unesp

UNIVERSIDADE ESTADUAL PAULISTA

Pós-Graduação em Engenharia Elétrica

MESTRADO

DESENVOLVIMENTO DE SENSORES À CORDA VIBRANTE: APLICAÇÕES EM ENGENHARIA DE REABILITAÇÃO

12100012294

Tony Inácio da Silva

Ilha Solteira - SP

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
CAMPUS DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE SENSORES À CORDA VIBRANTE: APLICAÇÕES EM ENGENHARIA DE REABILITAÇÃO

Tony Inácio da Silva

Proc. 071/2002 - NBD 055/02

UNESP - "CAMPUS DE ILHA SOLTEIRA" SERVIÇO TÉC. DE BIBLIOTECA E DOCUMENTAÇÃO	
DATA DE CHEGADA	DATA DE TOMBO
19.09.02	30.09.02
REGISTRADO POR	TOMBO
Ailza	Te. 1294
AQUISIÇÃO	CLASSIFICAÇÃO
Doação Autor R$ 10,00	S586d

30400007
ENGENHARIA ELÉTRICA

ILHA SOLTEIRA – SP, MAIO DE 2002

fys 199211
fys 55454

Dissertação submetida à Universidade Estadual Paulista – UNESP, Campus de Ilha Solteira, para a obtenção do título de Mestre em Engenharia Elétrica.

1210001294

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
CAMPUS DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE SENSORES À CORDA VIBRANTE: APLICAÇÕES EM ENGENHARIA DE REABILITAÇÃO

Tony Inácio da Silva

Dissertação submetida à Universidade Estadual Paulista – UNESP, Campus de Ilha Solteira, para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Aparecido Augusto de Carvalho

ILHA SOLTEIRA – SP, MAIO DE 2002

BCpIS - FEIS - UNESP

Desenvolvimento de Sensores à Corda Vibrante: Aplicações em Engenharia de Reabilitação

Tony Inácio da Silva

DISSERTAÇÃO SUBMETIDA À FACULDADE DE ENGENHARIA – CAMPUS DE ILHA SOLTEIRA – UNESP – COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA (ME).

Prof. Dr. Marcelo Carvalho Minhoro Teixeira
Vice-Coordenador em Exercício

COMISSÃO EXAMINADORA:

Prof. Dr. Aparecido Augusto de Carvalho – orientador

Prof. Dr. Alberto Cliquet Junior

Prof. Dr. Alexandre César Rodrigues da Silva

BCpIS - FEIS - UNESP

À Ruth Silva e

José Inácio da Silva,

meus pais,

Dedico

À minhas irmãs Marisa, Carolina, Ruth e Sílvia.

À minha família, meus amigos.

Ofereço

BCpIS - FEIS - UNESP

Agradecimentos

Quero agradecer primeiramente e principalmente a Deus, que me deu saúde, forças, e através do Seu Filho Jesus Cristo, a alegria e paz, mais que necessárias para seguir em frente no momentos difíceis, mostrando-me que não estava sozinho.

Aos meus pais, irmãs e toda minha família que me acompanharam, incentivaram, com muito entusiasmo desde os meus primeiros passos.

Ao professor Aparecido, pela orientação, confiança, estímulo, paciência e apoio em todos o momentos do projeto. Ao professor Alexandre Silva, pela atenção e dedicação que sem medir esforços, também muito contribuiu nesta realização.

Aos meus amigos Gercina Silva, Silvio Cesar, Ângelo Loureço, Rogério Blini, Jeferson Ponties, Adriano Cardoso, Débora Leister, Uender Faria, Josivaldo Godoy, Silvano Rossi, Marcelo Almeida, Jackson Tsukada, Carlos Lee, Giovanni Faria, Fábio Toshiaki, Wagner Peron, Simone Albuquerque, entre outros, que de muitas formas, através de uma forte amizade, deram valiosas contribuições durante a realização deste trabalho.

Aos professores do curso de pós-graduação em Engenharia Elétrica da FEIS, pela busca continua da excelência da qualidade dos trabalhos e condições de realização dos mesmos.

Aos técnicos Deoclécio e Carlos Alex, pelo apoio nos problemas de informática. Aos técnicos Adilson, Everaldo, Hidemassa, Aderson e Chaves pelo apoio técnico em muitas ocasiões deste trabalho. Ao técnico Cosmo pelo auxílio sem medida no desenvolvimento do trabalho.

À Maria Cristina, Sueli, Luzinete, Sandra, João, Cleide, Elaini, Marta, Neide, Terezinha, Arlindo, Maria de Fátima, Paulo da Silva, Onilda Akasaki, e Adelaide Passipieri que sempre se dispuseram a ajudar, não como funcionários da Unesp, mas como meus sinceros amigos.

À EMBRAER pela doação do material para confecção dos transdutores. À CAPES e à FAPESP pelo suporte financeiro.

RESUMO

Este trabalho descreve o desenvolvimento de um sensor a corda vibrante e dois diferentes circuitos de condicionamento de sinais para o mesmo. Foi projetado com o objetivo de instrumentar uma muleta canadense, para monitorar forças normais exercidas pelos membros superiores de pacientes paraplégicos, e um dinamômetro biomédico, visando medição de forças exercidas pelas mãos. Foi construído em dura alumínio, sendo constituído por uma corda fina de aço inoxidável e por uma bobina excitadora e sensora.

Os circuitos de condicionamento de sinais possuem uma parte analógica e uma digital. A analógica, que é a mesma nos dois circuitos de condicionamento, é constituída por um amplificador e por um filtro passa-baixas de oitava ordem. Em um dos circuitos a parte digital foi implementada com dispositivos lógicos programáveis (FPGA) e no outro com microcontroladores (MCU).

No circuito com FPGA utilizou-se componentes FLEX 10K e o Ambiente de Desenvolvimento Max+Plus II, ambos da Altera. A linguagem de descrição utilizada foi o AHDL, e também circuitos esquemáticos.

No circuito com MCU utilizou o 8031 da Intel, e o PIC 18C452 da Microchip, ambos com 8 bits, com periféricos auxiliares disponíveis. O ambiente de desenvolvimento utilizado para o 8031 foi o kit 153 da Degen. O ambiente de desenvolvimento utilizado para o PIC 18 foi o MPLAB da própria Microchip.

A resposta do sensor foi linear na faixa de 0 a 400 N, apresentando resolução de 1 N, precisão de 99,9% e histerese máxima de 6,5%. A faixa dinâmica do sensor implementado é adequada para sua utilização em um dinamômetro biomédico e também na muleta, desde que o paraplégico não tenha mais que 81 kg.

Foram realizados experimentos com alguns indivíduos, monitorando-se forças exercidas pelas mãos, simultaneamente com o dinamômetro com sensor a corda vibrante e com um dinamômetro com extensômetros metálicos, previamente calibrado com pesos conhecidos. A discrepância entre as leituras dos dois dinamômetros foi inferior a 5 N.

A tecnologia desenvolvida na implementação do sensor a corda vibrante e de seus circuitos de condicionamento de sinais poderá ser muito útil na Engenharia de Reabilitação e em outras aplicações, como, por exemplo, no monitoramento de deformações em grandes estruturas.

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

ABSTRACT

This work describes the development of a vibrating wire sensor and two different signal conditioning circuits for this. It was designed with the purpose of instrumenting a canadian crutch for monitoring forces exerted by upper members of paraplegic patients and for monitoring a biomedical dynamometer, in order to measure forces exerted by the hands. It was built in hard aluminum, being constituted by a steel wire and by a sensor and exciting coil.

The signal conditiong circuits have an analog and a digital part. The analog one is the same for the two signal conditiong circuits. It is constituted by an amplifier and by an eight order filter. One of the digital circuits was implemented with FPGA devices and the another one with microcontrollers (MCU).

10 K Flex components were used in the FPGA device with development in the environment of development Max+ Plus II, from Altera. The language of description was the AHDL.

The microcontrollers used were the 8031 from Intel and the PIC18C452 from Microchip. The environment of development were respectively a Degen kit number 153 and the MAPLAB, also from Microchip.

The sensor response was linear in the range of 0 to 400 N, it presented resolution of 1 N, precision of 99.9%, and maximum hysteresis of 6.5%. The sensor dynamic range is adequate for use in a biomedical dynamometer and in a canadian crutch, if the paraplegic weight is lower than 81 kg.

Experiments were done with some subjects, by monitoring the forces exerted by the hands simultaneouly with a strain gage biomedical dynamometer and with the developed vibrating wire dynamometer. The strain gage dynamometer was previously calibrated with known weights. The discrepancy between the response of the two dynamometers was lower than 5 N.

The technology developed in the implementation of the vibrating wire sensor and its signal conditioning circuits will be very useful in Rehabilitation Engineering and in other applications as, for example, in the monitoring of deformations in large structures.

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

ÍNDICE

AGRADECIMENTOS IV

RESUMO V

ABSTRACT VI

ÍNDICE VII

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABELAS XII

1. INTRODUÇÃO 1

1.1. APRESENTAÇÃO DO PROJETO 1
1.2. HISTÓRICO DO SENSOR 3

2. FUNDAMENTOS TEÓRICOS 4

2.1. O SENSOR 4
2.2. PRINCÍPIO DE FUNCIONAMENTO 4
2.2.1. Freqüência de Vibração 5
2.2.2. Relação entre a Freqüência e a Força Externa 9
2.2.3. A Captação da Freqüência 13
2.2.4. Cálculo do Campo Magnético Produzido por um Bobina com Núcleo Ferromagnético 14
2.2.5. Campo Magnético Residual 15
2.2.6. Correntes Induzidas por Variação de Indução 16
2.3. SISTEMA DE CONDICIONAMENTO E AQUISIÇÃO DE SINAIS 19
2.4. DISPOSITIVOS FPGA 21
2.4.1. Estrutura da FPGA 21
2.4.2. Tipos de FPGA – Altera 22
2.4.3. Técnicas de Descrição 23
2.4.4. O AHDL 24
2.5. OS MICROCONTROLADORES 26
2.5.1. A Família 8051 da Intel 27
2.5.2. A Família PIC18 da Microchip 29

3. MATERIAIS E MÉTODOS 32

3.1. INTRODUÇÃO 32
3.2. O SENSOR E SEUS COMPONENTES 33
3.3. O AMPLIFICADOR, O FILTRO E O COMPARADOR 34
3.4. A AQUISIÇÃO DE SINAIS COM O DISPOSITIVO FPGA 34
3.4.1. Medição do Período 34
3.5. A AQUISIÇÃO DE SINAIS COM O MICROCONTROLADOR 34
3.5.1. Medição do Período 34
3.5.2. A Aquisição de Sinais com o 8031 34
3.5.3. A Aquisição de Sinais com o PIC18C452 34
3.6. AS MEDIÇÕES 34

4. RESULTADOS OBTIDOS 34

4.1. O SENSOR E A PARTE ANALÓGICA 34
4.2. RESULTADOS COM O FPGA 34
4.3. RESULTADOS COM O MICROCONTROLADOR 8031 34
4.4. RESULTADOS COM O MICROCONTROLADOR PIC18 34

5. CONCLUSÃO 34

6. BIBLIOGRAFIA 34

APÊNDICES 77

APÊNDICE 1 – Descrição do Circuito Implementado em FPGA 78

APÊNDICE 2 – Programa em Assembly Gravado no PIC 18C452 102

APÊNDICE 3 – Artigo Submetido e Aceito em Congresso Internacional 161

APÊNDICE 4 – Precisão, Repetibilidade e Histerese 166

ÍNDICE DE FIGURAS

Figura 1.1 – Ilustração Simples do Dinamômetro. 2
Figura 1.2 - Muleta Instrumentada com Sensor à Corda Vibrante. 3
Figura 2.1 - O Sensor de Corda Vibrante e seus Componentes. 4
Figura 2.2 - Ilustração da Interação entre Sensor e o Sistema Condicionamento e Aquisição de Sinais. 5
Figura 2.3 - (a) Corda Fixa e Tensa, Deslocada por uma Força Externa do seu Eixo Central. (b) Porção dx Tomada para Análise. 6
Figura 2.4 - A Corda Vibrante e o Corpo do Sensor Sofrendo a Aplicação de uma Força. 9
Figura 2.5 – Diagrama de Forças sobre o Sensor sob Compressão. 10
Figura 2.6 – Diagrama de Forças sobre o Sensor sob Tração. 10
Figura 2.7 – Vista Lateral da Bobina Sensora e da Corda. 13
Figura 2.8 - Circuito Magnético Formado pela Bobina Sensora e a Corda Metálica. 14
Figura 2.9 - Comportamento do Campo Magnético Produzido pela Bobina Sensora. 15
Figura 2.10 – Curva de Magnetização Característica. 16
Figura 2.11 – (a) Espira Atravessada pela Indução B(t); (b) Representação Vetorial dos Componentes Envolvidos na Indução da Corrente. 17
Figura 2.12 - A f.e.m. Induzida em uma Espira sob Indução Magnética Variável no Tempo. 19
Figura 2.13 – Filtro Passa Baixa de 2ª Ordem. 20
Figura 2.14 – Filtro Passa Baixa de 8ª Ordem. 20
Figura 2.15 – Tela Inicial do Software Filter para Cálculo de Filtros Ativos. 21
Figura 2.16 – Estrutura dos Dispositivos FPGA. 22
Figura 2.17 - Fluxo de Dados no MAX+PLUS II. 24
Figura 2.18 – Diagrama de Arquivos do AHDL. 25
Figura 2.20 – Divisor Parametrizável. 25
Figura 2.20 – Tela do Software MAX PLUS II. 26
Figura 2.21 – A Estrutura Interna do 8051 da Intel. 28
Figura 2.22 – O MCU PIC18C4X2, em Encapsulamento DIP 40. 30
Figura 2.23 – A Estrutura Interna do PIC18C4X2. 31
Figura 3.1 - O Sensor a Corda Vibrante e o seu Sistema de Leitura de Sinal. 32
Figura 3.2 – O Corpo do Sensor (medidas de centímetros). 33
Figura 3.3 – Ilustração da Construção das duas Tampas Rosqueáveis do Sensor (medidas em centímetros). 34
Figura 3.4 – Ilustração Detalhada da Fixação da Corda no Sensor. 35
Figura 3.5 – As duas Vistas do Dinamômetro e seus Componentes (medidas em milímetros). 36
Figura 3.6 – Ilustração Detalhada da Fixação da Corda. 37
Figura 3.7 – Tela do Software Filter, com os Valores dos Componentes do Filtro. 38
Figura 3.8 – Esquemático do Circuito Analógico do Sistema de Aquisição e Condicionamento de Sinais. 38

Figura 3.9 – O Sensor Construído e o Circuito da Parte Analógica..... 39
Figura 3.10 - Módulo de Aprendizagem da Altera..... 40
Figura 3.11 – Geração dos Pulsos a Serem Contados..... 40
Figura 3.12 – Primeira Versão do Contador..... 41
Figura 3.13 – Ilustração da Imprecisão na Contagem de Pulsos. 42
Figura 3.14 – Fluxograma do Sistema de Condicionamento Aquisição e de Sinais Implementado em FPGA..... 43
Figura 3.15 – Frequencímetro Implementado em FPGA. 43
Figura 3.16 – Ilustração das Bordas de Subida Acionando duas Interrupções no Microcontrolador. 44
Figura 3.17 – Diagrama do kit Degem do 8031. 45
Figura 3.18 – O kit Degen com o Microcontrolador 8031..... 46
Figura 3.19 – Fluxograma do Programa Implementado no Microcontrolador PIC 18. 47
Figura 3.20 – Circuito Implementado para o Microcontrolador PIC 18. 48
Figura 3.21 – O Sensor Preso a Estrutura de Teste. 49
Figura 3.22 – Microcontrolador PIC 18 com o Display LCD..... 49
Figura 3.23 – O Dinamômetro Implementado. 50
Figura 3.24 – Fotos da Estrutura Utilizada nas Medições..... 51
Figura 3.25 – Foto da Utilização do Dinamômetro. 52
Figura 4.1 – Comportamento do Sensor com a Aplicação de Força Externa, e a Variação da Força Inicial Aplicada sobre a Corda. 54
Figura 4.2 – Medições de 500-1000 Hz com o FPGA..... 55
Figura 4.3 – Medições de 520-530 Hz com o FPGA..... 56
Figura 4.4 – Medições de 690-700 Hz com o FPGA..... 56
Figura 4.5 – Medições de 870-880 Hz com o FPGÀ..... 57
Figura 4.6 – Medições de 960-970 Hz com o FPGA..... 57
Figura 4.7 – Medições de 500-502 Hz com o FPGA..... 58
Figura 4.8 – Medições de 600-602 Hz com o FPGA..... 58
Figura 4.9 – Medições de 700-702 Hz com o FPGA..... 59
Figura 4.10 – Medições de 800-802 Hz com o FPGA..... 59
Figura 4.11 – Medições de 900-902 Hz com o FPGA..... 60
Figura 4.12 – Medições de 998-1.000 Hz com o FPGA..... 60
Figura 4.13 – Gráfico dos Resultados com o MCU 8031. 61
Figura 4.14 – Medições de 500-1000 Hz com o MCU PIC 18. 62
Figura 4.15 – Medições de 500-520 Hz com o MCU PIC 18. 63
Figura 4.16 – Medições de 700-720 Hz com o MCU PIC 18. 63
Figura 417 – Medições de 1000-1020 Hz com o MCU PIC 18. 64
Figura 4.18 – Medições de 500-502 Hz com o MCU PIC 18. 64
Figura 4.19 – Medições de 700-702 Hz com o MCU PIC 18. 65
Figura 4.20 – Medições de 950-952 Hz com o MCU PIC 18. 65
Figura 4.21 – Aplicação de Força de 0 – 250 N com o MCU PIC 18 (11 N de Força Inicial na Corda)... 66

Figura 4.22 – Teste de Repetibilidade do Sensor para 78,45 N. 67
Figura 4.23 – Aplicação de Força de 0 – 250 N com o MCU PIC 18 (8 N de Força Inicial na Corda). 68
Figura 4.22 – Teste de Repetibilidade do Sensor para 78,45 N. 69
Figura 4.23 – Conexão entre a Célula de Carga e o Sensor à Corda Vibrante. 71

ÍNDICE DE TABELAS

Tabela 2.1 – Classificação dos Componentes FPGA da Altera. 23
Tabela 3.1 – Componentes do Módulo da Altera. 39
Tabela 4.1 – Dados do Corpo do Sensor e da Corda. 53
Tabela 4.2 – Relações entre a Força Externa e Freqüência da Corda 54
Tabela 4.3 – Leituras da Freqüência do Teste de Repetibilidade 67
Tabela 4.3 – Leituras da Freqüência do Teste de Repetibilidade 69
Tabela 4.4 – Resultados com Indivíduos. 70

1. INTRODUÇÃO

O sensor de corda vibrante pode ser utilizado para mensurar força, deslocamento, torque, pressão, com grande precisão e confiabilidade[1]. Sendo um sensor de grande robustez, longa vida útil e de fácil instalação, ele é largamente empregado em barragens, minas e escavações, estruturas prediais, entre outras[2].

Com um princípio de funcionamento simples, toda a teoria do sensor de corda foi completamente estabelecida. Pesquisas recentes tem empregado esta técnica no desenvolvimento de novos sensores, com sensíveis modificações em relação aos primeiros dispositivos.

No entanto, para os transdutores de força e pressão, os modelos mais comuns, houve poucas modificações no decorrer dos anos, ao contrário dos seus sistemas de aquisição de dados, que evoluíram de sistemas acústicos[3] a computadores digitais, podendo realizar leituras de vários e diferentes transdutores.

O trabalho apresenta os conceitos fundamentais do sensor, descreve suas características construtivas e dois circuitos distintos de condicionamento de sinais, empregando microcontrolador e dispositivos lógicos programáveis (FPGA).

1.1. Apresentação do Projeto

Neste projeto, implementou-se um sensor à corda vibrante, e dois circuitos de condicionamento de sinais, visando aplicações em Engenharia de Reabilitação. O sensor possui estrutura de dura-alumínio, instrumentando um dinamômetro, objetivando-se medir a soma das forças exercidas pelos dedos das mãos e para calibrar sensores tácteis individuais, e uma muleta do tipo canadense, com o objetivo de medir as forças exercidas pelos membros superiores de pacientes paraplégicos. Trata-se de uma aplicação inovadora em Engenharia de Reabilitação.

O circuito de condicionamento e aquisição de sinais consiste em uma parte analógica e uma digital. Um amplificador e um filtro constituem a parte analógica. A digital (medidor de freqüência) foi implementada com dispositivos lógicos programáveis e com microcontroladores.

No sistema de aquisição com dispositivos lógicos programáveis, utilizou-se o ambiente Altera, que dispõem de um módulo didático para implementações práticas, com o componente da família FLEX 10K. Este sistema foi empregado para implementar um freqüencímetro.

No sistema de aquisição implementado com microcontroladores, utilizou-se o 8031 da Intel, instalado no kit Degen EB-153, e também o PIC18C452 da Microchip. O programa do 8031 foi implementado no próprio sistema da Degen, utilizando linguagem assembly do kit. Para o PIC 18, o programa monitor do chip foi implementado e compilado no ambiente MPLAB IDE, também da Microchip.

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

No projeto inicial de pesquisa, não estava previsto estudos com microcontroladores. Porém, com a grande importância que este singular componente vem alcançando, resolveu-se iniciar este estudo em paralelo com os dispositivos lógicos programáveis. Isto vem de encontro com os recentes desenvolvimentos na conexão de sensores com redes de dados, que visam o sensoriamento distribuído[4, 5, 6].

O dinamômetro consiste de duas hastes (haste 1 e 2) suportadas em uma base, o sensor é inserido entre a garra 2 e a haste 2, por meio de engates moveis (girantes). A garra 1 é engastada na haste 1. O paciente exercerá força de forma a aproximar as garras, aplicando um esforço longitudinal sobre o sensor, sem a influência de momentos fletores. A Figura 1.1 mostra uma idealização do dinamômetro.

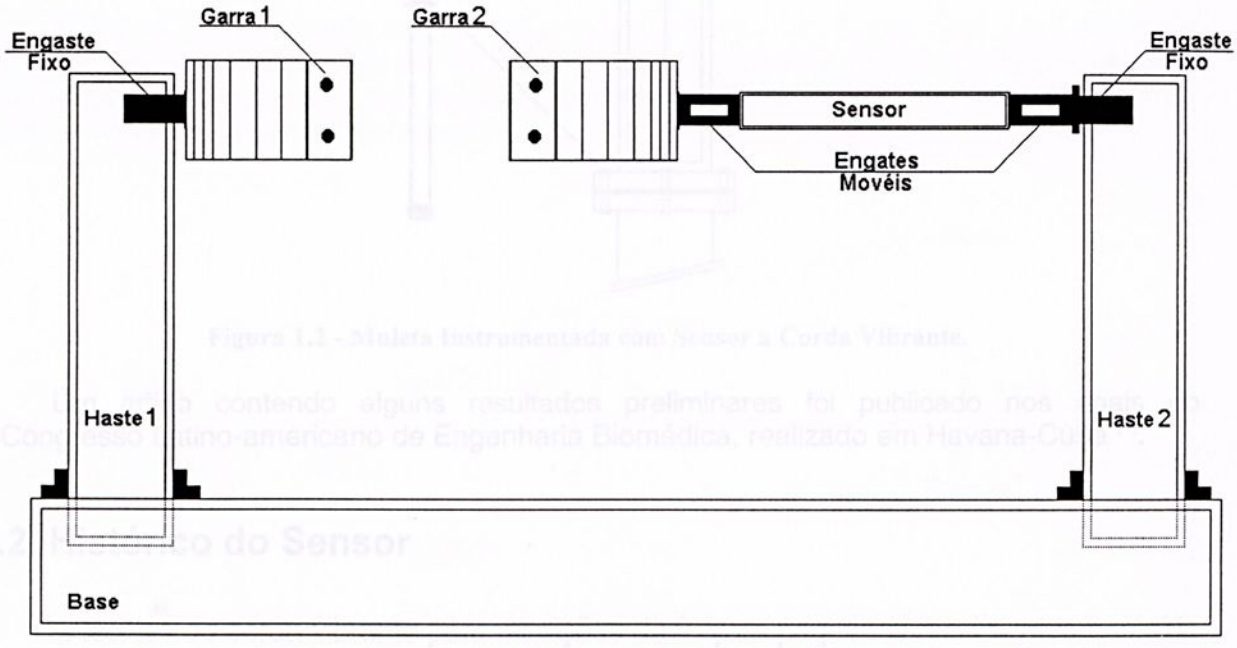


Figura 1.1 – Ilustração Simples do Dinamômetro.

Com a muleta (tipo canadense e em duralumínio) instrumentada com um sensor à corda vibrante, pode-se medir e monitorar os esforços normais exercidos pelos membros superiorers de pacientes sobre muletas, durante uma marcha. A Figura 1.2 ilustra a muleta instrumentada.

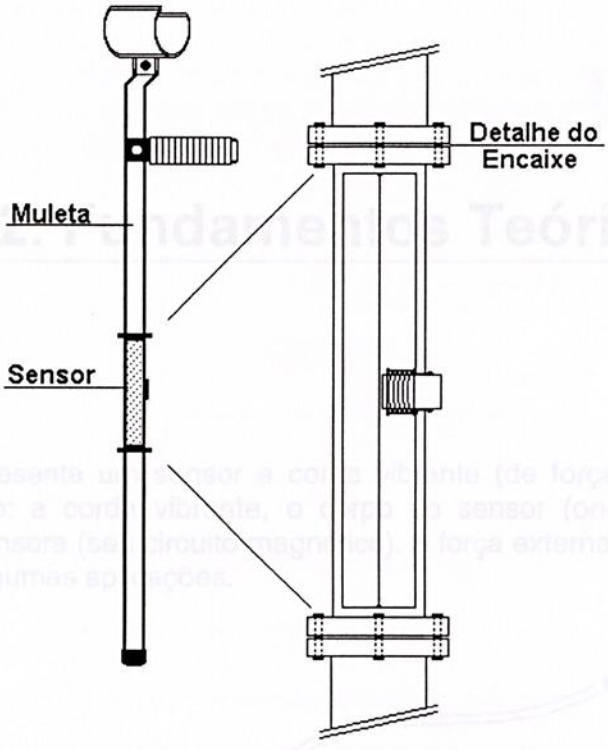


Figura 1.2 - Muleta Instrumentada com Sensor à Corda Vibrante.

Um artigo contendo alguns resultados preliminares foi publicado nos anais do II Congresso Latino-americano de Engenharia Biomédica, realizado em Havana-Cuba [7].

1.2. Histórico do Sensor

A técnica de corda vibrante para medições de força e pequenos movimentos tem sido usada há pelo menos 100 anos. O primeiro registro de uso comercial foi na França, onde, em 1928, Andre Coyne patenteou um medidor de força à corda vibrante.

Essa técnica, aplicada para medições de pressões de água, começou na Alemanha e na França, na década de 50, onde os requisitos para um transdutor de pressão eram capacidade de monitoramento remoto e boa estabilidade a longo prazo. Os transdutores de pressão eram instalados nas paredes de represas. Aproximadamente 3000 transdutores de corda vibrante foram instalados na década de 50, e 90% deles ainda funcionavam perfeitamente depois de 30 anos.

Posteriormente melhorias na miniaturização e no selamento dos transdutores foram feitas, na década de 80, por McRae e Sellers, utilizando uma técnica de fixação da corda, desenvolvida primeiramente pelo Nowegian Geotechnical Institute e depois modificado por Hawkes[8].

Uma grande inovação descrita por Feng et. al.[9] consistiu em conduzir o sinal do sensor ao sistema de controle por meio de fibra óptica, tornando o sensor praticamente imune a interferências eletromagnéticas. Pesquisas recentes tem empregado a técnica à corda vibrante no desenvolvimento de novos sensores (viscosímetros e densímetros)[10], com sensíveis modificações em relação aos primeiros dispositivos.

2. Fundamentos Teóricos

2.1. O Sensor

A Figura 2.1 apresenta um sensor a corda vibrante (de força) com seus principais componentes, que são: a corda vibrante, o corpo do sensor (onde é aplicada a força externa), e a bobina sensora (seu circuito magnético). A força externa representada pode ter sentido inverso para algumas aplicações.

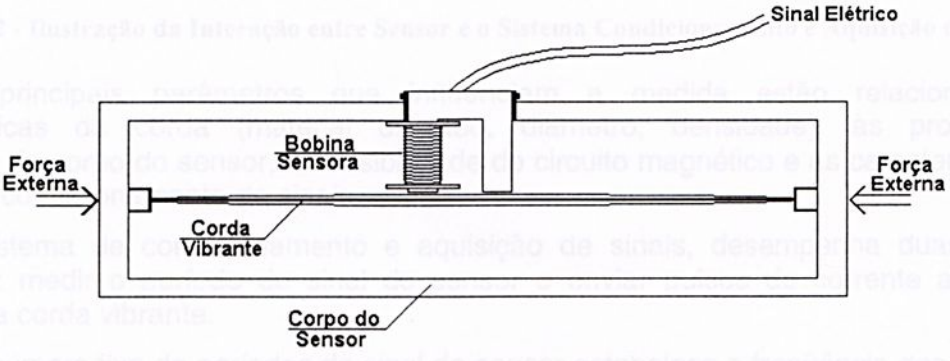


Figura 2.1 - O Sensor de Corda Vibrante e seus Componentes.

Os sensores a corda vibrante estão baseados no comportamento da oscilação da corda que, possuindo um comprimento L, densidade linear de massa ρ, e tencionada por uma força longitudinal **T**, terá uma freqüência f de oscilação, dada por (2.1).

$$f = \frac{1}{2L}\left(\frac{T}{\rho}\right)^{1/2} \qquad (2.1)$$

Com a variação de algum dos três parâmetros (**T**, **ρ** ou **L**) haverá mudança na freqüência de oscilação. Esta alteração pode ser "sentida" pelo circuito magnético instalado próximo da corda oscilante (Figura 2.1).

2.2. Princípio de Funcionamento

Nos sensores mais comuns a força aplicada na corda é o parâmetro variável, alterada por meio de uma força externa que atua no corpo do sensor.

A bobina sensora é utilizada para iniciar a oscilação da corda, através da inserção de

um pulso elétrico. A seguir, a própria bobina é usada para captar a freqüência de vibração da corda. O sinal de saída do sensor é uma onda senoidal, cuja freqüência f pode ser medida através de um sistema de aquisição de dados; o qual analisa os dados colhidos, associando-os a grandeza mensurada (força, pressão, deslocamento, temperatura, etc.). A Figura 2.2 ilustra a conexão do sensor com o sistema de condicionamento e aquisição de sinais.

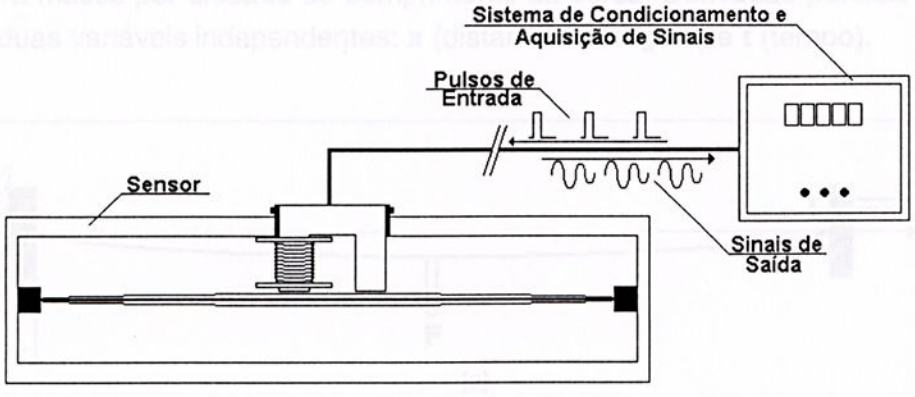


Figura 2.2 - Ilustração da Interação entre Sensor e o Sistema Condicionamento e Aquisição de Sinais.

Os principais parâmetros que influenciam a medida estão relacionados às características da corda (material utilizado, diâmetro, densidade), às propriedades mecânicas do corpo do sensor, à sensibilidade do circuito magnético e às características do circuito de condicionamento de sinais.

O sistema de condicionamento e aquisição de sinais, desempenha duas funções essenciais: medir o período do sinal do sensor e enviar pulsos de corrente ao sensor, excitando a corda vibrante.

Um número fixo de períodos do sinal do sensor estabelece a freqüência dos pulsos de excitação. Neste projeto o sensor possui um sinal de saída com freqüência na faixa de 500-1000 Hz, e o sinal de excitação com variação de 25-50 Hz, com a geração de aproximadamente um pulso de excitação a cada 20 ciclos do sinal do sensor.

2.2.1. Freqüência de Vibração

Para descrever o comportamento de uma corda fixa por seus extremos e tencionada por uma força, considera-se uma corda flexível, que não oferece resistência à flexão, nem ao cisalhamento e sua tensão é constante para pequenos deslocamentos.

A Figura 2.3a ilustra, uma corda de comprimento L, fixada por seus extremos e tensionada por uma força T; a Figura 2.3b mostra uma porção infinitesimal dx, distando de um dos extremos a distância x, deslocada do eixo central de uma medida y, formando os ângulos α e β com a horizontal.

Há um deslocamento de massa no sentido do eixo y, através de uma aceleração, logo:

$$\sum F = m\ddot{y} \qquad \text{(a)}$$

Colocando este princípio (a) para o eixo y da Figura 2.3, tem-se que,

$$-T\,\mathrm{sen}\,\beta + T\,\mathrm{sen}\,\alpha = (\rho\,dx)\frac{\partial^2 y}{\partial t^2}$$

sendo ρ é a massa por unidade de comprimento da corda. Derivadas parciais são usadas, porque há duas variáveis independentes: **x** (distância da origem) e **t** (tempo).

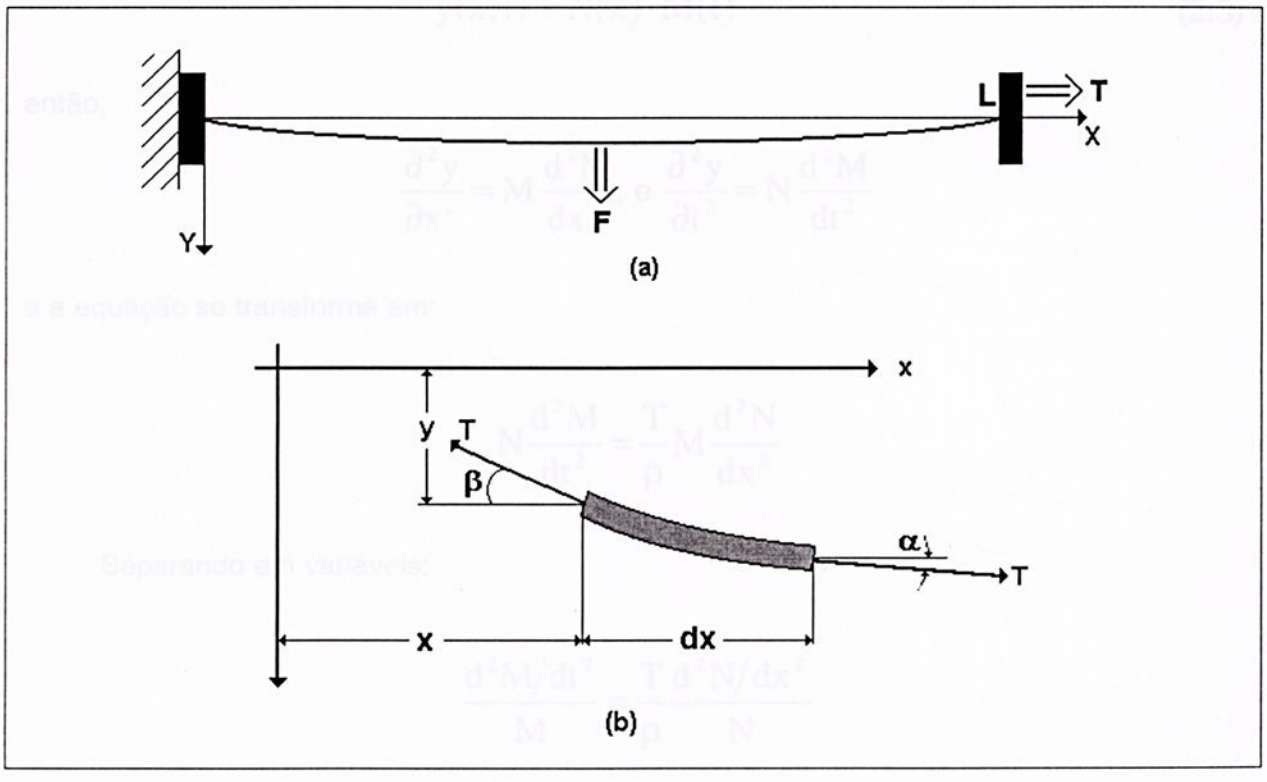


Figura 2.3 - (a) Corda Fixa e Tensa, Deslocada por uma Força Externa do seu Eixo Central. (b) Porção dx Tomada para Análise.

Mas $\left(\frac{\partial y}{\partial x}\right)_{x=X} = \mathrm{tg}\beta$, e $\left(\frac{\partial y}{\partial x}\right)_{x=X+dx} = \mathrm{tg}\alpha$, e para pequenos deslocamentos: $\mathrm{sen}\,\beta \cong \mathrm{tg}\beta$ e $\mathrm{sen}\,\alpha \cong \mathrm{tg}\alpha$. Logo:

$$-T\left(\frac{\partial y}{\partial x}\right)_{x=X} + T\left(\frac{\partial y}{\partial x}\right)_{x=X+dx} = (\rho\,dx)\frac{\partial^2 y}{\partial t^2}$$

$$\text{ou, } \rho\frac{\partial^2 y}{\partial t^2} = \frac{T(\partial y/\partial x)_{x=x+dx} - T(\partial y/\partial x)_{x=x}}{dx} \qquad \text{(b)}$$

Como o segundo termo de (b) é uma derivada, tem-se:

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\rho}\frac{\partial^2 y}{\partial x^2} \tag{2.2}$$

Esta é a equação diferencial de uma corda vibrante[11, 12]. Existem duas variáveis independentes. A solução desta equação pode ser obtida pelo *método das variáveis separadas*. Assim:

$$y(x,t) = N(x)\cdot M(t) \tag{2.3}$$

então,

$$\frac{\partial^2 y}{\partial x^2} = M\frac{d^2 N}{dx^2}, \text{ e } \frac{\partial^2 y}{\partial t^2} = N\frac{d^2 M}{dt^2}$$

e a equação se transforma em:

$$N\frac{d^2 M}{dt^2} = \frac{T}{\rho}M\frac{d^2 N}{dx^2}$$

Separando em variáveis:

$$\frac{d^2 M/dt^2}{M} = \frac{T}{\rho}\frac{d^2 N/dx^2}{N}$$

Como **N** e **M** são independentes um do outro, a expressão acima deve ser igual a uma constante, utilizando **- k²**. Isto conduz a duas equações diferenciais ordinárias:

$$\frac{d^2 M}{dt^2} + k^2 M = 0 \tag{c}$$

$$\frac{d^2 N}{dx^2} + \frac{k^2}{T/\rho}N = 0 \tag{d}$$

Conhecendo, pela *teoria de equações diferenciais*, que a solução para as duas é do tipo: **A sen (z t)**. Com (c) e (d), voltando a expressão (2.3), a solução será:

$$y(x,t) = \left(A\,\mathrm{sen}\frac{kx}{\sqrt{T/\rho}} + B\cos\frac{kx}{\sqrt{T/\rho}} \right)\left(C\,\mathrm{sen}\,kt + Dkt\right) \qquad (2.4)$$

Esta expressão traduz a posição de uma partícula da corda no eixo y (Figura 2.3b), localizada no ponto x, no instante t. Com esta interpretação conclui-se que nos extremos da corda:

$$y(0,t) = 0$$

$$y(L,t) = 0$$

pois os extremos estão fixos e imóveis em qualquer instante. Analisando a expressão (2.4), com estas condições, conclui-se que B é nulo, e que:

$$0 = A\,\mathrm{sen}\frac{kL}{\sqrt{T/\rho}}\left(C\,\mathrm{sen}\,kt + Dkt\right) \qquad (e)$$

Em (e), **A** é uma constante, não podendo ser sempre igual a zero, conclui-se que:

$$\mathrm{sen}\frac{kL}{\sqrt{T/\rho}} = 0 \qquad (f)$$

O valor **k** em (f) é a freqüência angular ω. Na busca pelo valor da freqüência da primeira harmônica, o seu valor é expresso por $\omega = 2\pi f$. Assim, para a primeira harmônica:

$$\frac{\omega L}{\sqrt{T/\rho}} = \frac{2\pi f\, L}{\sqrt{T/\rho}} = \pi \qquad (g)$$

Resolvendo (g) para a freqüência **f**:

$$f = \frac{1}{2L}\sqrt{\frac{T}{\rho}} \qquad (2.5)$$

A freqüência de vibração da corda é proporcional à raiz quadrada da força a que está submetida e inversamente proporcional ao seu comprimento e à raiz quadrada de sua densidade linear ρ.

A equação (2.5) será fundamental para o estudo, pois ela fornece a relação entre a

freqüência de vibração e a força **T** aplicada à corda. Se de alguma forma se produzir uma variação nesta força, pode-se "senti-la" através da variação da freqüência.

Observando o sentido de aplicação da força externa na Figura 2.1, a freqüência de vibração da corda será máxima quando não for aplicada nenhuma força externa no sensor, ficando a corda submetida a uma força **T**. Se houver uma força externa (**F**) aplicada sobre o sensor, que faça com que a corda seja tencionada com uma força $T - F$, sua nova freqüência de vibração será proporcional a esta diferença.

Se o sentido de força externa da Figura 2.1 for inverso, a freqüência máxima será quando for aplicada sobre o sensor a força externa máxima.

Este é o princípio de funcionamento do sensor à corda vibrante: qualquer grandeza que tenha relação e realize uma variação na vibração da corda, pode ser mensurada por este sensor. Com esse princípio, pode-se mensurar força, pressão, aceleração, temperatura, pequenos deslocamentos, viscosidades de líquidos e gases, entre outras grandezas.

2.2.2. Relação entre a Freqüência e a Força Externa

O objetivo desta abordagem é obter uma relação entre a força externa aplicada sobre o sensor e a freqüência de vibração da corda correspondente, tanto para a compressão quanto para a tração do corpo do sensor.

A Figura 2.4 está representando o corpo de sensor e sua reação a uma força externa de compressão F; para a tração o sentido da força externa seria o inverso. As Figuras 2.5 e 2.6 ilustram o diagrama de forças, para as duas situações.

Este corpo sensor é projetado para que sua reação consista de uma deformação linear não permanente (elástica), que altere a força que está sendo exercida sobre a corda, diminuindo-a, dentro de alguns limites, ou aumentando-a, no caso de tração sobre o sensor.

Assim, há duas características a serem estudadas em um projeto: a força inicial a ser aplicada na corda (definida durante a montagem do sensor) e a máxima força mensurável pelo sensor. Esta limitação ocorre porque o afrouxamento demasiado da corda permite que sua oscilação apresente componentes de ordem superior à primeira harmônica da freqüência, prejudicando a leitura do sinal.

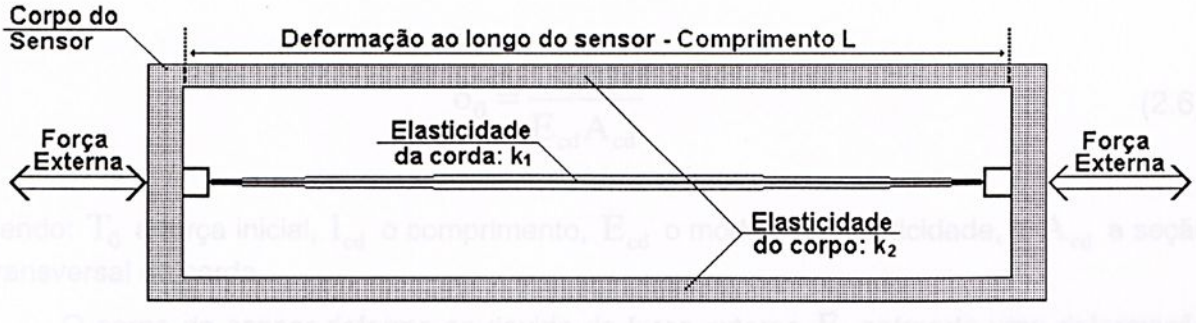


Figura 2.4 - A Corda Vibrante e o Corpo do Sensor Sofrendo a Aplicação de uma Força.

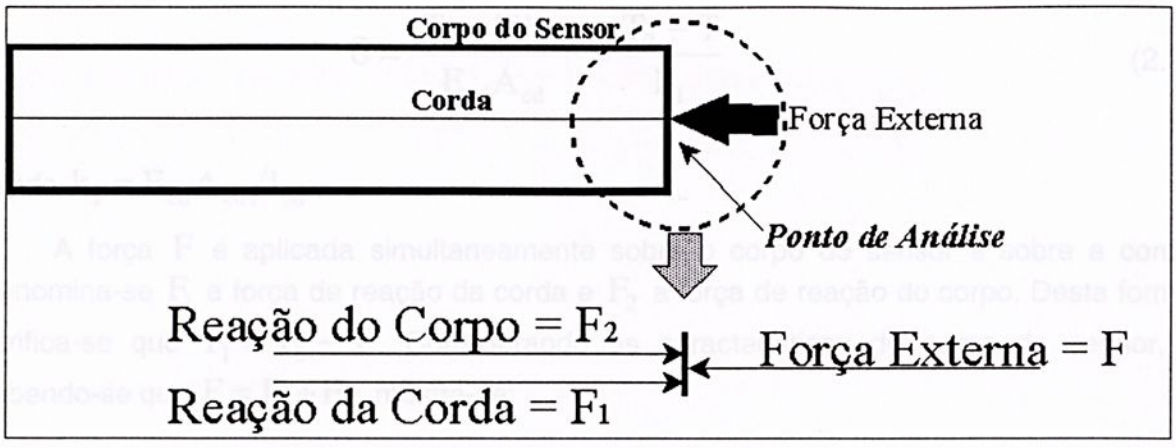


Figura 2.5 – Diagrama de Forças sobre o Sensor sob Compressão.

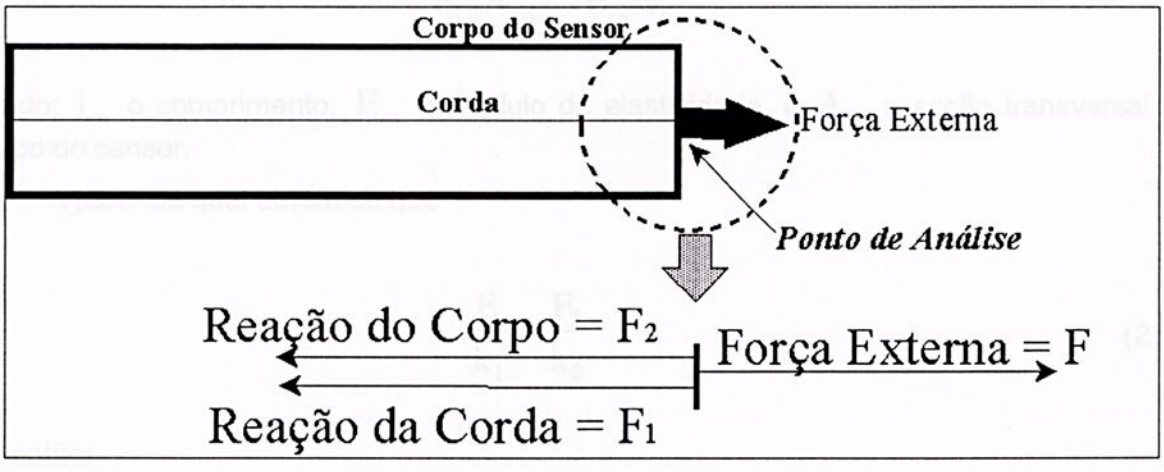


Figura 2.6 – Diagrama de Forças sobre o Sensor sob Tração.

Tendo como base as Figuras 2.4, 2.5 e 2.6, equaciona-se as deformações sofridas pela corda e pelo corpo do sensor e chega-se a relação entre freqüência de vibração da corda e a força aplicada externamente.

