word, High Byte      3F
Data, High Word, Low Byte       80
Data, Low Word, High Byte       00
Data, Low Word, Low Byte        00
Error Check Low                         F7
Error Check High                        CF
*/
/***********************************/
void Func_read_holding_register(void)           //0x03
/***********************************/
{
    USHORT          usRcvRegStart;
    USHORT          usRcvRegCount;
    if( gucPUCFrameLength == ( MB_PDU_FUNC_READ_SIZE + MB_PDU_SIZE_MIN ) )
    {
                usRcvRegStart = (gucRcvBuffer[2] << 8);
        usRcvRegStart |= gucRcvBuffer[3];
        usRcvRegCount = (gucRcvBuffer[4] << 8);
        usRcvRegCount |= gucRcvBuffer[5] ;
        /* verifica se o número de registradores a serem lidos é válido.
          Caso não seja, retorna uma exceção do tipo 'illegal data value'.
         */
        if      (
                        ( usRcvRegCount > 0 )
                        &&
                        ( usRcvRegCount <= MB_PDU_FUNC_READ_REGCNT_MAX )
                        )
        {
                        /* reutiliza o código da função no bilhete de resposta,
                         uma vez que este ainda continua válido e é o mesmo */
            /* O segundo byte da resposta contém o número de bytes. */
            gucRcvBuffer[2] = ( UCHAR ) ( usRcvRegCount * 2 );
```

```
            /* faz uma chamada a função abaixo para completar o buffer com zeros, caso necessário. */
            eMBRegHoldingCB( usRcvRegStart, usRcvRegCount, MB_REG_READ );
                        gucPUCFrameLength = 2 + usRcvRegCount * 2;
        }
        else
        {
            eMBException = MB_EX_ILLEGAL_DATA_VALUE;
        }
    }
    else
    {
        /* Não pode ser uma requisição válida porque o comprimento é incorreto. */
        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
    }
}
#endif
```

FUNC_READ_INPUT_REGISTER.C

```
#if FUNC_READ_INPUT_REGISTER_ENABLED
#ifndef _FUNC_READ_INPUT_REGISTER

        #define _FUNC_READ_INPUT_REGISTER TRUE
        #define REG_INPUT_REG_START 0
        #define REG_INPUT_REG_NREGS 42
        // definição das posições
        #define REG_INPUT_STRAIN_VALUE          0
        #define REG_INPUT_TEMPERATURE_VALUE     1
        #define REG_OFFSET_DIF                  2
        #define OFFSET_SAMPLE                   5 // início de onde serão armazenadas as amostras
        #ifdef _8051
                idata USHORT gusRegInputRegister[REG_INPUT_REG_NREGS];
        #endif

        #ifdef _PIC
                USHORT gusRegInputRegister[REG_INPUT_REG_NREGS];
        #endif

/*****************************************************************************/
void eMBRegInputCB( USHORT usAddress, USHORT usNRegs )
/*****************************************************************************/
{
        UCHAR cont_aux;
    USHORT iRegIndex;
        if      (
                ( usAddress >= REG_INPUT_REG_START )
                &&
                ( usAddress + usNRegs <= REG_INPUT_REG_START + REG_INPUT_REG_NREGS )
                )
        {
                iRegIndex = ( USHORT ) ( usAddress - REG_INPUT_REG_START );
                cont_aux=3;
                while( usNRegs)
                {
                        gucRcvBuffer[cont_aux++] = ( UCHAR )( gusRegInputRegister[iRegIndex] >> 8 );
                        gucRcvBuffer[cont_aux++] = ( UCHAR )( gusRegInputRegister[iRegIndex] & 0xFF );
                        iRegIndex++;
                        usNRegs--;
                }
        }
    else
    {
        eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
    }
}
/*****************************************/
void Func_read_input_register(void)      // 0X04
/*****************************************/
{
    USHORT          usRcvRegStart;
    USHORT          usRcvRegCount;
    if( gucPUCFrameLength == ( MB_PDU_FUNC_READ_SIZE + MB_PDU_SIZE_MIN ) )
    {
                usRcvRegStart = (gucRcvBuffer[2] << 8);
        usRcvRegStart |= gucRcvBuffer[3];
        usRcvRegCount = (gucRcvBuffer[4] << 8);
        usRcvRegCount |= gucRcvBuffer[5] ;
        /* verifica se o número de registradores a serem lidos é válido.
           caso não seja, retorna uma exceção do tipo 'illegal data value'.
         */
                if      (
                        ( usRcvRegCount  )
                        &&
                        ( usRcvRegCount < MB_PDU_FUNC_READ_REGCNT_MAX )
                        )
                {
            gucRcvBuffer[2] = ( UCHAR ) ( usRcvRegCount * 2 );