Realizando primeiramente o caso de compressão do sensor, a deformação inicial da corda ao longo de seu comprimento, durante a montagem do sensor é dado por[13]:

$$\delta_0 = \frac{T_0 l_{cd}}{E_{cd} A_{cd}} \tag{2.6}$$

sendo: T_0 a força inicial, l_{cd} o comprimento, E_{cd} o módulo de elasticidade, e A_{cd} a seção transversal da corda.

O corpo do sensor deforma-se devido da força externa F, sofrendo uma deformação δ ao longo de seu comprimento. A corda sofrerá a mesma deformação δ, diminuindo a força inicial para um valor T. Por sua vez, esta deformação é resultante da diferença $T_0 - T$, logo, (2.6) converte-se em:

$$\delta = \frac{(T_0 - T)\,l_{cd}}{E_{cd}A_{cd}} = \frac{T_0 - T}{k_1} \tag{2.7}$$

sendo $k_1 = E_{cd}A_{cd}/l_{cd}$.

A força F é aplicada simultaneamente sobre o corpo do sensor e sobre a corda. Denomina-se F_1 a força de reação da corda e F_2 a força de reação do corpo. Desta forma, verifica-se que $F_1 = T_0 - T$. Considerando as características do corpo do sensor, e sabendo-se que $F = F_1 + F_2$, mostra-se:

$$\delta = \frac{F_1 l_{cd}}{E_{cd}A_{cd}} = \frac{F_2 l_{cs}}{E_{cs}A_{cs}} \tag{2.8}$$

sendo: l_{cs} o comprimento, E_{cs} o módulo de elasticidade, e A_{cs} a seção transversal do corpo do sensor.

A partir da qual obtém-se que

$$\frac{F_1}{k_1} = \frac{F_2}{k_2} \tag{2.9}$$

ou então:

$$F_1 = \frac{F_2 k_1}{k_2} \tag{2.10}$$

sendo $k_2 = E_{cs}A_{cs}/l_{cs}$.

Assim, substituindo $F_1 = F - F_2$ em (2.10), mostra-se que:

$$F_2 = \frac{F k_2}{k_1 + k_2} \tag{2.11}$$

Com o auxílio de (2.7), (2.9) e (2.11), verifica-se que a expressão para a deformação sofrida pelo sensor, que é igual a deformação da corda, será:

$$\delta = \frac{F}{k_1 + k_2} = \frac{T_0 - T}{k_1} \tag{2.12}$$

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

sendo: T a nova força aplicada sobre a corda com a presença de F.

Isolando-se T, em (2.12) tem-se:

$$T = T_0 - \frac{k_1}{k_1 + k_2} F \tag{2.13}$$

Substituindo-se (2.13) em (2.5), obtém-se finalmente:

$$f = \frac{1}{2L} \sqrt{\frac{T_0 - \frac{k_1}{k_1 + k_2} F}{\rho}} \tag{2.14}$$

A expressão (2.14) traduz a freqüência de oscilação da corda quando sobre o corpo do sensor é aplicada uma força externa, considerando as características da corda, do corpo e a força inicial aplicada sobre a corda.

Para a tração do corpo do sensor, tem-se que a corda terá aumento da sua força de tração, aumentando assim sua freqüência de vibração. O corpo do sensor deforma-se devido a força externa F, sofrendo uma deformação δ ao longo de seu comprimento. A corda sofrerá a mesma deformação δ, aumentando a força inicial para um valor T. Por sua vez, esta deformação é resultante da diferença $T - T_0$. Assim, da equação (2.12) tem-se:

$$\delta = \frac{F}{k_1 + k_2} = \frac{T - T_0}{k_1} \tag{2.15}$$

Sabendo-se que F_1 (a força de reação da corda, tal como antes) é a força que produz a nova deformação da corda, verifica-se que na situação de tração, $F_1 = T - T_0$, sendo T a nova força aplicada sobre a corda com a presença de F. Isolando-se T, em (2.15) tem-se:

$$T = T_0 + \frac{k_1}{k_1 + k_2} F \tag{2.16}$$

Substituindo-se (2.16) em (2.5), obtém-se:

$$f = \frac{1}{2L}\sqrt{\frac{T_0 + \frac{k_1}{k_1 + k_2}F}{\rho}} \qquad (2.17)$$

Semelhante a expressão (2.14), a (2.17) traduz o aumento na freqüência de vibração da corda com uma força externa tracionando o corpo do sensor.

Estas duas equações (2.14) e (2.17) devem ser utilizadas pelo sistema de condicionamento e aquisição de sinais para interpretar de maneira mais conveniente o sinal do sensor, podendo, assim, apresentar ao usuário, um valor mensurado em Newtons, ou em outras unidades de força.

2.2.3. A Captação da Freqüência

A estrutura básica do captador é mostrada na Figura 2.7. Uma bobina com várias espiras é enrolada em torno de um pequeno imã ou um eletroimã. O campo magnético estático do imã produz um pólo norte e um pólo sul na porção da corda metálica mais próxima, pois é constituída de material ferromagnético. Esta porção torna-se um imã, com campo magnético próprio.

Quando a corda oscila, seu movimento relativo à bobina varia o fluxo magnético através dela, induzindo uma corrente elétrica. Como a corda oscila aproximando-se e afastando-se da bobina (no eixo **y**, conforme Figura 2.3), a corrente induzida varia de sentido com a mesma freqüência da oscilação da corda. Desse modo, a freqüência do sinal elétrico senoidal produzido pela bobina sensora será a mesma da oscilação da corda, que será transmitido ao sistema de aquisição[14].

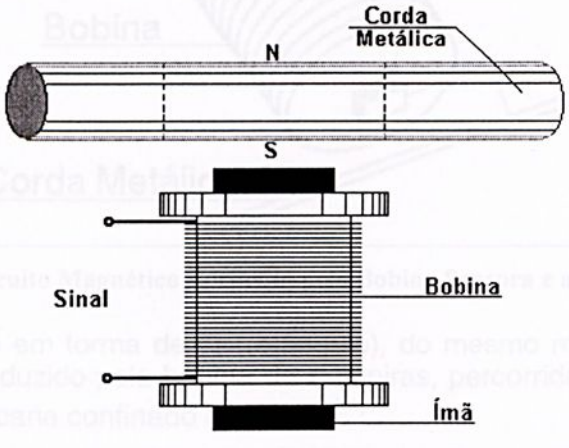


Figura 2.7 – Vista Lateral da Bobina Sensora e da Corda.

Para se iniciar um movimento oscilatório na corda, uma segunda bobina recebe um pulso de corrente que traciona momentaneamente a corda, e assim inicia-se sua oscilação.

O eletroimã, constituído pela segunda bobina e o núcleo, que produz um campo magnético estático sobre a corda metálica, pode estar localizado sobre outro suporte, não necessariamente sobre o mesmo da primeira bobina.

Uma outra forma de produzir este campo magnético estático consistiria da corda sendo percorrida por uma corrente DC, gerando um campo ao longo de seu comprimento. Uma das desvantagens deste método é a necessidade do isolamento galvânico entre a

corda e o corpo do sensor, o que implica no acréscimo de condutores até as extremidades do sensor. Porém, uma das vantagens seria a produção de um campo magnético mais intenso, com maior controle, produzindo-se, assim, um sinal de maior precisão[15].

Em uma abordagem teórica, no circuito magnético do sensor à corda vibrante deve-se fazer menção de várias expressões, deduzidas a partir das equações de Maxwell, pois as duas bobinas e a corda metálica interagem em campos magnéticos diferentes.

2.2.4. Cálculo do Campo Magnético Produzido por um Bobina com Núcleo Ferromagnético

A seqüência empregada aqui será o calculo do campo magnético produzido pelo pulso de corrente para o iniciar o movimento da corda, calculo do campo magnético residual na corda metálica, e finalmente a obtenção de uma expressão que forneça a f.e.m. induzida na bobina sensora, sendo o sinal enviado para o circuito de condicionamento e aquisição de sinais.

Considere a bobina da Figura 2.8. Admite-se que o campo magnético se concentra no interior do núcleo ferromagnético em toda a sua extensão, pois possui permeabilidade magnética muito superior ao ar que o envolve[16].

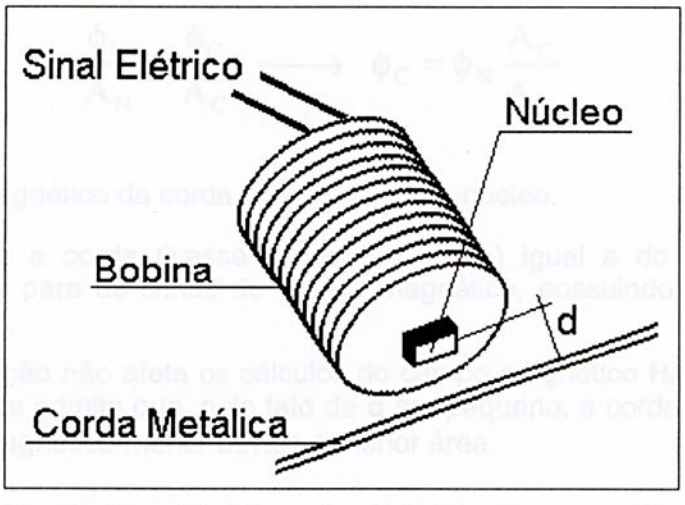


Figura 2.8 - Circuito Magnético Formado pela Bobina Sensora e a Corda Metálica.

Se o núcleo fosse em forma de □ (retângulo), do mesmo material ferromagnético, o campo magnético **H** produzido pela bobina de **n** espiras, percorrida pela corrente **I** seria de intensidade $H = nI$ e ficaria confinado ao núcleo[17].

No projeto do sensor à corda vibrante, o campo magnético produzido pelo pulso de corrente é guiado até as proximidades da corda metálica por meio de um núcleo ferromagnético, de forma que o valor nI pode ser considerado o valor do campo magnético na extremidade do núcleo. Entre esta extremidade e a corda, existe um entreferro formado por ar, conforme representado na Figura 2.8, tendo um comprimento **d**.

Na Figura 2.9, observa-se com mais detalhes o comportamento do campo magnético do circuito, inclusive algumas linhas de campo que atravessam o ar e a corda.

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

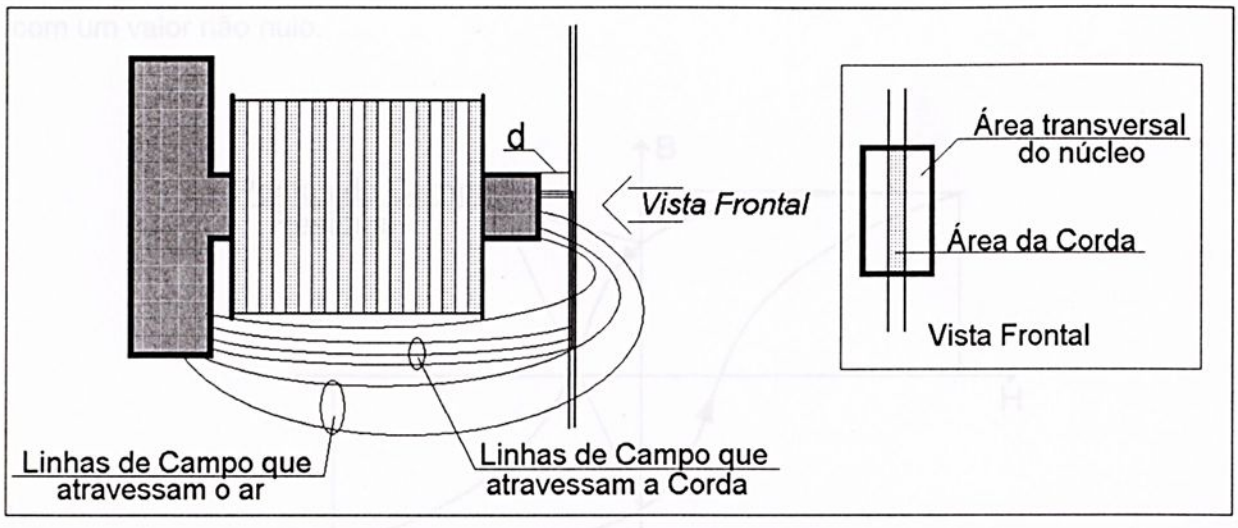


Figura 2.9 - Comportamento do Campo Magnético Produzido pela Bobina Sensora.

Tendo em vista que a distância **d** é muito menor que a distância percorrida pelas linhas de campo pelo ar, considera-se que a indução magnética B do núcleo é a mesma da corda. Assim, denominando-se A_N a área do núcleo, e A_C a área da corda, tem-se:

$$B = \frac{\phi_N}{A_N} = \frac{\phi_C}{A_C} \longrightarrow \phi_C = \phi_N \frac{A_C}{A_N} \tag{2.18}$$

sendo ϕ_C o fluxo magnético da corda e ϕ_N o fluxo do núcleo.

Resumindo, se a corda tivesse dimensões (área) igual a do núcleo, ela seria o caminho preferencial para as linhas de campo magnético, possuindo um fluxo magnético idêntico ao do núcleo.

Esta consideração não afeta os cálculos do campo magnético H_C com grandes erros, pois ela simplesmente admite que, pelo fato de **d** ser pequeno, a corda faz parte do núcleo, mas com um fluxo magnético menor devido a menor área.

2.2.5. Campo Magnético Residual

Uma das propriedades dos materiais ferromagnéticos, é a de "reter" uma porção do campo magnético aplicado sobre si. Na verdade, ao ser aplicado um campo magnético sobre um material com alta permeabilidade, suas moléculas polares se alinham, e o campo magnético de cada uma é somado ao das outras, e assim formam não somente em caminho de excelente permeabilidade, mas também um gerador de campo magnético.

Quando o campo magnético original, o que ocasionou o alinhamento das moléculas é retirado, as mesmas não retornam ao estado inicial em sua totalidade, mas algumas permanecem alinhadas, gerando campo magnético tal como um imã. Este campo é chamado de campo magnético residual.

A Figura 2.10 traz uma curva de magnetização característica deste efeito que o campo magnético produz. O campo **H** é aplicado sobre o material, que dependendo de sua permeabilidade, produz a indução magnética **B**. Ao ser anulado **H**, a indução permanece

com um valor não nulo.

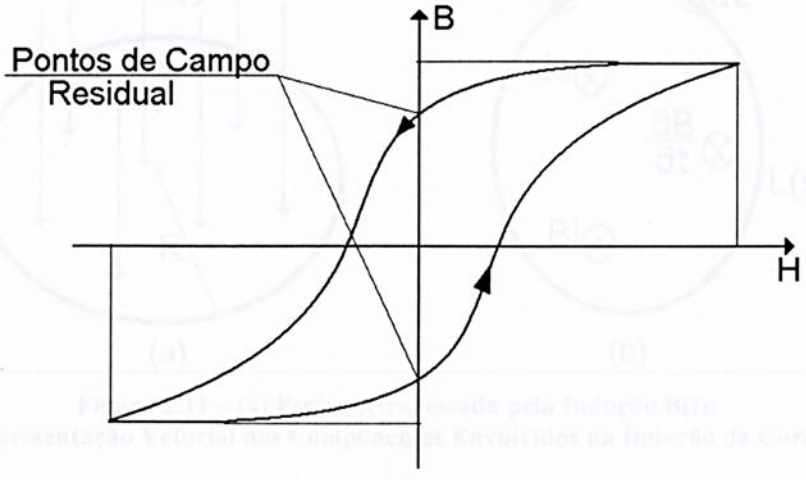


Figura 2.10 – Curva de Magnetização Característica.

De posse da curva de magnetização do material constituinte da corda, e conhecendo-se o valor da intensidade de campo magnético aplicado sobre ela, calcula-se o campo magnético residual, mensurando-se sua magnetização.

2.2.6. Correntes Induzidas por Variação de Indução

A indução magnética está expressa em relação ao campo magnético por:

$$B = \mu H \tag{2.19}$$

sendo μ a permeabilidade magnética do meio, **B** a indução magnética, e **H** o campo magnético.

A interpretação desta expressão pode ser entendida como: sob um campo magnético **H** em um meio de permeabilidade magnética μ, sua influência será igual a **B**.

Em uma espira, tal como traz a Figura 2.11a, a qual define uma superfície **s**, uma indução variável no tempo $B(t)$, dado por (2.20), atravessa em toda sua superfície. Em se tratando de uma espira condutora, ela "reagirá" de contra a variação temporal da indução magnética, produzindo uma indução B_i, contrária em sentido, mas na mesma direção de $B(t)$. Pode-se observar a representação deste fenômeno na Figura 2.11b.

$$B(t) = B_0 \operatorname{sen}(2\pi f \cdot t) \tag{2.20}$$

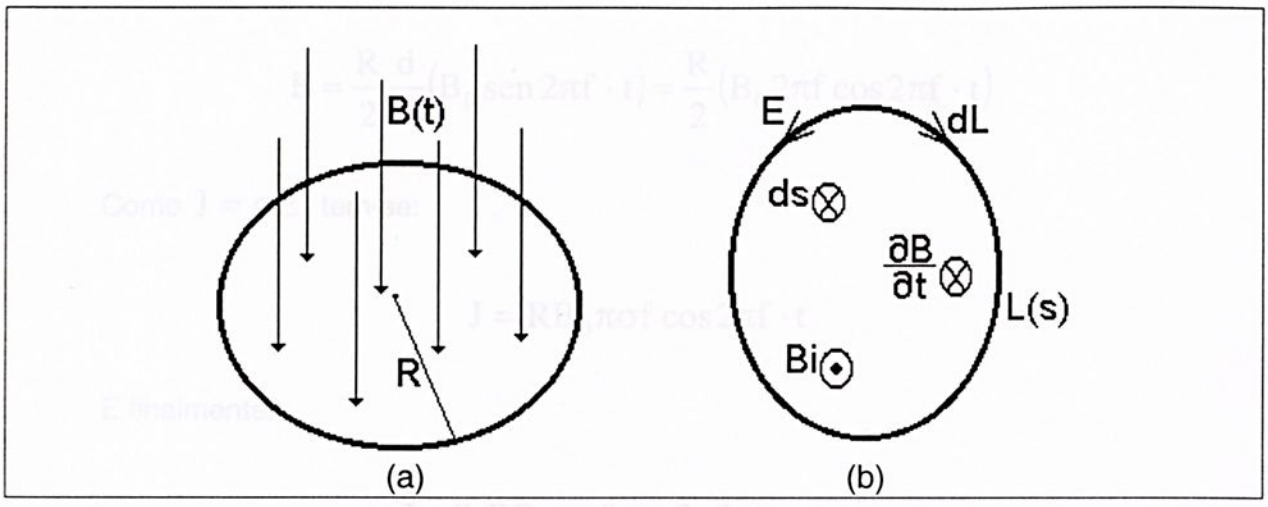


Figura 2.11 – (a) Espira Atravessada pela Indução B(t); (b) Representação Vetorial dos Componentes Envolvidos na Indução da Corrente.

Para calcular a corrente induzida na espira, que produz a indução B_i, são admitidas duas aproximações: a seção da espira é pequena em relação ao raio **R**, e assim, o campo elétrico **E**, e consequentemente $J = \sigma E$ (onde **J** é a densidade de corrente elétrica e σ é condutividade elétrica) são constantes em toda a espira; a outra aproximação, é que B_i da bobina é muito menor que $B(t)$ imposto pelo meio externo.

Pela terceira lei de Maxwell[16]:

$$\text{Rot } E = -\frac{\partial B}{\partial t} \tag{2.21}$$

se obtém a relação entre a geração do campo elétrico (o lado direito da expressão) e a variação da indução magnética no tempo (o lado esquerdo da expressão acima). O sinal negativo está representado na Figura 2.11 com o sentido invertido entre B_i e $\partial B/\partial t$.

Integrando a expressão (2.21) na superfície e ao longo da espira da Figura 2.11, tem-se:

$$\oint_{L(s)} E dL = \iint_S -\frac{\partial B}{\partial t} ds \tag{2.22}$$

onde **S** é a superfície da espira e **L(s)** é o caminho que envolve **S**.

Nota-se que **B** depende somente do tempo e que os pares de vetores **E**, **dL**, e $\partial B/\partial t$, **ds** são colineares, assim:

$$E 2\pi R = -\frac{dB(t)}{dt} \pi R^2$$

unesp

$$E = \frac{R}{2}\frac{d}{dt}\left(B_0 \operatorname{sen} 2\pi f \cdot t\right) = \frac{R}{2}\left(B_0 2\pi f \cos 2\pi f \cdot t\right)$$

Como $J = \sigma E$, tem-se:

$$J = RB_0\pi\sigma f \cos 2\pi f \cdot t$$

E finalmente:

$$I = S_e RB_0\pi\sigma f \cos 2\pi f \cdot t \qquad (2.23)$$

sendo S_e a seção transversal do fio que forma a espira.

O circuito de condicionamento e aquisição de sinais do sensor à corda vibrante trabalha com a f.e.m. induzida na bobina, assim, pode-se obter a expressão (2.23) de outra forma, e portanto, se ter uma melhor visualização do tensão induzida na bobina pela corda metálica magnetizada em seu movimento oscilatório. A f.e.m. criada na bobina será:

$$\text{f.e.m.} = \oint_{L(s)} E dL = \iint_S -\frac{\partial B}{\partial t} ds$$

E assim:

$$\text{f.e.m.} = \frac{dB}{dt}\pi R^2 = -\left(B_0 2\pi f \cos 2\pi f \cdot t\right)\pi R^2$$

$$\text{f.e.m.} = -2R^2 B_0 \pi^2 f \cos 2\pi f \cdot t \qquad (2.24)$$

Esta interpretação pode ser observada na Figura 2.12.

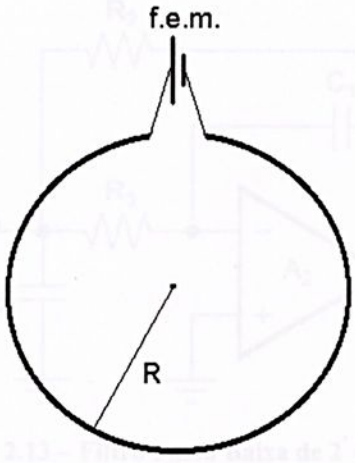


Figura 2.12 - A f.e.m. Induzida em uma Espira sob Indução Magnética Variável no Tempo.

Pela lei de Ohm, $f.e.m. = I/R_{ele}$ (a representação R_{ele} da resistência elétrica é para a distinção com **R** do raio). O valor de R_{ele} intrínseca ao material condutor é dada por;

$$R_{ele} = \frac{1}{\sigma}\frac{2\pi R}{S}$$

Realizando as substituições chega-se a mesma expressão (2.23) da corrente elétrica. Deve-se considerar cada espira, assim, para **n** espiras, a expressão (2.24) é alterada para a expressão (2.25).

$$f.e.m. = -2nR^2B_0\pi^2 f \cos 2\pi f \cdot t \qquad (2.25)$$

2.3. Sistema de Condicionamento e Aquisição de Sinais

Como o sinal produzido pelo sensor tem baixa amplitude, da ordem de milivolts, há necessidade de um circuito de condicionamento de sinais de qualidade, com fator de ganho alto e filtro de ruídos.

No projeto se utilizou dois amplificadores inversores em cascata, o que não afeta a forma de onda, pois um inversor compensa o outro.

O sinal do sensor, por ter baixa amplitude, é influenciado por vários sinais de ruído, e uma vez amplificado, os ruídos também o são. Fazendo o uso de um filtro ativo se amplifica o sinal do sensor e, ao mesmo tempo, se atenua os ruídos, já que se tratam de sinais de freqüências diferentes.

O circuito da Figura 2.13 representa um filtro passa baixas de 2ª ordem, na configuração Butterworth. Para um filtro de maior ordem, usam-se estes circuitos em cascata, como na Figura 2.14, onde as seções B e C são semelhantes as seções A e D; utilizando-se de fatores de otimização encontrados em livros específicos[17], calcula-se os valores dos resistores e dos capacitores de todo o conjunto.

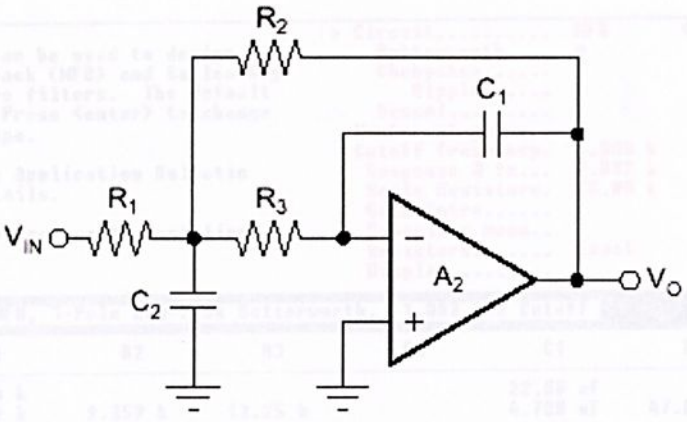


Figura 2.13 – Filtro Passa Baixa de 2ª Ordem.

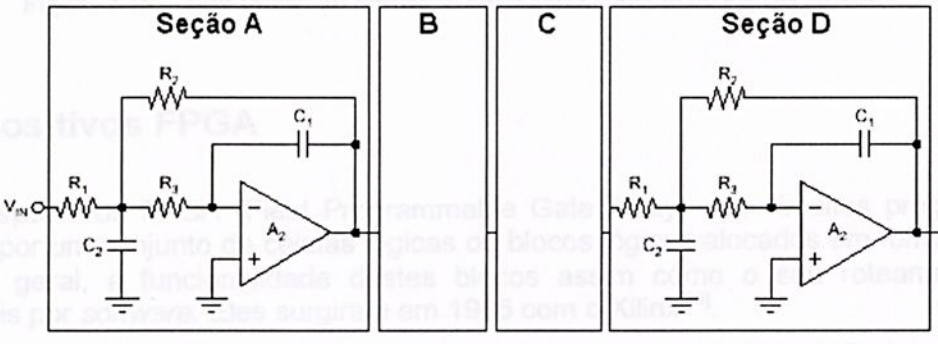


Figura 2.14 – Filtro Passa Baixa de 8ª Ordem.

Para o cálculo dos componentes que formam o filtro passa baixas, existem fórmulas pré-estabelecidas que combinam valores de ganho do filtro e freqüência de corte, bem como fatores de qualidade. O uso destas fórmulas se torna trabalhoso quando se trata de filtros de ordem elevadas. Por essa razão, existem softwares que realizam estes cálculos.

Um desses softwares, o Filter, distribuído no site da Burr-Brown Corporation[18], foi utilizado nesta pesquisa para projetar um filtro de 8ª ordem, com ganho 6 em cada seção da Figura 2.14, utilizando a configuração Butterworth.

A Figura 2.15 mostra a tela de apresentação inicial do software, já com alguns valores de componentes pré-carregados como exemplo.

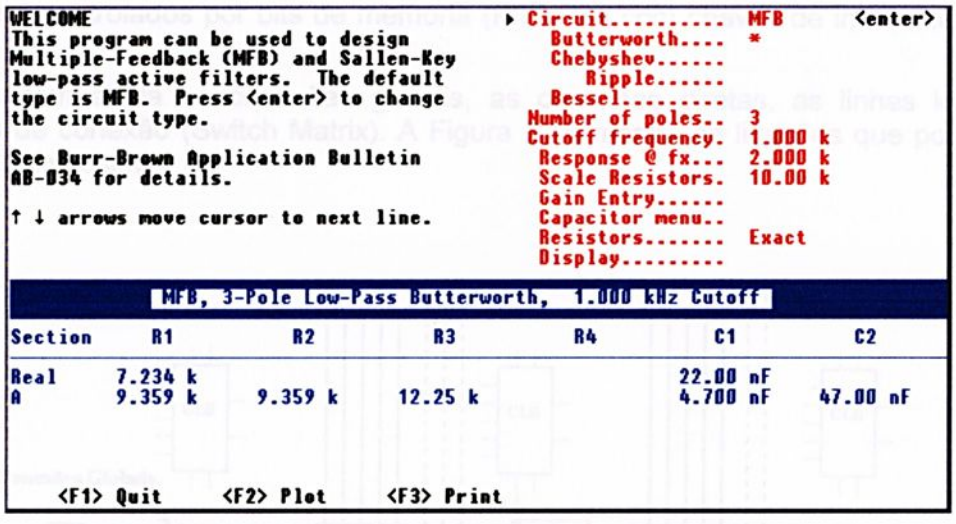


Figura 2.15 – Tela Inicial do Software Filter para Cálculo de Filtros Ativos.

2.4. Dispositivos FPGA

Os dispositivos FPGA (Field Programmable Gate Array) são circuitos programáveis compostos por um conjunto de células lógicas ou blocos lógicos alocados em forma de uma matriz. Em geral, a funcionalidade destes blocos assim como o seu roteamento, são configuráveis por *software*. Eles surgiram em 1985 com o Xilinx[19].

A utilização de FPGA se justifica por suas grandes vantagens. Pode-se realizar projetos em menor espaço de tempo, pois se elimina arranjos com componentes discretos, provendo uma descrição em alto-nível para projetar lógicas complexas, que requerem milhares de portas lógicas, além de suportar metodologia de projeto modular e vários níveis de hierarquia. O sistema FPGA utilizado neste trabalho foi o ambiente de projeto Max+Plus II da Altera, com linguagem AHDL. Com este sistema pode-se efetuar a descrição do circuito digital e sua simulação[20].

As principais companhias que desenvolvem a tecnologia FPGA são a Actel, Altera, Plessey, Plus, AMD, QuickLogic, Xilinx.

2.4.1. Estrutura da FPGA

Os dispositivos FPGA são formados por três elementos básicos, sendo eles: os CLB's (Configurable Logic Block) que são blocos lógicos configuráveis, os IOB's (In/Out Block) que são blocos de entrada e saída, e os SB's (Switch Box) que são as caixas de conexão.

Os CLB's compõem as unidades lógicas do FPGA, responsáveis pela capacidade de processamento. Cada CLB possui três geradores de função combinacional, dois Flip-Flop tipo D, lógica de controle e lógica aritmética dedicada. Cada bloco CLB pode ser configurado como um somador de 2 bits.

Os IOB's são a interface entre uma FPGA e o resto do sistema, e se localizam na periferia do chip. Os SB's (Switch Box) têm por finalidade permitir a interconexão entre os CLB's através dos canais de roteamento.

Roteamento é a interconexão entre os blocos, e é feita através de uma rede de camadas de metal. As conexões físicas entre os fios são feitas, ora com transistores de

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

passagem controlados por bits de memória (PIP), ora com chaves de interconexão (Switch Matrix).

Existem ainda as conexões globais, as conexões diretas, as linhas longas, e as matrizes de conexão (Switch Matrix). A Figura 2.16 mostra as ligações que podem ocorrer dentro do dispositivo.

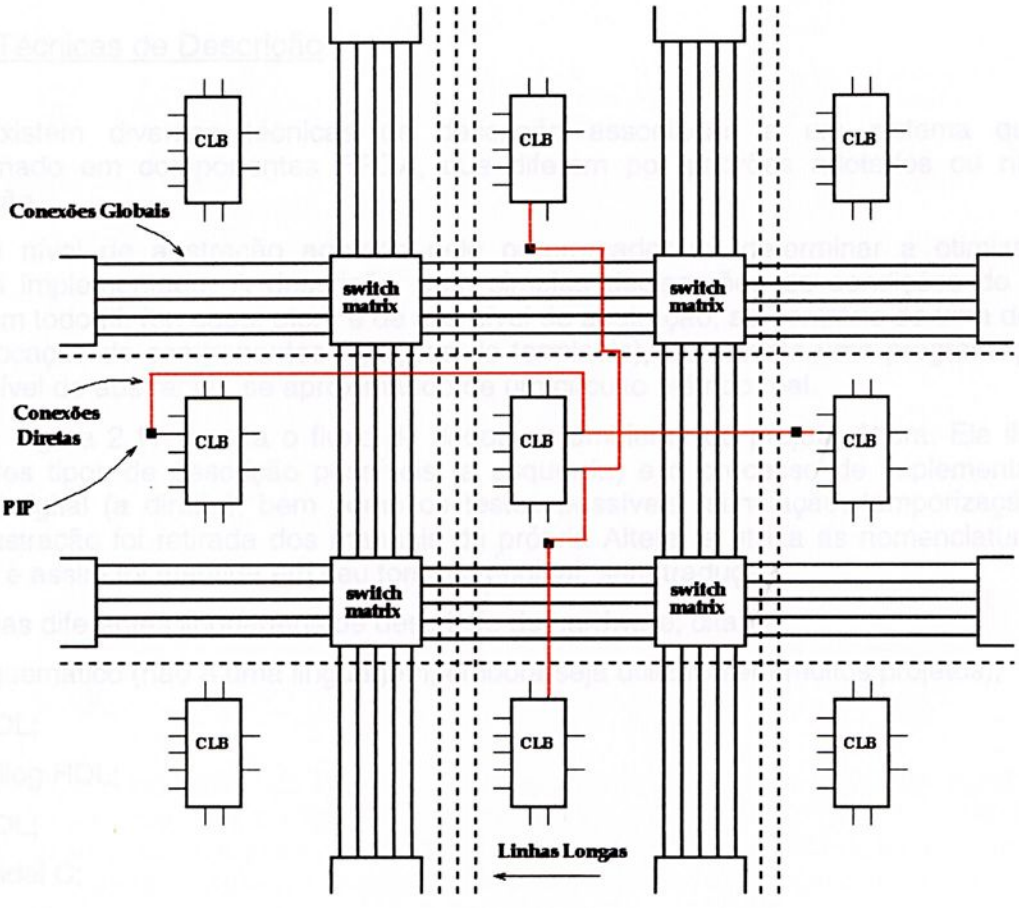


Figura 2.16 – Estrutura dos Dispositivos FPGA.

2.4.2. Tipos de FPGA – Altera

Dentre os fabricantes de componentes FPGA, a Altera tem se destacado com sua família de componentes, mas principalmente com os módulos de aprendizagem (Design Laboratory Package), distribuídos com baixo custo às universidades engajadas no Projeto Universitário Altera.

Neste projeto foram utilizados dois componentes da Altera, instalados em dois módulos de aprendizagem.

Sendo componentes programáveis, estes dispositivos são classificados tal como as memórias de microcontroladores. A Tabela 2.1 mostra a classificação da Altera para os seus componentes. Esta classificação determina se o programa gravado é "volátil" ou não.

Tabela 2.1 – Classificação dos Componentes FPGA da Altera.

Família do Componente	Tecnologia de Programação
FLEX 10K	SRAM
FLEX 8K	SRAM
MAX 9K	EEPROM
MAX 7K	EEPROM
MAX 5K	EEPROM

2.4.3. Técnicas de Descrição

Existem diversas técnicas de descrição associadas a um sistema que será programado em componentes FPGA, que diferem por padrões adotados ou níveis de abstração.

O nível de abstração adotado pelo programador irá determinar a otimização do sistema implementado. A descrição, com simples declarações de condições do sistema como um todo (if, for, case, etc.), é de alto nível de abstração, ao contrário de uma descrição com alocação de componentes (ligações de terminais), que revela uma programação com baixo nível de abstração, se aproximando de um circuito elétrico real.

A Figura 2.17 mostra o fluxo de dados no ambiente de projeto Altera. Ela ilustra os diferentes tipos de descrição possíveis (a esquerda) e o processo de implementação do circuito digital (a direita), bem como os testes possíveis (simulação, temporização, etc.). Esta ilustração foi retirada dos manuais da própria Altera, e utiliza as nomenclaturas mais usuais, e assim foi mantida em seu formato original, sem tradução.

Das diferentes linguagens de descrição de hardware, cita-se:

- Esquemático (não é uma linguagem, embora seja utilizado em muitos projetos);
- VHDL;
- Verilog HDL;
- AHDL;
- Handel C;
- VDL/I 6;
- M[21].

A VHDL (**V**ery Hight Speed Integrated Circuit **H**ardware **D**escription **L**inguage) é uma forma de se descrever, através de uma linguagem de programação, o comportamento de um circuito ou componente digital. Sua importância é tal que se tornou um padrão do IEEE (IEEE-1076) em 1987.

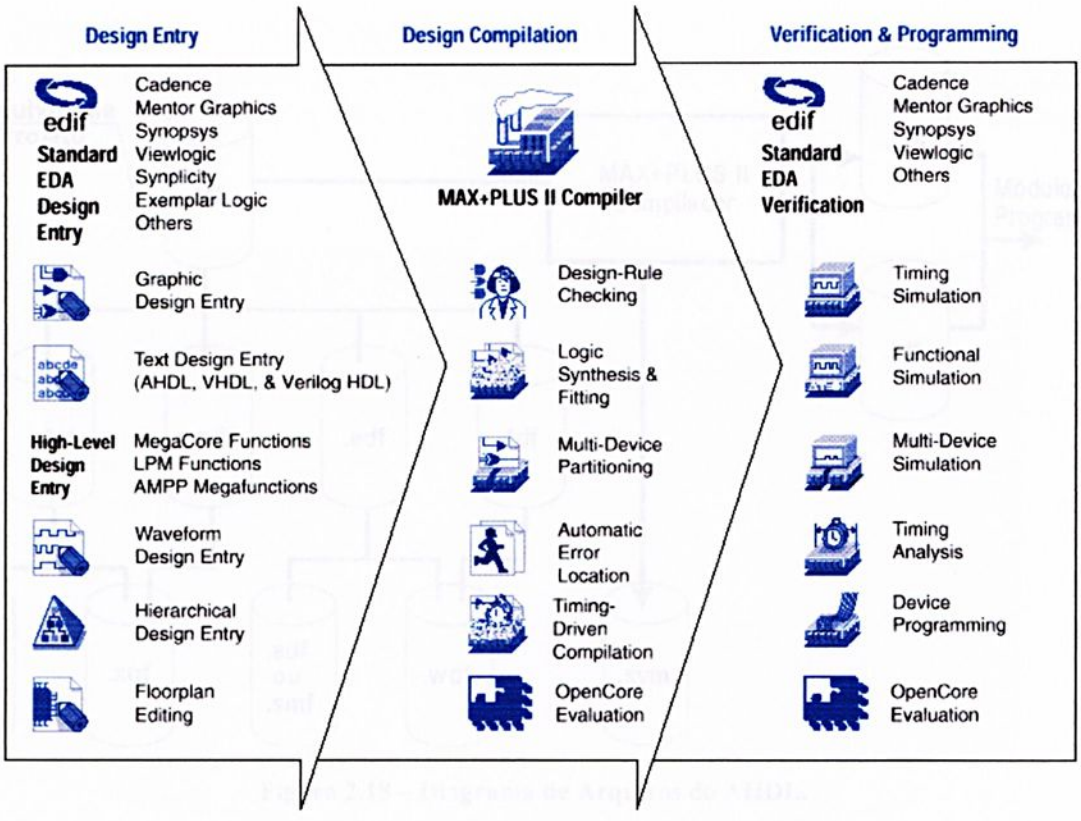


Figura 2.17 - Fluxo de Dados no MAX+PLUS II.

2.4.4. O AHDL

A AHDL (**L**inguagem de **D**escrição de **H**ardware da **A**ltera) é uma linguagem de alto nível e modular, que está completamente integrada dentro do sistema MAX+PLUS II. Ela é especialmente adaptada para o desenvolvimento de lógicas combinacionais complexas, operações de grupo, máquinas de estado, tabelas verdade e lógica parametrizada.

Pode-se usar o Editor de Texto do MAX+PLUS II ou qualquer outro editor para criar Arquivos Textos de Projeto AHDL (**.tdf**). Pode-se compilar os arquivos com extensão tdf's, para se criar arquivos de saída para simulação, análise de tempo, e programação de componentes.

Os elementos e as instruções do AHDL são úteis, versáteis e muito fáceis de usar. Consegue-se criar projetos hierárquicos inteiros com ele, ou mesmo, um conjunto de arquivos tdf's com outros tipos de arquivos de projetos, como em VHDL, por exemplo.

Projetos em AHDL são facilmente incorporados em outros, hierarquicamente. Ao se descrever um circuito digital, pode-se criar um símbolo que o represente, e assim, pode-se utilizá-lo em projetos gráficos (**.gdf**). Com esse princípio, o MAX+PLUS II incorpora em suas bibliotecas muitas funções, megafunções e macrofunções, que são circuitos prontos para serem utilizados. A Biblioteca de Módulos Parametrizáveis (Library of Parameterized Modules – LPM) possui megafunções que podem ter seus parâmetros ajustáveis para as diferentes aplicações do usuário.

A Figura 2.18 mostra como é realizada a integração entre os diferentes arquivos de entradas e saídas do sistema MAX+PLUS II.

unesp

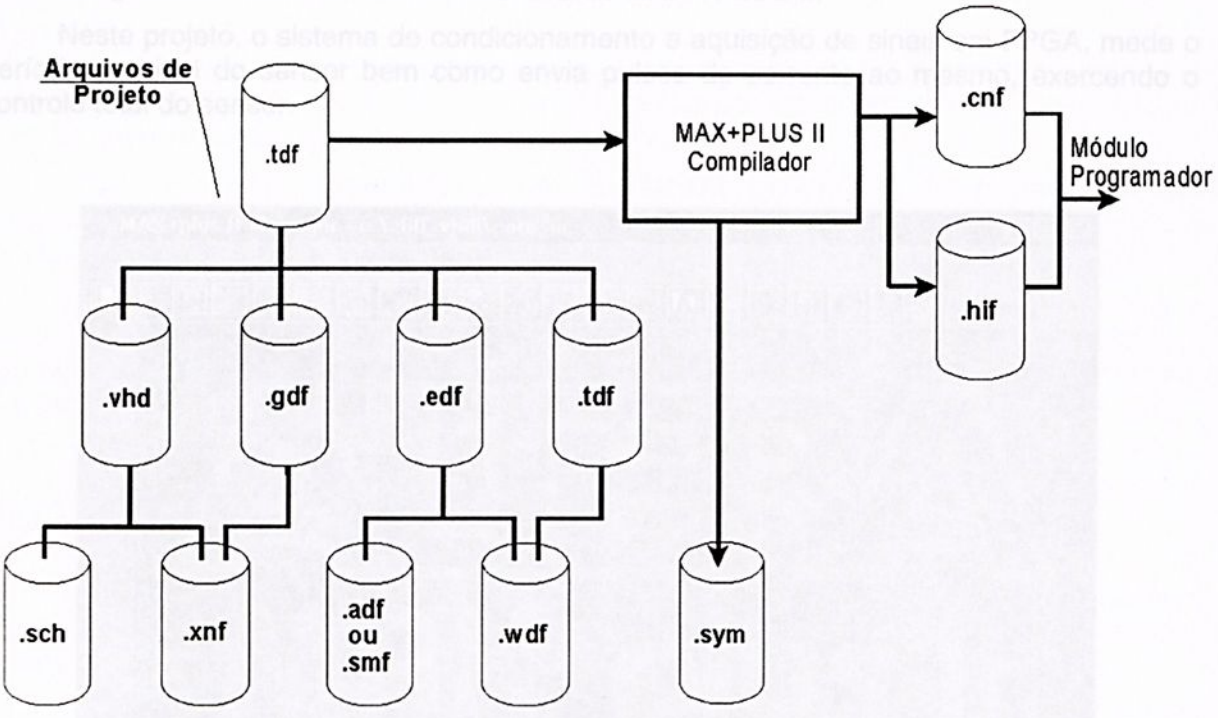


Figura 2.18 – Diagrama de Arquivos do AHDL.

Um projeto hierárquico pode conter arquivos de entrada dos tipos TDF, GDF, EDF (Edif), SCH (arquivos do ORCAD, da MicroSim), VHD (VHDL) entre outros. E as saídas podem ter muitos formatos que poderão ser utilizados por outros Sistemas de Descrição de Hardware, e não somente o da Altera como por exemplo, o formato EDIF. Um projeto desenvolvido no Ambiente Altera poderá ser reaproveitado em outros ambientes de projeto, de outros fabricantes.

A Altera lançou o conceito de Megafunção, onde se cria um componente lógico que pode ser alterado através de parâmetros, para atender às necessidades de uma determinada parte do projeto, simplesmente reconfigurando o mesmo. A Figura 2.19 mostra uma das megafunções parametrizáveis disponíveis na biblioteca da Altera.

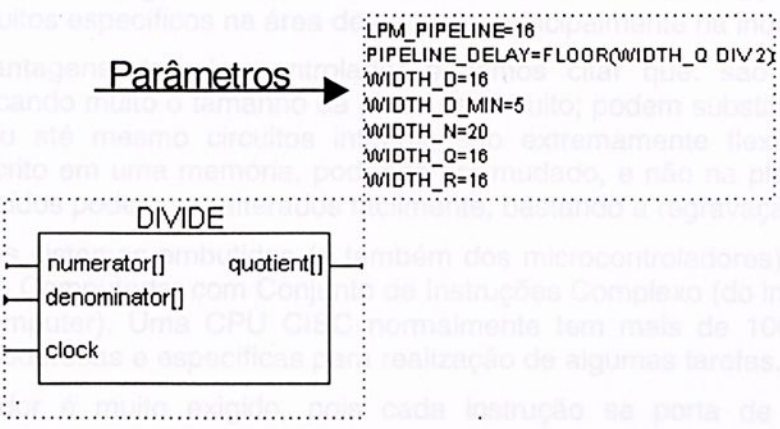


Figura 2.19 – Divisor Parametrizável.

A Figura 2.20 mostra a tela de abertura do MAX +PLUS II.

Neste projeto, o sistema de condicionamento e aquisição de sinais em FPGA, mede o período do sinal do sensor bem como envia pulsos de corrente ao mesmo, exercendo o controle total do sensor.

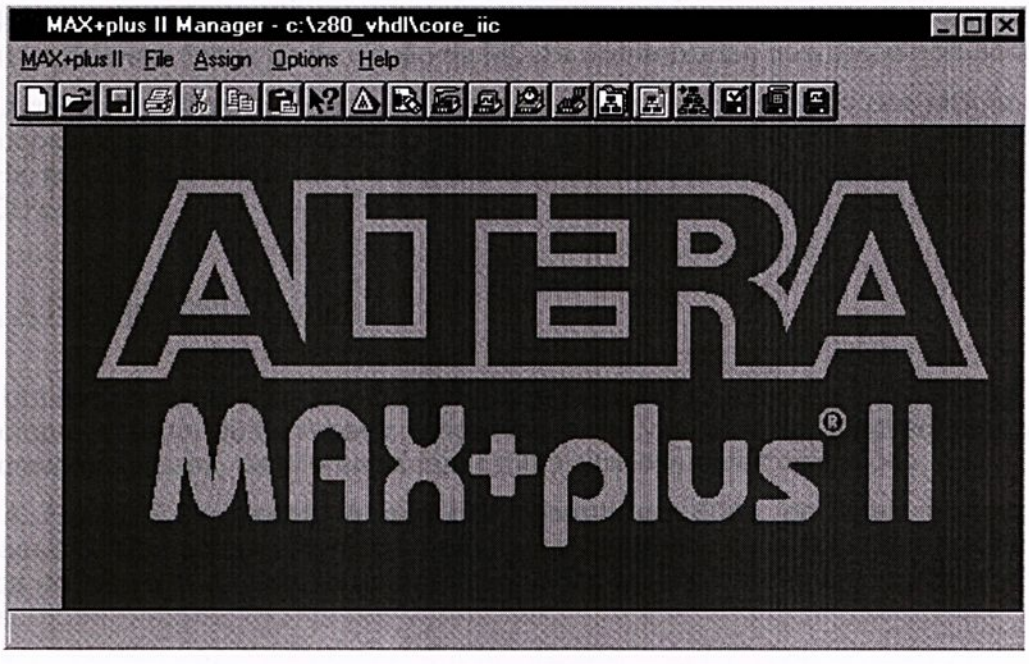


Figura 2.20 – Tela do Software MAX PLUS II.

2.5. Os Microcontroladores

Um microcontrolador (MCU) é essencialmente um computador em um único chip, e semelhante a um computador pessoal, microcontroladores tem memória, unidades de processamento e circuitos de entrada e saída. Porém ao contrário dos computadores pessoais os microcontroladores não tem monitores, teclados, discos rígidos e não podem carregar programas como jogos, utilitários, etc. De fato os microcontroladores são usados com programas muitos específicos na área de controle, principalmente na indústria.

Entre as vantagens do microcontrolador podemos citar que: são extremamente pequenos, simplificando muito o tamanho da placa de circuito; podem substituir com grande eficiência parte ou até mesmo circuitos inteiros; são extremamente flexíveis já que o programa está escrito em uma memória, podendo ser mudado, e não na placa de circuito; os produtos produzidos podem ser alterados facilmente, bastando a regravação[22].

A maioria dos sistemas embutidos (e também dos microcontroladores) está baseada no conceito CISC – Computador com Conjunto de Instruções Complexo (do inglês "Complex Instruction Set Computer). Uma CPU CISC normalmente tem mais de 100 instruções e muitas delas são poderosas e específicas para realização de algumas tarefas.

O programador é muito exigido, pois cada instrução se porta de uma maneira específica. Algumas operam somente em certos espaços de endereços ou registradores e outras podem somente reconhecer um certo tipo de modo de endereçamento. Este tipo de CPU está restrita às CPUs de baixo desempenho, pois uma grande quantidade da área do CI é gasta para implementar a lógica de controle.

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

O conceito RISC, que é o complementar de CISC, está se espalhando pelos sistemas dedicados. O termo RISC significa Computador com Conjunto de Instruções Reduzido (do inglês "Reduced Instructions Set Computer"). Essas máquinas oferecem poucas instruções e, por isso, sua unidade de controle é mais simples permitindo que se logre uma melhor otimização. Os benefícios do RISC, além do melhor desempenho, são um menor CI, uma menor quantidade de pinos e um menor consumo de energia.

Tal como em microprocessadores, os MCU's podem ser fabricados com duas tecnologias: CISC e RISC, sendo esta última praticamente um "sinônimo" de alta velocidade. E existem diversas linhas de produção de MCU's diferenciadas pela tecnologia adotada.

Apesar de serem baseados em microprocessadores, o objetivo dos MCU's não é ter uma capacidade de processamento superior ou igual a eles, por isso, velocidades de 20 MHz, e processamentos de 5 MIPS (milhões de instruções por segundo) são ótimas para os MCU's.

2.5.1. A Família 8051 da Intel

Um dos primeiros microcontroladores que surgiram foi a família 8051 da Intel, baseado no microprocessador 8085. A partir desse dispositivo versátil e útil, surgiram uma variedade de microcontroladores, de diferentes fabricantes para diferentes aplicações.

A Figura 2.21 mostra a estrutura interna do 8051 da Intel. Mesmo estando ultrapassado, ele traz em si o modelo da grande gama dos seus sucessores. Esta figura foi retirada do catálogo da Intel, e foram preservadas as expressões em inglês, que são mais usuais.

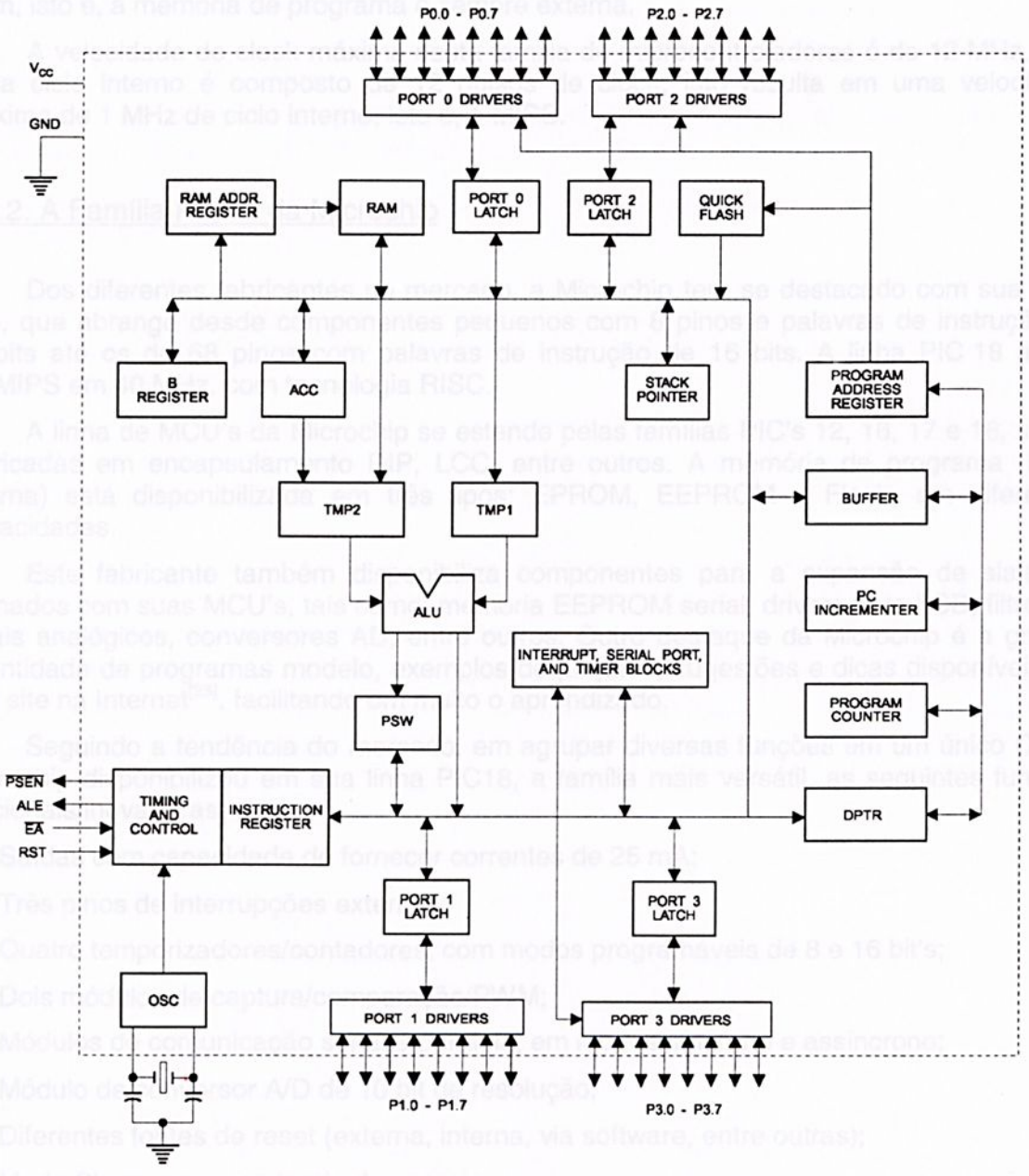


Figura 2.21 – A Estrutura Interna do 8051 da Intel.

Os modelos 80C51 e o 80C31 têm uma capacidade de 128B na RAM interna, Os microcontroladores da família do 80C51 têm a capacidade de acessar uma RAM externa que ao contrário da interna não é tão limitada e pode ter até 64 kB.

O acesso a esta memória é feito por instruções específicas para esta função, porém o seu uso externo resulta em uso demasiado de linhas de I/O. No caso do 80C51, são usadas 18 linhas para que se possa acessar os 64 kB de RAM externa.

Na família do 8051 o acesso à memória de dados é separado do acesso à memória de programa, o que facilita e muito a manipulação e o acesso em oito bits da RAM, tanto interna quanto externa. Porém isto não limita o acesso em dezesseis bits que pode ser feito através do DPTR, um registrador que pode ser usado para manipulação de endereços da memória de dados e leitura de tabelas na memória de programa entre outras funções.

A diferença entre os modelos 8051 e 8031 é que este último não possui memória

Rom, isto é, a memória de programa é sempre externa.

A velocidade de clock máxima desta família de microcontroladores é de 12 MHz, mas cada ciclo interno é composto de 12 pulsos de clock, isto resulta em uma velocidade máxima de 1 MHz de ciclo interno, isto é, 1 MIPS.

2.5.2. A Família PIC18 da Microchip

Dos diferentes fabricantes no mercado, a Microchip tem se destacado com sua linha PIC, que abrange desde componentes pequenos com 8 pinos e palavras de instrução de 12 bits até os de 68 pinos com palavras de instrução de 16 bits. A linha PIC 18 atinge 10 MIPS em 40 MHz, com tecnologia RISC.

A linha de MCU's da Microchip se estende pelas famílias PIC's 12, 16, 17 e 18, sendo fabricadas em encapsulamento DIP, LCC, entre outros. A memória de programa (ROM interna) está disponibilizada em três tipos: EPROM, EEPROM e Flash, em diferentes capacidades.

Este fabricante também disponibiliza componentes para a expansão de sistemas formados com suas MCU's, tais como: memória EEPROM serial, drivers para LCD, filtros de sinais analógicos, conversores AD, entre outros. Outro destaque da Microchip é a grande quantidade de programas modelo, exemplos de projetos, sugestões e dicas disponíveis em seu site na Internet[23], facilitando em muito o aprendizado.

Seguindo a tendência do mercado, em agrupar diversas funções em um único C.I., a Microchip disponibilizou em sua linha PIC18, a família mais versátil, as seguintes funções adicionais/inovadoras:

- Saídas com capacidade de fornecer correntes de 25 mA;
- Três pinos de interrupções externas;
- Quatro temporizadores/contadores, com modos programáveis de 8 e 16 bit's;
- Dois módulos de captura/comparação/PWM;
- Módulos de comunicação serial e paralela, em modos síncrono e assíncrono;
- Módulo de conversor A/D de 10 bit de resolução;
- Diferentes fontes de reset (externa, interna, via software, entre outras);
- Modo Sleep, para economia de energia.

Existem outras funções internas que visam facilitar a operação, bem como ampliar a gama de aplicações do MCU. A Microchip disponibiliza todos os equipamentos e softwares necessários para a programação de seus microcontroladores.

Os microcontroladores Microchip PIC 18 possuem vários modos de operação de clock, o mais comum e também, o utilizado nesta pesquisa, é o clock de fonte externa com cristal de quartzo. Além disso, um circuito "Phase Locked Loop" (PLL) está inserido no microcontrolador para fornecer uma opção programável para o usuário, se desejar, multiplicar a freqüência fornecida pelo cristal externo por 4. Isto significa que um sinal de clock com 10 MHz de freqüência será multiplicado para 40 MHz. Esta opção é muito útil para prevenção de problemas com EMI (Interferência Eletromagnética) devido a alta freqüência dos cristais.

Para a maioria das instruções do PIC18, tem-se quatro estágios de execução (estágios do tipo interpretação da instrução, leitura dos operadores, entre outras), isto significa, que cada instrução leva 4 ciclos de clock para ser executada. Assim pode-se

cm 1 2 3 4 5 6 7 unesp 11 12 13 14 15 16 17 18

definir como freqüência interna a freqüência com que são executadas as instruções, e no caso do PIC 18, ela é igual a ¼ da freqüência do clock. No 8051 a freqüência interna é igual a 1/12 da freqüência de clock

Com a utilização do circuito PLL, a freqüência do cristal será igual a freqüência interna. No caso de um cristal de 10 MHz, tem-se uma instrução executada a cada 0,1 μs.

Algumas instruções, tais como, *CALL*, *GOTO*, são executadas em 02 ciclos, devido a mudança de leitura do programa.

O microcontrolador PIC18C452 possui 32 kBytes de memória de programa do tipo EPROM, isto é, apagável com luz ultra violeta. Como cada instrução ocupa 2 Bytes (16 bit's) de memória, o PIC18C452 é capaz de armazenar 16 kB de instruções (um total de 16.384 palavras de instrução).

O kit PICSTART PLUS é disponibilizado para a programação de todos os componentes da linha PIC. O sistema de programação MPLAB IDE é um completo grupo de softwares para a programação dos MCU's, realizando uma integração completa do PC com o MCU.

Para a linha PIC 17 e PIC 18, a Microchip disponibiliza o MPLAB-C 17 e 18, que integrado ao sistema MPLAB IDE, possibilita a programação em linguagem C, amplamente conhecida e utilizada pelos desenvolvedores de softwares.

A Figura 2.22 mostra um componente PIC 18, com seus pinos e suas respectivas funções, de forma abreviada. A Figura 2.23 mostra a estrutura interna deste componente. Estas duas ilustrações foram retiradas do catálogo da Microchip, e aqui foram preservadas as nomenclaturas em inglês, bem como as terminologias dos pinos e portas entrada e saída.

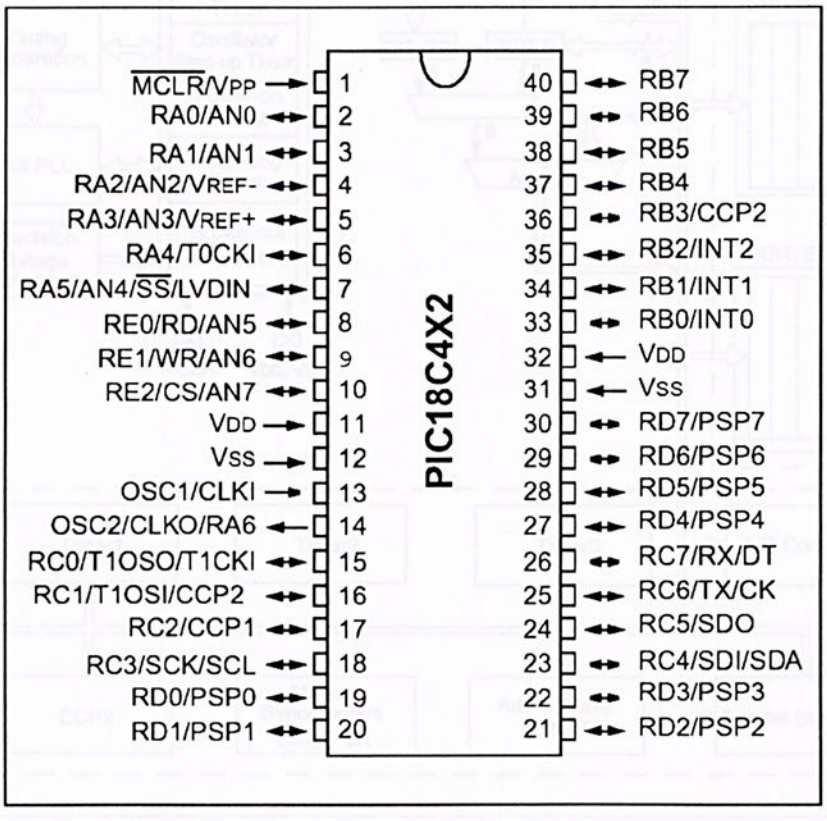


Figura 2.22 – O MCU PIC18C4X2, em Encapsulamento DIP 40.

unesp

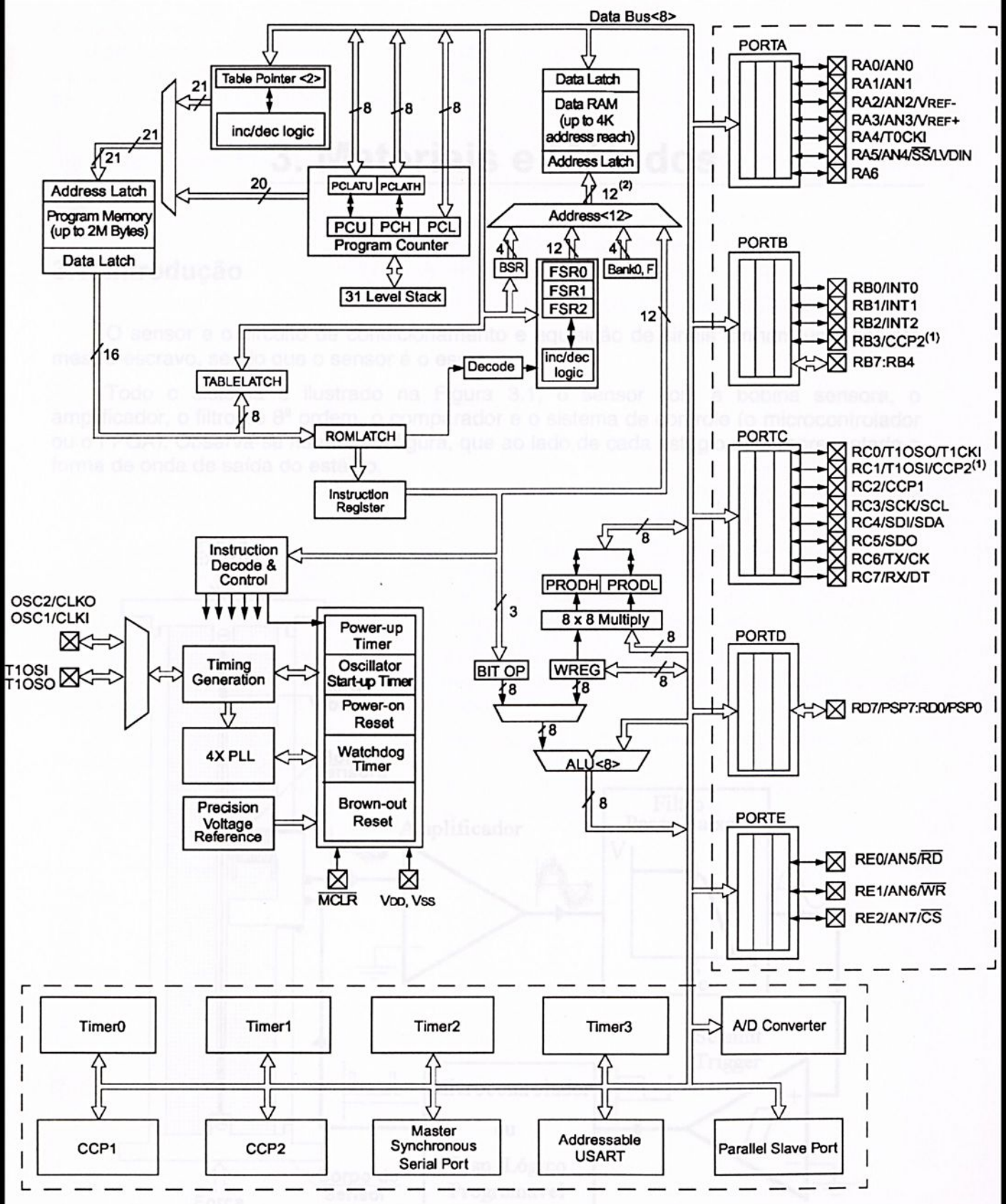


Figura 2.23 – A Estrutura Interna do PIC18C4X2.

unesp

3. Materiais e Métodos

3.1. Introdução

O sensor e o circuito de condicionamento e aquisição de sinais formam um conjunto mestre-escravo, sendo que o sensor é o escravo.

Todo o sistema é ilustrado na Figura 3.1, o sensor com a bobina sensora, o amplificador, o filtro de 8ª ordem, o comparador e o sistema de controle (o microcontrolador ou o FPGA). Observa-se na mesma figura, que ao lado de cada estágio está representada a forma de onda de saída do estágio.

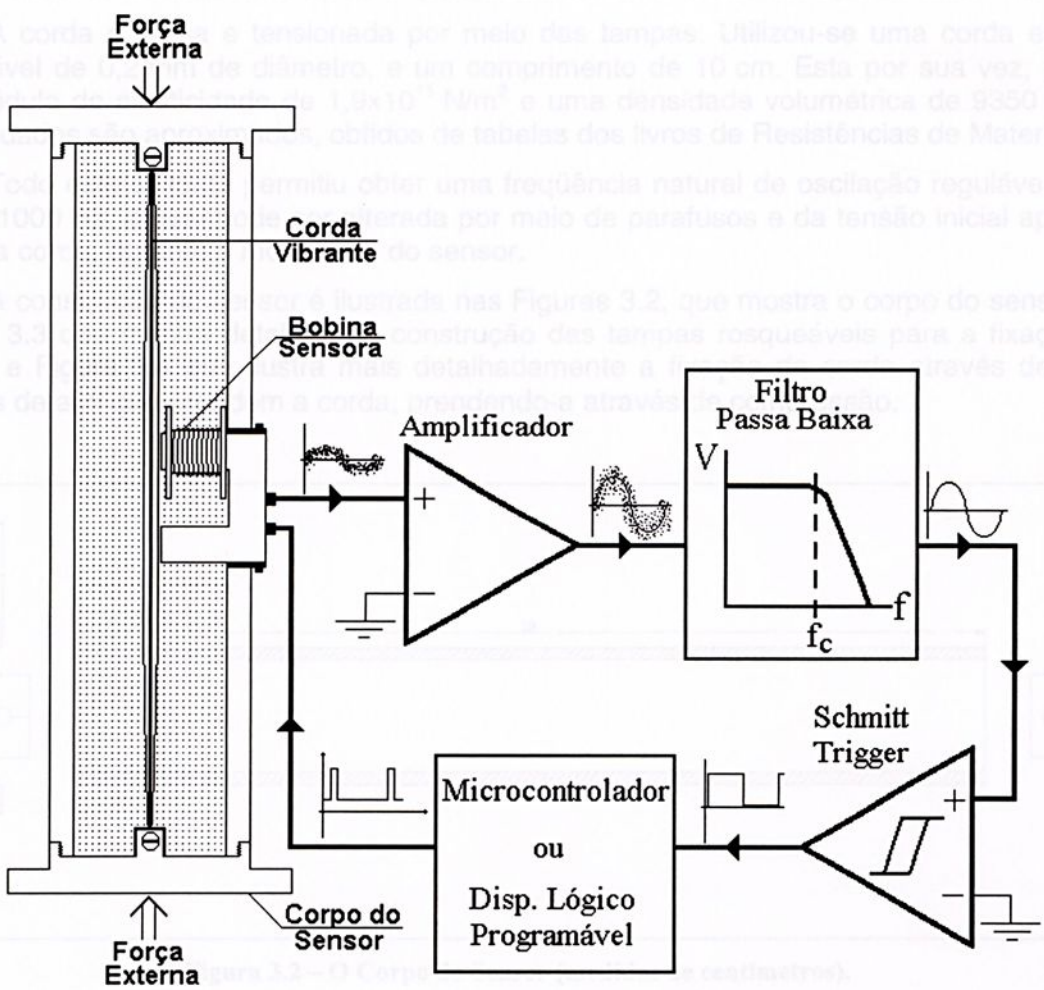


Figura 3.1 - O Sensor a Corda Vibrante e o seu Sistema de Leitura de Sinal.

Para a corda começar a oscilar, necessita de um pulso elétrico a partir do circuito de condicionamento e aquisição de sinais. Este circuito inicia o movimento na corda metálica e, ao mesmo tempo, injeta uma pequena corrente na bobina sensora, que produz, então, um campo magnético estático sobre a mesma. Notar na Figura 3.1 que o sinal que sai do microcontrolador ou Dispositivo Lógico Programável não se anula em nenhum instante.

A corda, em seu movimento oscilatório, com uma pequena porção polarizada, produz um campo magnético variável na bobina sensora, onde é induzida uma tensão. Este pequeno sinal elétrico possui a mesma freqüência de oscilação da corda.

O sinal senoidal é amplificado e filtrado pela parte analógica do circuito de condicionamento e aquisição de sinais. Passando por um Disparador Schmitt, o sinal é transformado em uma onda quadrada, de mesmo período que o sinal senoidal.

Este sinal digital atua como fonte de interrupção para o microcontrolador bem como sinal de habilitação para o dispositivo FPGA.

3.2. O Sensor e seus Componentes

Para instrumentar a muleta e o dinamômetro, construiu-se um sensor em dura alumínio, de formato cilíndrico, com paredes de espessura 1,4 mm, com duas tampas rosqueáveis. O dura alumínio utilizado possui um módulo de elasticidade de $7x10^{10}$ N/m^2 [24].

A corda é presa e tensionada por meio das tampas. Utilizou-se uma corda em aço inoxidável de 0,2 mm de diâmetro, e um comprimento de 10 cm. Esta por sua vez, possui um módulo de elasticidade de $1,9x10^{11}$ N/m^2 e uma densidade volumétrica de 9350 kg/m^3. Esses dados são aproximados, obtidos de tabelas dos livros de Resistências de Materiais[12].

Todo este sistema permitiu obter uma freqüência natural de oscilação regulável entre 500 e 1000 Hz, a qual pode ser alterada por meio de parafusos e da tensão inicial aplicada sobre a corda, durante a montagem do sensor.

A construção do sensor é ilustrada nas Figuras 3.2, que mostra o corpo do sensor; na Figura 3.3 que mostra detalhes da construção das tampas rosqueáveis para a fixação da corda; e Figura 3.4 que ilustra mais detalhadamente a fixação da corda através de duas chapas de aço que prendem a corda, prendendo-a através de compressão.

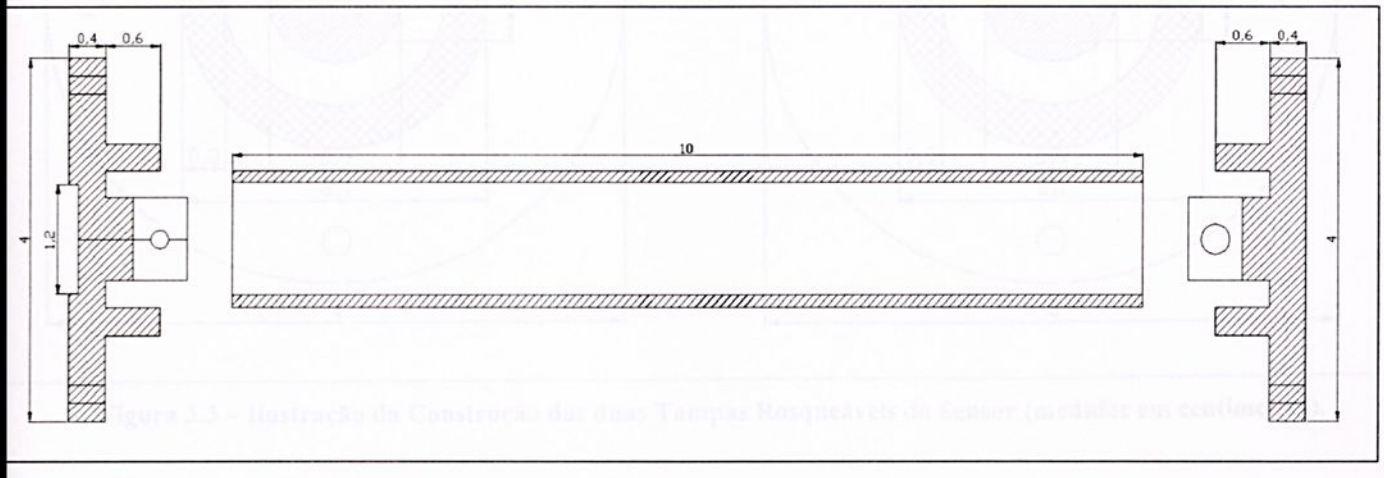


Figura 3.2 – O Corpo do Sensor (medidas de centímetros).

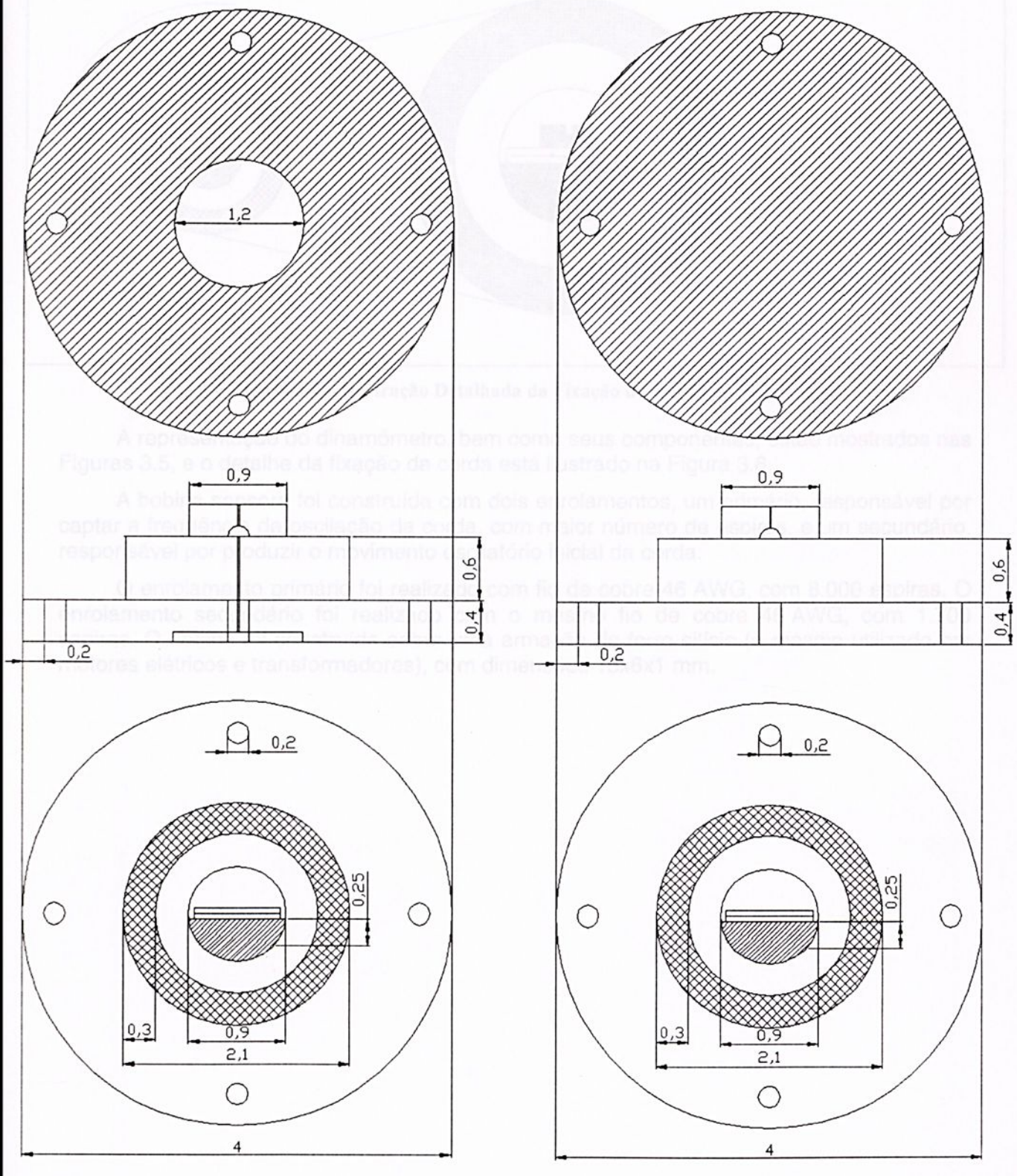


Figura 3.3 – Ilustração da Construção das duas Tampas Rosqueáveis do Sensor (medidas em centímetros).

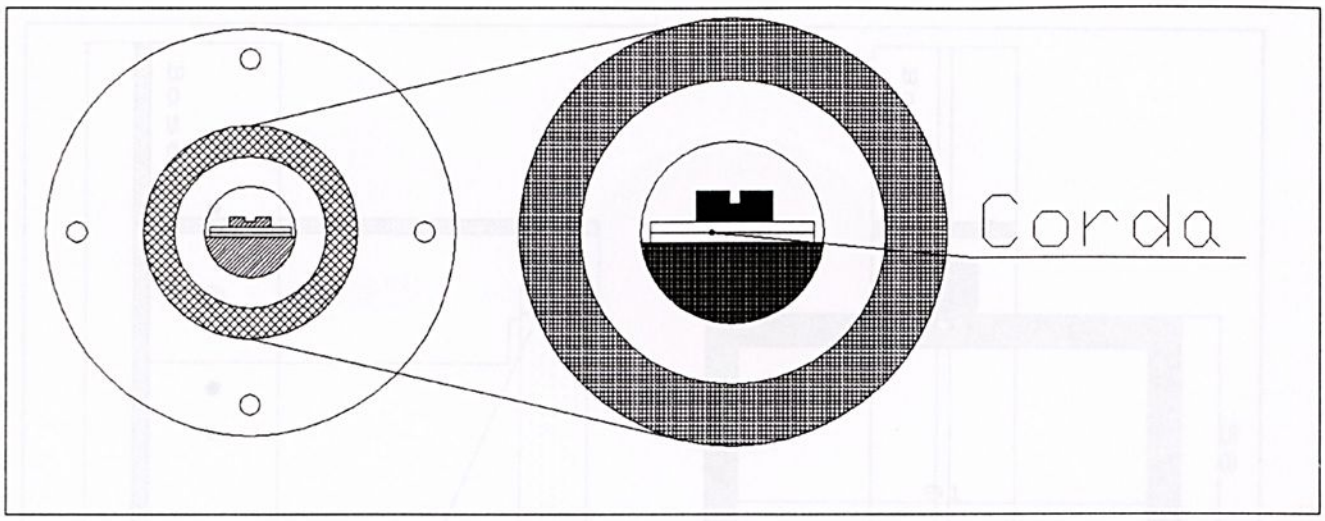


Figura 3.4 – Ilustração Detalhada da Fixação da Corda no Sensor.

A representação do dinamômetro, bem como seus componentes, estão mostrados nas Figuras 3.5, e o detalhe da fixação da corda está ilustrado na Figura 3.6.

A bobina sensora foi construída com dois enrolamentos, um primário, responsável por captar a freqüência de oscilação da corda, com maior número de espiras, e um secundário, responsável por produzir o movimento oscilatório inicial da corda.

O enrolamento primário foi realizado com fio de cobre 46 AWG, com 8.000 espiras. O enrolamento secundário foi realizado com o mesmo fio de cobre 46 AWG, com 1.700 espiras. O bobina foi construída sobre uma armação de ferro silício (o mesmo utilizado em motores elétricos e transformadores), com dimensões 15x6x1 mm.

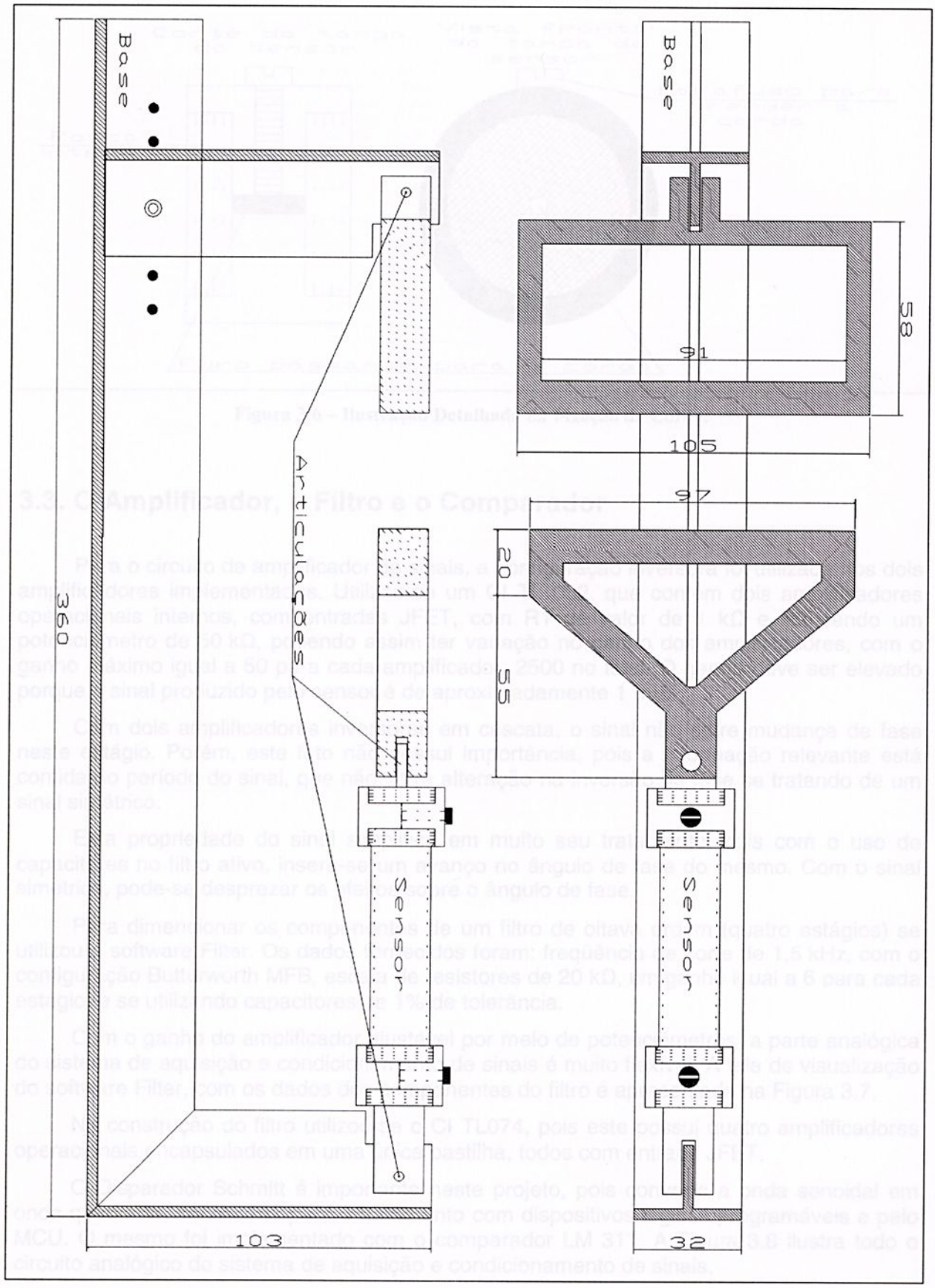


Figura 3.5 – As duas Vistas do Dinamômetro e seus Componentes (medidas em milímetros).

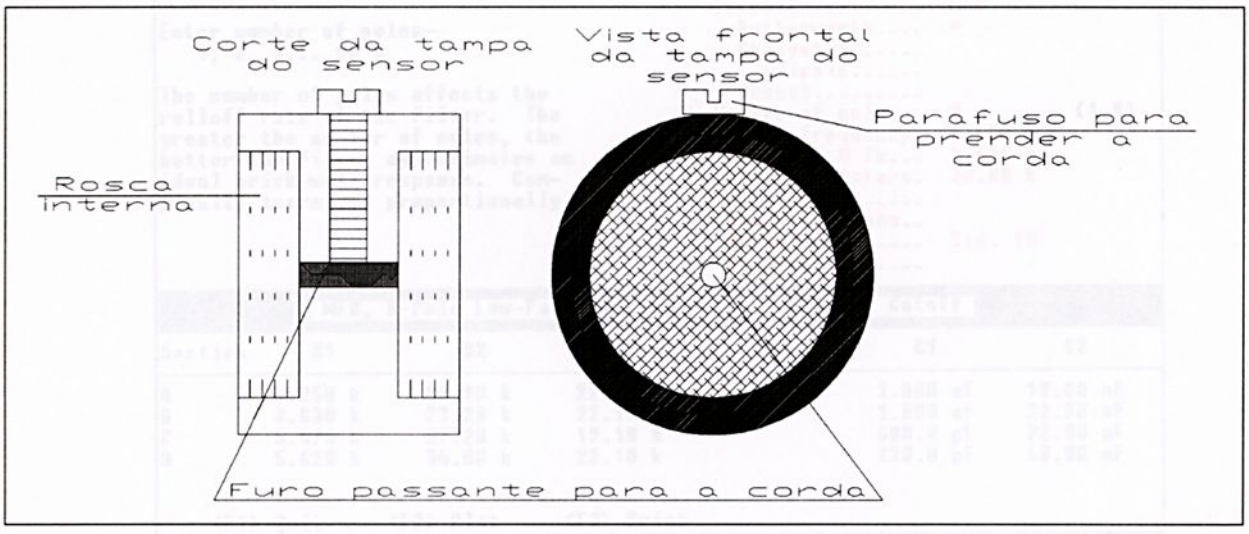


Figura 3.6 – Ilustração Detalhada da Fixação da Corda.

3.3. O Amplificador, o Filtro e o Comparador

Para o circuito de amplificador de sinais, a configuração inversora foi utilizada nos dois amplificadores implementados. Utilizou-se um CI TL 082, que contém dois amplificadores operacionais internos, com entradas JFET, com R1 de valor de 1 kΩ e R2 sendo um potenciômetro de 50 kΩ, podendo assim ter variação no ganho dos amplificadores, com o ganho máximo igual a 50 para cada amplificador, 2500 no total. O ganho deve ser elevado porque o sinal produzido pelo sensor é de aproximadamente 1 mV.

Com dois amplificadores inversores em cascata, o sinal não sofre mudança de fase neste estágio. Porém, este fato não possui importância, pois a informação relevante está contida no período do sinal, que não sofre alteração na inversão de fase se tratando de um sinal simétrico.

Esta propriedade do sinal simplifica em muito seu tratamento, pois com o uso de capacitores no filtro ativo, insere-se um avanço no ângulo de fase do mesmo. Com o sinal simétrico, pode-se desprezar os efeitos sobre o ângulo de fase.

Para dimensionar os componentes de um filtro de oitava ordem (quatro estágios) se utilizou o software Filter. Os dados fornecidos foram: freqüência de corte de 1,5 kHz, com o configuração Butterworth MFB, escala de resistores de 20 kΩ, um ganho igual a 6 para cada estágio, e se utilizando capacitores de 1% de tolerância.

Com o ganho do amplificador ajustável por meio de potenciômetros, a parte analógica do sistema de aquisição e condicionamento de sinais é muito flexível. A tela de visualização do software Filter, com os dados dos componentes do filtro é apresentada na Figura 3.7.

Na construção do filtro utilizou-se o CI TL074, pois este possui quatro amplificadores operacionais encapsulados em uma única pastilha, todos com entrada JFET.

O Disparador Schmitt é importante neste projeto, pois converte a onda senoidal em onda quadrada, necessária para o tratamento com dispositivos lógicos programáveis e pelo MCU. O mesmo foi implementado com o comparador LM 311. A Figura 3.8 ilustra todo o circuito analógico do sistema de aquisição e condicionamento de sinais.

unesp

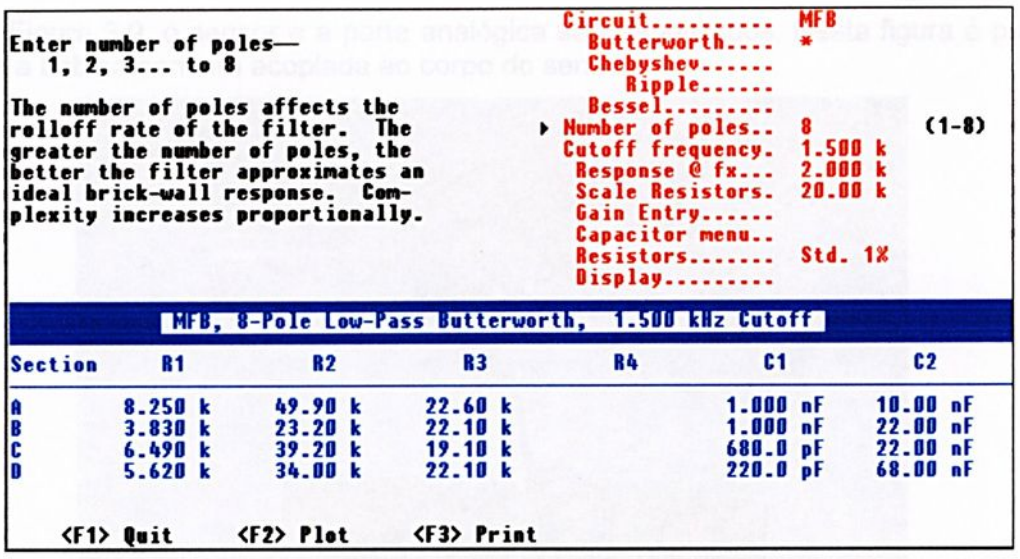


Figura 3.7 – Tela do Software Filter, com os Valores dos Componentes do Filtro.

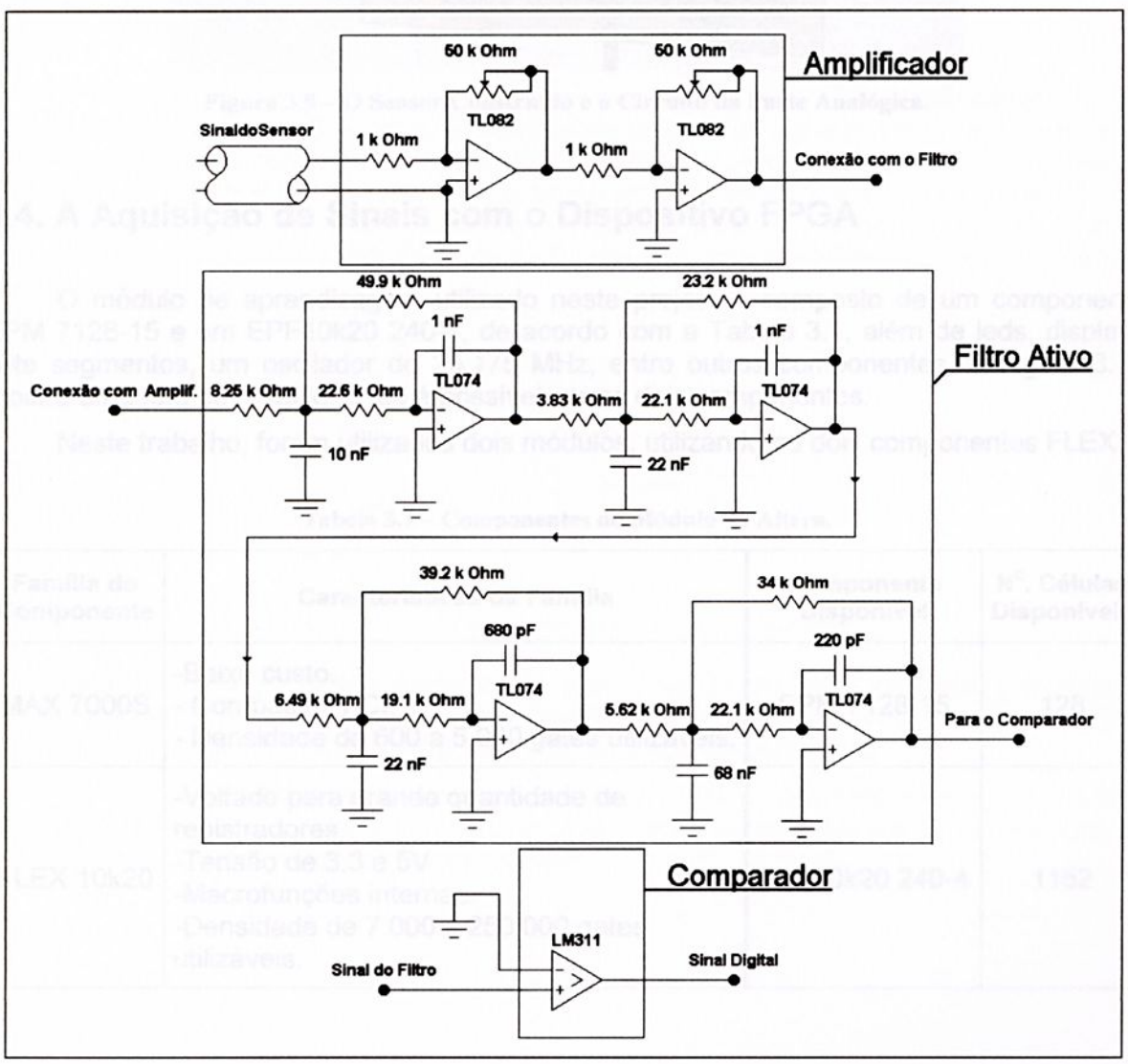


Figura 3.8 – Esquemático do Circuito Analógico do Sistema de Aquisição e Condicionamento de Sinais.

Na Figura 3.9, o sensor e a parte analógica são visualizados. Nesta figura é possível visualizar a bobina sensora acoplada ao corpo do sensor.