                        eMBRegInputCB( usRcvRegStart, usRcvRegCount );

                        gucPUCFrameLength = 2 + usRcvRegCount * 2;
                }
                else
                {
                        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
                }
        }
    else
    {
        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
    }
}
#endif
#endif
```

```
FUNC_WRITE_SINGLE_COIL.C
#if FUNC_WRITE_SINGLE_COIL_ENABLED
#include "func_utils.c"
/*
Recepção
        Function code           1 Byte          0x05
        Output Address          2 Bytes         0x0000 to 0xFFFF
        Output Value            2 Bytes         0x0000 or 0xFF00
Exemplo de um bilhete Force Single Coil
Slave Address 247 (F7)
Function Code 5 (05)
Coil Address High Order 0 (00)
Coil Address Low Order 9 (09)
Force Data High Order 255 (FF)
Force Data Low Order 0 (00)
Error Check (LRC or CRC)--
Resposta
        Function code           1 Byte          0x05
        Output Address          2 Bytes         0x0000 to 0xFFFF
        Output Value            2 Bytes         0x0000 or 0xFF00
Erro
        Error code                      1 Byte          0x85
        Exception code          1 Byte          01 or 02 or 03 or 04
*/
/*******************************/
void Func_write_single_coil(void)               // 0x05
/*******************************/
{
        USHORT          usRegAddress;
        UCHAR                   ucByteOffset;
        UCHAR                   *pointer;
        usRegAddress = (gucRcvBuffer[2] << 8);
        usRegAddress |= gucRcvBuffer[3];
        if      (
                ( usRegAddress >= REG_COILS_START )
                &&
                ( usRegAddress  <= REG_COILS_START + REG_COILS_SIZE )
                )
        {
                ucByteOffset = (UCHAR)  (usRegAddress  - REG_COILS_START);
                if      (
                        ( gucRcvBuffer[5] == 0x00 )
                        &&
                        (
                                ( gucRcvBuffer[4] == 0xFF )
                                ||
                                ( gucRcvBuffer[4] == 0x00 )
                        )
                        )
                {
                        pointer = gucRegCoil + ucByteOffset;
                        if ( gucRcvBuffer[4] == 0xFF )
                                *pointer = 1;
                        else
                                *pointer = 0;
                        gucPUCFrameLength = 5 ;
                }
                else
                {
                        eMBException = MB_EX_ILLEGAL_DATA_VALUE;
                }
        }
        else
        {
        /* Não pode ser uma requisição correta, uma vez que o comprimento é incorreto. */
        eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
        }
}
#endif
```

FUNC_WRITE_SINGLE_REGISTER.C