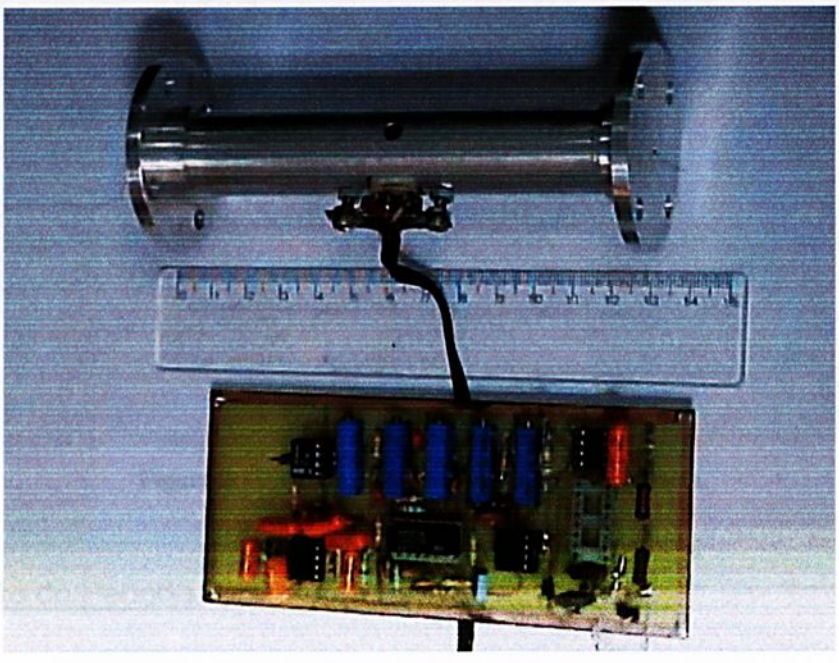


Figura 3.9 – O Sensor Construído e o Circuito da Parte Analógica.

3.4. A Aquisição de Sinais com o Dispositivo FPGA

O módulo de aprendizagem utilizado neste projeto é composto de um componente EPM 7128-15 e um EPF10k20 240-4, de acordo com a Tabela 3.1, além de leds, displays sete segmentos, um oscilador de 25,175 MHz, entre outros componentes. A Figura 3.10 mostra uma foto do módulo, onde é possível ver os dois componentes.

Neste trabalho, foram utilizados dois módulos, utilizando-se dois componentes FLEX.

Tabela 3.1 – Componentes do Módulo da Altera.

Família do Componente	Características da Família	Componente Disponível	Nº. Células Disponíveis
MAX 7000S	-Baixo custo. - Compatível PCI. - Densidade de 600 a 5.000 gates utilizáveis.	EPM 7128-15	128
FLEX 10k20	-Voltado para grande quantidade de registradores. -Tensão de 3.3 e 5V. -Macrofunções internas. -Densidade de 7.000 a 250.000 gates utilizáveis.	EPF10k20 240-4	1152

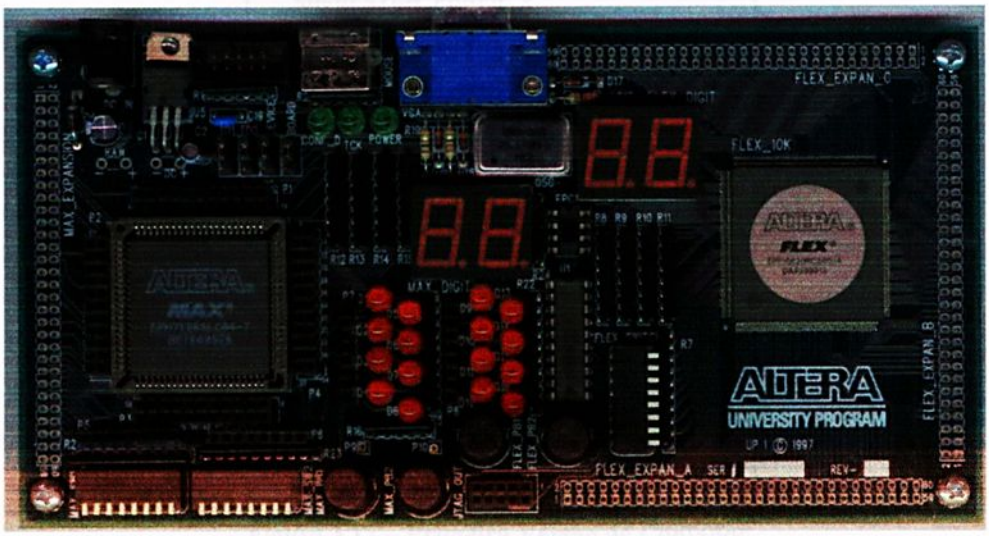


Figura 3.10 - Módulo de Aprendizagem da Altera.

3.4.1. Medição do Período

A medição de período de um sinal desconhecido é feita com a associação de um contador e uma porta AND. Uma das entradas da porta AND é a entrada para os pulsos de freqüência desconhecida, e a outra é a entrada para os pulsos conhecidos. Para sinais de desconhecidos de freqüência baixa, utiliza-se um sinal de freqüência maior e conhecida, que forma a base de tempo. Assim, quando o nível do sinal desconhecido for "1", o sinal conhecido gera pulsos na saída da porta AND, que são enviados para o contador, conforme ilustra a Figura 3.11.

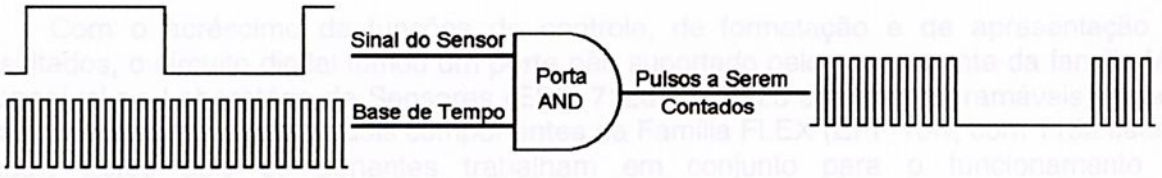


Figura 3.11 – Geração dos Pulsos a Serem Contados.

Este número de pulsos multiplicado pelo período do sinal conhecido fornece o tamanho de meio período do sinal desconhecido. Multiplicando-se por 2, tem-se o período do sinal, consequentemente a freqüência do sinal, antes desconhecida.

A Figura 3.12 apresenta um modelo descrito em AHDL, que implementa um contador de pulsos. Este foi o primeiro contador implementado nesta pesquisa, e foi gravado no componente MAX do módulo de aprendizagem.

```
  SUBDESIGN cont8_2
(
  enable,clk   : INPUT;
  TIME[7..0]              : OUTPUT;
      VARIABLE
      count[7..0]         : DFF;
      clear               : node;
  BEGIN
      clear=count7&count6&count3;
      count[].clk =clk ;
      time[]      =count[];
      count[].clrn =!clear;

      IF enable THEN
      count[]  = count[]+1;
      else
      count[]= count[];
      end IF ;
  end;
```

Figura 3.12 – Primeira Versão do Contador.

Nesta primeira versão do contador (Figura 3.12) tem-se a declaração das entradas e saídas do contador, bem como o número de bits que as compõem, seguidas de INPUT e OUTPUT, respectivamente. Nota-se a facilidade de se aumentar o número de bits simplesmente alterando o valor declarado.

Ao sinal proveniente do sensor atribuiu-se o nome de ENABLE e, ao sinal do oscilador (freqüência conhecida) atribuiu-se o nome CLK.

Em seguida, tem-se o início da descrição do comportamento do modelo (BEGIN) o equivalente a ligarmos cada componente às suas respectivas entradas. A lógica de contagem está descrita no final com o comando IF...THEN. Neste modelo a variável de saída é de 8 bits (TIME[7..0]).

No modelo, só há contagem enquanto (IF...THEN) o sensor (ENABLE) estiver enviando sinais de nível alto. Para a contagem utilizou-se FLIP-FLOP's D, denominados de DFF no modelo.

Com o acréscimo de funções de controle, de formatação e de apresentação de resultados, o circuito digital tomou um porte não suportado pelo componente da família Max disponível no Laboratório de Sensores (EPM 7128, com 128 células programáveis), sendo, então, necessário o uso de dois componentes da Família FLEX (EPF 10K, com 1152 células cada). Estes dois componentes trabalham em conjunto para o funcionamento do frequencímetro e apresentação de resultados.

Nesta etapa, concluiu-se que seria mais adequado separar o modelo já implementado em vários sub-modelos/blocos, facilitando a compreensão e diagramação, além de otimizar a implementação. Foi necessário utilizar-se uma versão mais atual do ambiente de projeto da Altera (versão 10.0), já que a versão estudante (7.21) utilizada até então, não possuía algumas funções requeridas pelo frequencímetro implementado.

Um expressivo crescimento do circuito em FPGA ocorreu também por causa do acréscimo de um bloco que realiza uma média entre a nova leitura e leituras anteriores. Com a imprecisão e variação do sinal, decorrente de ruídos, ocorrem diferentes contagem de pulsos, dando imprecisão ao resultado; com o cálculo de uma média entre as leituras, consegue-se maior precisão.

Este processo se assemelha a constante de tempo na resposta de medidores analógicos (inércia do ponteiro de um galvanômetro é considerada como uma constante de tempo inserida ao sistema, por exemplo). Esta imprecisão na contagem de pulsos é ilustrada na Figura 3.13.

unesp

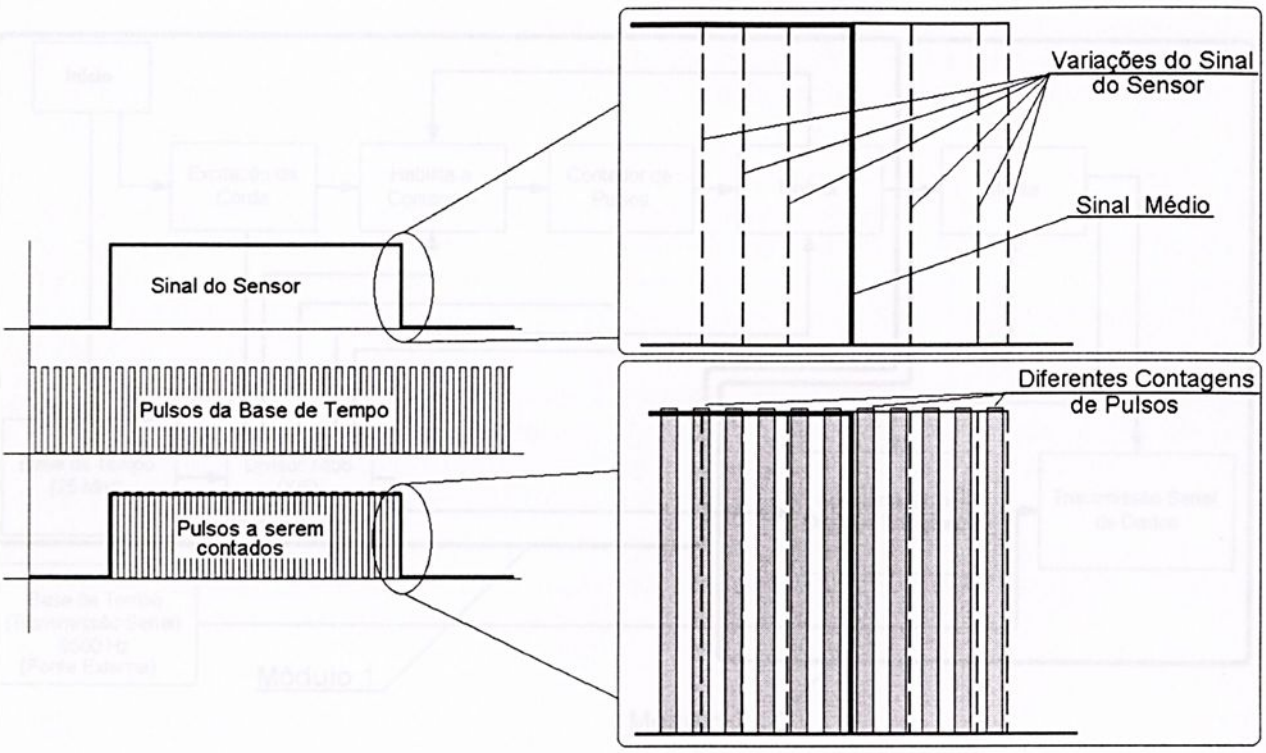


Figura 3.13 – Ilustração da Imprecisão na Contagem de Pulsos.

O bloco MédiaTotal implementado no Circuito FPGA realiza a média com 16 leituras, e considerando um sinal do sensor de período de 1 ms (1000 Hz), uma mudança na força externa tem seu real valor apresentado em 16 ms teoricamente. Este valor pode ser maior, devido a atrasos, se ocorrer uma excitação da corda há a suspensão da leitura do sinal por 2 ciclos.

Do circuito digital implementado nesta pesquisa, um fluxograma ilustrativo é apresentado na Figura 3.14, e na Figura 3.15, estão mostradas as interligações externas dos dois módulos.

Na condição do laboratório dispor de um componente de maior capacidade, estes dois circuitos, gravados em dois componentes, podem ser agrupados, formando um sistema de aquisição completo em um único CI.

Cada um dos símbolos, mostrados na Figura 3.15, é composto de vários blocos internos. A maioria dos blocos foi confeccionada na linguagem de descrição AHDL, para cada um desses blocos foi gerado um símbolo com entradas e saídas, e através das mesmas foram interligados no modo esquemático. Cada um destes, que formam a descrição de hardware do frequencímentro utilizado no sensor, tem sua função devidamente explanada no Apêndice 1.

Devido a limitação de hardware dos atuais dispositivos FPGA's do laboratório, não foi possível realizar operações matemáticas, de acordo com as expressões (2.14) e (2.17), assim, a resposta apresentada nos displays, é o valor da freqüência do sinal do sensor em hexadecimal, com uma casa decimal de precisão.

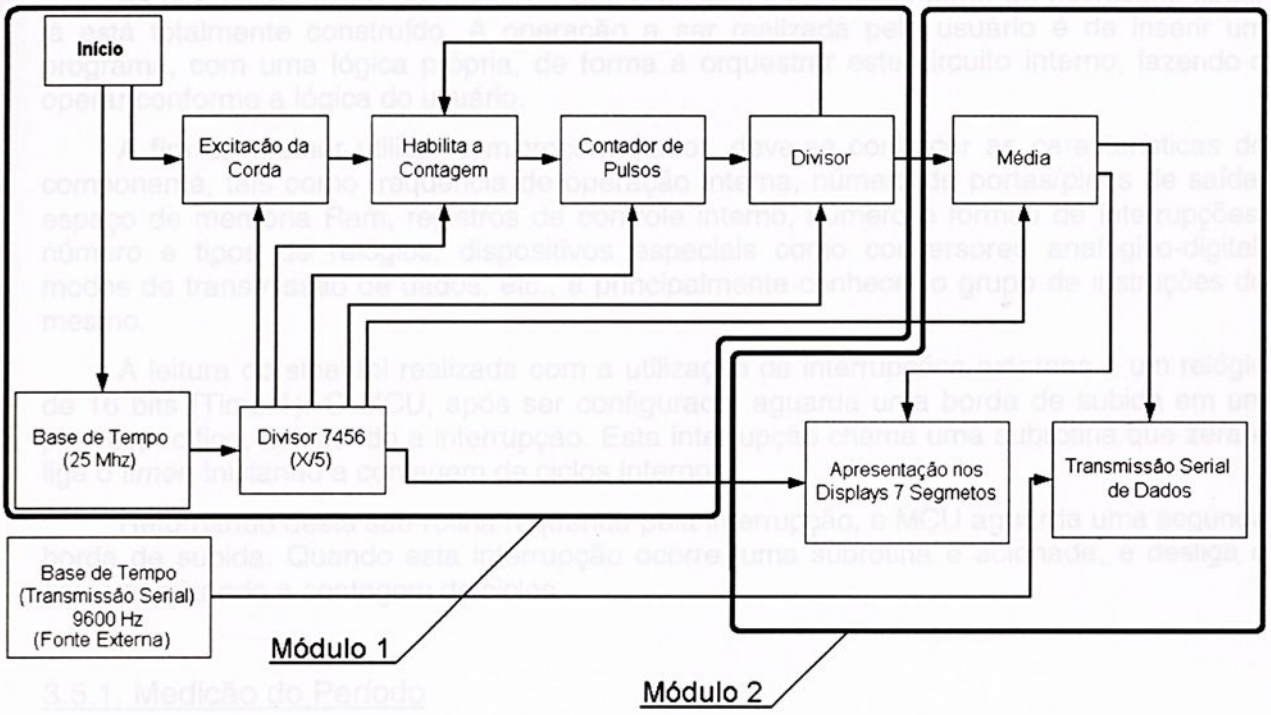


Figura 3.14 – Fluxograma do Sistema de Condicionamento Aquisição e de Sinais Implementado em FPGA.

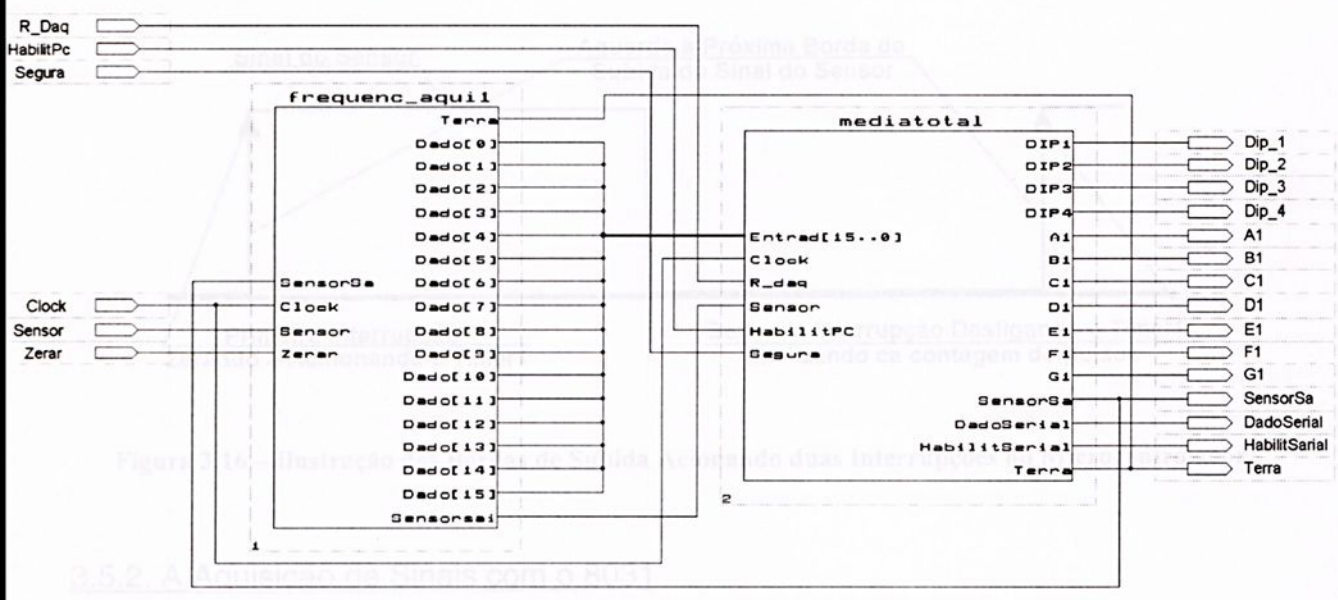


Figura 3.15 – Frequencímetro Implementado em FPGA.

3.5. A Aquisição de Sinais com o Microcontrolador

A aquisição de dados com o microcontrolador foi realizada de forma similar à implementada com os dispositivos FPGA.

As diferenças estão por conta de que o circuito eletrônico interno ao microcontrolador já está totalmente construído. A operação a ser realizada pelo usuário é de inserir um programa, com uma lógica própria, de forma a orquestrar este circuito interno, fazendo-o operar conforme a lógica do usuário.

A fim de melhor utilizar o microcontrolador, deve-se conhecer as características do componente, tais como freqüência de operação interna, número de portas/pinos de saída, espaço de memória Ram, registros de controle interno, número e formas de interrupções, número e tipos de relógios, dispositivos especiais como conversores analógico-digital, modos de transmissão de dados, etc., e principalmente conhecer o grupo de instruções do mesmo.

A leitura do sinal foi realizada com a utilização de interrupções externas e um relógio de 16 bits (Timer1). O MCU, após ser configurado, aguarda uma borda de subida em um pino específico, acionando a interrupção. Esta interrupção chama uma subrotina que zera e liga o *timer*, iniciando a contagem de ciclos internos.

Retornando desta sub-rotina requerida pela interrupção, o MCU aguarda uma segunda borda de subida. Quando esta interrupção ocorre, uma subrotina é acionada, e desliga o *timer*, finalizando a contagem de ciclos.

3.5.1. Medição do Período

Fazendo uma analogia com a lógica empregada no dispositivo FPGA, a operação AND entre o sinal do sensor e os pulsos da base de tempo, conforme a Figura 3.11, é realizada através das duas interrupções. Os pulsos da base de tempo são os ciclos internos contados pelo *timer*. Esta lógica está ilustrada na Figura 3.16.

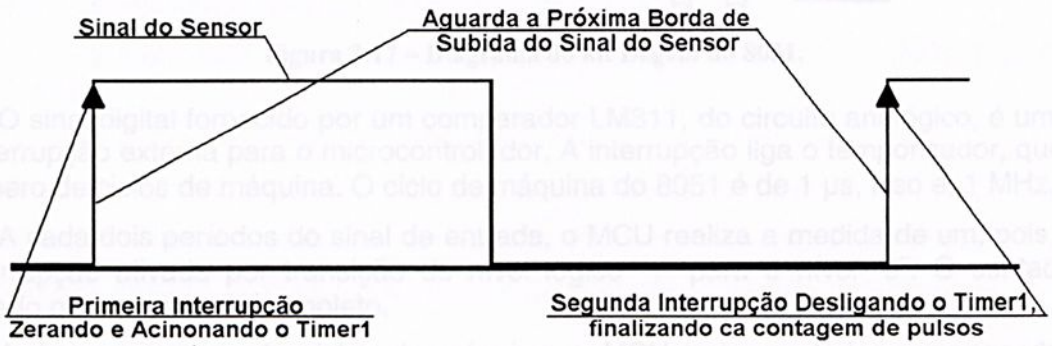


Figura 3.16 – Ilustração das Bordas de Subida Acionando duas Interrupções no Microcontrolador.

3.5.2. A Aquisição de Sinais com o 8031

O primeiro microcontrolador utilizado neste projeto foi um 8031, da mesma família do 8051, mas sem ROM interna. Este MCU está montado em forma de kit de aprendizagem, fornecido pela Degen, sob o nome: Curso EB-153, Microcontrolador 8051.

Este kit é limitado quanto aos recursos disponibilizados ao usuário, pois, de todas as portas do 8031, somente duas são disponibilizadas, assim como é muito reduzido o tamanho da memória RAM disponível ao usuário. O kit possui um programa monitor próprio que gerencia e utiliza os recursos do MCU. O diagrama do kit é mostrado na Figura 3.17.

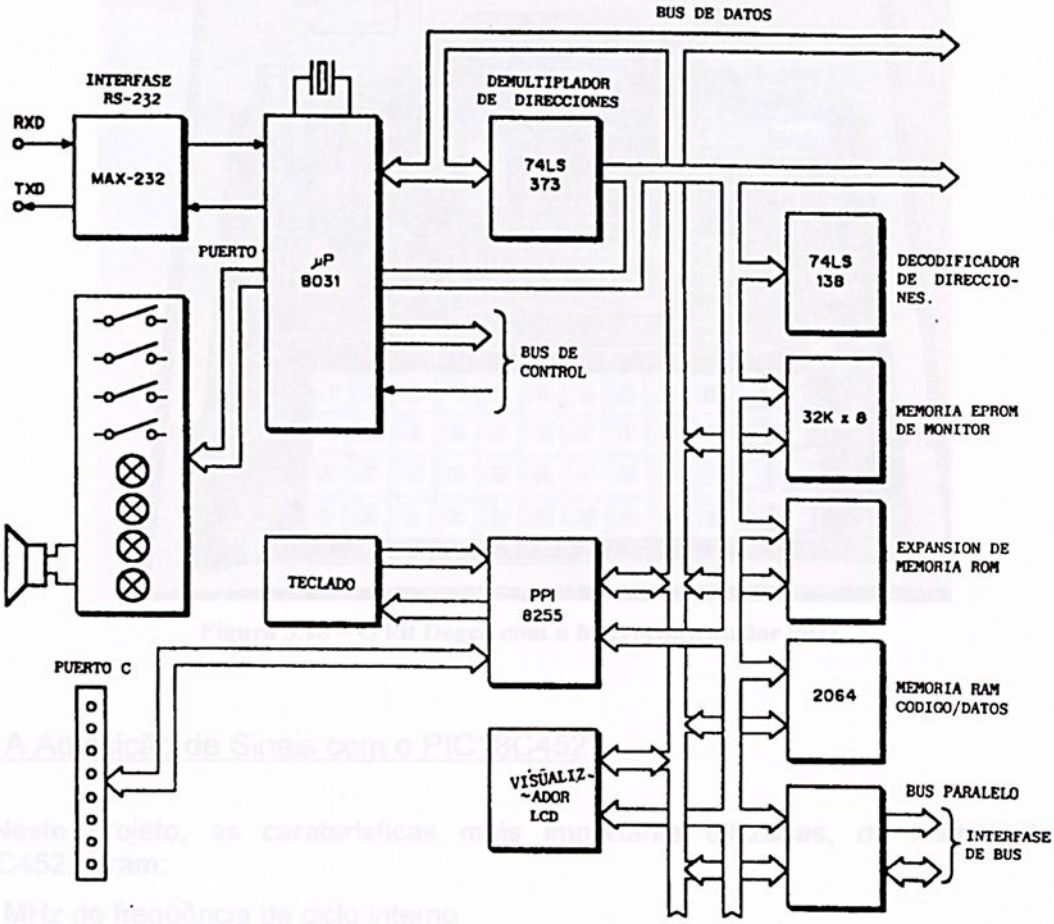


Figura 3.17 – Diagrama do kit Degem do 8031.

O sinal digital fornecido por um comparador LM311, do circuito analógico, é uma fonte de interrupção externa para o microcontrolador. A interrupção liga o temporizador, que conta o número de ciclos de máquina. O ciclo de máquina do 8051 é de 1 µs, isso é, 1 MHz.

A cada dois períodos do sinal de entrada, o MCU realiza a medida de um, pois, sendo a interrupção ativada por transição do nível lógico "1" para o nível "0". O contador fica acionado por um período completo.

Após a contagem de ciclos de máquina, o MCU trata os dados armazenados pelo contador e os apresenta em um display, em formato hexadecinal. A Figura 3.18 apresenta o kit Degen utilizado.

Os resultados com o microcontrolador foram obtidos com o auxílio de pesos conhecidos.

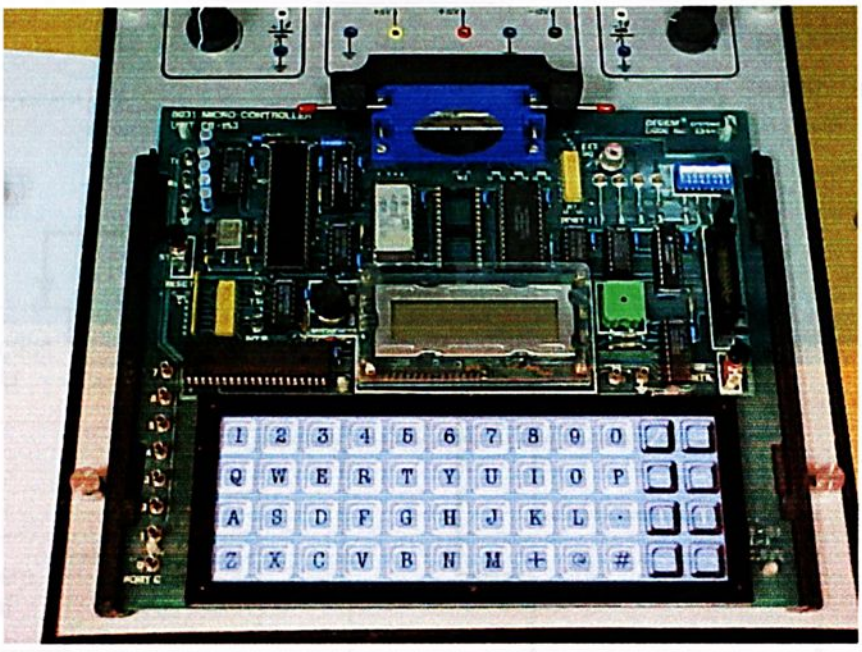


Figura 3.18 – O kit Degen com o Microcontrolador 8031.

3.5.3. A Aquisição de Sinais com o PIC18C452

Neste projeto, as caraterísticas mais importante utilizadas, do microcontrolador PIC18C452, foram:

- 10 MHz de freqüência de ciclo interno
- 5 portas de entrada/saída, totalizando 33 pinos (6 pinos da porta A, 8 da B, 8 da C, 8 da D, 3 da E).
- 1536 bytes de memória Ram, distribuídos em 6 bancos de 256 bytes cada.
- mais de 40 registros internos, utilizados em diferentes operações do MCU.
- mais de 10 fontes diferentes de interrupções, sendo 8 acessadas externamente.
- 4 relógios (timers) com funcionamento independente, podendo trabalhar com 8 ou 16 bits.

Toda as operações realizadas pelo microcontrolador estão ilustradas no fluxograma da Figura 3.19.

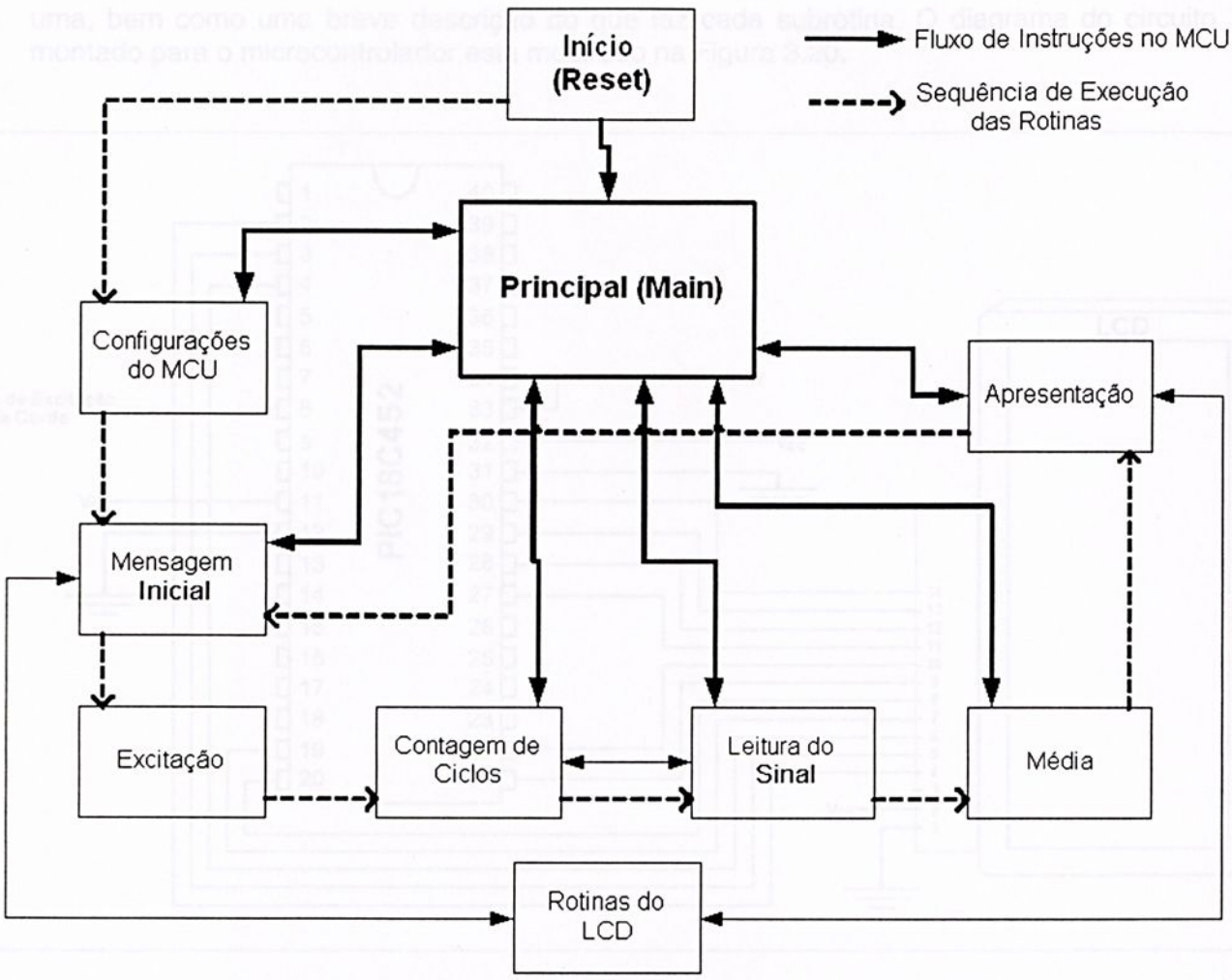


Figura 3.19 – Fluxograma do Programa Implementado no Microcontrolador PIC 18.

O programa possui várias subrotinas, que são controladas por uma rotina principal (Main). Ao final da execução de cada subrotina, o microcontrolador retorna a rotina principal, que decide qual o próximo passo a ser dado (setas cheias da Figura 3.19).

Mas a lógica que o programa segue é uma seqüência de sub-rotinas para ler, formatar e apresentar a contagem de ciclos internos do MCU (setas tracejadas da Figura 3.19). Esta seqüência é praticamente idêntica a utilizada no dispositivo FPGA.

Um diferencial entre o sistema implementado no MCU e o implementado com FPGA é que o MCU faz a contagem de um período inteiro, enquanto o FPGA faz a contagem de pulsos em meio período (parte alta do período). Por isso, o FPGA é capaz de realizar uma leitura a cada ciclo do sinal do sensor, enquanto que o MCU seria capaz de realizar uma leitura a cada dois ciclos.

Com a presença das Rotinas do LCD, que é muito mais lento que o MCU, não é possível implementar uma leitura a cada dois ciclos, mas sim, a cada 20 ms, isto é, a cada 20 ciclos do sinal do sensor, considerando-se 1000 Hz a freqüência do sinal do sensor

Devido a mesma imprecisão apresentada na Figura 3.13, se faz necessário o calculo de uma média das leituras. Esta média de leituras, com a disponibilidade de um maior espaço de memória RAM, é realizada com 64 leituras. Assim, uma nova força aplicada sobre o sensor será apresentada no LCD 1,28 s após sua ocorrência.

Todas as linhas de programa estão no Apêndice 2, com a descrição do que faz cada

uma, bem como uma breve descrição do que faz cada subrotina. O diagrama do circuito montado para o microcontrolador está mostrado na Figura 3.20.

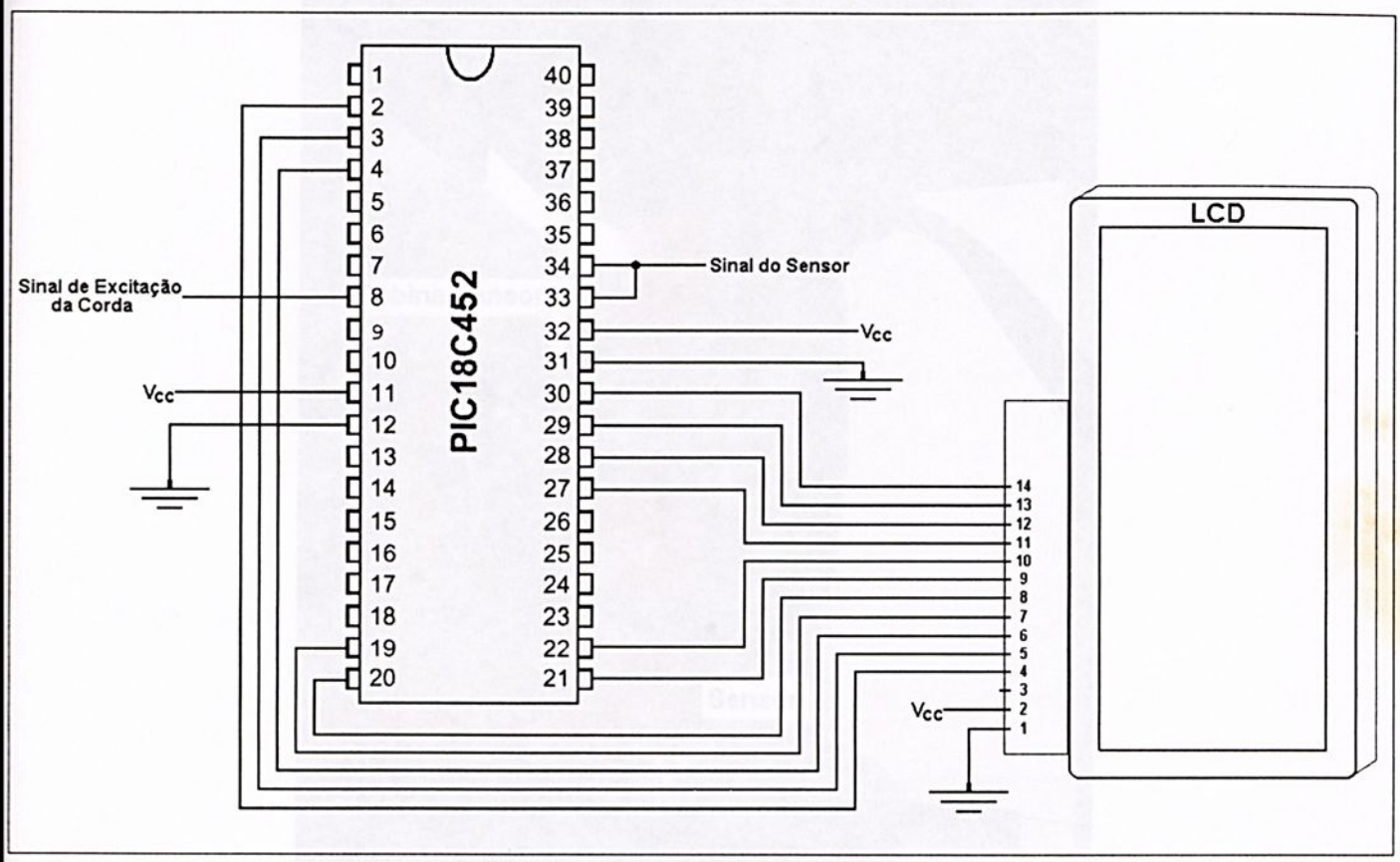


Figura 3.20 – Circuito Implementado para o Microcontrolador PIC 18.

As Figuras 3.21, 3.22, 3.23 e 3.24 mostram o sensor preso a estrutura de testes, o microcontrolador PIC 18 com o display LCD, e o dinamômetro implementado, respectivamente.

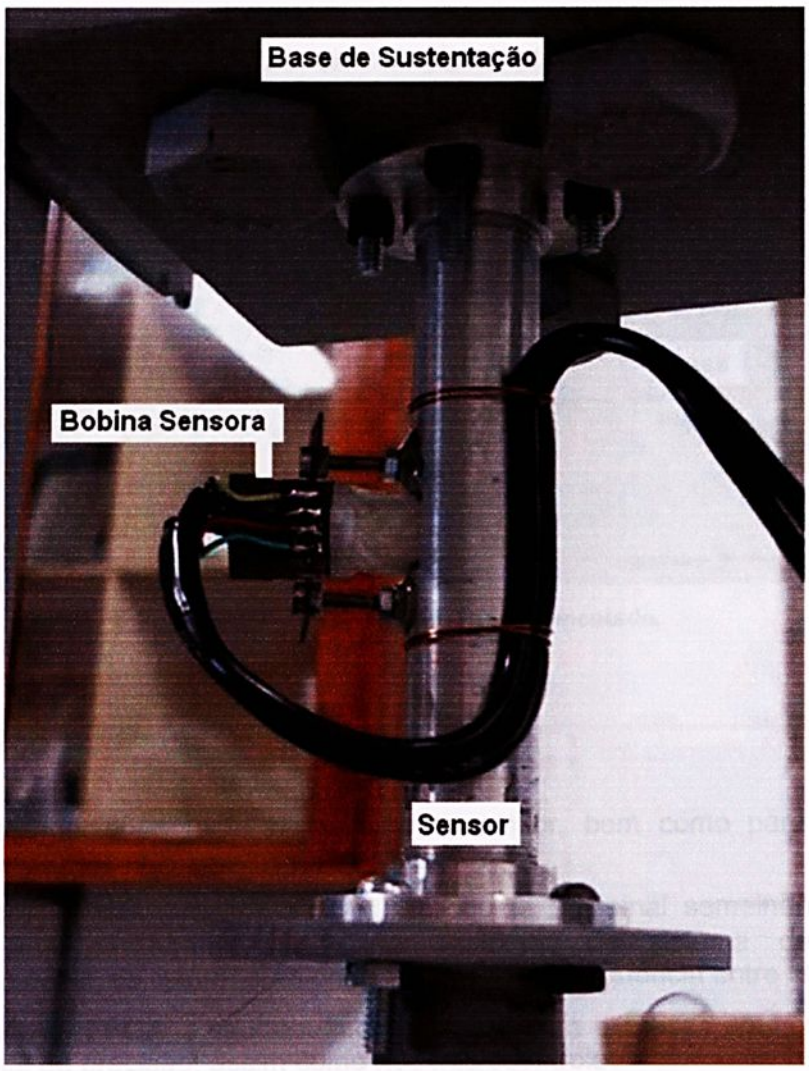


Figura 3.21 – O Sensor Preso a Estrutura de Teste.

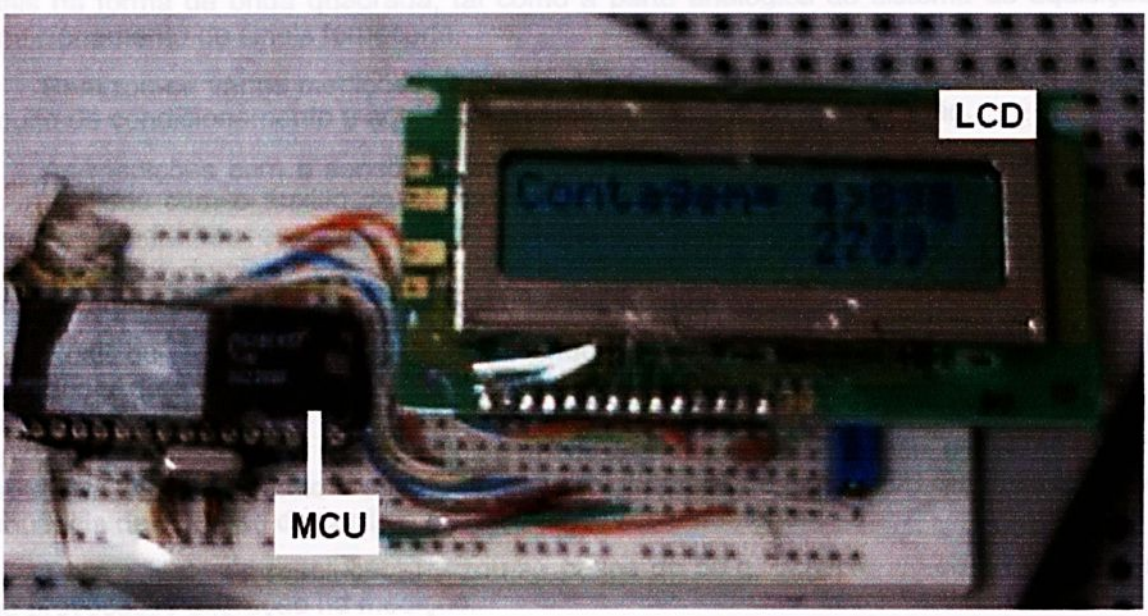


Figura 3.22 – Microcontrolador PIC 18 com o Display LCD.

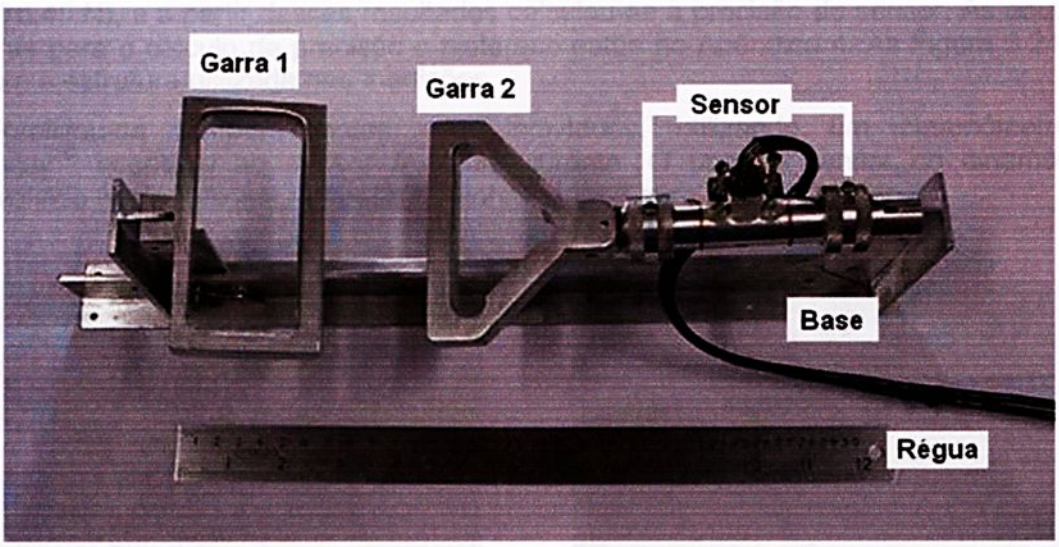


Figura 3.23 – O Dinamômetro Implementado.

3.6. As Medições

Realizou-se medições tanto para testar o sensor, bem como para calibração do sistema de aquisição e condicionamento de sinais.

Para calibração dos circuitos digitais, aplicou-se um sinal semelhante ao sinal do sensor, após sua passagem pela parte analógica do sistema de aquisição e condicionamento de sinais, isto é, uma onda quadrada, de freqüência entre 500 à 1100 Hz.

No o circuito de FPGA, o seu resultado é a freqüência expressa em hexadecimal, com uma casa decimal de precisão, assim como nos microcontroladores.

Utilizou-se de um gerador de sinais da marca Sony/Tektronix, modelo AFG310, com resolução de 0.01 Hz, na freqüência de 1 kHz. Este gerador substitui o sensor, gerando sinais na forma de onda quadrada, tal como a parte analógica do sistema de aquisição e condicionamento de sinais forneceria.

Realizou-se várias medições em diferentes faixas de freqüência, para assim avaliar o circuito de condicionamento e aquisição de sinais em sua faixa de aplicação.

As medições com o sensor foram realizadas somente com os microcontroladores, e foram obtidos com o auxílio de pesos conhecidos. O sensor foi fixado em uma estrutura rígida, os programas respectivos foram carregados nos dispositivos. Interligou-se o sensor a parte analógica do circuito de aquisição e condicionamento de sinais. A parte analógica foi interligada aos pinos de interrupção dos microcontroladores.

Realizou-se dois tipos de teste, aplicando-se forças de compressão e de tração sobre o sensor. Adotou-se pesos de 1 kg, realizando o carregamento e o descarregamento do sensor. Os pesos foram aferidos numa balança eletrônica (Metter, Modelo T5N) com capacidade de 5000 g e de resolução de 0,01 g. Os resultados são apresentados nos displays LCD, apresentado a contagem de ciclos internos de cada MCU em hexadecimal. A freqüência de vibração da corda é obtida a partir desta contagem.

Para o teste de resolução, utilizou-se pesos de 50 g, observando-se a mínima carga que provocasse alteração na medição.

unesp

Para a medição da repetibilidade realizou-se oito medições para o mesmo valor de força (78,45 N), e a partir desses resultados calculou-se a precisão do sensor. As equações utilizadas para o cálculo da precisão e histerese estão no Apêndice 4. As Figura 3.24a e b mostram a estrutura utilizada nas medições.

Realizou-se também testes com indivíduos, utilizando um dinamômetro de referência[25] acoplado ao sensor à corda vibrante. O uso do sensor do dinamômetro desenvolvido nesta pesquisa é realizado conforme a Figura 3.25.