```
#if FUNC_WRITE_SINGLE_REGISTER_ENABLED
/*
(0x06) Write Single Register
Esta função é utilizada para escrever um único registrador de memória interna no dispositivo remoto.
O bilhete especifica o endereço em que deve ser escrito.
Os registradores devem ser endereçados começando em zero.
Dessa maneira o registrador 1 é endereçado como 0.
A resposta normal é uma cópia do bilhete de requisição,
retornada após o conteúdo do registrador ter sido escrito.
Requisição
        Function code           1 Byte          0x06
        Register Address        2 Bytes         0x0000 to 0xFFFF
        Register Value          2 Bytes         0x0000 or 0xFFFF
Resposta
        Function code           1 Byte          0x06
        Register Address        2 Bytes         0x0000 to 0xFFFF
        Register Value          2 Bytes         0x0000 or 0xFFFF
Erro
        Error code                      1 Byte          0x86
        Exception code          1 Byte          01 or 02 or 03 or 04
*/
/***********************************/
void Func_write_single_register(void) // 0x06
/***********************************/
{
    USHORT          usRcvRegAddress;
    USHORT          usRcvRegValue;
        USHORT                  usByteOffset;
    if( gucPUCFrameLength == ( MB_PDU_FUNC_READ_SIZE + MB_PDU_SIZE_MIN ) )
    {
                usRcvRegAddress = (gucRcvBuffer[2] << 8);
        usRcvRegAddress |= gucRcvBuffer[3];
        usRcvRegValue = (gucRcvBuffer[4] << 8);
        usRcvRegValue |= gucRcvBuffer[5] ;
                usByteOffset =  usRcvRegAddress  - REG_HOLDING_START;
                if      (
                        ( usRcvRegAddress >= REG_HOLDING_START )
                        &&
                        ( usRcvRegAddress  <= REG_HOLDING_START + REG_HOLDING_NREGS )
                        )
                {
                        gusRegHolding[usByteOffset]= usRcvRegValue;
                        gucPUCFrameLength = 5 ;
                }
                else
                {
                        eMBException = MB_EX_ILLEGAL_DATA_ADDRESS;
                }
        }
        else
        {
                eMBException = MB_EX_ILLEGAL_DATA_VALUE;
        }
}
#endif
```

SPI.C

```
#ifndef _SPI

        #define          _SPI     TRUE

        #ifndef spi_sck
                // Definições dos pinos de comunicação
                #define spi_sck          pin_c3          // pino de clock
                #define spi_sdi          pin_c4
                #define spi_sdo          pin_c5
                #define LOAD             output_low(PIN_B1); delay_us(1); output_high(PIN_B1);
                #define LATCH            output_low(PIN_C2); delay_us(1); output_high(PIN_C2);
                // L-H latch outputs
        #endif
/***********************************/
void spi_escreve_bit (boolean bit_wr)
/***********************************/
// escreve um bit na interface SPI
{
        output_low (spi_sck);           // desativa a linha de clock
        output_bit (spi_sdo,bit_wr);    // coloca o dado na saída
        output_high (spi_sck);          // ativa a linha de clock
}
/*********************************/
void spi_escreve_byte (UCHAR dado)
/*********************************/
// escreve um byte na interface SPI
{
        UCHAR conta = 8;
        // envia primeiro o MSB
        while (conta)
        {
                spi_escreve_bit ((shift_left(&dado,1,0)));
                conta--;
        }
        LATCH
}
/***********************/
boolean spi_le_bit (void)
/***********************/
// le um bit na interface SPI
{
        output_low (spi_sck);
        output_high (spi_sck);
        return input (spi_sdi);
}
/***********************/
UCHAR spi_le_byte (void)
/***********************/
// lê um byte na interface SPI
{
        UCHAR conta = 7, dado = 0;
        LOAD
        shift_left (&dado, 1, input(spi_sdi));  // o bit Qh já está disponível
        while (conta)
        {
                shift_left (&dado, 1, spi_le_bit());
                conta--;
        }
        return dado;
}
/***********************/
void Liga_LED(UCHAR mask)
/***********************/
{
// led ligado em low (0)
var_output &= ~mask;
spi_escreve_byte (var_output);
}
/*************************/
void Desliga_LED(UCHAR mask)
/*************************/
{
// led desligado em high (1)
var_output |= mask;
spi_escreve_byte (var_output);
}
// definições de comandos SPI para memória
//#define spi_read_cmd  0x03
//#define spi_write_cmd 0x02
//#define spi_wren_cmd  0x06
//#define spi_wrdi_cmd  0x04
//#define spi_rdsr_cmd  0x05
//#define spi_wrsr_cmd  0x01
//void spi_escreve_eeprom ( long int endereco, byte dado)
//// escreve um dado em uma determinada posição da memória SPI
//{
//      output_low (spi_cs);                    // seleciona a memória
//      spi_escreve_byte (spi_write_cmd);       // envia comando de escrita
//      spi_escreve_byte (endereco >>8);        // envia endereço MSB
//      spi_escreve_byte (endereco);            // envia endereço LSB
//      spi_escreve_byte (dado);                // envia dado
//      output_high (spi_cs);           // desativa linha CS e inicia a escrita
//      // lembre-se de aguardar 5ms antes de realizar outra escrita ou leitura
//}
//
//byte spi_le_eeprom ( long int endereco )
//// lê um dado de uma determinada posição da memória SPI
//{
```