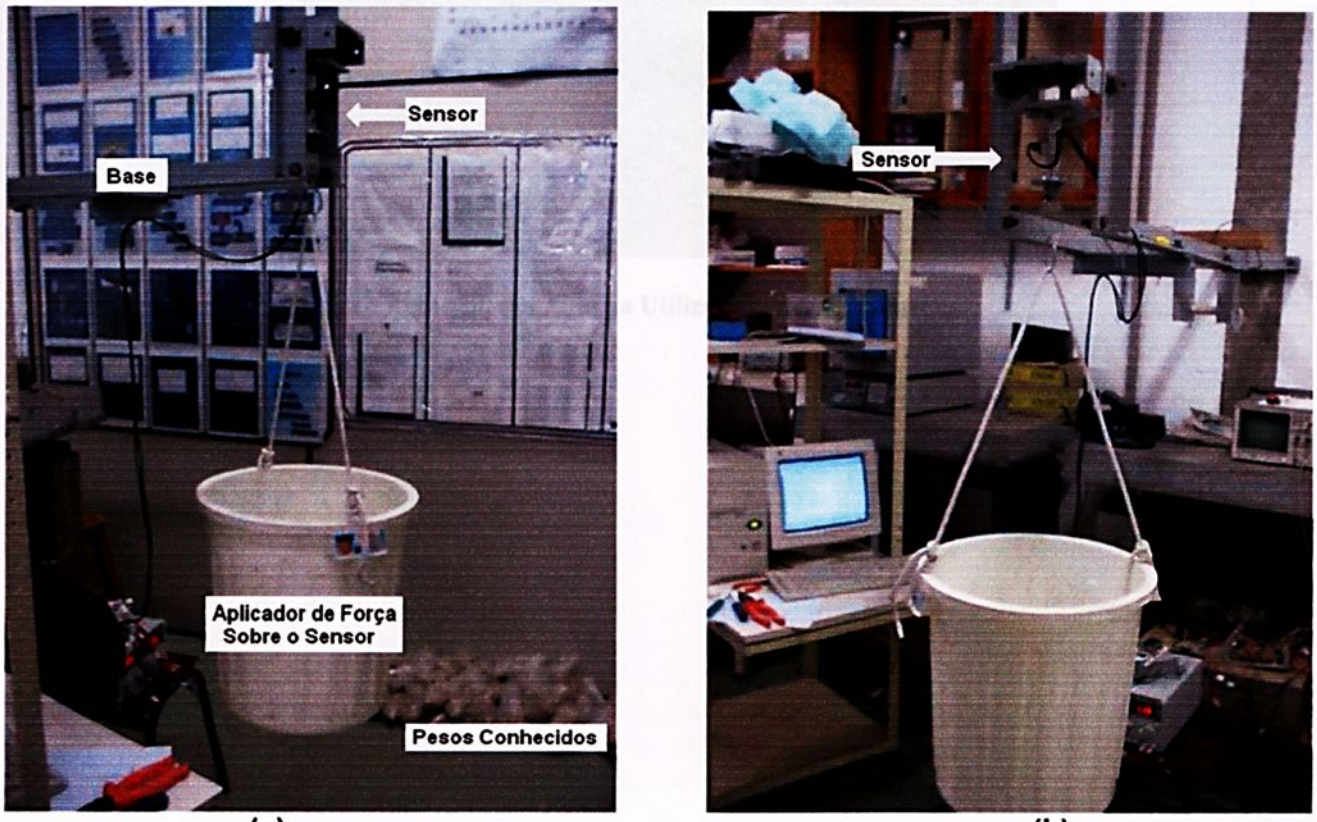


Figura 3.24 – Fotos da Estrutura Utilizada nas Medições.

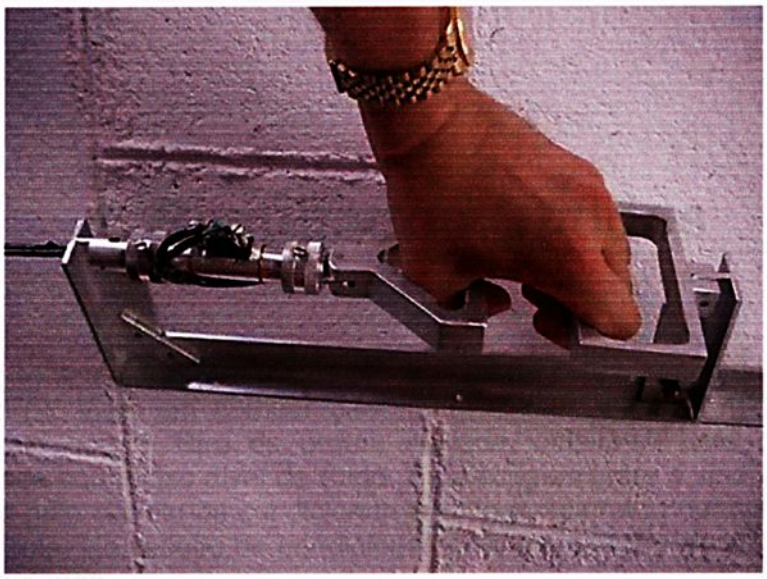


Figura 3.25 – Foto da Utilização do Dinamômetro.

4. Resultados Obtidos

A combinação dos amplificadores de ganho variável com o ganho do filtro forneceu um sinal amplificado (25 V_{PP}), em comparação ao sinal menor que 1 mV_{PP} fornecido pelo sensor. Juntamente com o filtro de ordem elevada, obteve-se uma onda senoidal de amplitude adequada com baixo nível de ruído. A defasagem entre o sinal de entrada e o de saída, introduzido pelo circuito de condicionamento, não acarretou nenhuma distorção para o circuito de controle. No circuito de condicionamento e aquisição de sinais, foram obtidos os resultados esperados.

A excitação da corda, realizada por pulso de corrente, introduziu ruídos no sistema pois as duas bobinas estão sobre o mesmo núcleo. Este tipo de ruído não é filtrado pelo circuito de condicionamento e aquisição de sinais, mas consegui-se minimiza-los produzindo-se alguns atrasos nas leituras.

4.1. O Sensor e a Parte Analógica

De acordo com a expressão (2.14), pode-se simular o comportamento do sensor quando submetido a uma força externa. Com as especificações de construção do sensor, conforme mostrado na Tabela 4.1, obtém-se o gráfico da Figura 4.1, onde a legenda indica Ten10, Ten9, Ten8, Ten7, Ten6 e Ten5 que significa força inicial de 10, 9, 8, 7, 6 e 5 N respectivamente. Com a modificação da tensão inicial aplicada sobre a corda, durante a montagem do sensor, obtém-se uma família de curvas parametrizadas.

Tabela 4.1 – Dados do Corpo do Sensor e da Corda.

Dados do Corpo		Dados da Corda	
Seção do Corpo (m^2)	$5{,}98159 \times 10^{-05}$	Seção da Corda (m^2)	$3{,}14159 \times 10^{-08}$
Mód. Elast. (N/m^2)	$7{,}00 \times 10^{10}$	Mód. Elast. (N/m^2)	$1{,}90 \times 10^{11}$
Comprimento (m)	0,10	Comprimento (m)	0,088
Diâmetro Ext. (m)	0,015	Densidade (kg/m^3)	9350
Espessura (mm)	1,4	Diâmetro (mm)	0,2

unesp

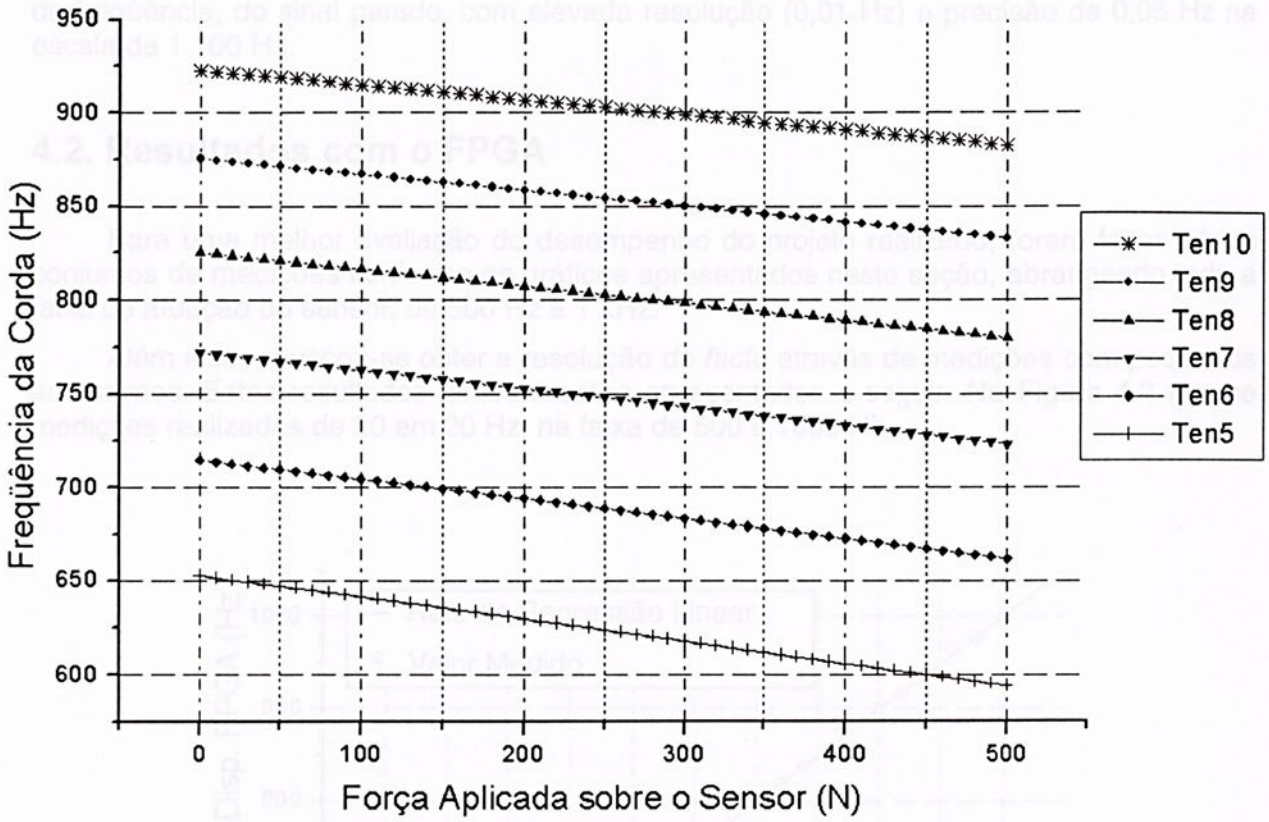


Figura 4.1 – Comportamento do Sensor com a Aplicação de Força Externa, e a Variação da Força Inicial Aplicada sobre a Corda.

Nota-se que, apesar das expressões (2.14) e (2.17) ter comportamento exponencial, verifica-se uma ótima linearidade na faixa de força utilizada, tendo em vista os materiais utilizados na construção do sensor.

Nas curvas apresentadas na Figura 4.1 tem-se uma variação de freqüência com a variação da força externa aplicada ao sensor, apresentados na Tabela 4.2. A faixa de força aplicada é de 0-500 N.

Tabela 4.2 – Relações entre a Força Externa e Freqüência da Corda.

Força Inicial da Corda (N)	Freqüência Inicial (Hz)	Variação de Freqüência (Hz)	Relação Força/Freqüência (N/Hz)
10	922,55	40,27	12,416
9	875,21	42,55	11,751
8	825,15	45,28	11,042
7	771,86	48,61	10,286
6	714,60	52,80	9,470
5	652,34	58,31	8,575

Com a relação entre a variação de força e a variação de freqüência obtida, empregando-se um *display* de quatro dígitos, representando a centena, dezena, unidade e uma casa decimal, os resultados apresentados terão uma resolução média de um Newton por décimo de Hertz (1 N/0,1 Hz).

Este resultado obtido atende as especificações propostas no projeto inicial.

Para se avaliar a exatidão deste medidor utilizou-se sinais produzidos por um gerador de sinais Sony/Tektronix, modelo AFG310. Ele possui um mostrador que apresenta o valor

unesp

da freqüência, do sinal gerado, com elevada resolução (0,01 Hz) e precisão de 0,05 Hz na escala de 1.100 Hz.

4.2. Resultados com o FPGA

Para uma melhor avaliação do desempenho do projeto realizado, foram feitos vários conjuntos de medições conforme os gráficos apresentados neste seção, abrangendo toda a faixa de atuação do sensor, de 500 Hz a 1 kHz.

Além disso, buscou-se obter a resolução *de facto* através de medições com pequenos acréscimos. Estes resultados também estão apresentados a seguir. Na Figura 4.2 tem-se medições realizadas de 20 em 20 Hz, na faixa de 500 à 1000 Hz.

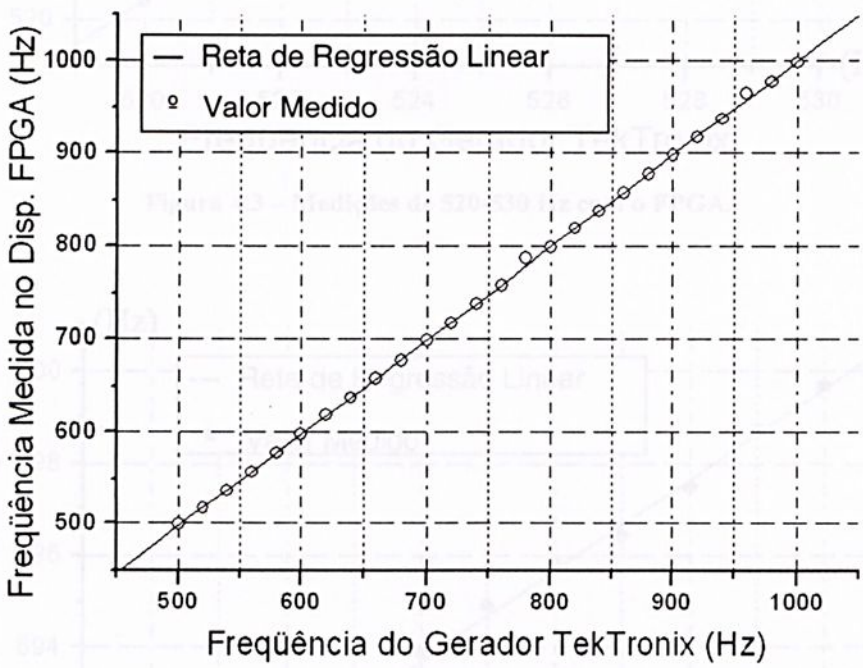


Figura 4.2 – Medições de 500-1000 Hz com o FPGA.

Foram feitas medições de 1 em 1 Hz, nas faixas 520 à 530 Hz, de 690 à 700 Hz, de 870 à 880 Hz e de 960 à 970 Hz, e estão apresentados nas Figuras 4.3, 4.4, 4.5 e 4.6.

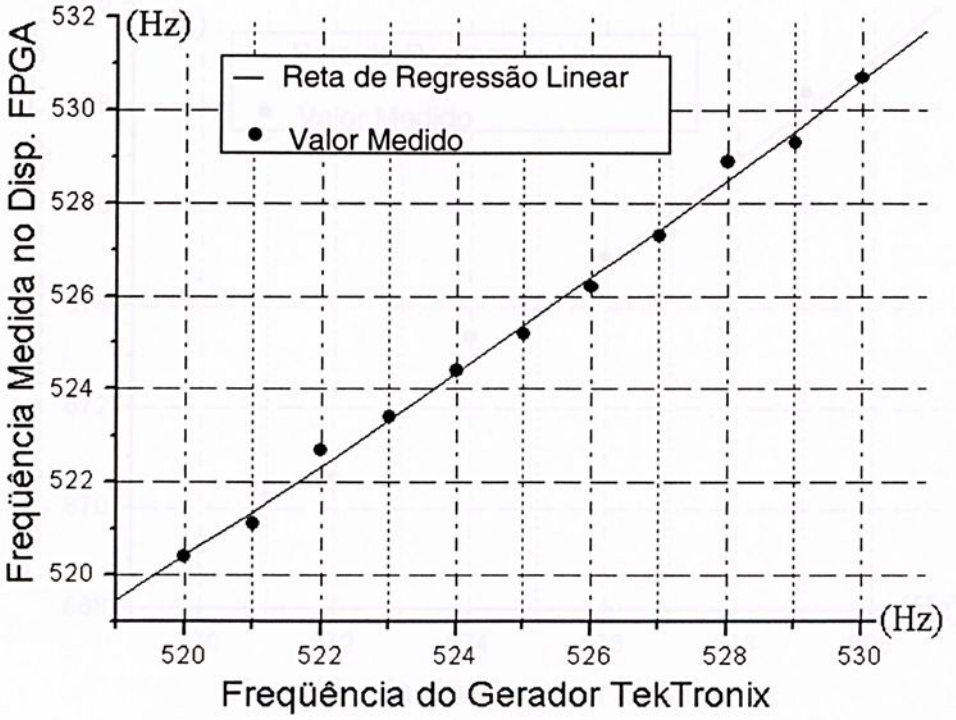


Figura 4.3 – Medições de 520-530 Hz com o FPGA.

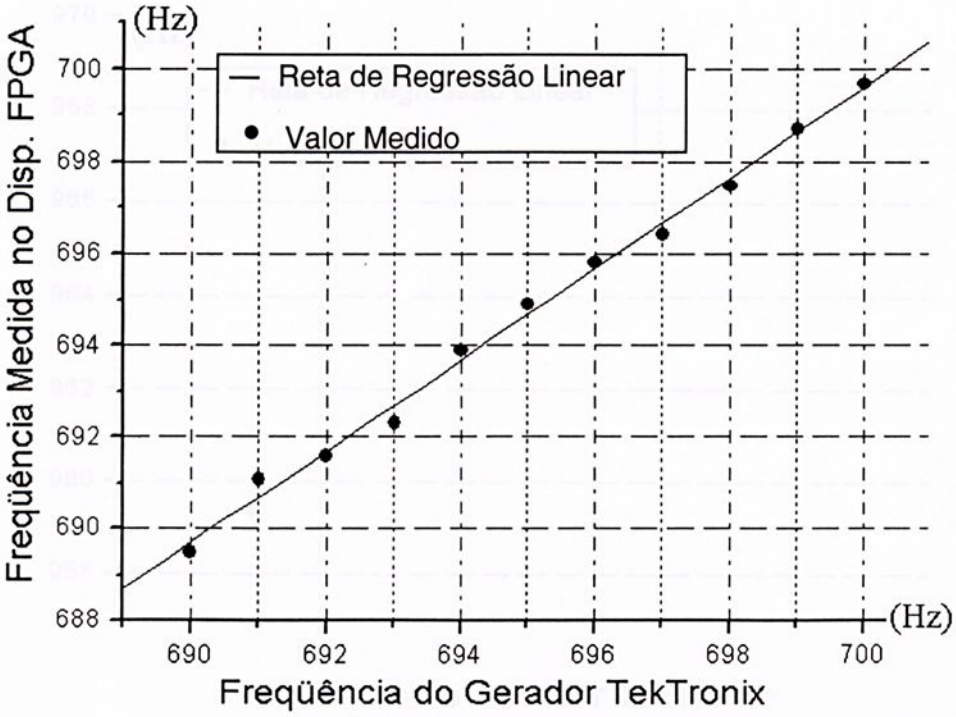


Figura 4.4 – Medições de 690-700 Hz com o FPGA.

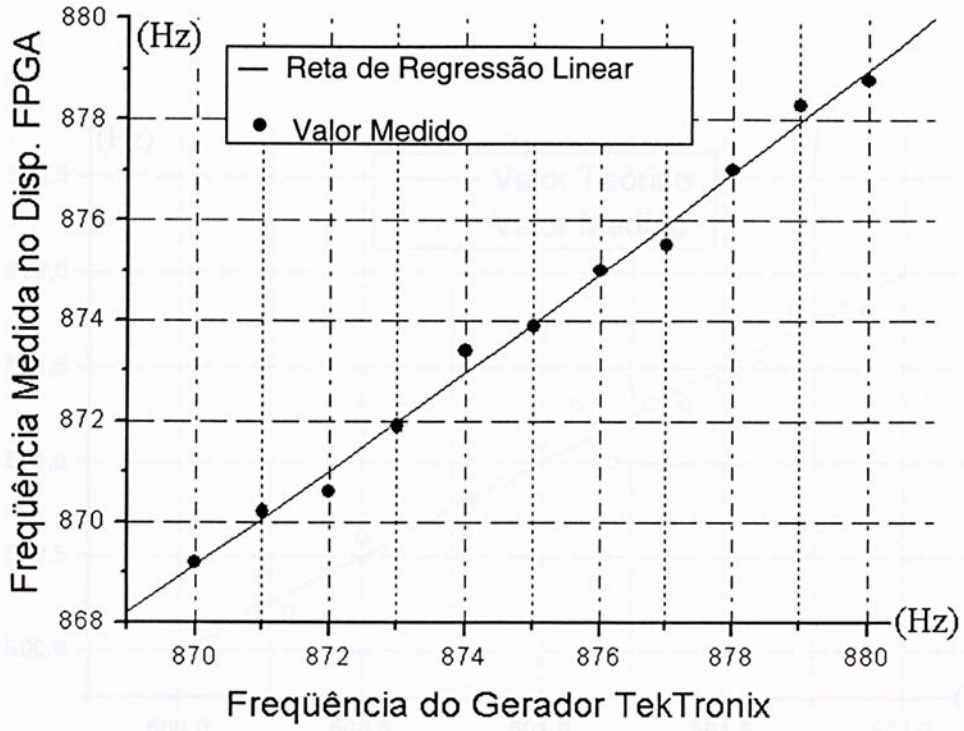


Figura 4.5 – Medições de 870-880 Hz com o FPGA.

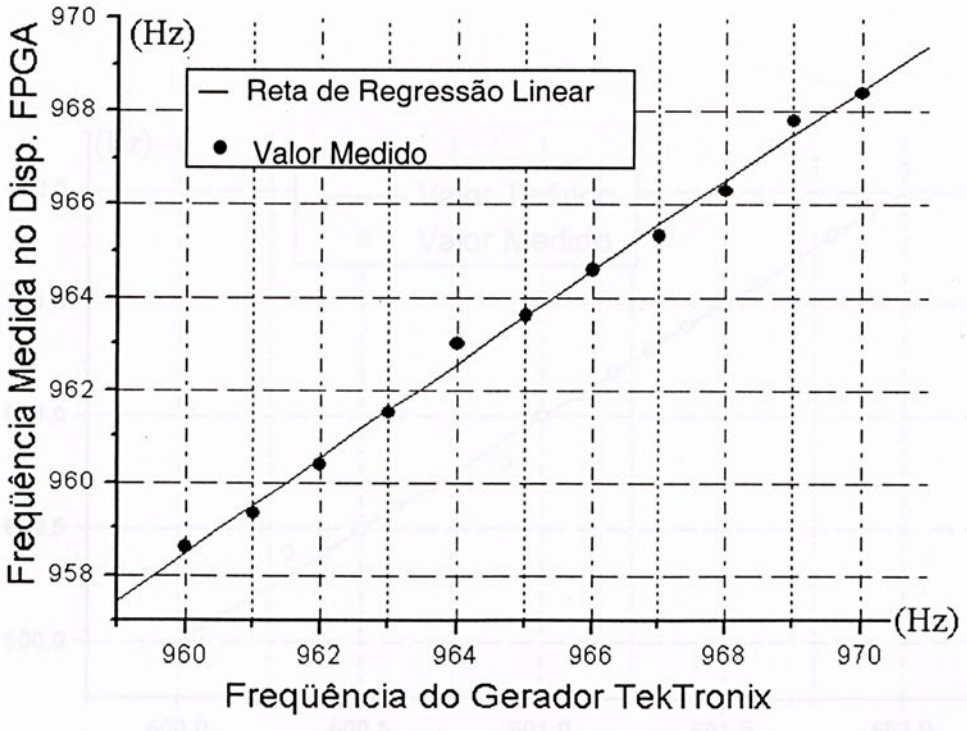


Figura 4.6 – Medições de 960-970 Hz com o FPGA.

A seguir, foram realizadas medições de 0,1 em 0,1 Hz, nas faixas 500 à 502 Hz, de 600 à 602 Hz, de 700 à 702 Hz, de 800 à 802, de 900 à 902, e de 998 à 1.000 Hz, e estão apresentados nas Figuras 4.7, 4.8, 4.9, 4.10, 4.11 e 4.12. Para a realização destas medições, se utilizou o bloco **mediatotal** gravado na segunda FLEX.

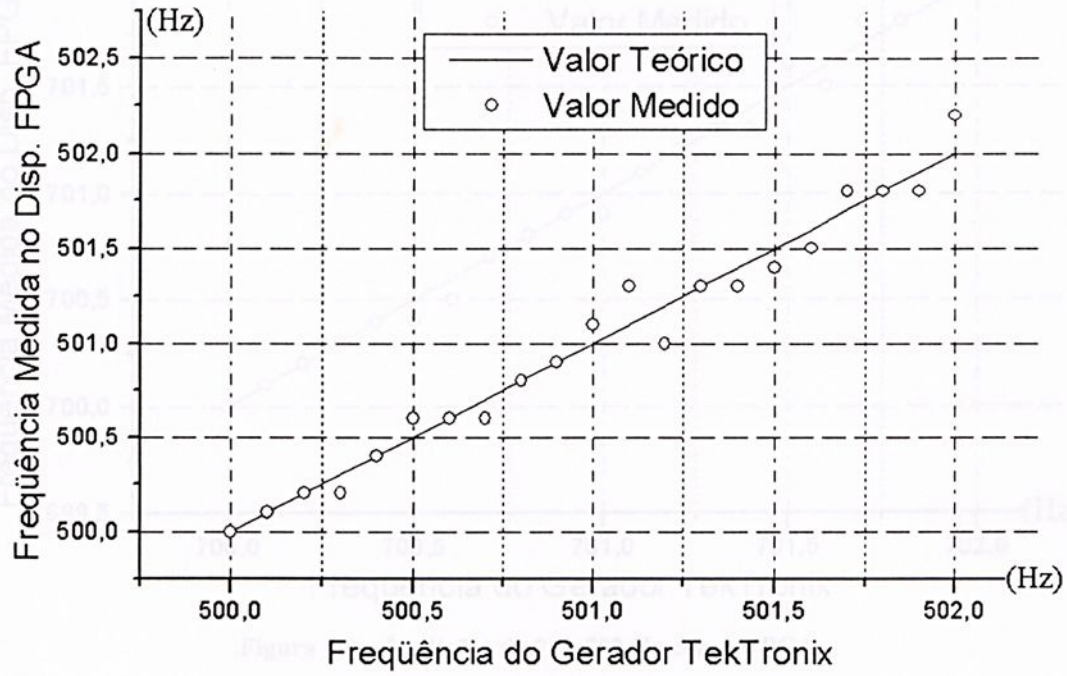


Figura 4.7 – Medições de 500-502 Hz com o FPGA.

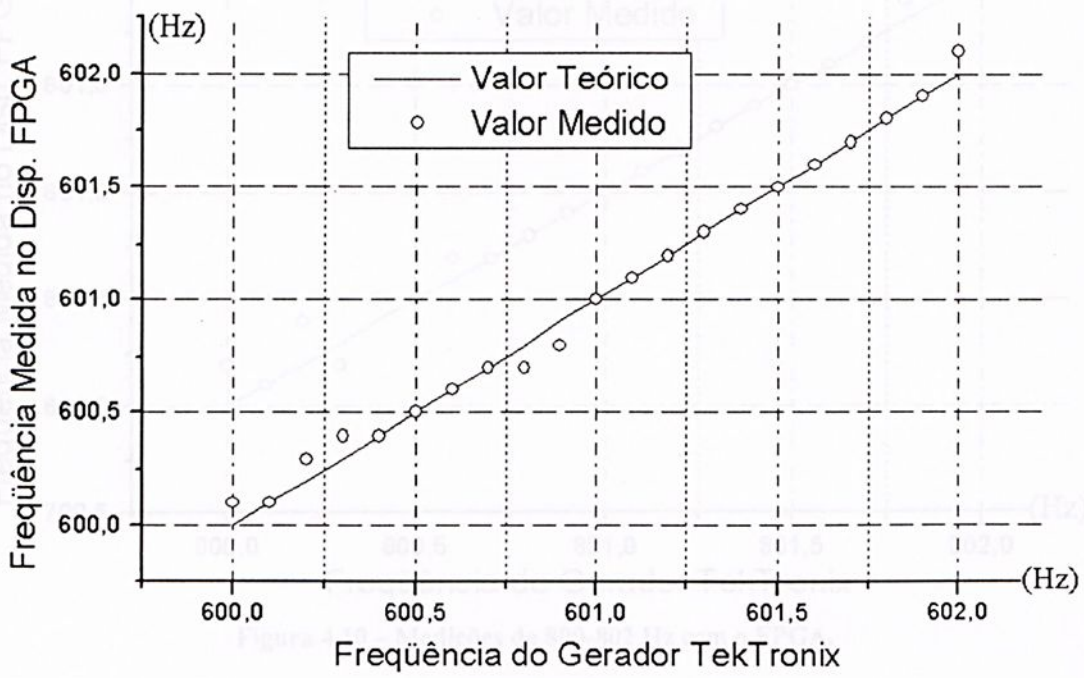


Figura 4.8 – Medições de 600-602 Hz com o FPGA.

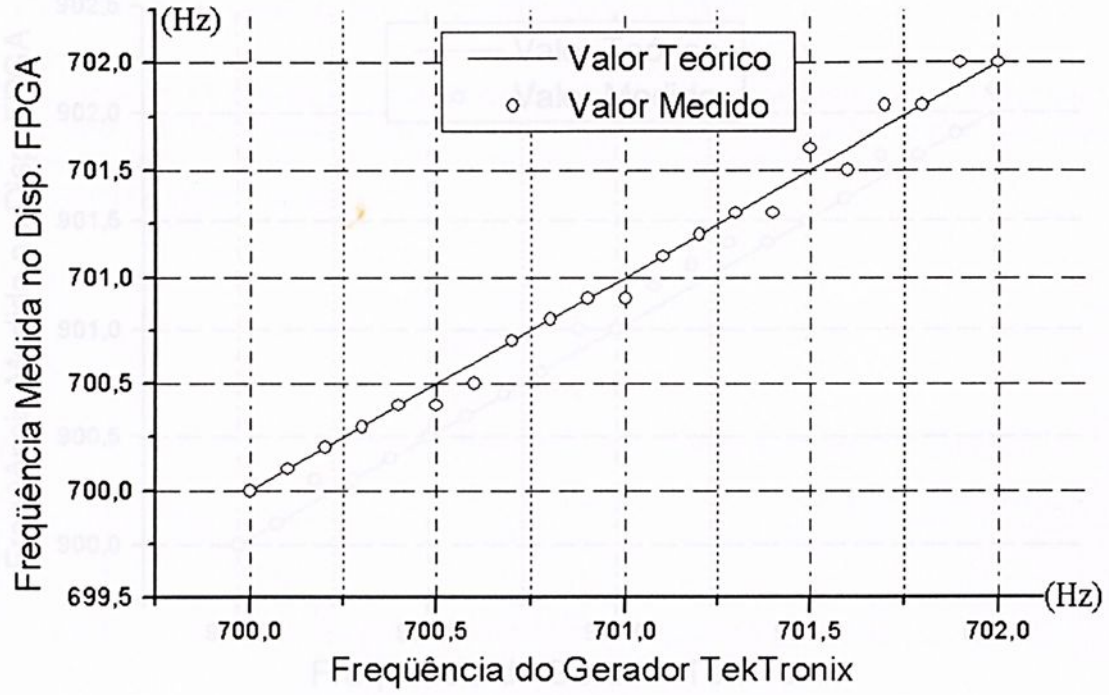


Figura 4.9 – Medições de 700-702 Hz com o FPGA.

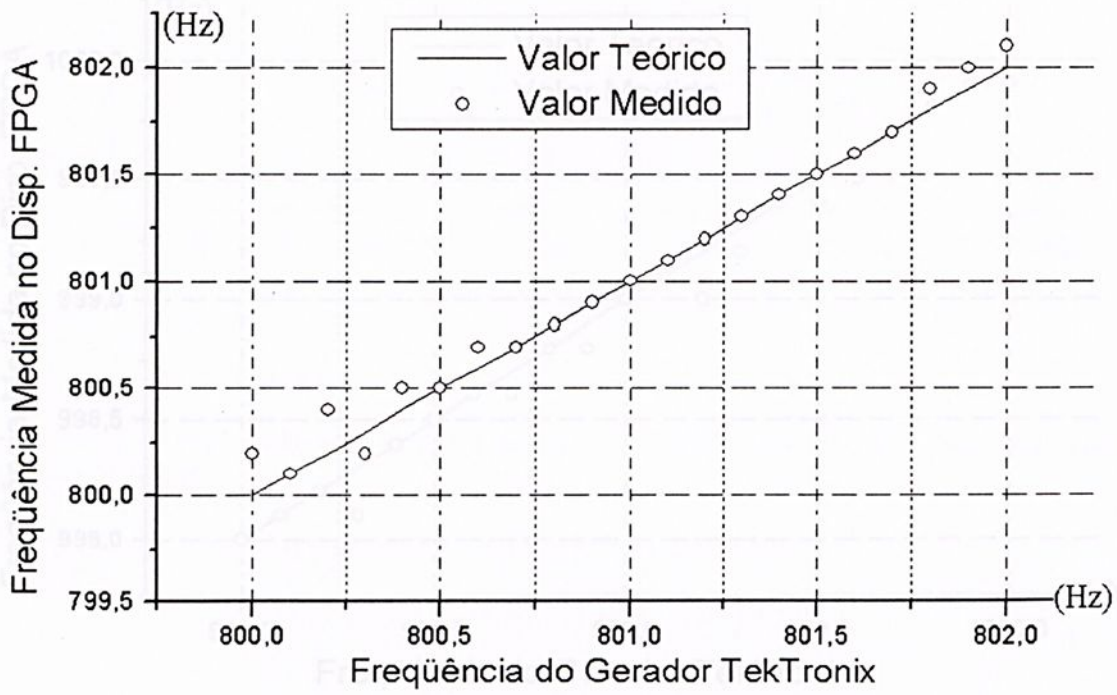


Figura 4.10 – Medições de 800-802 Hz com o FPGA.

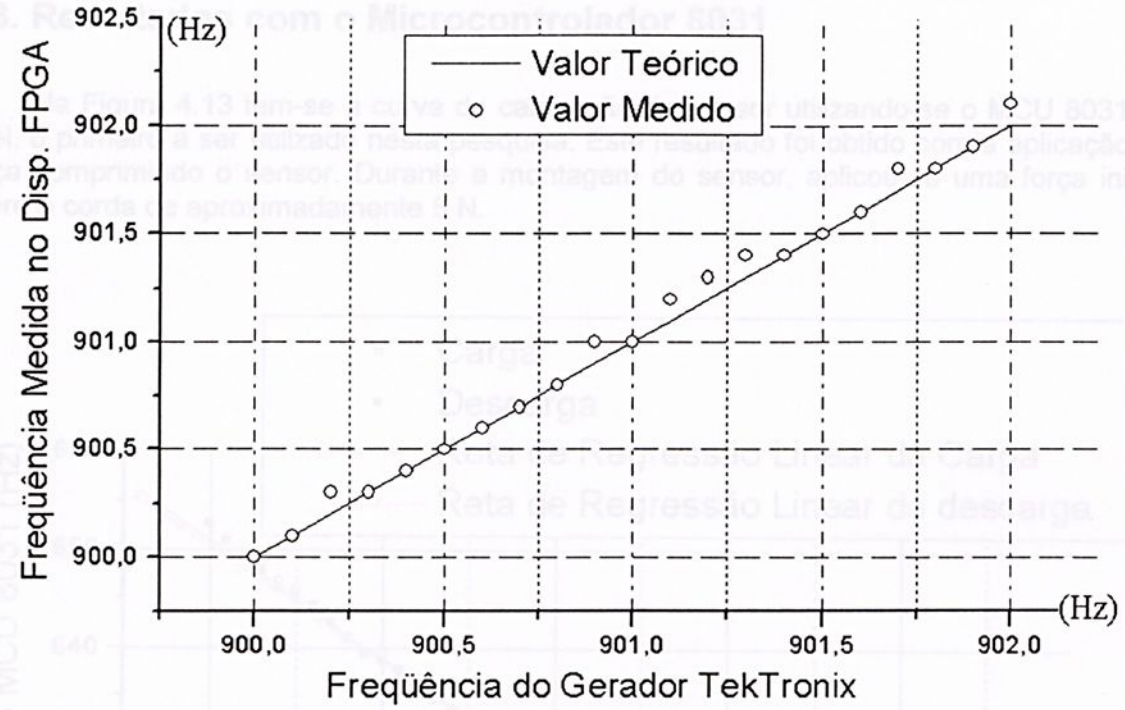


Figura 4.11 – Medições de 900-902 Hz com o FPGA.

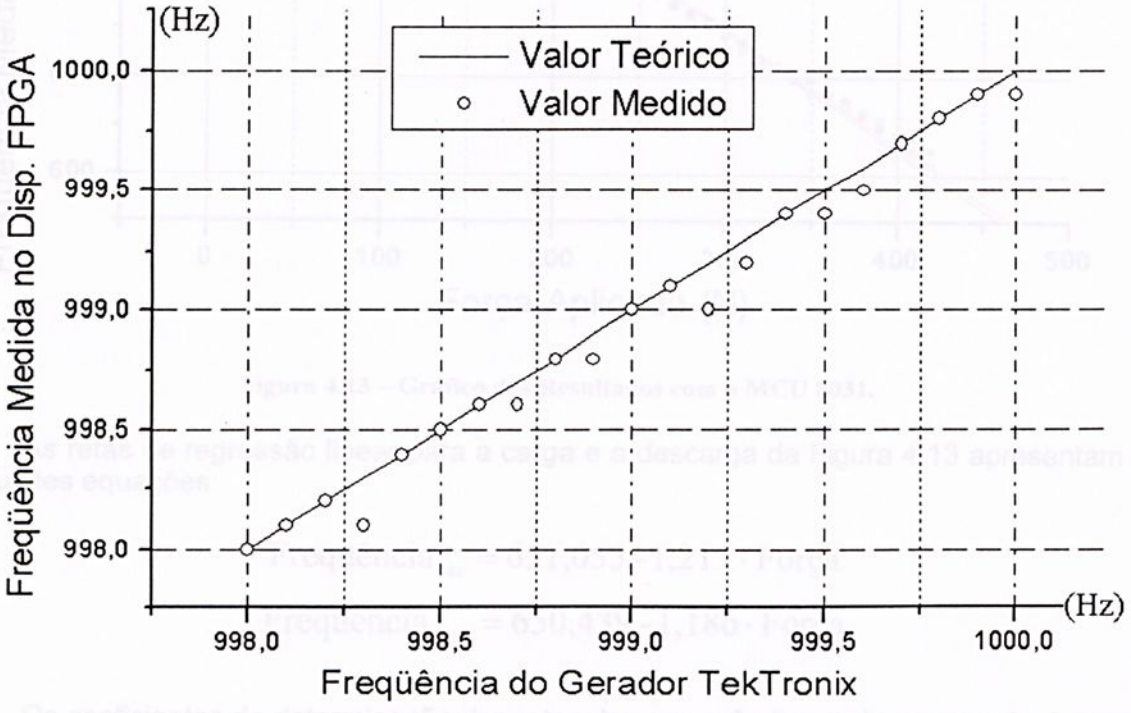


Figura 4.12 – Medições de 998-1.000 Hz com o FPGA.

4.3. Resultados com o Microcontrolador 8031

Na Figura 4.13 tem-se a curva de calibracão do sensor utilizando-se o MCU 8031 da Intel, o primeiro a ser utilizado nesta pesquisa. Este resultado foi obtido com a aplicação de força comprimindo o sensor. Durante a montagem do sensor, aplicou-se uma força inicial sobre a corda de aproximadamente 5 N.

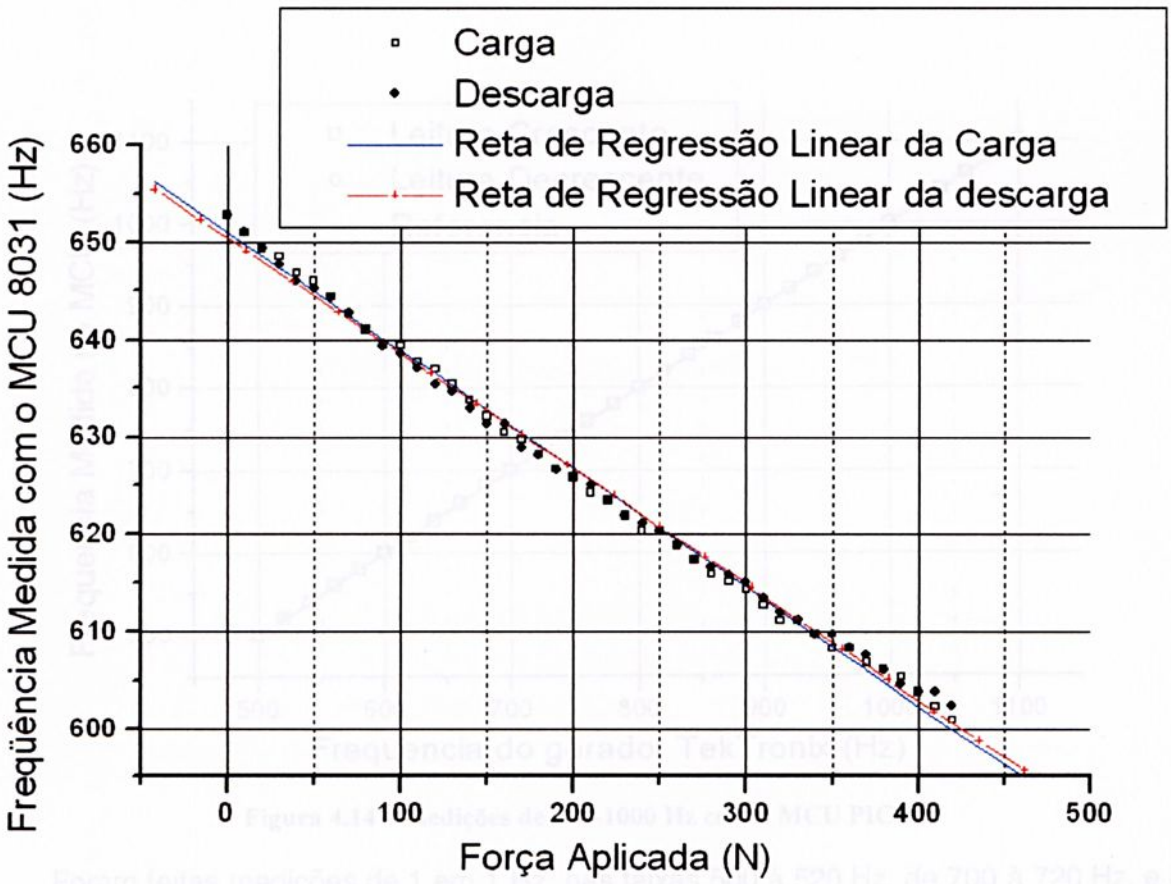


Figura 4.13 – Gráfico dos Resultados com o MCU 8031.

As retas de regressão linear para a carga e a descarga da Figura 4.13 apresentam as seguintes equações:

$$\text{Freqüência}_{car} = 651{,}055 - 1{,}217 \cdot \text{Força}$$

$$\text{Freqüência}_{desc} = 650{,}439 - 1{,}186 \cdot \text{Força}$$

Os coeficientes de determinação das retas de regressão linear da carga e da descarga são, respectivamente, 0,9963 e 0,9958.

Portanto, consta-se a linearidade da resposta do sensor e uma histerese, cujo valor máximo é de 3,17 %.

4.4. Resultados com o Microcontrolador PIC18

Para uma melhor avaliação do desempenho do sistema de medição com o PIC 18, foram feitos vários conjuntos de medições conforme os gráficos apresentados nesta seção, abrangendo toda a faixa de atuação do sensor, de 500 Hz a 1,1 kHz.

Além disso, buscou-se obter a resolução *de facto* através de medições com pequenos acréscimos. Estes resultados também estão apresentados a seguir. Na Figura 4.14, tem-se as medições realizadas de 20 em 20 Hz, na faixa de 500 à 1100 Hz.

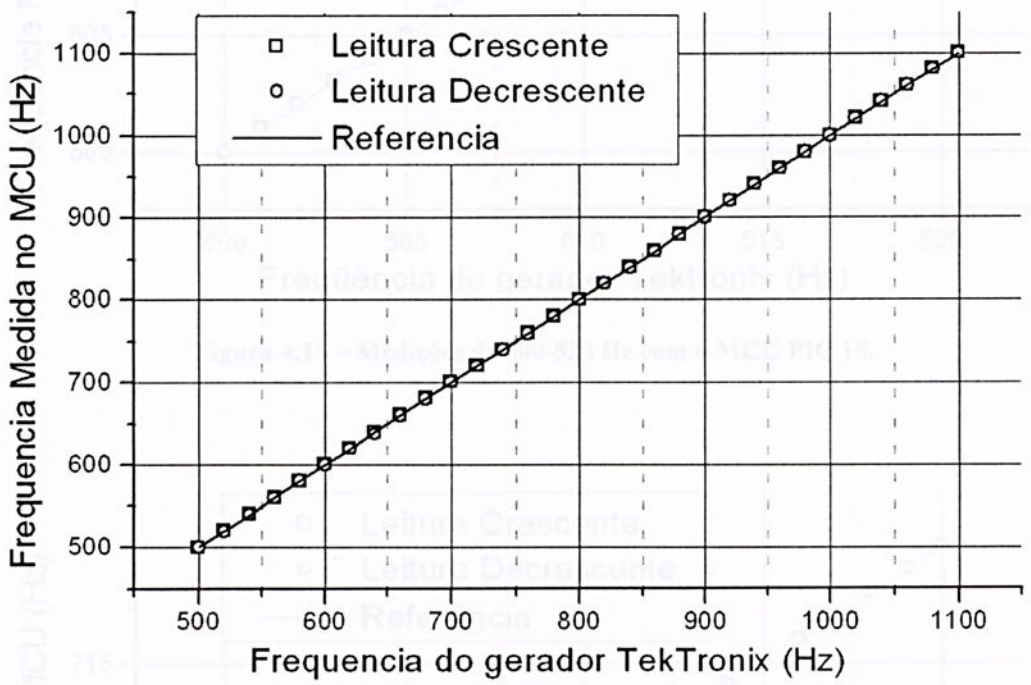


Figura 4.14 – Medições de 500-1000 Hz com o MCU PIC 18.

Foram feitas medições de 1 em 1 Hz, nas faixas 500 à 520 Hz, de 700 à 720 Hz, e de 1000 à 1020 Hz, e estão apresentados nas Figuras 4.15, 4.16, e 4.17.

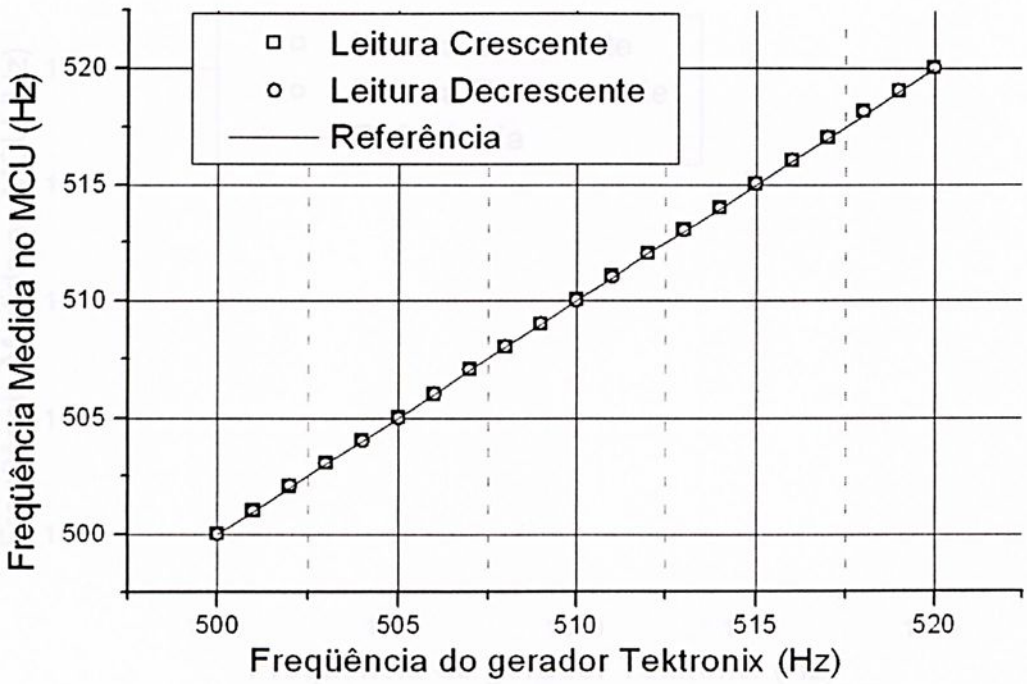


Figura 4.15 – Medições de 500-520 Hz com o MCU PIC 18.