```
//      byte dado;
//      output_low (spi_cs);
//      spi_escreve_byte (spi_read_cmd);        // envia comando de leitura
//      spi_escreve_byte (endereco >> 8);       // envia endereço MSB
//      spi_escreve_byte (endereco);            // envia endereço LSB
//      dado = spi_le_byte ();                  // lê o dado
//      output_high (spi_cs);           // desativa linha CS e termina leitura
//}
//
//void spi_ativa_escrita_eeprom (void)
//// habilita a escrita na memória
//{
//      output_low (spi_cs);
//      spi_escreve_byte (spi_wren_cmd);
//      output_high (spi_cs);
//}
//
//void spi_desativa_escrita_eeprom (void)
//// desabilita a escrita na memória SPI
//{
//      output_low (spi_cs);
//      spi_escreve_byte (spi_wrdi_cmd);
//      output_high (spi_cs);
//}
//
//byte spi_le_status_eeprom (void)
//// le o registrador de estado da memória
//{
//      byte dado;
//      output_low (spi_cs);
//      spi_escreve_byte (spi_rdsr_cmd);
//      dado = spi_le_byte ();
//      output_high (spi_cs);
//}
//
//void spi_escreve_status_eeprom (byte dado)
//// escreve o dado no registrador de estado da memória SPI
//{
//      output_low (spi_cs);
//      spi_escreve_byte (spi_wrsr_cmd);
//      spi_escreve_byte (dado);
//      output_high (spi_cs);
//}
#endif
```

MB_CONFIG.H

```
#ifndef _CONFIG
        #define _CONFIG
        #ifdef ENABLED
                #undef ENABLED
        #endif
        #define         ENABLED         1
        #define         DISABLED        0

/* ----------------------- Defines ------------------------------------------*/
/* quais tipos de protocolos estão ativados */
#define MB_RTU_ENABLED                          ENABLED
/* Se o protocolo RTU está habilitado */
#define MB_ASCII_ENABLED                        DISABLED
/* Se o protocolo ASCII está habilitado */
#define MB_TCP_ENABLED                          DISABLED
/* o valor do tempo limite para o protocolo Modbus ASCII.
 *
 * Este tempo não é fixo para o Modbus ASCII e é deste modo, uma opção a ser configurada
 * Deve ser informado o tempo máximo a ser aguardado pelo terminal.
 */
#define MB_ASCII_TIMEOUT_SEC                    1

#define MB_FUNC_HANDLERS_MAX                    16
#define PIN_ENABLE_RX_EXISTS                    FALSE
#define FUNC_READ_COILS_ENABLED                 ENABLED // fcn 01
#define FUNC_READ_DISCRETE_INPUTS_ENABLED       ENABLED // fcn 02
#define FUNC_READ_HOLDING_REGISTER_ENABLED      ENABLED // fcn 03
#define FUNC_READ_INPUT_REGISTER_ENABLED        ENABLED // fcn 04
#define FUNC_WRITE_SINGLE_COIL_ENABLED          ENABLED // fcn 05
#define FUNC_WRITE_SINGLE_REGISTER_ENABLED      ENABLED // fcn 06
#define FUNC_DIAG_READ_EXCEPTION_ENABLED        DISABLED // fcn 07
#define FUNC_DIAG_DIAGNOSTIC_ENABLED            DISABLED
#define FUNC_DIAG_GET_COM_EVENT_CNT_ENABLED     DISABLED
#define FUNC_DIAG_GET_COM_EVENT_LOG_ENABLED     DISABLED
#define FUNC_WRITE_MULTIPLE_COILS_ENABLED       DISABLED  // fcn 15
#define FUNC_WRITE_MULTIPLE_REGISTERS_ENABLED   DISABLED  // fcn 16
#define FUNC_OTHER_REPORT_SLAVEID_ENABLED       DISABLED
#define FUNC_READWRITE_MULTIPLE_REGISTERS_ENABLED DISABLED
#define FUNC_READ_DEVICE_IDENTIFICATION_ENABLED DISABLED
#endif
```