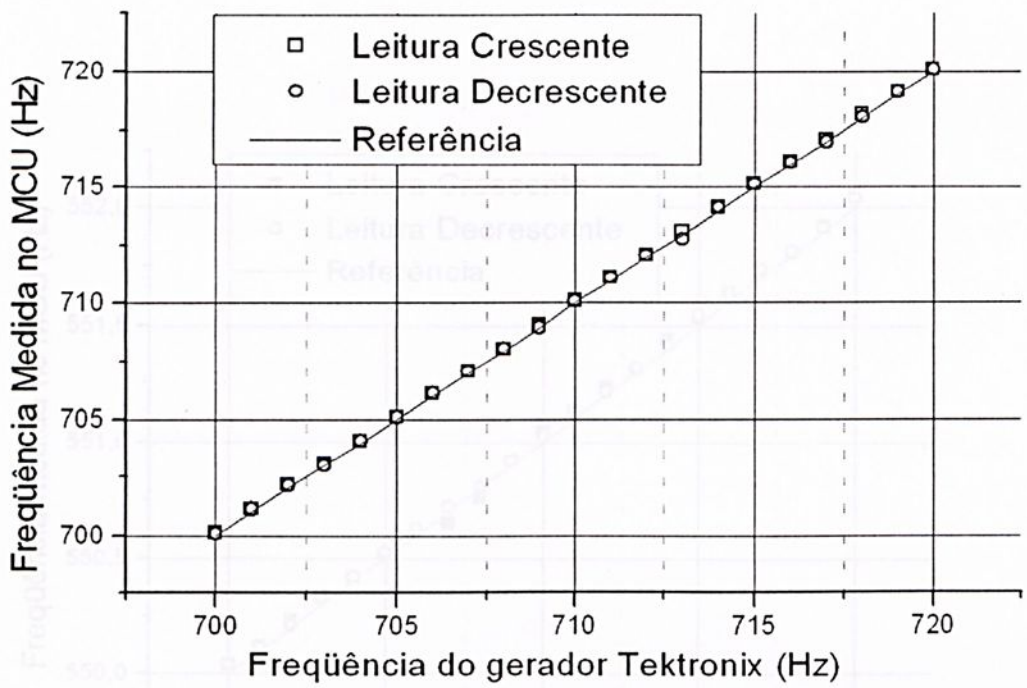


Figura 4.16 – Medições de 700-720 Hz com o MCU PIC 18.

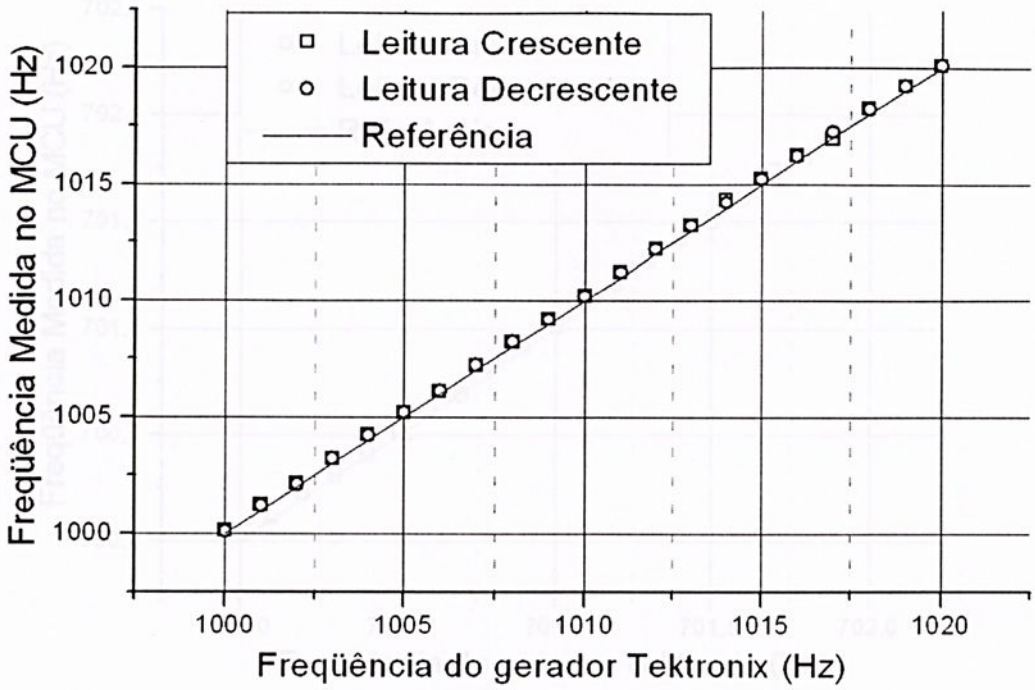


Figura 417 – Medições de 1000-1020 Hz com o MCU PIC 18.

A seguir, foram realizadas medições de 0,1 em 0,1 Hz, nas faixas 500 à 502 Hz, de 700 à 702 Hz, e de 950 à 952 Hz, e estão sendo apresentadas nas Figuras 4.18, 4.19, e 4.20.

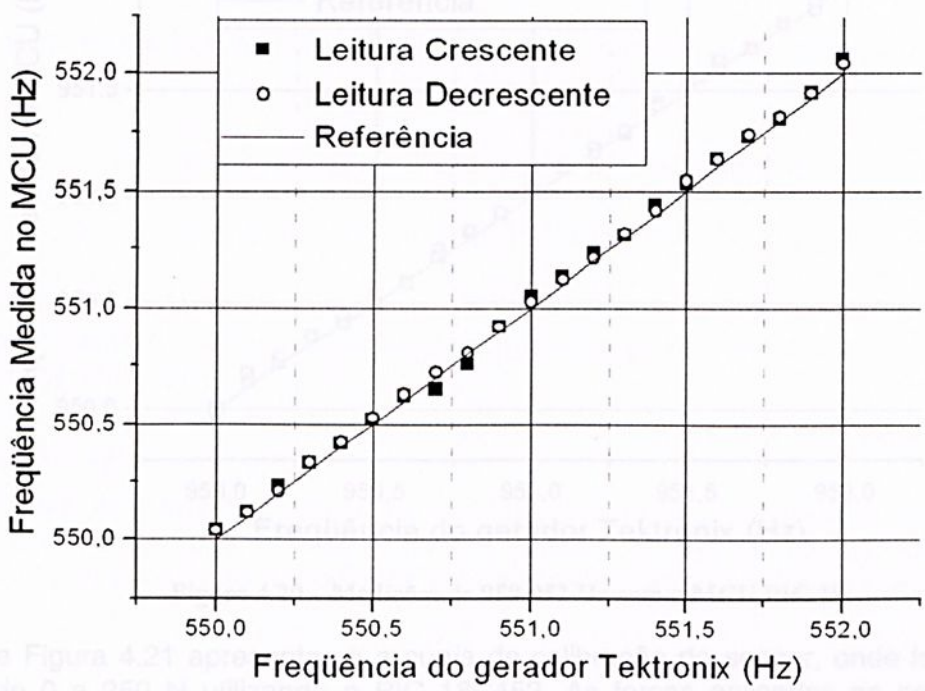


Figura 4.18 – Medições de 500-502 Hz com o MCU PIC 18.

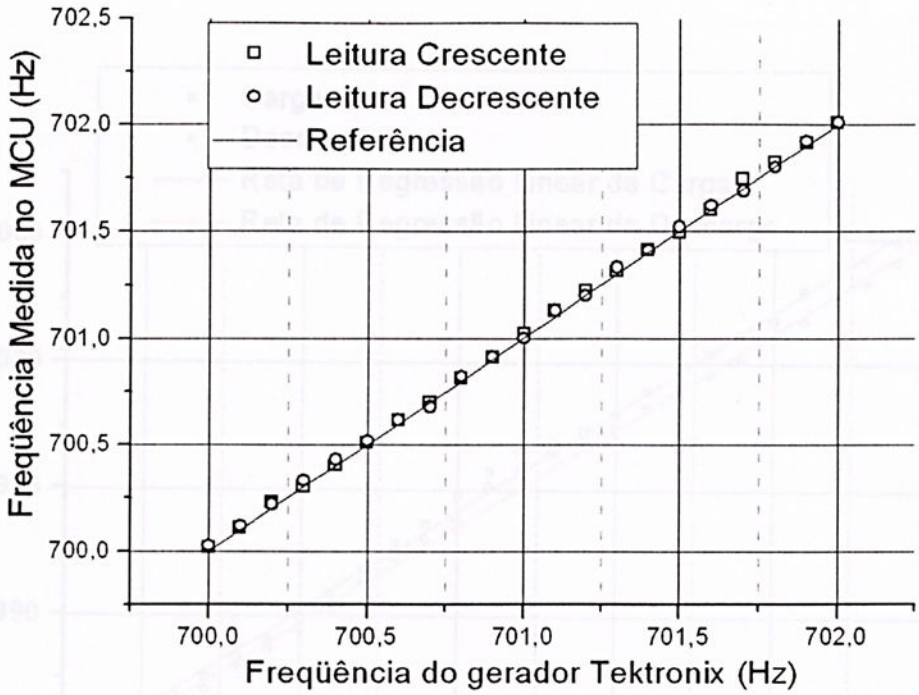


Figura 4.19 – Medições de 700-702 Hz com o MCU PIC 18.

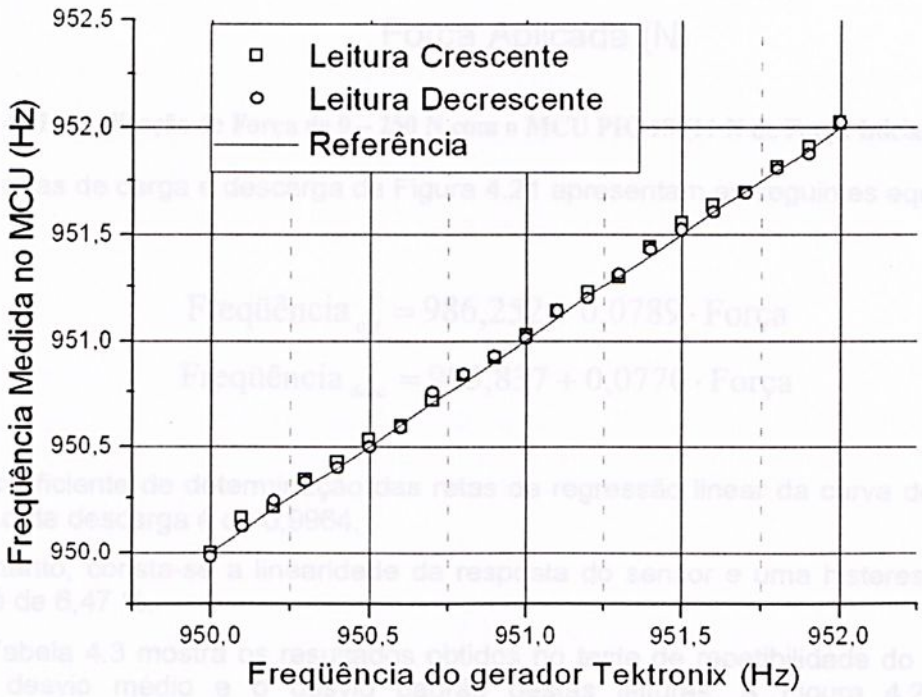


Figura 4.20 – Medições de 950-952 Hz com o MCU PIC 18.

Na Figura 4.21 apresenta-se a curva de calibração do sensor, onde foram aplicadas forças de 0 a 250 N utilizando o PIC 18C452. As forças aplicadas no sensor foram de tração. Durante a montagem do sensor, aplicou-se uma força inicial sobre a corda de aproximadamente 11 N.

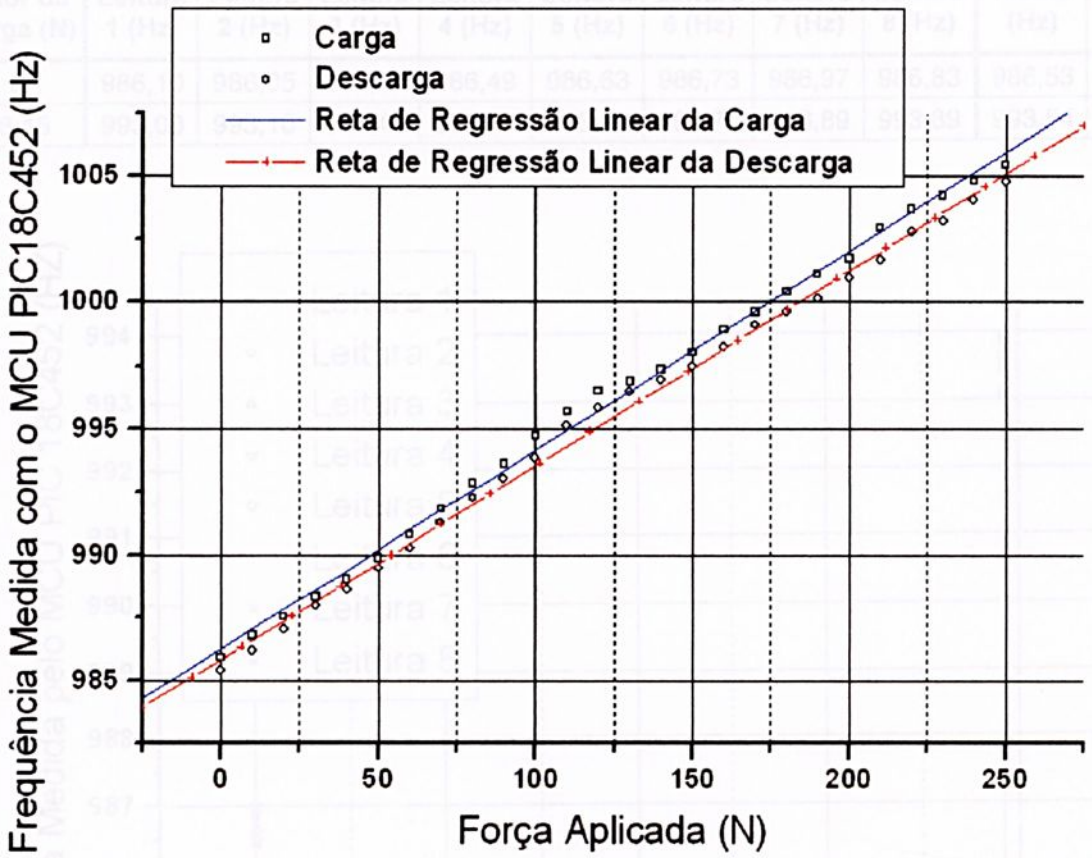


Figura 4.21 – Aplicação de Força de 0 – 250 N com o MCU PIC 18 (11 N de Força Inicial na Corda).

As retas de carga e descarga da Figura 4.21 apresentam as seguintes equações:

$$\text{Freqüência}_{car} = 986{,}252 + 0{,}0789 \cdot \text{Força}$$

$$\text{Freqüência}_{desc} = 985{,}837 + 0{,}0770 \cdot \text{Força}$$

O coeficiente de determinação das retas de regressão linear da curva de carga é de 0,9968 e o da descarga é de 0,9964.

Portanto, consta-se a linearidade da resposta do sensor e uma histerese, cujo valor máximo é de 6,47 %.

A Tabela 4.3 mostra os resultados obtidos no teste de repetibilidade do sensor, bem como o desvio médio e o desvio padrão destas leituras. A Figura 4.22 mostra a repetibilidade do sensor em 8 medições com a mesma força. O valor de carga adotado foi de 78,45 N.

Tabela 4.3 – Leituras da Freqüência do Teste de Repetibilidade.

Valor da Carga (N)	Leitura 1 (Hz)	Leitura 2 (Hz)	Leitura 3 (Hz)	Leitura 4 (Hz)	Leitura 5 (Hz)	Leitura 6 (Hz)	Leitura 7 (Hz)	Leitura 8 (Hz)	Média (Hz)	Desvio Padrão (Hz)
0	986,10	986,05	986,44	986,49	986,63	986,73	986,97	986,83	986,53	0,33
78,45	993,00	993,10	993,44	993,54	993,69	993,79	993,89	993,89	993,54	0,34

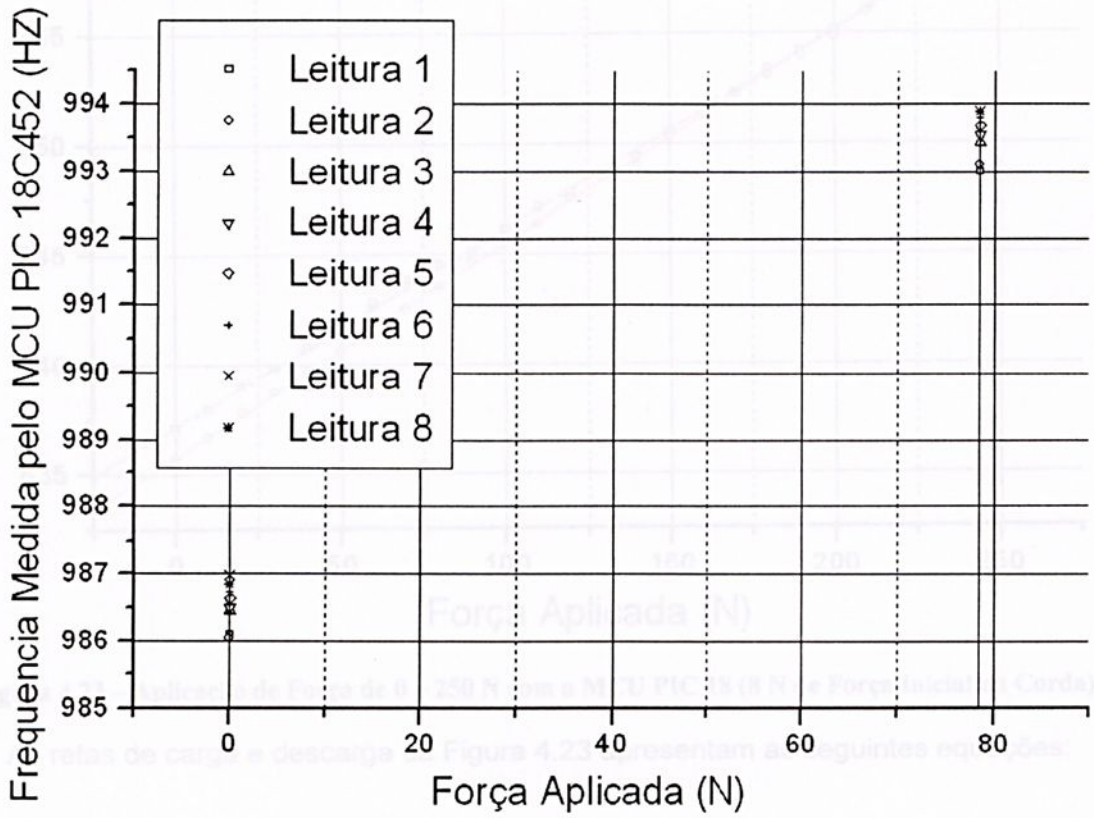


Figura 4.22 – Teste de Repetibilidade do Sensor para 78,45 N.

A precisão é muito boa, com valor superior a 99,93 %.

Na Figura 4.23 apresenta-se a curva de calibração do sensor onde aplicou-se uma força inicial sobre a corda de aproximandamente 8 N. Foram aplicadas forças de 0 a 250 N utilizando o PIC 18C452. As forças aplicadas no sensor foram de tração.

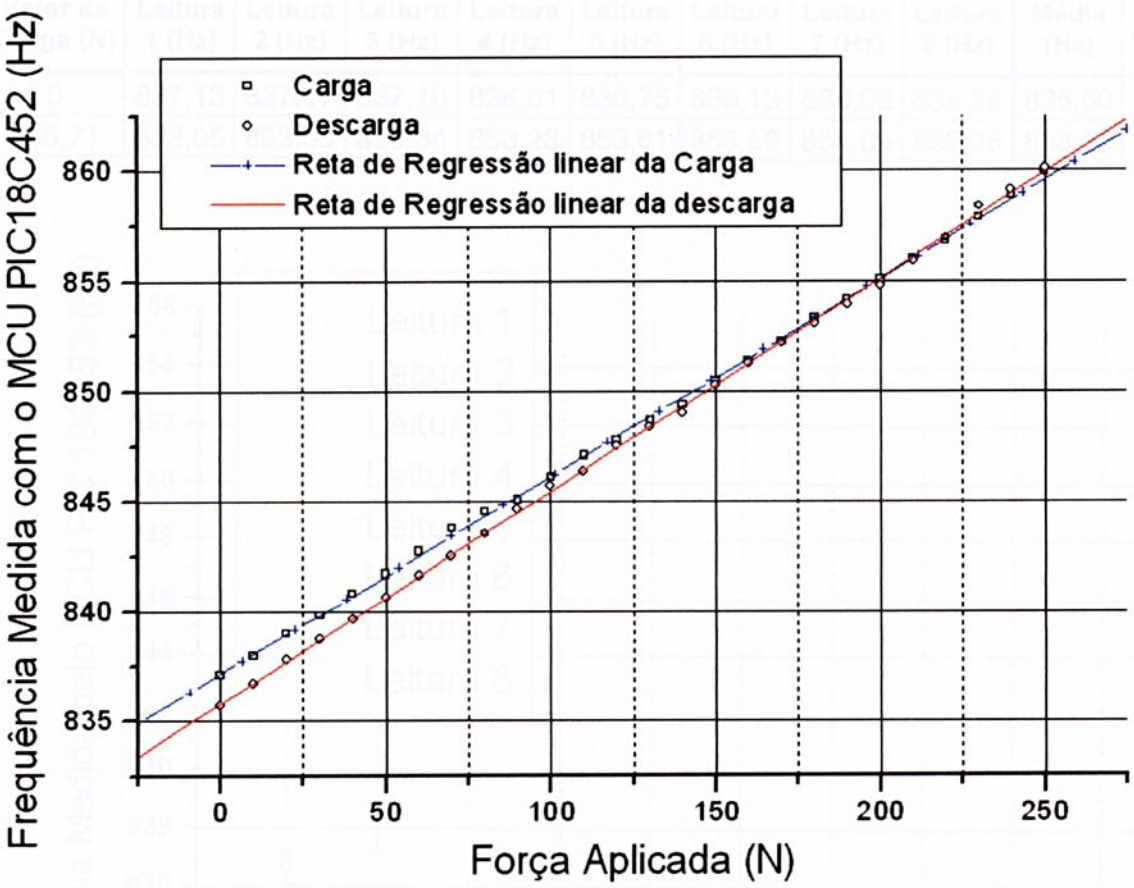


Figura 4.23 – Aplicação de Força de 0 – 250 N com o MCU PIC 18 (8 N de Força Inicial na Corda).

As retas de carga e descarga da Figura 4.23 apresentam as seguintes equações:

$$\text{Freqüência}_{car} = 837{,}155 + 0{,}0896 \cdot \text{Força}$$

$$\text{Freqüência}_{desc} = 835{,}817 + 0{,}0964 \cdot \text{Força}$$

O coeficiente de determinação das retas de regressão linear da curva de carga é de 0,99943 e o da descarga é de 0,99956.

Portanto, consta-se a linearidade da resposta do sensor e uma histerese, cujo valor máximo é de 5,89 %.

A Tabela 4.4 mostra os resultados obtidos no teste de repetibilidade do sensor, bem como o desvio médio e o desvio padrão destas leituras. A Figura 4.24 mostra a repetibilidade do sensor em 8 medições com a mesma força. O valor de carga adotado foi de 166,71 N.

Tabela 4.3 – Leituras da Freqüência do Teste de Repetibilidade.

Valor da Carga (N)	Leitura 1 (Hz)	Leitura 2 (Hz)	Leitura 3 (Hz)	Leitura 4 (Hz)	Leitura 5 (Hz)	Leitura 6 (Hz)	Leitura 7 (Hz)	Leitura 8 (Hz)	Média (Hz)	Desvio Padrão (Hz)
0	837,13	837,17	837,10	836,61	836,75	836,13	836,02	835,88	836,60	0,53
166,71	853,05	853,56	853,34	853,23	853,81	853,56	854,06	853,38	853,50	0,32

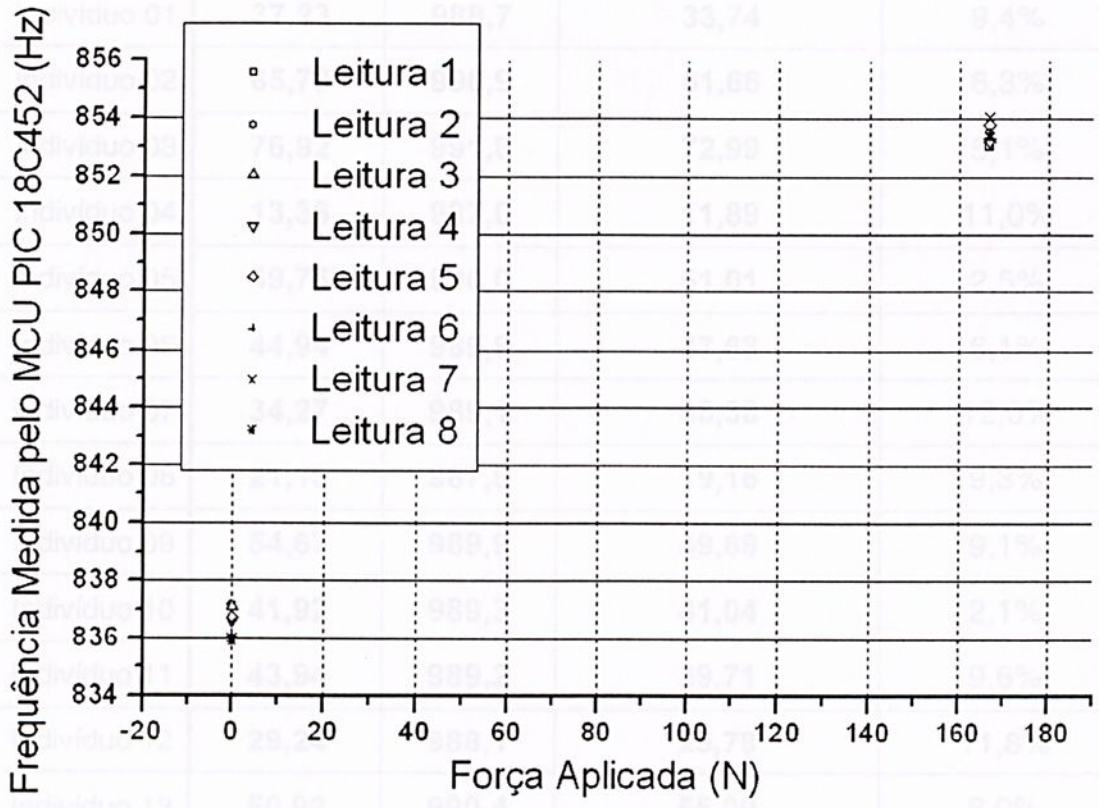


Figura 4.22 – Teste de Repetibilidade do Sensor para 78,45 N.

A precisão é muito boa, com valor superior a 99,998 %.

Para ambas montagens, a resolução é definida como a mínima variação do sinal de entrada que produz uma variação observável na saída do sensor. A resolução observada foi de 1 N, que corresponde a uma variação na saída do sistema de medição, de 0,1 Hz.

O tempo de resposta do sensor é diminuto devido às características mecânicas do mesmo. A limitação do tempo de resposta está no sistema eletrônico de medição. Tomando por base o circuito com o MCU PIC18, com o sinal do sensor em torno de 1 kHz, tem-se a excitação da corda a cada 20 ms. Como o sistema realiza uma leitura a cada sinal de excitação, o tempo de resposta é de 20 ms. Para o calculo da média das leituras, são utilizadas 64 leituras, logo, são necessários 1,28 s para que uma mudança na força aplicada sobre o sensor seja notada no display LCD. Portanto, o sistema implementado é mais adequado para medições de forças estáticas ou variantes lentamente com o tempo.

Foram realizados testes com 13 indivíduos, voluntários do Laboratório de Sensores. Nestas medições uma célula de carga com strain gages metálicos[24] foi utilizada como referência. A conexão entre os dois sensores foi realizada conforme ilustra a Figura 4.23. Os

resultados estão mostrados na Tabela 4.4. Na conversão de freqüência (Hz) para força (N) utilizou-se a Figura 4.21, e também uma média entre as equações obtidas a partir dela.

Tabela 4.4 – Resultados com Indivíduos.

	Célula de Carga (N)	Freqüência (Hz)	Força Correspondente a freqüência (N)	Diferença entre as Forças (%)
Indivíduo 01	**37,23**	**988,7**	**33,74**	9,4%
Indivíduo 02	**65,79**	**990,9**	**61,66**	6,3%
Indivíduo 03	**76,92**	**991,8**	**72,99**	5,1%
Indivíduo 04	**13,36**	**987,0**	**11,89**	11,0%
Indivíduo 05	**49,75**	**990,0**	**51,01**	2,5%
Indivíduo 06	**44,94**	**989,8**	**47,68**	6,1%
Indivíduo 07	**34,27**	**989,1**	**38,38**	12,0%
Indivíduo 08	**21,13**	**987,5**	**19,16**	9,3%
Indivíduo 09	**54,67**	**989,9**	**49,68**	9,1%
Indivíduo 10	**41,92**	**989,3**	**41,04**	2,1%
Indivíduo 11	**43,94**	**989,2**	**39,71**	9,6%
Indivíduo 12	**29,24**	**988,1**	**25,78**	11,8%
Indivíduo 13	**50,92**	**990,4**	**55,00**	8,0%

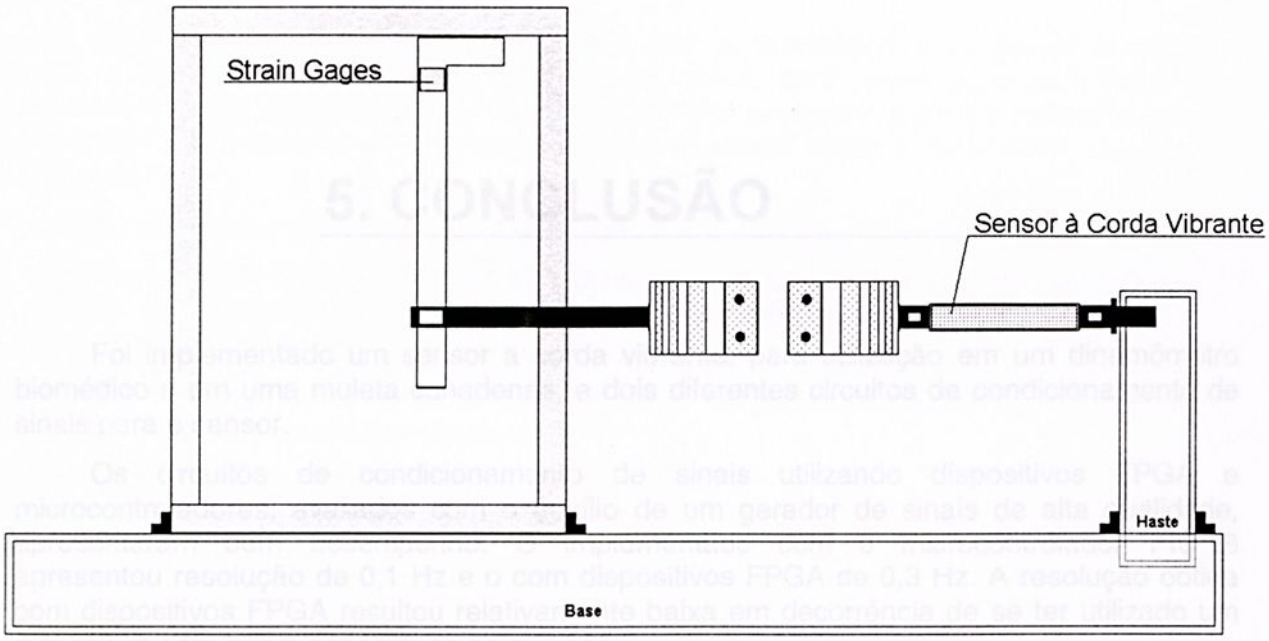


Figura 4.23 – Conexão entre a Célula de Carga e o Sensor à Corda Vibrante.

5. CONCLUSÃO

Foi implementado um sensor a corda vibrante, para utilização em um dinamômetro biomédico e em uma muleta canadense, e dois diferentes circuitos de condicionamento de sinais para o sensor.

Os circuitos de condicionamento de sinais utilizando dispositivos FPGA e microcontroladores, avaliados com o auxílio de um gerador de sinais de alta qualidade, apresentaram bom desempenho. O implementado com o microcontrolador PIC 18 apresentou resolução de 0,1 Hz e o com dispositivos FPGA de 0,3 Hz. A resolução obtida com dispositivos FPGA resultou relativamente baixa em decorrência de se ter utilizado um dispositivo com baixa capacidade de gravação.

Utilizando-se dispositivos FPGA pode-se fazer processamento paralelo, o que possibilita que os sinais sejam processados muito mais rapidamente que com os microcontroladores. Em conseqüência haverá diminuição do tempo de resposta do sistema eletrônico, o que possibilitará a utilização do sensor a corda vibrante em medições dinâmicas.

Foi realizada uma simulação computacional da resposta do sensor na qual se mostra que, na faixa de freqüência utilizada, sua resposta é linear.

A construção do sensor é relativamente simples, mas exige-se precisão na sua construção mecânica.

A faixa dinâmica do sensor construído é de 400 N. Esta faixa é adequada tanto para o dinamômetro como para a muleta, desde que o paciente não pese mais que 81 kg. Deve-se ressaltar que este foi o primeiro trabalho no qual se utilizou o sensor a corda vibrante em aplicações biomédicas, onde as forças medidas são de baixa intensidade.

Os resultados experimentais evidenciaram a resposta linear do sensor na faixa mencionada. A precisão é muito boa, sendo superior a 99,9%. Igualmente a resolução medida, de 1 N, é excelente.

A histerese de 6,5%, observada quando utilizou-se o PIC 18, pode ser considerada alta. Este problema poderá ser minimizado se a construção do sensor for realizada com técnicas de mecânica de precisão.

Observou-se uma discrepância inferior a 5 N quando se comparou as forças exercidas pelas mãos de indivíduos, medidas simultaneamente com o dinamômetro com sensor a corda vibrante e com um dinamômetro biomédico, com extensômetros metálicos. Esta discrepância pode ser atribuída ao sistema mecânico implementado para o paciente aplicar força simultaneamente nos dois dinamômetros e também às características construtivas do sensor a corda vibrante.

A tecnologia desenvolvida neste trabalho poderá ser muito útil para o desenvolvimento de sensores a corda vibrante para outras aplicações como, por exemplo, no monitoramento de forças e deformações em grandes estruturas. No Brasil, pelo que é de nosso conhecimento, não se fabrica sensores a corda vibrante. Há uma demanda grande para este tipo de sensor, principalmente para o monitoramento de esforços mecânicos em barragens

unesp

de usinas hidroelétricas.

A experiência adquirida com este projeto, nas implementações do sensor a corda vibrante e dos circuitos de condicionamento de sinais, será muito importante para o desenvolvimento de outros trabalhos de instrumentação eletrônica a serem desenvolvidos no Laboratório de Sensores do Departamento de Engenharia Elétrica, da UNESP, Campus de Ilha Solteira.

Como sugestão para trabalhos futuros sugere-se:

a) aprimoramento do projeto mecânico e da construção do sensor com a implementação do circuito de condicionamento de sinais utilizando dispositivos FPGA com maior capacidade de gravação;

b) a implementação de sensores a corda vibrante para monitoramento de esforços e deformações em grandes estruturas.

6. Bibliografia

[01] Catálogo 71.001E/88.09 da Maihak - Vibrating-Wire Principle de Measurement MSD.

[02] PALLÁS-ARENY, R. and WEBSTER, J. G., "Sensors and Signal Conditioning"; John Willey & Sons, Inc, 1991.

[03] FIALHO, J. Lobo; "Análise Experimental de Tensões; Laboratório Nacional de Engenharia Civil"; Lisboa:1969.

[04] WARRIOR, Jay, "Smart Sensors Networks of the Future", Sensors Magazine, Março de 1997. Site: http://www.sensorsmag.com.

[05] JOHNSON, Robert N., "Building Plug-and-Play Networked Smart Transducers", Sensors Magazine: Outubor de 1997. Site: http://www.sensorsmag.com.

[06] LEE, Kang, Schneeman, Richard, "Distributed Measurement and Control Based on the IEEE 1451 Smart Transducer Interface Standards", Instrumentation and Measurement Technical Conference 1999, Venice, Italy, May 24-26, 1999.

[07] SILVA, T. I., Carvalho, A. A., Leister, D. D., Silva, A. C. R., "Desenvolvimento de um Sensor à Corda Vibrante para Aplicações em Engenharia de Reabilitação", II Congresso Latino-americano de Engenharia Biomédica, Havana, 23 a 25 de Maio de 2001, Cuba.

[08] NACHATIGAL; Chester L.; "Instrumentation and Control: Fundamentals and Applications"; Wiley-Interscience: 1990.

[09] FENG, M. Q., SUZUKI, H. and YOKOI, I., "Development of Optical Sensing Systems for Smart Civil Infrastructure", Smart Mater. Struct, vol. 4, 1995, pp. A114-A120.

unesp

[10] OWERS-BRADLEY, J. R. et. al., "Damping of a Vibrating Wire in Spin Polarized Liquid 3He-4He Mixtures", Journal of Low Temperature Physics, vol. 110, nos. I/2, 1998, pp. 321-326.

[11] SETO, Willian W.; "Vibrações Mecânicas"; Trad. Negrão, Nege e Silva, Fernando V. H.; Barueri: Mcgraw-Hill do Brasil, 1979.

[12] THOMSON, Willian T.; "Teoria da Vibração com Aplicações"; tradução Sigaud, Cássio; Rio de Janeiro: Interciência, 1978.

[13] TIMOSHENKO, S., "Resistência dos Materiais", Rio de Janeiro: Livros Técnicos e Científicos,1978.

[14] HALLIDAY, D.; RESNICK, R.; WALTER, J.; "Fundamentos de Física"; Tradução Sotero, D. e Constamilan, G.; Vol. 3, 3ª ed.; Rio de Janeiro: Livros Técnicos e Científicos, 1996.

[15] ASTI, G. and SOLZI, M., "Vibrating Wire Magnetic Susceptometer", Rer. Sci. Instrum., Vol. 67, no. 10, Outubro de 1996, pp. 3543-3352.

[16] BASTOS, João Pedro Assumpção, "Eeltromagnetismo e Cálculo de Campos", 3ª Ed.Florrianopolis: Ed. da UFSC, 1996.

[17] PERTENCE JR., Antonio, "Amplificadores Operacionais e Filtros Ativos, Teoria, Projetos, Aplicações e Laboratório, 2ª Edição, São Paulo: McGraw Hill, 1988.

[18] http://www.burr-brown.com, site da Burr-Bronw.

[19] TERROSO, Anderson R., "Dispositivo Lógico Programável (FPGA), Linguagem de Descrição de Hardware (VHDL)", Faculdade de Engenharia: PUCRS, Abril de 2000.

[20] MAX+PLUS II, Programmable Logic Development Sytem, AHDL, Altera Corporation: Sao Jose, Version 6.0, November/1995.

[21] MARGINOT, Serge, "Evaluation Criteria of HDLs: VHDL Composed to Verilog, VDL/I & M", 0-8186-2780, 1992 IEEE, pp. 746-751.

cm 1 2 3 4 5 6 7 unesp 1 12 13 14 15 16 17 18

[22] SILVA, Vidal P., "Aplicações Práticas do Microcontrolador 8051", São Paulo: Érica, 1994.

[23] http://www.microchip.com, Site da Microchip.

[24] HIBBERLER, R. C., "Resistência dos Materiais", Rio de janeiro: Livros Técnicos e Científicos, 2000.

[25] SILVA, J. G., CARVALHO, A. A., RODRIGUES, R. O., "Desenvolvimento de um Dinamômetro para Avaliações Clínicas Funcionais das Mãos" IBERDISCAP 2000, Madri, Outubro de 20001.

APÊNDICES

APÊNDICE 1

A1. DESCRIÇÃO DE HARDWARE DO FPGA

A3.1. Introdução

Como visto na seção 3.4 do presente trabalho, o circuito em FPGA foi implementado em dois módulos, contendo um dispositivo FLEX cada um. Conforme a Figura A.1, a qual é idêntica a Figura 3.15, tem-se os blocos gravados em cada dispositivo FLEX. Os dois circuitos implementados para cada FLEX foi **frequec_aqui1** e **mediatotal**.

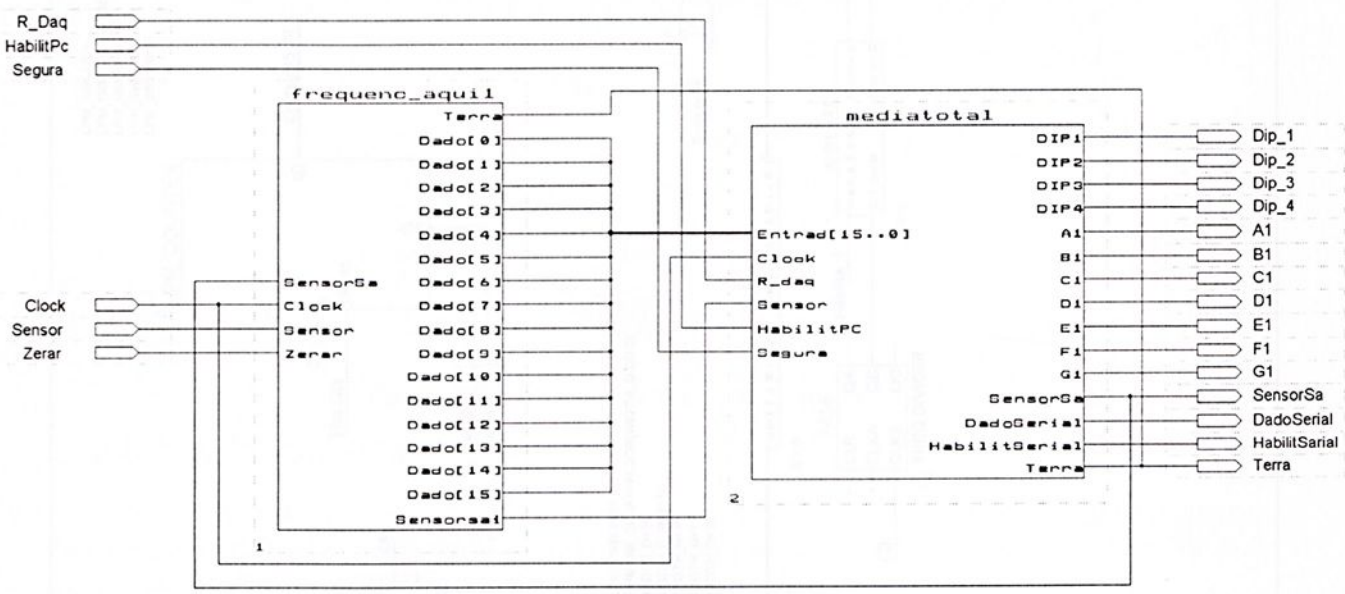


Figura A.1 – Frequencímetro implementado em FPGA.

Neste Apêndice 1 estão descritos todos os blocos implementados, bem como a função de cada um, suas características, entradas e saídas.

Por se ter duas FLEX, chama-se **Primeira FLEX** o componente que contém o circuito **frequec_aqui1**, e **Segunda FLEX** o componente que contém o circuito **mediatotal**.

Os nomes atribuídos a cada circuito foram atribuídos pelo autor, com relação a função do bloco.

A3.2. Blocos Gravados na Primeira FLEX

Os blocos da Figura A.2, onde mostra o circuito **frequec_aqui1**, e suas especificações estão descritos nos próximos itens. Foram utilizados componentes parametrizáveis, que são megafunções da biblioteca do ambiente Altera.

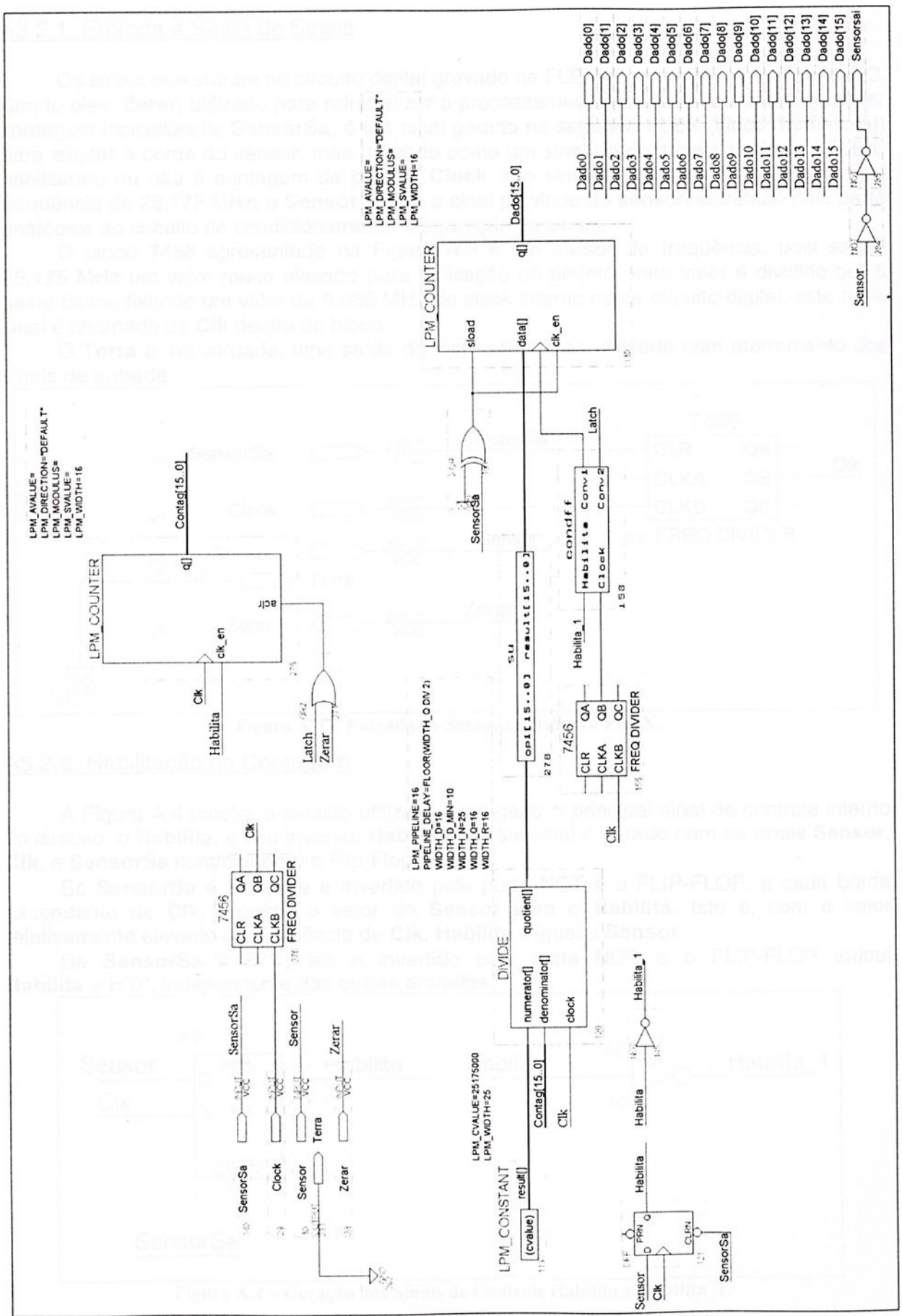


Figura A.2 – Circuito frequec_aqui1 gravado na primeira FLEX.

A3.2.1. Entrada e Saída de Sinais

Os sinais que entram no circuito digital gravado na FLEX são mostrados na Figura A.3. Sendo eles: **Zerar**, utilizado para reinicializar o processamento do sinal, zerando a qualquer contagem incicailizada; **SensorSa**, é um sinal gerado na segunda FLEX (bloco **mediatotal**) para excitar a corda do sensor, mas utilizado como um sinal de controle na primeira FLEX, habilitando ou não a contagem de pulsos; **Clock**, é o sinal de base de tempo, sendo de freqüência de 25,175 MHz; e **Sensor**, que é o sinal provindo do sensor, já tratado pela parte analógica do circuito de condicionamento e aquisição de sinais.

O bloco 7456 apresentado na Figura A.3 é um divisor de freqüência, pois sendo 25,175 MHz um valor muito elevado para aplicação no projeto, este valor é dividido por 5 neste bloco, fixando um valor de 5,035 MHz de clock interno neste circuito digital, este novo sinal é chamado de **Clk** dentro do bloco.

O **Terra** é, na verdade, uma saída do dispositivo, mas utilizada com aterramento dos sinais de entrada.

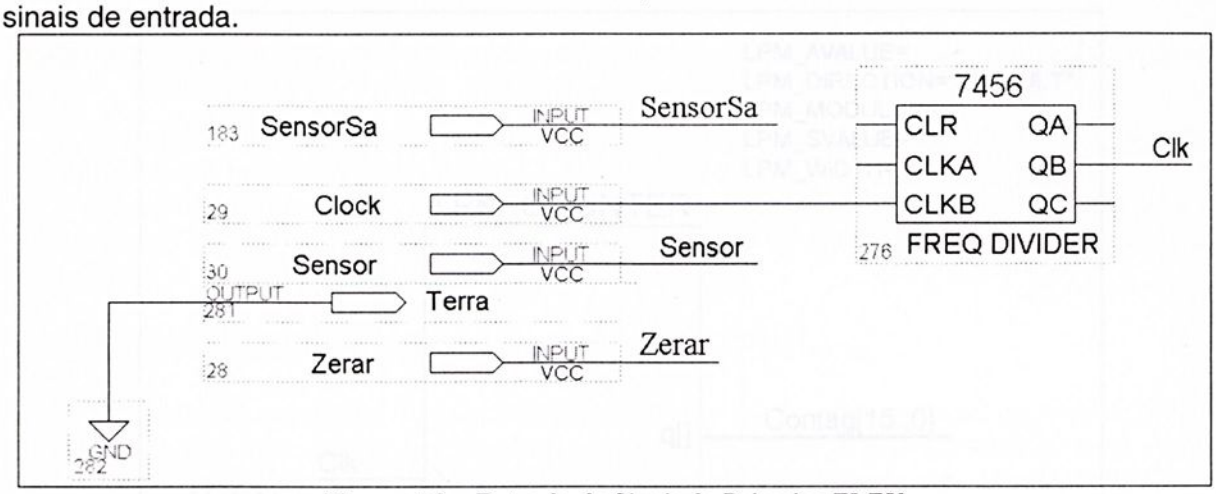


Figura A.3 – Entrada de Sinais da Primeira FLEX.

A3.2.2. Habilitação da Contagem

A Figura A.4 mostra o circuito utilizado para gerar o principal sinal de controle interno do circuito, o **Habilita**, e seu inverso, **Habilita_1**. Este sinal é gerado com os sinais **Sensor**, **Clk**, e **SensorSa** controlando um Flip-Flop D.

Se **SensorSa** é B"0", ele é invertido pela porta NOT e o FLIP-FLOP, a cada borda ascendente de **Clk**, repassa o valor do **Sensor** para o **Habilita**. Isto é, com o valor relativamente elevado da freqüência de **Clk**, **Habilita** é igual a **Sensor**.

Se **SensorSa** é B"1", ele é invertido pela porta NOT e o FLIP-FLOP atribui **Habilita** = B"0", independente das outras entradas.

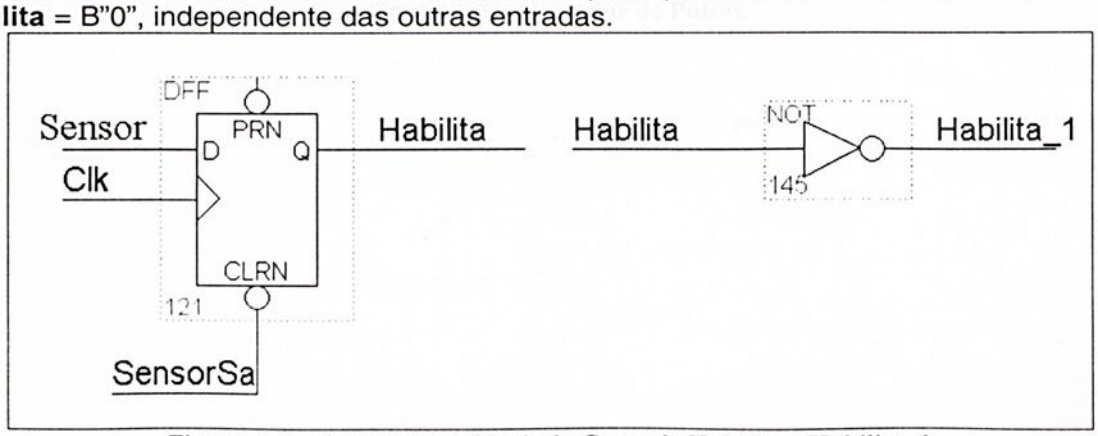


Figura A.4 – Geração dos Sinais de Controle Habilita e Habilita_1.

A3.2.3. Contador dos pulsos - LPM_COUNTER

A Figura A.5 mostra o contador de pulsos. Neste bloco é realizada a contagem de pulsos, através do janelamento efetuado pelo sinal do sensor, em cima da base de tempo, proveniente do clock de 5,035 MHz.

Utilizou-se um contador parametrizável, cujos parâmetros ajustáveis facilitam a descrição do projeto. Nesta megafunção, as entradas e as saídas podem ser suprimidas, utilizando somente as mais convenientes. A Figura A.6 mostra a tabela da verdade com todas as entradas e saídas possíveis do bloco.

No bloco utilizado estão presentes as entradas: **Clk** que é a base de tempo, **Habilita** que corresponde ao sinal do sensor, cuja freqüência corresponde à força aplicada sobre o sensor, este entra no pino **clk_en** do contador. O pino **aclr** é ligado a uma porta OR, em cujas entradas estão ligados **Zerar** e **Latch**; isto é, se um deste dois sinais estiver em B"1", **aclr** receberá B"1" zerando a contagem. O sinal **Latch** será visto posteriormente.

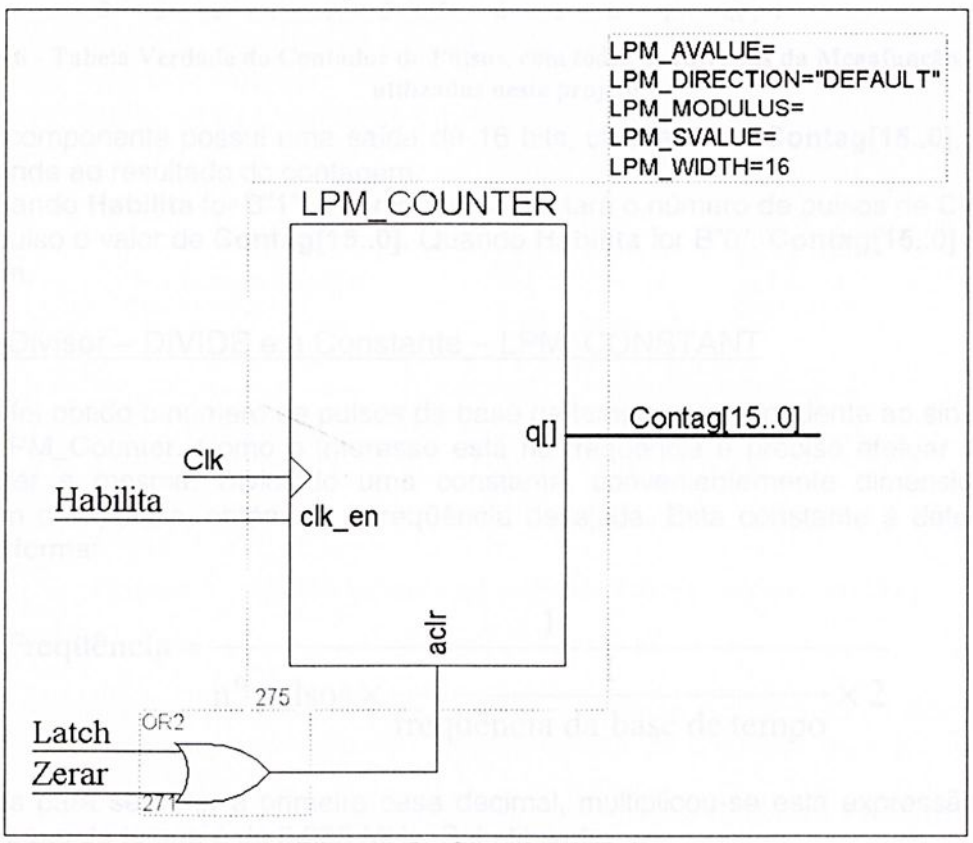


Figura A.5 – Contador de Pulsos.

Inputs										Outputs
aclr		aload		clock		sset		cnt_en		
	aset		clk_en		sclr		sload		updown	q[LPM_WIDTH-1..0]
1	X	X	X	X	X	X	X	X	X	000...
0	1	X	X	X	X	X	X	X	X	111...
0	1	X	X	X	X	X	X	X	X	LPM_AVALUE
0	0	1	X	X	X	X	X	X	X	data[]
0	0	0	0	X	X	X	X	X	X	q[]
0	0	0	1	∫	1	X	X	X	X	000...
0	0	0	1	∫	0	1	X	X	X	111...
0	0	0	1	∫	0	1	X	X	X	LPM_SVALUE
0	0	0	1	∫	0	0	0	0	0	q[]
0	0	0	1	∫	0	0	1	X	X	data[]
0	0	0	1	∫	0	0	0	1	1	q[]+1
0	0	0	1	∫	0	0	0	1	0	q[]-1

Figura A.6 - Tabela Verdade do Contador de Pulsos, com todas as entradas da Megafunção, algumas não utilizadas neste projeto.

O componente possui uma saída de 16 bits, chamada de **Contag[15..0]**, e seu valor corresponde ao resultado do contagem.

Quando **Habilita** for B"1", o componente contará o número de pulsos de **Clk**, alterando a cada pulso o valor de **Contag[15..0]**. Quando **Habilita** for B"0", **Contag[15..0]** armazena a contagem.

A3.2.4. Divisor – DIVIDE e a Constante – LPM_CONSTANT

Já foi obtido o número de pulsos da base de tempo correspondente ao sinal do sensor com o LPM_Counter. Como o interesse está na freqüência é preciso efetuar uma divisão para obter a mesma. Dividindo uma constante, convenientemente dimensionada, pela contagem dos pulsos, obtém-se a freqüência desejada. Esta constante é determinada da seguinte forma:

$$\text{Freqüência} = \frac{1}{\text{nº pulsos} \times \dfrac{1}{\text{freqüência da base de tempo}} \times 2} \qquad \text{(A1)}$$

Mas para se obter a primeira casa decimal, multiplicou-se esta expressão por 10. O valor da base de tempo é de 5,035 MHz. Substituindo:

$$10 \cdot \text{Freqüência} = \frac{10}{\text{nº pulsos} \times \dfrac{1}{5{,}035 \times 10^6} \times 2} \qquad \text{(A2)}$$

$$10 \cdot \text{Freqüência} = \frac{25{,}175 \times 10^6}{\text{nº pulsos}} \qquad \text{(A3)}$$

Para descrever a expressão (A3) em um circuito digital, utilizou-se de duas megafunções disponíveis no Ambiente de Projeto Altera, apresentadas na Figura A.7.

A megafunção utilizada, **DIVIDE**, possui parâmetros a serem configurados. **LPM_PIPELINE** e **LPM_DELAY** são constantes que determinam o número de ciclos de

clock necessários para o valor da divisão se estabilize na saída. **WIDTH_N**, **WIDTH_D**, **WIDTH_Q**, **WIDTH_R** são respectivamente o tamanho, em bits, do numerador, do denominador, do quociente, e do resto da divisão. O número de bits relaciona o valor máximo que se consegue representar, 16 bits representa no máximo 65.536 em decimal. **WIDTH_D_MIN** é o valor mínimo do denominador.

Neste projeto utilizou-se 25 bits para o numerador, 16 bits para o denominador e para o quociente, não se utilizando o resto.

O **LPM_CONSTANT** fornece o valor de uma constante, que conforme (15), deve ter o valor de $25{,}175 \times 10^6$; este valor necessita de 25 bits para ser representado na forma binária. Estes são os dois parâmetros a serem configurados deste componente.

O valor do denominador do **DIVIDE** é fornecido pelo **Contag[15..0]**, já o numerador por **LPM_CONSTANT**, a entrada **Clock** é fornecido pelo clock interno **Clk**. O **quotient[ ]** é conectado diretamente ao bloco **Ajuste**, que será visto em seguida. A cada alteração no valor de **Contag[15..0]** o valor da divisão se altera. Para a estabilização do valor da divisão são necessários 16 ciclos de **Clk** após a entrada do valor no denominador.

No decorrer do desenvolvimento do projeto, chegou-se a implementação de um divisor, mas não se alcançou uma velocidade semelhante ao apresentado por esta megafunção. Seu único inconveniente é a utilização de muitas células do dispositivo FPGA.

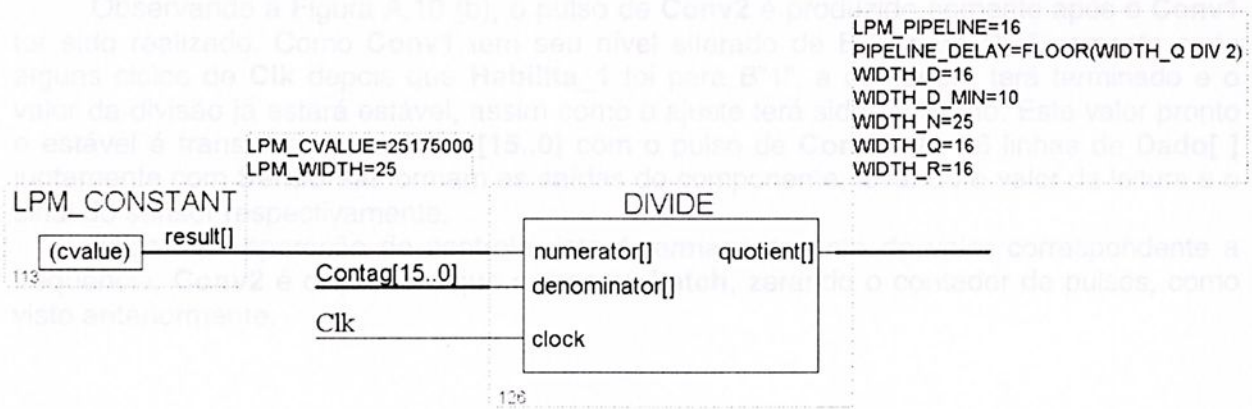
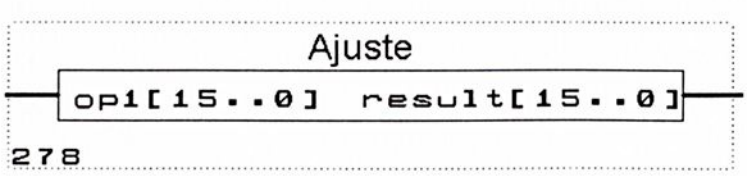


Figura A.7 – Circuito utilizado para realizar a divisão expressa em (A3).

A3.2.5. Ajuste

Através de experimentos práticos, foi verificado um erro constante de leitura, em torno de 2 unidades do valor da freqüência, isto representa 32 unidades do valor resultante da divisão.

Diante deste erro constante, optou-se por implementar um bloco de ajuste, que corrije este erro. A Figura A.8 (a) mostra o símbolo do componente apresentado na Figura A.2, e em A.8 (b) a descrição do circuito em linguagem AHDL.



(a)

```
SUBDESIGN ajuste
  ( op1[15..0] : INPUT;
    result[15..0] : OUTPUT;)
BEGIN
  result[]=op1[] + 32;
END;
```

(b)

Figura A.8 – (a) O símbolo do componente; (b) A descrição do seu circuito digital.

A3.2.6. Latch e Controle da Saída

A Figura A.9 mostra o circuito do Sina Lactch e do controle dos dados de saída, implementado com um **LPM_COUNTER**, um divisor 7456, o bloco **condff**, e uma porta

XOR.

O sinal interno Latch, que é responsável pela controle do fluxo de dados no circuito digital por inteiro. Na verdade, ele controla a saída de dados, não permitindo que as alterações da contagem a cada ciclo de **Clk**, e também do **quotient[ ]**, reflita na saída, pois durante a contagem, estes dados não corresponde a real leitura do sensor, sendo valores intermediários. e também a contagem dos pulsos, como visto anteriormente.

O bloco de **condff** é formado por duas entradas e duas saídas. A entrada **Clk** é o mesmo clock de todo circuito, mas aqui ele atravessa um divisor de freqüência 7456, tornando-se um sinal de aproximadamente 1 MHz. O sinal **Habilita_1**, como visto anteriormente, é o inverso do **Habilita**, consequentemente, o inverso do sinal do sensor. As saídas do bloco **controle** são o **Conv1** e o **Conv2**, sendo este último chamado de **Latch**, que é utilizado para o controle do contador de pulsos, como já visto O circuito interno de **condff** é mostrado na Figura A.10 (a).

Este bloco simplesmente cria dois pulsos, **Conv1** e **Conv2**, quando **Habilita_1** vai de B"0" para B"1", conforme é mostrado na Figura A.10 (b). Esta ação é executada por meio FLIP-FLOPs em cascata, que e em conjunto com uma porta XOR, cujas as entradas são **SensoeSa** e Vcc, produz formas de onda desejadas para o controle de **LPM_COUNTER**., realizando o controle da saída de dados.

Observando a Figura A.10 (b), o pulso de **Conv2** é produzido somente após o **Conv1** ter sido realizado. Como **Conv1** tem seu nível alterado de B"0" para B"1" somente após alguns ciclos de **Clk** depois que **Habilita_1** foi para B"1", a contagem terá terminado e o valor da divisão já estará estável, assim como o ajuste terá sido realizado. Este valor pronto e estável é transferido para **Dado[15..0]** com o pulso de **Conv1**. As 16 linhas de **Dado[ ]** juntamente com **Sensorsai** formam as saídas do componente, levando o valor da leitura e o sinal do sensor respectivamente.

Após esta operação de controle, isto é, armazenamento do valor correspondente a freqüência, **Conv2** é disparado, que o mesmo **Latch**, zerando o contador de pulsos, como visto anteriormente.

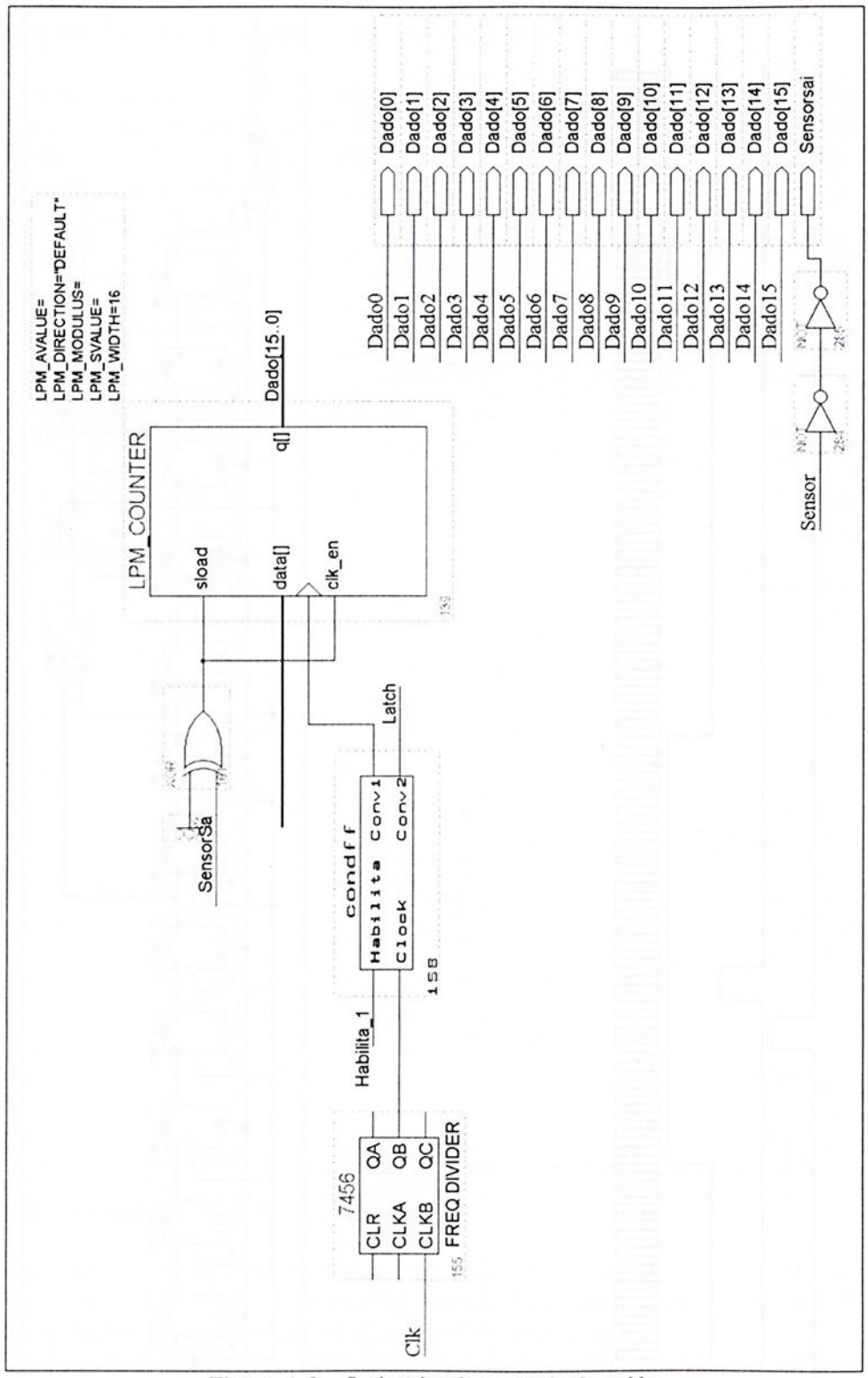
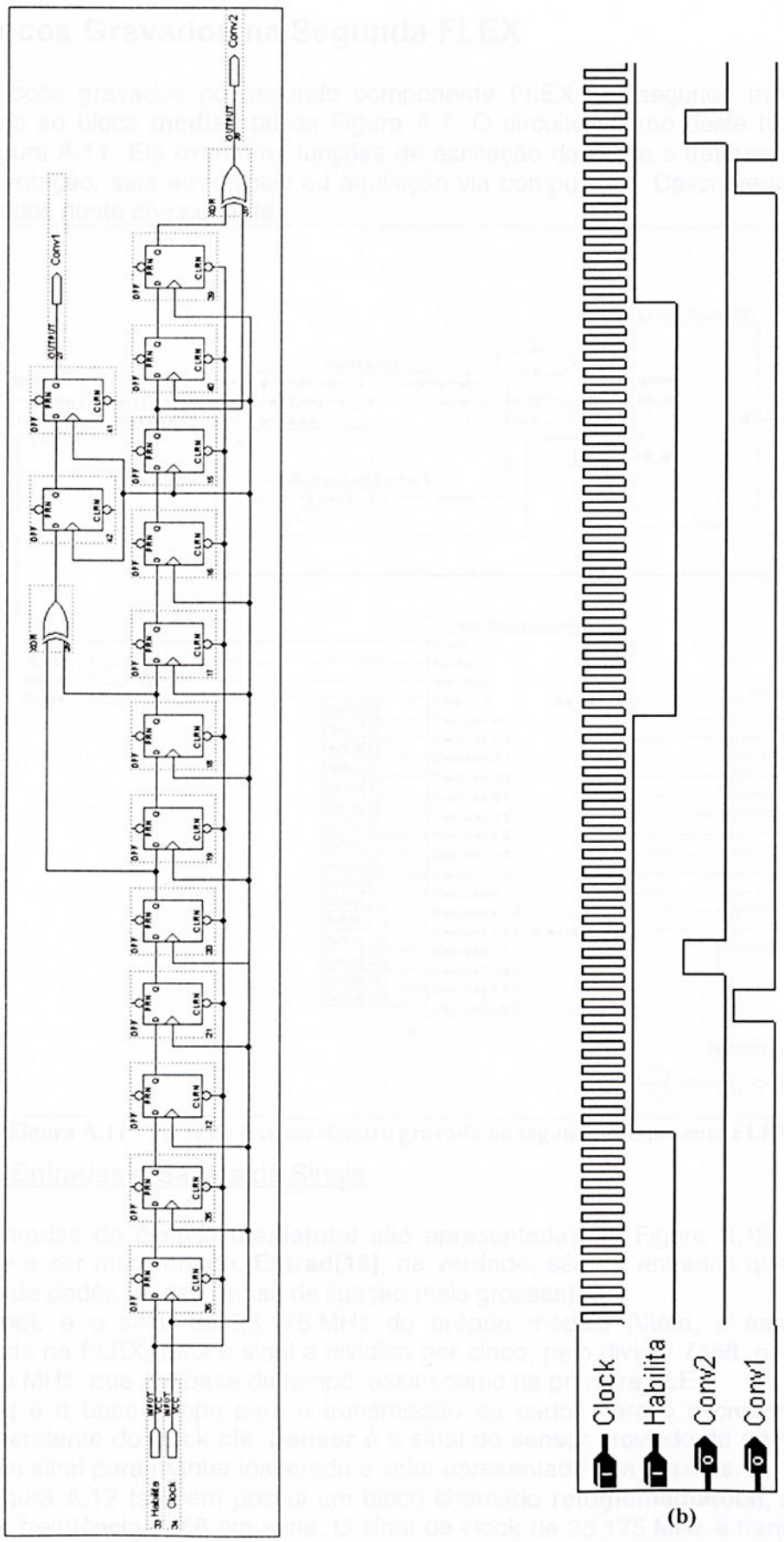


Figura A.9 – O circuito de controle da saída.



(a) (b)

Figura A.10 – (a) Circuito interno do Bloco condff; (b) as formas de onda do bloco.

A3.3. Blocos Gravados na Segunda FLEX

Os blocos gravados no segundo componente FLEX, do segundo módulo, são os pertencentes ao bloco **mediatotal** da Figura A.1. O circuito interno deste bloco pode ser visto na Figura A.11. Ele exerce as funções de excitação da corda e tratamento dos dados para apresentação, seja em display ou aquisição via computador. Descreve-se a seguir os blocos contidos neste componente.

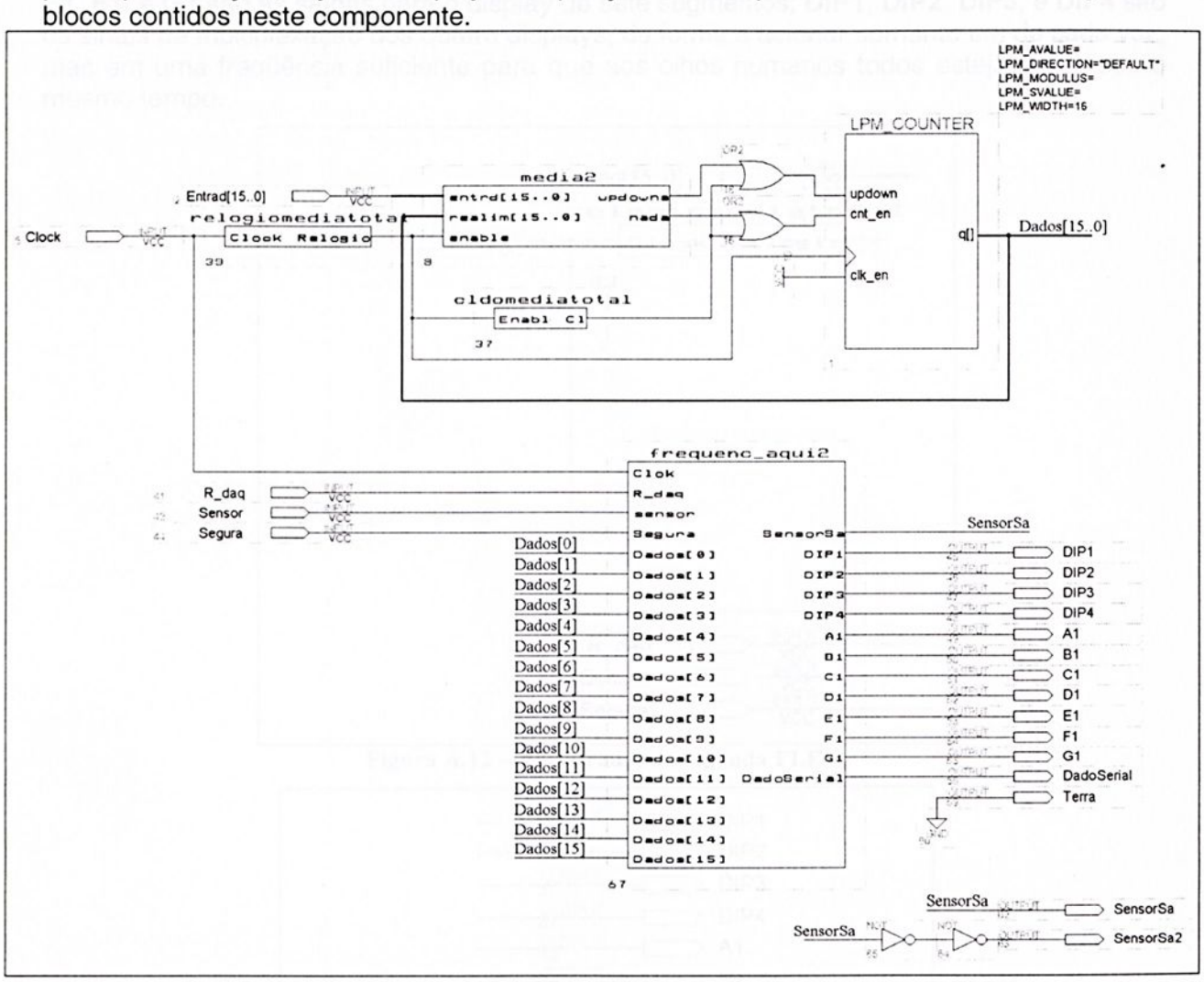


Figura A.11 - Parte do frequencímetro gravado no segundo componente FLEX.

A3.3.1. As Entradas e Saídas de Sinais

As entradas do circuito **mediatotal** são apresentadas na Figura A.12. Um aspecto interessante a ser mencionado, **Entrad[15]**, na verdade, são 16 entradas que formam um barramento de dados (possui linhas de ligação mais grossas).

O **Clock** é o sinal de 25.175 MHz do próprio módulo Altera, e assim como foi implementado na FLEX, aqui o sinal é dividido por cinco, pelo divisor 7456, gerando o sinal **clk** de 5,035 MHz, que é a base de tempo, assim como na primeira FLEX.

R_daq é a base tempo para a transmissão de dados para o microcomputador, de forma independente do clock **clk**. **Sensor** é o sinal do sensor provindo do primeiro módulo. **Segura** é um sinal para manter inalterado o valor apresentado nos displays.

Na Figura A.12 também possui um bloco chamado **relogiomediatotal**, que possui 7 divisores de freqüência 7456 em série. O sinal de clock de 25,175 MHz é transformado em um sinal de 322,24 Hz, que será o sinal de Enable do bloco **media2**, e do **LMP_Counter** . Esta baixa freqüência é necessária para acionar o clock do contador e os Latchs internos de

media2.

As saídas, apresentadas na Figura A.13, são: **SensorSa** e **SensorSa2** que é o sinal de excitação da corda e o sinal de controle que vai para a Primeira FLEX. As duas portas NOT colocadas na saída **SensorSa2** foram necessárias porque o ambiente MAX + Plus II não aceita que o mesmo sinal tenha duas saídas.

Dadoserial é a saída de dados na forma serial, no padrão RS-232; **A1**, **B1**, **C1**, **D1**, **E1**, **F1**, e **G1** são as saídas para o display de sete segmentos; **DIP1**, **DIP2**, **DIP3**, e **DIP4** são os sinais de multiplexação dos quatro displays, de forma a acionar somente um de cada vez, mas em uma freqüência suficiente para que aos olhos humanos todos estejam ligados ao mesmo tempo.

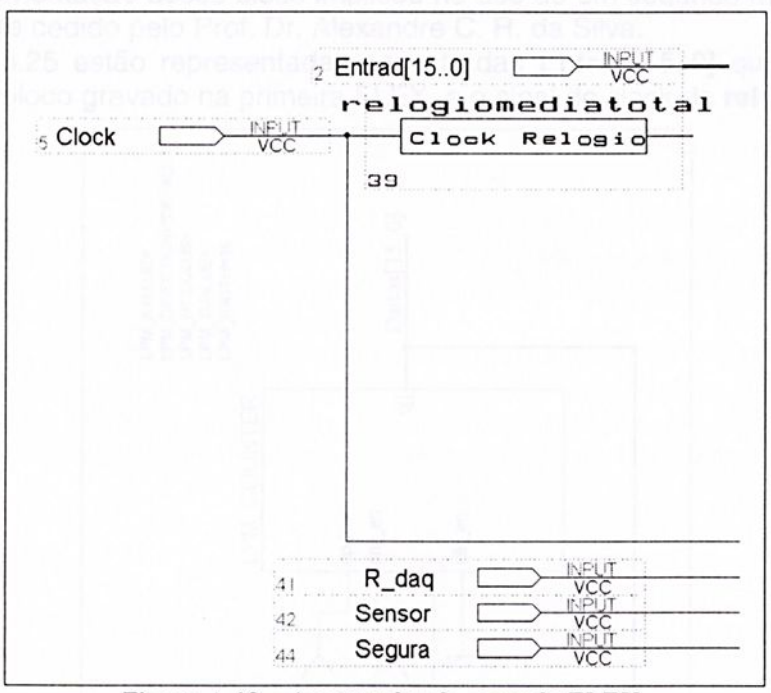


Figura A.12 – As entradas da segunda FLEX.

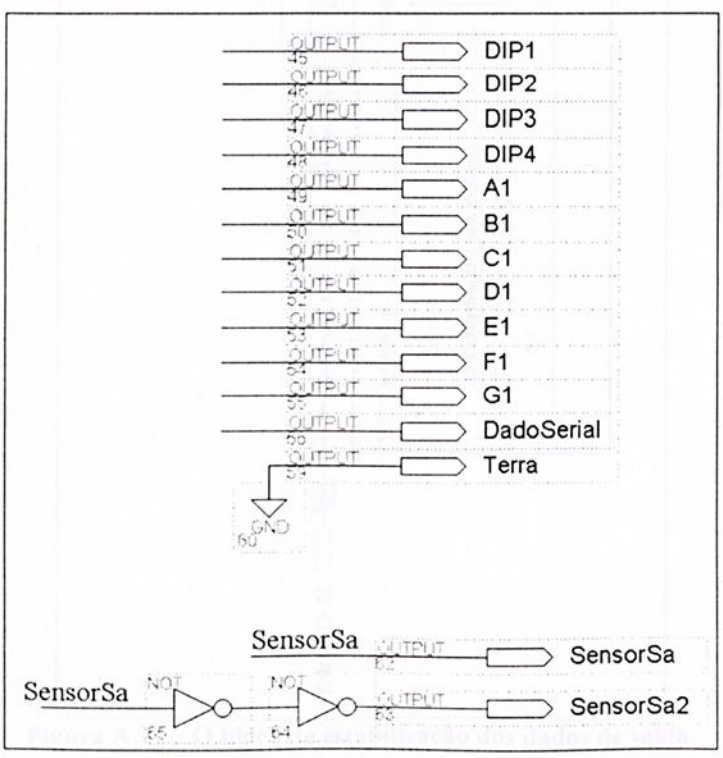


Figura A.13 – As saídas da segunda FLEX.

A3.3.2. Blocos LPM_COUNTER e media2

Os blocos implementados para a realização da média etapa são mostrados na Figura A.14. Sua função é melhorar a apresentação dos resultados, apresentando a média de 16 leituras consecutivas. Isto representa uma maior latência na mudança do valor apresentado nos displays, aumentando o tempo de resposta.

Este bloco foi implementado para a realização de medidas de maior resolução, pois sem o mesmo a leitura do dígito menos significativo era imprecisa, devido a sua grande variação. A implementação desse bloco implicou no uso de um segundo módulo didático da Altera, gentilmente cedido pelo Prof. Dr. Alexandre C. R. da Silva.

Na Figura 3.25 estão representadas as entradas **Entrad[15..0]** que recebe a saída **Dados[15..0]** do bloco gravado na primeira FLEX, e o sinal de clock de **relogiomedtotal**.

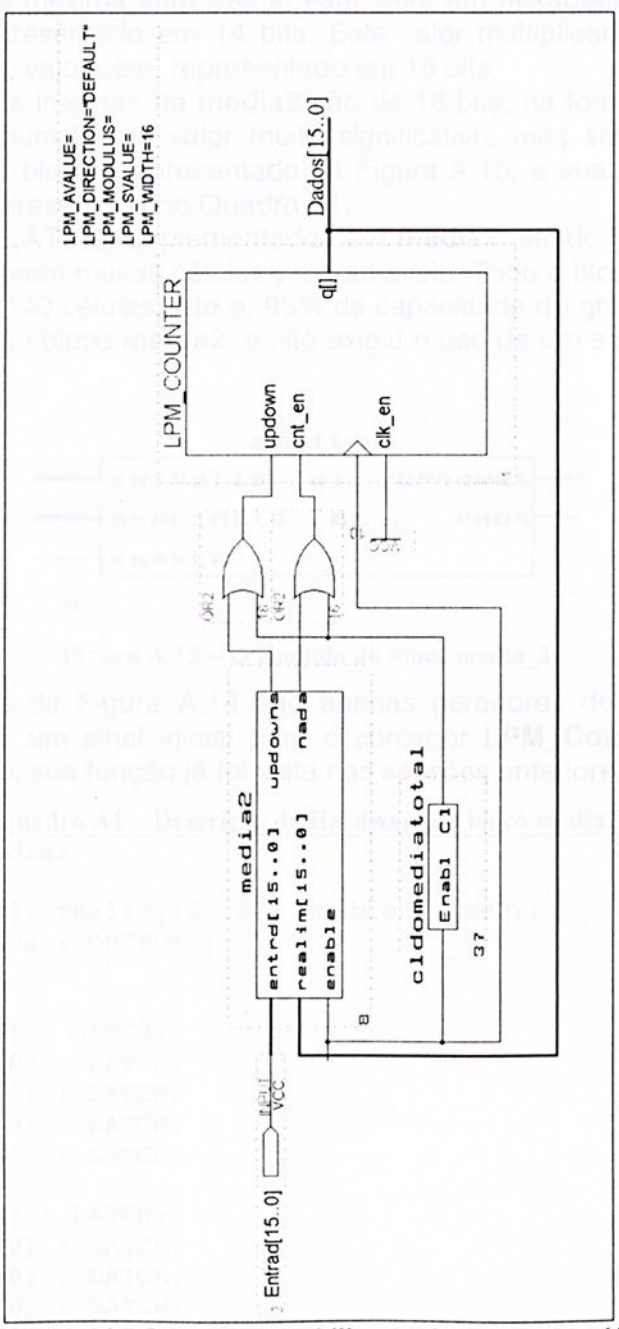


Figura A.14 – O bloco de estabilização dos dados de saída.

O bloco principal do circuito é o **media2**, pois ele é responsável pela leitura de 16 resultados consecutivos, somando-os e dividindo pelo número de leituras.

O princípio desta divisão está alicerçado em um fundamento de circuitos digitais e, em uma característica do valor resultante da leitura, explicados a seguir.

Em representações binárias, realizando a transladação dos 15 bits mais significativos para as 15 posições menos significativas, e zerando o décimo sexto bit, resulta na metade do valor inicial, pois se trata de números de base 2. Assim, com o mesmo número de 16 bits, com a transladação dos 14 bits mais significativos para as 14 posições menos significativas, e zerando os dois bits mais significativos, divide-se o valor inicial por quatro.

Este princípio de divisão por múltiplos de 2 é utilizado no circuito **media2**, que soma 16 leituras em quatro diferentes variáveis, cujos valores divididos por 4, individualmente, e armazenados em outras 4 variáveis. Estas são somadas em uma última variável, que é dividida por 4 novamente, cujo resultado é a saída do circuito.

O valor da leitura máxima será 999,9, este valor em hexadecimal será igual a 270F, que por sua vez é representado em 14 bits. Este valor multiplicado por quatro é igual a 9C3C, em hexadecimal, valor este, representado em 16 bits.

Como as variáveis internas de **media2** são de 16 bits, na forma de divisão proposta, não será perdido nenhum bit de valor muito significativo, mas somente os dois valores menos significativos. O bloco é apresentado na Figura A.15, e sua descrição de hardware (sua programação) é apresentada no Quadro A1.

Ao todo, são 26 **LATCH**s implementados em **media2**, sendo todos de 16 bits. Estas variáveis internas requerem muitas células programáveis. Todo o bloco gravado na segunda FLEX ocupa mais de 1140 células, isto é, 98% da capacidade de gravação do componente, devido principalmente ao bloco **media2**, e isto exigiu o uso de um segundo módulo didático da Altera.

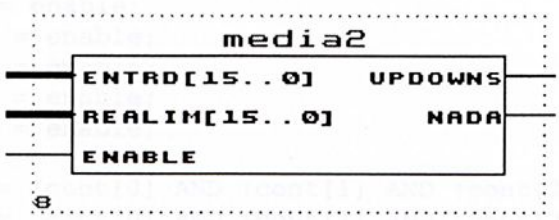


Figura A.15 – O símbolo do bloco media_2.

Os outros blocos da Figura A.14 são apenas geradores de sinais de controle. O **cldomediatotal** produz um sinal inicial para o contador **LPM_Counter**, no início de sus operação, já este último, sua função já foi vista nas sessões anteriores.

Quadro A1 – Descrição de Hardware do bloco media2.

```
SUBDESIGN media2

(entrd[15..0], realim[15..0], enable : INPUT;
 updowns, nada : OUTPUT;)

VARIABLE
 somat[15..0] : LATCH;
 soma1a[15..0] : LATCH;
 soma1b[15..0] : LATCH;
 soma1c[15..0] : LATCH;
 soma1d[15..0] : LATCH;

 medbt[15..0] : LATCH;
 medb1a[15..0] : LATCH;
 medb1b[15..0] : LATCH;
 medb1c[15..0] : LATCH;
 medb1d[15..0] : LATCH;
```

```
num_0[15..0] : LATCH;
num_1[15..0] : LATCH;
num_2[15..0] : LATCH;
num_3[15..0] : LATCH;
num_4[15..0] : LATCH;
num_5[15..0] : LATCH;
num_6[15..0] : LATCH;
num_7[15..0] : LATCH;
num_8[15..0] : LATCH;
num_9[15..0] : LATCH;
num_A[15..0] : LATCH;
num_B[15..0] : LATCH;
num_C[15..0] : LATCH;
num_D[15..0] : LATCH;
num_E[15..0] : LATCH;
num_F[15..0] : LATCH;

ver : NODE;
ver_2 : NODE;
cont[3..0] : DFF;

BEGIN
somat[].ena = enable;
soma1a[].ena = enable;
soma1b[].ena = enable;
soma1c[].ena = enable;
soma1d[].ena = enable;

medbt[].ena = enable;
medb1a[].ena = enable;
medb1b[].ena = enable;
medb1c[].ena = enable;
medb1d[].ena = enable;

num_0[].ena = !cont[0] AND !cont[1] AND !cont[2] AND !cont[3];
num_1[].ena =  cont[0] AND !cont[1] AND !cont[2] AND !cont[3];
num_2[].ena = !cont[0] AND  cont[1] AND !cont[2] AND !cont[3];
num_3[].ena =  cont[0] AND  cont[1] AND !cont[2] AND !cont[3];
num_4[].ena = !cont[0] AND !cont[1] AND  cont[2] AND !cont[3];
num_5[].ena =  cont[0] AND !cont[1] AND  cont[2] AND !cont[3];
num_6[].ena = !cont[0] AND  cont[1] AND  cont[2] AND !cont[3];
num_7[].ena =  cont[0] AND  cont[1] AND  cont[2] AND !cont[3];
num_8[].ena = !cont[0] AND !cont[1] AND !cont[2] AND  cont[3];
num_9[].ena =  cont[0] AND !cont[1] AND !cont[2] AND  cont[3];
num_A[].ena = !cont[0] AND  cont[1] AND !cont[2] AND  cont[3];
num_B[].ena =  cont[0] AND  cont[1] AND !cont[2] AND  cont[3];
num_C[].ena = !cont[0] AND !cont[1] AND  cont[2] AND  cont[3];
num_D[].ena =  cont[0] AND !cont[1] AND  cont[2] AND  cont[3];
num_E[].ena = !cont[0] AND  cont[1] AND  cont[2] AND  cont[3];
num_F[].ena =  cont[0] AND  cont[1] AND  cont[2] AND  cont[3];

cont[].clk = enable;

cont[] = cont[] + 1;

num_0[] = entrd[];
num_1[] = entrd[];
num_2[] = entrd[];
num_3[] = entrd[];
num_4[] = entrd[];
```


cm 1 2 3 4 5 6 7 unesp 1 12 13 14 15 16 17 18

```
num_5[] = entrd[];
num_6[] = entrd[];
num_7[] = entrd[];
num_8[] = entrd[];
num_9[] = entrd[];
num_A[] = entrd[];
num_B[] = entrd[];
num_C[] = entrd[];
num_D[] = entrd[];
num_E[] = entrd[];
num_F[] = entrd[];
soma1a[] = num_0[] + num_1[] + num_2[] + num_3[];
soma1b[] = num_4[] + num_5[] + num_6[] + num_7[];
soma1c[] = num_8[] + num_9[] + num_A[] + num_B[];
soma1d[] = num_C[] + num_D[] + num_E[] + num_F[];

medb1a[13..0] = soma1a[15..2];
medb1a[14] = B"0";
medb1a[15] = B"0";

medb1b[13..0] = soma1b[15..2];
medb1b[14] = B"0";
medb1b[15] = B"0";

medb1c[13..0] = soma1c[15..2];
medb1c[14] = B"0";
medb1c[15] = B"0";

medb1d[13..0] = soma1d[15..2];
medb1d[14] = B"0";
medb1d[15] = B"0";

somat[] = medb1a[] + medb1b[] + medb1c[] + medb1d[];

medbt[13..0] = somat[15..2];
medbt[14] = B"0";
medbt[15] = B"0";

IF medbt[] > realim[] THEN
 ver = B"1";
 ver_2 = B"1";
END IF;
IF medbt[] < realim[] THEN
 ver = B"0";
 ver_2 = B"1";
END IF;
IF medbt[] == realim[] THEN
 ver = B"0";
 ver_2 = B"0";
END IF;


CASE VER IS
WHEN B"1" =>
 updowns = B"1";
WHEN B"0" =>
 updowns = B"0";
WHEN OTHERS =>
 updowns = B"0";
END CASE;
```


BCpIS - FEIS - UNESP

```
        CASE VER_2 IS
        WHEN B"1" => nada = B"1";
        WHEN B"0" => nada = B"0";
        END CASE;

        END;
```

A3.3.3. Apresentação dos resultados

A apresentação dos resultados é realizada pelo bloco **frequec_aqui2** da Figura A.11. Seu circuito interno é apresentado na Figura A.16. Cada um dos seus blocos é descrito a seguir.

Observando as Figuras A.11 e A.16, nota-se que a entrada **Clock** possui freqüência de 25,175 MHz, e é divido por cinco pelo bloco 7456.

Os sinais internos **Habilita** e **Habilita_1** são novamente gerados por um flip-flop, igual a Primeira FLEX.

O bloco **relogiomedtotal**, composto de 7 divisores 7456 em série é novamente utilizado aqui, com a mesma função.

R_daq, **Sensor**, **Segura**, **Dados[0]**, ..., **Dados[15]** são simplesmente repassados do bloco **mediatotal** para dentro do **frequec_aqui2**. Os outro blocos da figura são tem sua função explanada a seguir.

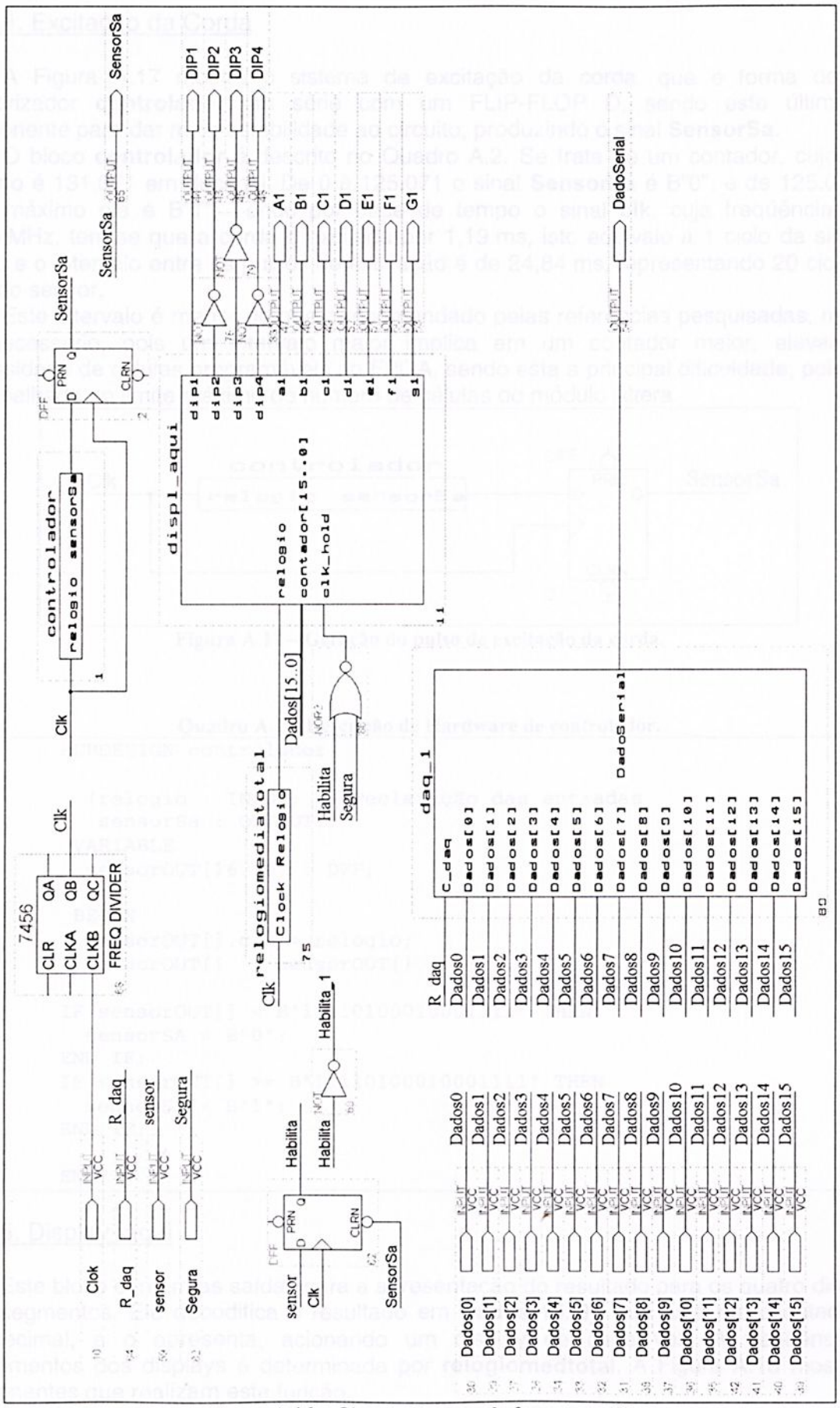


Figura A.16 – Circuito interno de frquec_aqui2.

A3.3.4. Excitação da Corda

A Figura A.17 mostra o sistema de excitação da corda, que é forma do pelo temporizador **controlador** em série com um FLIP-FLOP D, sendo este último um componente para dar mais estabilidade ao circuito, produzindo o sinal **SensorSa**.

O bloco **controlador** é descrito no Quadro A.2. Se trata de um contador, cujo valor máximo é 131.071 em decimal. De 0 à 125.071 o sinal **SensorSa** é B"0", e de 125.072 ao valor máximo ele é B"1". Tendo por base de tempo o sinal **clk**, cuja freqüência é de 5,035 MHz, tem-se que a corda é excitada por 1,19 ms, isto eqüivale a 1 ciclo da sinal da corda, e o intervalo entre os pulsos de excitação é de 24,84 ms, representando 20 ciclos do sinal do sensor.

Este intervalo é menor do que o recomendado pelas referências pesquisadas, mas se faz necessário, pois um intervalo maior implica em um contador maior, elevando a necessidade de células programáveis no FPGA, sendo esta a principal dificuldade, pois tem-se trabalhado no limite máximo do número de células do módulo Altera.

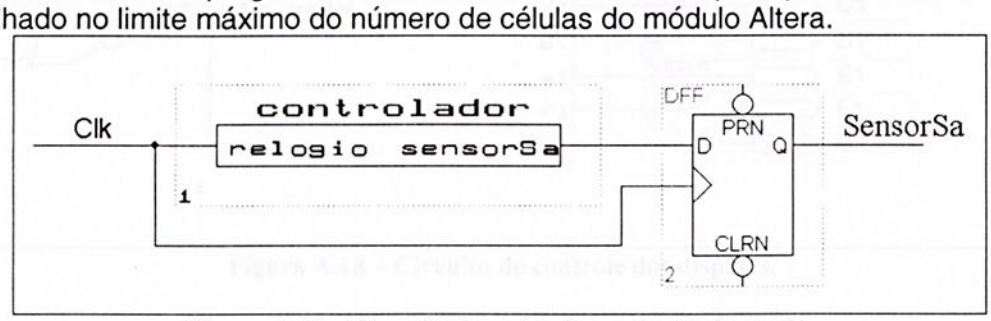


Figura A.17 – Geração do pulso de excitação da corda.

Quadro A.2 – Descrição de Hardware de controlador.

```
SUBDESIGN controlador

  (relogio : INPUT;  --Declaração das entradas
   sensorSa : OUTPUT;)
 VARIABLE
  sensorOUT[16..0] : DFF;

 BEGIN
  sensorOUT[].clk = relogio;
  sensorOUT[]  = sensorOUT[] + 1;

IF sensorOUT[] < B"11110100010001111" THEN
  sensorSA = B"0";
END IF;
IF sensorOUT[] >= B"11110100010001111" THEN
  sensorSA = B"1";
END IF;

END;
```

A3.3.5. Display_aqui

Este bloco contém as saídas para a apresentação do resultado para os quatro displays de 7 segmentos. Ele decodifica o resultado em **Dados[15..0]**, que está representado em hexadecimal, e o apresenta, acionando um display de cada vez. A freqüência de acionamentos dos displays é determinada por **relogiomedtotal**. A Figura A.18 mostra os componentes que realizam esta função.

O sinal **Habilita_1**, que é uma das entradas de **displ_aqui** permite a apresentação

somente quando a contagem de pulso estiver terminado, isto é, **Habilita_1** = B"1".

As saídas **A1**, **B1**, **C1**, **D1**, **E1**, **F1**, **G1**, **DIP1**, **DIP2**, **DIP3**, e **DIP4** atravessam buffers, não apresentados no presente trabalho; estes se fazem necessários para acionar os leds dos displays, sem sobrecorrentes no FPGA. A descrisão de hardware do bloco **displ_aqui** é mostrada no Quadro A.3.

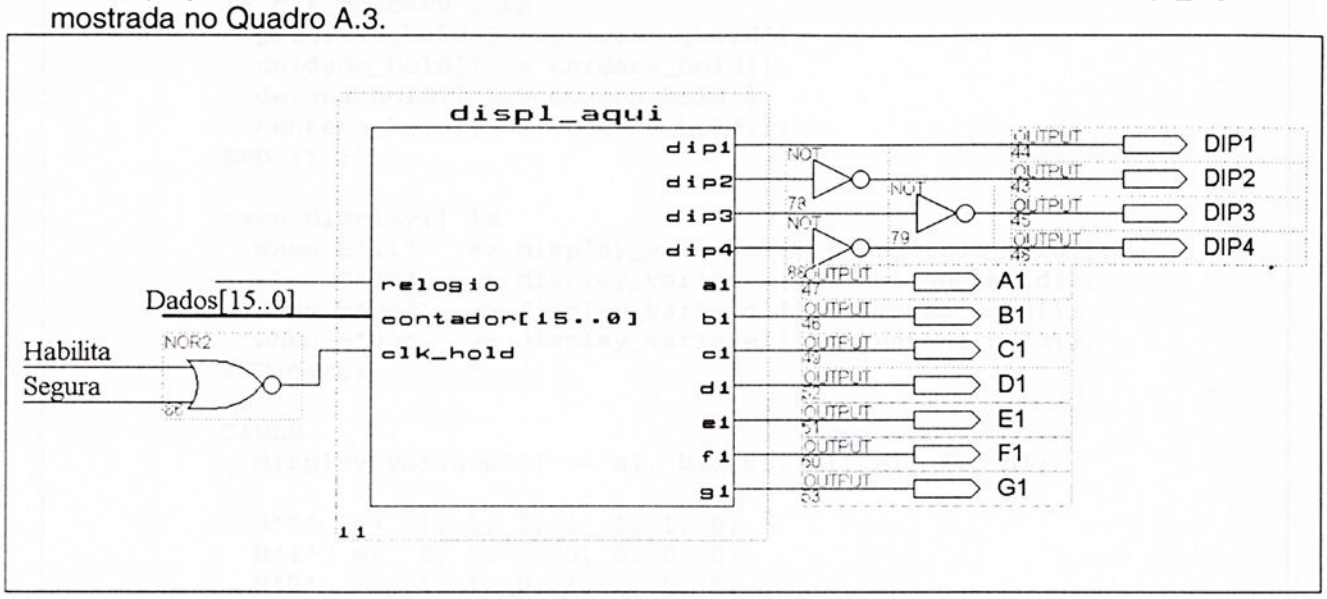


Figura A.18 – Circuito de controle dos displays.

Quadro A.3 – Descrição de Hardware do bloco displ_aqui.

```
SUBDESIGN displ_aqui
  (relogio, contador[15..0], clk_hold : INPUT;
   dip1, dip2, dip3, dip4    : OUTPUT;
   a1, b1, c1, d1, e1, f1, g1 : OUTPUT;)

 VARIABLE

   display[1..0] : DFF;
   display_variavel[3..0] : LATCH;
   precisao_hold[3..0] : LATCH;
   unidade_hold[3..0]  : LATCH;
   dezena_hold[3..0]   : LATCH;
   centena_hold[3..0]  : LATCH;

 BEGIN
   display_variavel[].ena  = relogio;
   display[].clk = relogio;
   dip1 = display1 AND display0;
   dip2 = display1 AND !display0;
   dip3 = !display1 AND display0;
   dip4 = display1 OR display0;
   precisao_hold[].ena = clk_hold;
   unidade_hold[].ena  = clk_hold;
   dezena_hold[].ena   = clk_hold;
   centena_hold[].ena  = clk_hold;

   display[] = display[] + 1;

 IF clk_hold==1 THEN
   precisao_hold[] = contador[3..0];
   unidade_hold[]  = contador[7..4];
```

```
    dezena_hold[]   = contador[11..8];
    centena_hold[]  = contador[15..12];
  END IF;

  IF clk_hold==0 THEN
    precisao_hold[] = precisao_hold[];
    unidade_hold[]  = unidade_hold[];
    dezena_hold[]   = dezena_hold[];
    centena_hold[]  = centena_hold[];
  END IF ;

  case display[] is
    when B"11"   => display_variavel[] = precisao_hold[];
    when B"10"   => display_variavel[] = unidade_hold[];
    when B"01"   => display_variavel[] = dezena_hold[];
    when B"00"   => display_variavel[] = centena_hold[];
  END case;

  TABLE
    display_variavel[] => a1, b1, c1, d1, e1, f1, g1;

    H"0"  =>  1, 1, 1, 1, 1, 1, 0;
    H"1"  =>  0, 1, 1, 0, 0, 0, 0;
    H"2"  =>  1, 1, 0, 1, 1, 0, 1;
    H"3"  =>  1, 1, 1, 1, 0, 0, 1;
    H"4"  =>  0, 1, 1, 0, 0, 1, 1;
    H"5"  =>  1, 0, 1, 1, 0, 1, 1;
    H"6"  =>  1, 0, 1, 1, 1, 1, 1;
    H"7"  =>  1, 1, 1, 0, 0, 0, 0;
    H"8"  =>  1, 1, 1, 1, 1, 1, 1;
    H"9"  =>  1, 1, 1, 1, 0, 1, 1;
    H"A"  =>  1, 1, 1, 0, 1, 1, 1;
    H"B"  =>  0, 0, 1, 1, 1, 1, 1;
    H"C"  =>  1, 0, 0, 1, 1, 1, 0;
    H"D"  =>  0, 1, 1, 1, 1, 0, 1;
    H"E"  =>  1, 0, 0, 1, 1, 1, 1;
    H"F"  =>  1, 0, 0, 0, 1, 1, 1;
  END TABLE;

END;
```

A3.3.6. O bloco daq_1 – aquisição via microcomputador

O circuito implementado para a comunicação serial é o **daq_1**, o qual é mostrado na Figura A.19.

A aquisição de dados por meio de um microcomputador será feita lendo os mesmos valores apresentados nos displays. Contudo, o micro será capaz de traçar um gráfico com o desenvolvimento das forças aplicadas sobre o sensor. A transmissão de dados será feita de forma serial, se utilizando a porta serial do microcomputador.

Este circuito digital recebe o valor da leitura realizada, este em 16 bits, particiona em dois bytes de 8 bits, envia dois bytes de identificação (o valor FFh duas vezes seguidas) e depois os dois bytes de dados, sendo o primeiroo menos significativo.

Por ser em modo assíncrono, após o segundo byte, o circuito espera a próxima leitura, mas o nível de sinal é mantido em alto, indicando que nenhum dado está sendo enviado. A freqüência de transmissão é determinada por **R_daq**, cujo valor utilizado neste projeto de 9.600 Hz.

cm 1 2 3 4 5 6 7 unesp 1 12 13 14 15 16 17 18

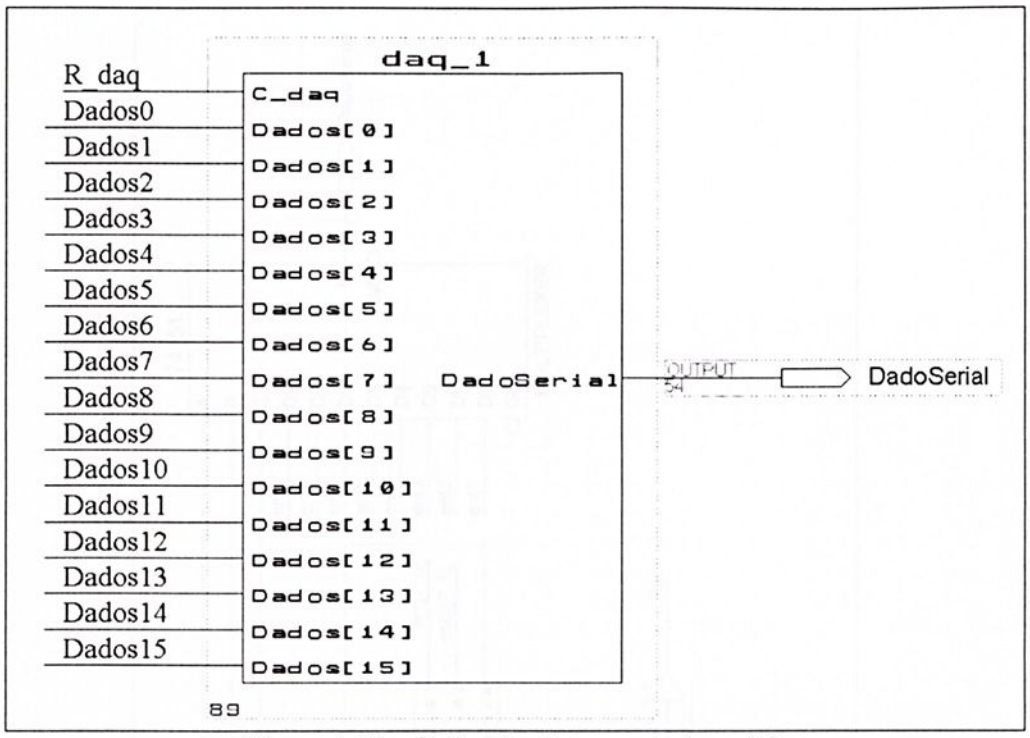


Figura A.19 – O circuito de transmissão serial.

O componente **Daq_1** é composto por outros circuitos internos, que são mostrados na Figura A.20. O bloco **daq** é responsável pela ordenação dos bits a serem enviados, o Quadro A.4 traz a descrição de seu hardware.

Observando ainda a Figura A.20, o conjunto de três FLIP-FLOPs gera os sinais de seleção do multiplexador 74151, de oito entradas e uma saída. Este multiplexador seleciona o bit que deve ser enviado, não somente os bits de dados, mas também o bit de paridade, gerado pelo componente 74280, o bit de início, e os dois de parada, estes são gerados por **daq**.

Daq envia o byte selecionado para o multiplexador por meio do barramento **ent[7..0]** e para o gerador de paridade através de **entp[7..0]**. Esta seleção é feita a partir dos dados disponíveis em **Dados[15..0]**. A estrutura interna de **daq** revela um contador, que para cada incremento, atribui valores para as diferentes saídas do circuito.

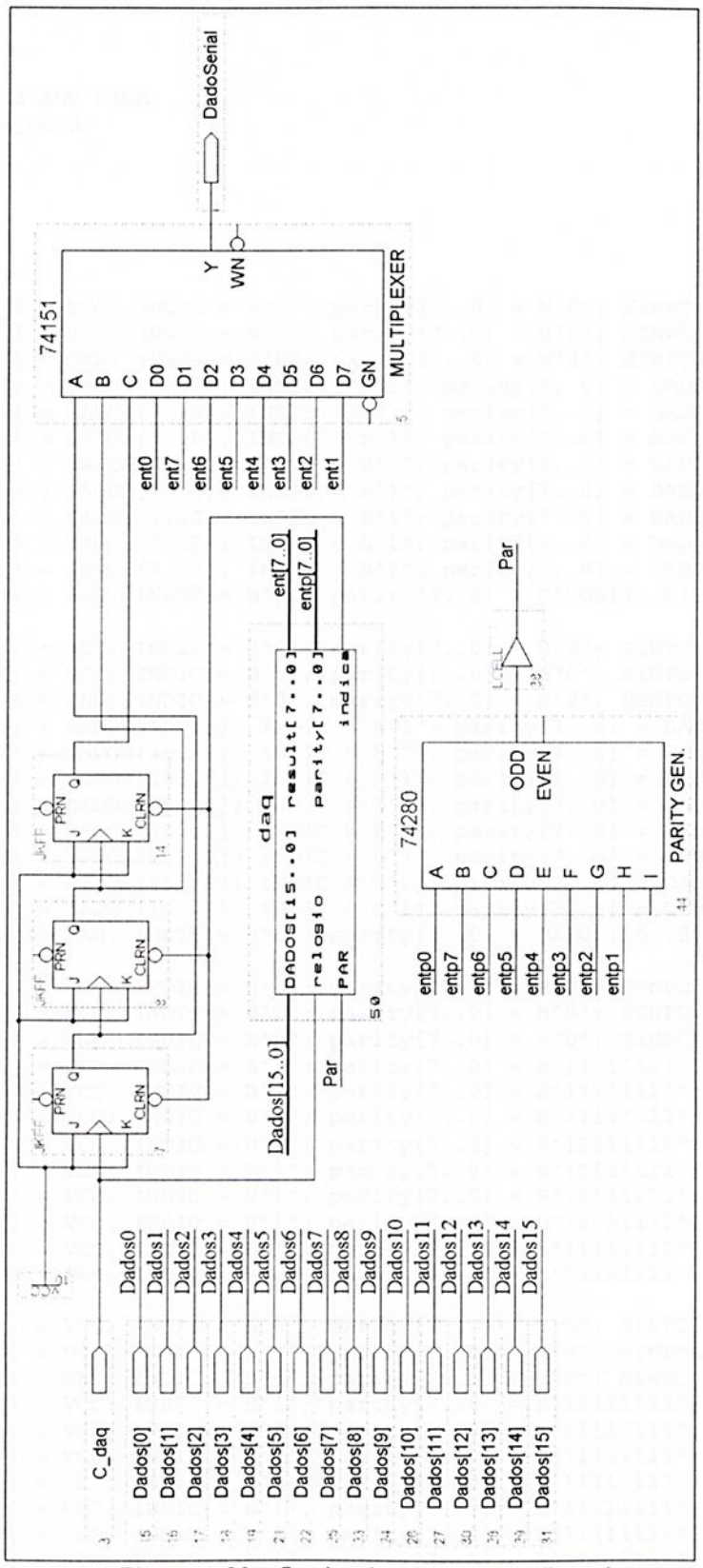


Figura A.20 – Os circuitos internos de Daq_1.

Quadro A.4 – Descrição de hardware do bloco daq.

```
UBDESIGN DAQ
  (DADOS[15..0], relogio, LIGA, clock_hold, PAR : INPUT;
  result[7..0], parity[7..0], INDIC, SINPC : OUTPUT;)
VARIABLE
```

```
  cont[5..0] : DFF;

BEGIN
  cont[].clk = relogio AND LIGA;
  cont[].clrn = clock_hold;
  cont[] = cont[]+1;

IF clock_hold then

CASE cont[] IS

 WHEN H"01" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"0";
 WHEN H"02" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"03" => result[] = GND; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"04" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"05" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"06" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"07" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"08" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"09" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"0A" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"0B" => result[] = DADOS[7..0]; INDIC = B"1"; parity[7..0] = DADOS[7..0]; SINPC = B"1";
 WHEN H"0C" => result[] = PAR; INDIC = B"0"; parity[7..0] = DADOS[7..0]; SINPC = B"1";

 WHEN H"0D" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"0E" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"0F" => result[] = GND; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"10" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"11" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"12" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"13" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"14" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"15" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"16" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"17" => result[] = DADOS[15..8]; INDIC = B"1"; parity[7..0] = DADOS[15..8]; SINPC = B"1";
 WHEN H"18" => result[] = PAR; INDIC = B"0"; parity[7..0] = DADOS[15..8]; SINPC = B"1";

 WHEN H"19" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"1A" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"1B" => result[] = GND; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"1C" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"1D" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"1E" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"1F" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"20" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"21" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"22" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"23" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"24" => result[] = PAR; INDIC = B"0"; parity[7..0] = B"11111111"; SINPC = B"1";

 WHEN H"25" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"26" => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"27" => result[] = GND; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"1";
 WHEN H"28" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"29" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2A" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2B" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2C" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2D" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2E" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"2F" => result[] = VCC; INDIC = B"1"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN H"30" => result[] = PAR; INDIC = B"0"; parity[7..0] = B"11111111"; SINPC = B"1";

 WHEN H"31" => result[] = VCC; INDIC = B"0"; parity[7..0] = B"11111111"; SINPC = B"1";
 WHEN OTHERS => result[] = VCC; INDIC = B"0"; parity[7..0] = H"0"; SINPC = B"0";
 END CASE;
 END IF;
```


BCpIS - FEIS - UNESP



```
IF !clock_hold then
  result[]=H"00";
END IF;

END;
```

APÊNDICE 2

A2. O PROGRAMA DO MICROCONTROLADOR PIC 18C452

A3.1. Introdução

O programa gerado para esse trabalho foi escrito em linguagem Assembley, baseado nas instruções internas do próprio microcontrolador PIC18C452 da Microchip.

O compliador usado para gerar os comandos em hexadecimal, para a gravação do componente, para simulação do programa é o MPLAB, também da Micorchip.

Para a implementação deste programa, se utilizou as informações do datasheet do componente como base teórica, bem como o manual do MPLAB. A descrição de cada instrução também é obtida do datasheet.

A seguir, estão colocadas todas as linhas de comando do programa implementado, bem como descrições detalhadas da ação da instrução. Quando necessário, para melhor compreensão, é feita uma descrição da rotina implementada, de como trata os dados e onde são armazenados os resultados.

A3.2. O Programa

```
list P=18C452
#include <P18C452.INC>

; Declaração das variiáveis

; Varáveis do Banco 0 - Endereço de Memória ocupados pelas Variáveis

cont_1 equ 0x00              ; cont_1 = 00h
cont_2 equ 0x01              ; cont_2 = 01h
cont_3 equ 0x02              ; cont_3 = 02h
temp equ 0x04                ; temp = 04h, variável da interrupção
atras_1 equ 0x05             ; atras_1 = 05h
atras_2 equ 0x06             ; atras_1 = 06h
atras_3 equ 0x07             ; atras_1 = 07h
cont1 equ 0x08               ; cont1 = 08h
lcdtemp equ 0x09             ; lcdtemp = 09h
ciclo equ 0x0A               ; ciclo = 0Ah

; Variáveis do LCD
RS equ 5                     ; no programa, RS = 5
RW equ 6                     ; RW = 6
RE equ 7                     ; RE = 7

; Varáveis do Banco 1

leitL equ 0x1F               ; leitL = 1Fh
leitH equ 0x1E               ; leitH = 1Eh
```


unesp

```
; variáveis da média (de 20h à FFh)

contmed equ 0x1D              ; contmed = 1Dh

m_0L equ 0x20                 ; m_0L = 20h
m_0H equ 0x21                 ; m_0H = 21h
m_1L equ 0x22                 ; m_1L = 22h
m_1H equ 0x23                 ; m_1H = 23h
m_2L equ 0x24                 ; m_2L = 24h
m_2H equ 0x25                 ; m_2H = 25h
m_3L equ 0x26                 ; m_3L = 26h
m_3H equ 0x27                 ; m_3H = 27h
m_4L equ 0x28                 ; m_4L = 28h
m_4H equ 0x29                 ; m_4H = 29h
m_5L equ 0x2A                 ; m_5L = 2Ah
m_5H equ 0x2B                 ; m_5H = 2Bh
m_6L equ 0x2C                 ; m_6L = 2Ch
m_6H equ 0x2D                 ; m_6H = 2Dh
m_7L equ 0x2E                 ; m_7L = 2Eh
m_7H equ 0x2F                 ; m_7H = 2Fh
m_8L equ 0x30                 ; m_8L = 30h
m_8H equ 0x31                 ; m_8H = 31h
m_9L equ 0x32                 ; m_9L = 32h
m_9H equ 0x33                 ; m_9H = 33h

m_10L equ 0x34                ; m_10L = 34h
m_10H equ 0x35                ; m_10H = 35h
m_11L equ 0x36                ; m_11L = 36h
m_11H equ 0x37                ; m_11H = 37h
m_12L equ 0x38                ; m_12L = 38h
m_12H equ 0x39                ; m_12H = 39h
m_13L equ 0x3A                ; m_13L = 3Ah
m_13H equ 0x3B                ; m_13H = 3Bh
m_14L equ 0x3C                ; m_14L = 3Ch
m_14H equ 0x3D                ; m_14H = 3Dh
m_15L equ 0x3E                ; m_15L = 3Eh
m_15H equ 0x3F                ; m_15H = 3Fh
m_16L equ 0x40                ; m_16L = 40h
m_16H equ 0x41                ; m_16H = 41h
m_17L equ 0x42                ; m_17L = 42h
m_17H equ 0x43                ; m_17H = 43h
m_18L equ 0x44                ; m_18L = 44h
m_18H equ 0x45                ; m_18H = 45h
m_19L equ 0x46                ; m_19L = 46h
m_19H equ 0x47                ; m_19H = 47h

m_20L equ 0x48                ; m_20L = 48h
m_20H equ 0x49                ; m_20H = 49h
m_21L equ 0x4A                ; m_21L = 4Ah
m_21H equ 0x4B                ; m_21H = 4Bh
m_22L equ 0x4C                ; m_22L = 4Ch
m_22H equ 0x4D                ; m_22H = 4Dh
m_23L equ 0x4E                ; m_23L = 4Eh
m_23H equ 0x4F                ; m_23H = 4Fh
m_24L equ 0x50                ; m_24L = 50h
m_24H equ 0x51                ; m_24H = 51h
m_25L equ 0x52                ; m_25L = 52h
m_25H equ 0x53                ; m_25H = 53h
m_26L equ 0x54                ; m_26L = 54h
m_26H equ 0x55                ; m_26H = 55h
```




```
m_27L equ 0x56              ; m_27L = 56h
m_27H equ 0x57              ; m_27H = 57h
m_28L equ 0x58              ; m_28L = 58h
m_28H equ 0x59              ; m_28H = 59h
m_29L equ 0x5A              ; m_29L = 5Ah
m_29H equ 0x5B              ; m_29H = 5Bh

m_30L equ 0x5C              ; m_30L = 5Ch
m_30H equ 0x5D              ; m_30H = 5Dh
m_31L equ 0x5E              ; m_31L = 5Eh
m_31H equ 0x5F              ; m_31H = 5Fh
m_32L equ 0x60              ; m_32L = 60h
m_32H equ 0x61              ; m_32H = 61h
m_33L equ 0x62              ; m_33L = 62h
m_33H equ 0x63              ; m_33H = 63h
m_34L equ 0x64              ; m_34L = 64h
m_34H equ 0x65              ; m_34H = 65h
m_35L equ 0x66              ; m_35L = 66h
m_35H equ 0x67              ; m_35H = 67h
m_36L equ 0x68              ; m_36L = 68h
m_36H equ 0x69              ; m_36H = 69h
m_37L equ 0x6A              ; m_37L = 6Ah
m_37H equ 0x6B              ; m_37H = 6Bh
m_38L equ 0x6C              ; m_38L = 6Ch
m_38H equ 0x6D              ; m_38H = 6Dh
m_39L equ 0x6E              ; m_39L = 6Eh
m_39H equ 0x6F              ; m_39H = 6Fh

m_40L equ 0x70              ; m_40L = 70h
m_40H equ 0x71              ; m_40H = 71h
m_41L equ 0x72              ; m_41L = 72h
m_41H equ 0x73              ; m_41H = 73h
m_42L equ 0x74              ; m_42L = 74h
m_42H equ 0x75              ; m_42H = 75h
m_43L equ 0x76              ; m_43L = 76h
m_43H equ 0x77              ; m_43H = 77h
m_44L equ 0x78              ; m_44L = 78h
m_44H equ 0x79              ; m_44H = 79h
m_45L equ 0x7A              ; m_45L = 7Ah
m_45H equ 0x7B              ; m_45H = 7Bh
m_46L equ 0x7C              ; m_46L = 7Ch
m_46H equ 0x7D              ; m_46H = 7Dh
m_47L equ 0x7E              ; m_47L = 7Eh
m_47H equ 0x7F              ; m_47H = 7Fh
m_48L equ 0x80              ; m_48L = 80h
m_48H equ 0x81              ; m_48H = 81h
m_49L equ 0x82              ; m_49L = 82h
m_49H equ 0x83              ; m_49H = 83h

m_50L equ 0x84              ; m_50L = 84h
m_50H equ 0x85              ; m_50H = 85h
m_51L equ 0x86              ; m_51L = 86h
m_51H equ 0x87              ; m_51H = 87h
m_52L equ 0x88              ; m_52L = 88h
m_52H equ 0x89              ; m_52H = 89h
m_53L equ 0x8A              ; m_53L = 8Ah
m_53H equ 0x8B              ; m_53H = 8Bh
m_54L equ 0x8C              ; m_54L = 8Ch
m_54H equ 0x8D              ; m_54H = 8Dh
m_55L equ 0x8E              ; m_55L = 8Eh
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
m_55H equ 0x8F              ; m_55H = 8Fh
m_56L equ 0x90              ; m_56L = 90h
m_56H equ 0x91              ; m_56H = 91h
m_57L equ 0x92              ; m_57L = 92h
m_57H equ 0x93              ; m_57H = 93h
m_58L equ 0x94              ; m_58L = 94h
m_58H equ 0x95              ; m_58H = 95h
m_59L equ 0x96              ; m_59L = 96h
m_59H equ 0x97              ; m_59H = 97h

m_60L equ 0x98              ; m_60L = 98h
m_60H equ 0x99              ; m_60H = 99h
m_61L equ 0x9A              ; m_61L = 9Ah
m_61H equ 0x9B              ; m_61H = 9Bh
m_62L equ 0x9C              ; m_62L = 9Ch
m_62H equ 0x9D              ; m_62H = 9Dh
m_63L equ 0x9E              ; m_63L = 9Eh
m_63H equ 0x9F              ; m_63H = 9Fh


som_0L equ 0xA1             ; som_0L = A1h
som_0H equ 0xA2             ; som_0H = A2h
som_1L equ 0xA3             ; som_1L = A3h
som_1H equ 0xA4             ; som_1H = A4h
som_2L equ 0xA5             ; som_2L = A5h
som_2H equ 0xA6             ; som_2H = A6h
som_3L equ 0xA7             ; som_3L = A7h
som_3H equ 0xA8             ; som_3H = A8h
som_4L equ 0xA9             ; som_4L = A9h
som_4H equ 0xAA             ; som_4H = AAh
som_5L equ 0xAB             ; som_5L = ABh
som_5H equ 0xAC             ; som_5H = ACh
som_6L equ 0xAD             ; som_6L = ADh
som_6H equ 0xAE             ; som_6H = AEh
som_7L equ 0xAF             ; som_7L = AFh
som_7H equ 0xB0             ; som_7H = B0h
som_8L equ 0xB1             ; som_8L = B1h
som_8H equ 0xB2             ; som_8H = B2h
som_9L equ 0xB3             ; som_9L = B3h
som_9H equ 0xB4             ; som_9H = B4h

som_10L equ 0xB5            ; som_10L = B5h
som_10H equ 0xB6            ; som_10H = B6h
som_11L equ 0xB7            ; som_11L = B7h
som_11H equ 0xB8            ; som_11H = B8h
som_12L equ 0xB9            ; som_12L = B9h
som_12H equ 0xBA            ; som_12H = BAh
som_13L equ 0xBB            ; som_13L = BBh
som_13H equ 0xBC            ; som_13H = BCh
som_14L equ 0xBD            ; som_14L = BDh
som_14H equ 0xBE            ; som_14H = BEh
som_15L equ 0xBF            ; som_15L = BFh
som_15H equ 0xC0            ; som_15H = C0h
som_16L equ 0xC1            ; som_16L = C1h
som_16H equ 0xC2            ; som_16H = C2h
som_17L equ 0xC3            ; som_17L = C3h
som_17H equ 0xC4            ; som_17H = C4h
som_18L equ 0xC5            ; som_18L = C5h
som_18H equ 0xC6            ; som_18H = C6h
som_19L equ 0xC7            ; som_19L = C7h
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
som_19H equ 0xC8            ; som_19H = C8h

som_20L equ 0xC9            ; som_20L = C9h
som_20H equ 0xCA            ; som_20H = CAh
som_21L equ 0xCB            ; som_21L = CBh
som_21H equ 0xCC            ; som_21H = CCh
som_22L equ 0xCD            ; som_22L = CDh
som_22H equ 0xCE            ; som_22H = CEh
som_23L equ 0xCF            ; som_23L = CFh
som_23H equ 0xD0            ; som_23H = D0h
som_24L equ 0xD1            ; som_24L = D1h
som_24H equ 0xD2            ; som_24H = D2h
som_25L equ 0xD3            ; som_25L = D3h
som_25H equ 0xD4            ; som_25H = D4h
som_26L equ 0xD5            ; som_26L = D5h
som_26H equ 0xD6            ; som_26H = D6h
som_27L equ 0xD7            ; som_27L = D7h
som_27H equ 0xD8            ; som_27H = D8h
som_28L equ 0xD9            ; som_28L = D9h
som_28H equ 0xDA            ; som_28H = DAh
som_29L equ 0xDB            ; som_29L = DBh
som_29H equ 0xDC            ; som_29H = DCh

som_30L equ 0xDD            ; som_30L = DDh
som_30H equ 0xDE            ; som_30H = DEh
som_31L equ 0xDF            ; som_31L = DFh
som_31H equ 0xE1            ; som_31H = E1h

med_0L equ 0xE2             ; med_0L = E2h
med_0H equ 0xE3             ; med_0H = E3h
med_1L equ 0xE4             ; med_1L = E4h
med_1H equ 0xE5             ; med_1H = E5h
med_2L equ 0xE6             ; med_2L = E6h
med_2H equ 0xE7             ; med_2H = E7h
med_3L equ 0xE8             ; med_3L = E8h
med_3H equ 0xE9             ; med_3H = E9h
med_4L equ 0xEA             ; med_4L = EAh
med_4H equ 0xEB             ; med_4H = EBh
med_5L equ 0xEC             ; med_5L = ECh
med_5H equ 0xED             ; med_5H = EDh
med_6L equ 0xEE             ; med_6L = EEh
med_6H equ 0xEF             ; med_6H = EFh
med_7L equ 0xF0             ; med_7L = F0h
med_7H equ 0xF1             ; med_7H = F1h
med_8L equ 0xF2             ; med_7L = F2h
med_8H equ 0xF3             ; med_7H = F3h
med_9L equ 0xF4             ; med_7L = F4h
med_9H equ 0xF5             ; med_7H = F5h
med_AL equ 0xF6             ; med_7L = F6h
med_AH equ 0xF7             ; med_7H = F7h
med_BL equ 0xF8             ; med_7L = F8h
med_BH equ 0xF9             ; med_7H = F9h
med_CL equ 0xFA             ; med_7L = FAh
med_CH equ 0xFB             ; med_7H = FBh
med_DL equ 0xFC             ; med_7L = FCh
med_DH equ 0xFD             ; med_7H = FDh
med_EL equ 0xFE             ; med_7L = FEh
med_EH equ 0xFF             ; med_7H = FFh
med_FL equ 0x00             ; med_7L = 00h
```


unesp

```
med_FH equ 0x01                 ; med_7H = 01h

med_T0L equ 0x02                ; med_T0L = 02h
med_T0H equ 0x03                ; med_T0H = 03h
med_T1L equ 0x04                ; med_T1L = 04h
med_T1H equ 0x05                ; med_T1H = 05h
med_T2L equ 0x06                ; med_T2L = 06h
med_T2H equ 0x07                ; med_T2H = 07h
med_T3L equ 0x08                ; med_T3L = 08h
med_T3H equ 0x09                ; med_T3H = 09h
med_T4L equ 0x0A                ; med_T4L = 0Ah
med_T4H equ 0x0B                ; med_T4H = 0Bh
med_T5L equ 0x0C                ; med_T5L = 0Ch
med_T5H equ 0x0D                ; med_T5H = 0Dh
med_T6L equ 0x0E                ; med_T6L = 0Eh
med_T6H equ 0x0F                ; med_T6H = 0Fh
med_T7L equ 0x10                ; med_T7L = 10h
med_T7H equ 0x11                ; med_T7H = 11h

medi_0L equ 0x12                ; medi_1L = 12h
medi_0H equ 0x13                ; medi_1H = 13h
medi_1L equ 0x14                ; medi_1L = 14h
medi_1H equ 0x15                ; medi_1H = 15h
medi_2L equ 0x16                ; medi_2L = 16h
medi_2H equ 0x17                ; medi_2H = 17h
medi_3L equ 0x18                ; medi_1L = 18h
medi_3H equ 0x19                ; medi_1H = 19h


med_TL equ 0x1A                 ; med_TL = 1Ah
med_TH equ 0x1B                 ; med_TH = 1Bh

; Varáveis do Banco 2

; variáveis da média (de 00h à 19h)


medi_0L equ 0x12                ; medi_1L = 12h
medi_0H equ 0x13                ; medi_1H = 13h
medi_1L equ 0x14                ; medi_1L = 14h
medi_1H equ 0x15                ; medi_1H = 15h
medi_2L equ 0x16                ; medi_2L = 16h
medi_2H equ 0x17                ; medi_2H = 17h
medi_3L equ 0x18                ; medi_1L = 18h
medi_3H equ 0x19                ; medi_1H = 19h

media_0L equ 0x00               ; media_0L = 00h
media_0H equ 0x01               ; media_0H = 01h
media_1L equ 0x02               ; media_1L = 02h
media_1H equ 0x03               ; media_1H = 01h


; O Programa - Início das Instruções do Programa

 org 00000h                     ; endereço de reset
 goto main                      ; vai para a rotina/programa principal
```


unesp

```
 org 00008h                ; endereço de interrupção maior prioridade
 bcf intcon,gie,0          ; desligas as interrupções
 bcf intcon3,int1if,0      ; desliga a interrupção INT0
 bsf temp,0,0              ; seta o bit 0 de temp
 retfie                    ; retorna da interrupção

 org 00018h                ; endereço de interrupção menor prioridade
 bcf intcon,gie,0          ; desligas as interrupções
 bcf intcon3,int1if,0      ; desliga a interrupção INT0
 bsf temp,0,0              ; seta o bit 0 de temp
 retfie                    ; retorna da interrupção

 org 00030h                ; endereço onde inicia o programa principal
main

 clrf intcon,0             ; zera Intcom, desabilitando todas as interrupções
 movlw 0xA0                ; W = A0h
 movwf intcon2,0           ; Intcon2 = W = A0h, configurando os pinos RB0 e
                           ; RB1 com borda de subida
 movlw 0x48                ; WREG = 48h
 movwf intcon3,0           ; Intcon3 = W = 48h, habilitando Int1 e colocando
                           ; como interrupção de alta prioridade
 bcf rcon,ipen             ; zera Ipen de Rcom, desabilitando os níveis de
                           ; interrupções
 clrf ipr1,0               ; zera o registro de niveis de interrupções de
 clrf ipr2,0               ; periféricos

 movlw 0x30                ; WREG = 30h
 call atraso1              ; chama a rotina de atraso

 call lcdinicio            ; chama a rotina de inicialização do LCD
 call lcdliga              ; chama a rotina de ligar o LCD
 call msginicial           ; chama a rotina de aprtesentação inicial do LCD
 movlw 0x6f                ; WREG = 6Fh
 call atraso1              ; chama a rotina de atraso1
 call apaga                ; chama a rotina de apagar o LCD
 call msgapres             ; chama a rotina de apresentação da contagem

 movlb 0x01                ; defini o Banco 1 da Ram como local de trabalho
 clrf contmed,1            ; zera o contador das leituras

; Rotinas de espera da interrupção
loop                       ; rotina de loop

 incf ciclo,1,0            ; incrementa ciclo a cada ciclo do loop
 clrf temp,0               ; zera a variável temp
 bsf intcon3,int1ie,0      ; liga a interrupção Interrupção INT0
 bsf intcon,gie,0          ; liga as interrupções

espr1
 nop
 btfss temp,0,0            ; testa o bit 0, se setado, pula
 goto espr1                ; volta a espr1

 clrf tmr3h,0
 clrf tmr3l,0
 clrf tmr3h,0
 clrf tmr3l,0
 bsf t3con,rd16,0          ; T1CON = WREG
```


unesp

```
 bsf t3con,tmr3on,0          ; set o bit TMR1ON, ligando o Timer 1
 clrf temp,0                 ; incrementa a variável temp
 bsf intcon,gie,0            ; liga as interrupções

espr2
 nop
 btfss temp,0,0              ; testa o bit 0, se setado, pula
 goto espr2

 bcf t3con,tmr3on,0          ; zera o bit TMR1ON, desligando o Timer 1

 movlb 0x01                  ; define o banco 1 da Ram como local de trabalho
 movf tmr3l,0,0              ; WREG = TMR3l
 movwf leitL,1               ; leitL = WREG
 movwf cont_2,0              ; cont_2 = WREG
 movf tmr3h,0,0              ; WREG = TMR3h
 movwf leitH,1               ; leitH = WREG
 movwf cont_1,0              ; cont_1 = WREG

 call msgcont2               ; chama a rotina de apresentação da contagem

 call media                  ; chama a rotina de média

 movlb 0x01                  ; o banco 1 da Ram é o banco em uso
 movf med_TH,0,1             ; WREG = med_TH
 movwf cont_1,0              ; cont_1 = WREG
 movf med_TL,0,1             ; WREG = med_TL
 movwf cont_2,0              ; cont_2 = WREG

 call msgcont                ; chama a rotina de apresentação da média

 call excit                  ; vai para a rotina de exitação da corda

 goto loop                   ; retorna a rotina de Loop


; Excitação da Corda
excit                        ; Excitação da corda
 clrf ciclo,0                ; ciclo = 00h
 bcf trise,trise0,0          ; defini o bit 0 da porta e como saída
 bsf porte,re0,0             ; atribui "1"para o pino Re0
 movlw 0x0c                  ; vamos chamar a rotina atraso1, para um delay de
 call atraso1,0              ; aproximadamente 0,6 milisegundos
 bcf porte,re0,0             ; atribui "0" para o pino Re0, terminando o pulso
 movlw 0x1Bh                 ; WREG = 1Bh
 call atraso1                ; chama a rotina de atraso, para 2,9 milisegundos
 nop
 return                      ; retorna da rotina de exictação.
; fim da rotina excit


; Mensagem do resultado da contagem, sem o calculo da média
msgcont2
 clrf trisd,0                ; coloca a porta d como saida
 clrf trisc,0                ; coloca a porta a como saida
 movlw 0xca                  ; WREG = CAh
 movwf portd,0               ; atribui o valor de W para a porta d
 bcf portc,RS,0              ; desabilita a escrita do dado
 bcf portc,RW,0              ; desabilita o LCD
 nop                         ; não faz nada
```


unesp

```
 nop
 nop
 nop
 nop
 bsf portc,RE,0             ; habilita o LCD
 nop
 nop
 nop
 nop
 nop
 nop
 bcf portc,RE,0             ; desabilita o LCD
 movlw 0x11                 ; WREG = 11h
 call atraso1               ; chama a rotina de atraso, para permitir que o LCD '

 movlb 0x01                 ; aciona o Banco 1 da Ram
 movf cont_1,0,0            ; WREG = cont_1
 movwf cont_3,0             ; cont_3 = W
 movlw 0xF0                 ; W = F0h
 andwf cont_3,1,0           ; cont_3 = W AND cont_3
 swapf cont_3,1,0           ; SWAP cont_3, rotaciona os nibbles de cont_3
 movlw 0x30                 ; W = 30h
 addwf cont_3,0,0           ; W = W + cont_3
 movwf lcdtemp,0            ; lcdtemp = W
 call lcdescv               ; chama a rotina que escreve no LCD

 movf cont_1,0,0            ; W = cont_1
 movwf cont_3,0             ; cont_3 = W
 movlw 0x0F                 ; W = 0Fh
 andwf cont_3,1,0           ; cont_3 = W AND cont_3
 movlw 0x30                 ; W = 30h
 addwf cont_3,0,0           ; W = W + cont_3
 movwf lcdtemp,0            ; lcdtemp = W
 call lcdescv               ; chama a rotina que escreve no LCD

 movf cont_2,0,0            ; W = cont_2
 movwf cont_3,0             ; cont_3 = W
 movlw 0xF0                 ; W = F0h
 andwf cont_3,1,0           ; cont_3 = W AND cont_3
 swapf cont_3,1,0           ; SWAP cont_3, rotaciona os nibbles de cont_3
 movlw 0x30                 ; W = 30h
 addwf cont_3,0,0           ; W = W + cont_3
 movwf lcdtemp,0            ; lcdtemp = W
 call lcdescv               ; chama a rotina que escreve no LCD

 movf cont_2,0,0            ; W = cont_2
 movwf cont_3,0             ; cont_3 = W
 movlw 0x0F                 ; W = 0Fh
 andwf cont_3,1,0           ; cont_3 = W AND cont_3
 movlw 0x30                 ; W = 30h
 addwf cont_3,0,0           ; W = W + cont_3
 movwf lcdtemp,0            ; lcdtemp = W
 call lcdescv               ; chama a rotina que escreve no LCD
 return                     ; retorna da rotina
; fim da rotina de msgcont2


; Mensagem do resultado da contagem, com a média
msgcont
 clrf trisd,0               ; coloca a porta d como saida
```


unesp

```
clrf trisc,0              ; coloca a porta a como saida
movlw 0x8a                ; WREG = 8Ah
movwf portd,0             ; atribui o valor de W para a porta d
bcf portc,RS,0            ; desabilita a escrita do dado
bcf portc,RW,0            ; desabilita o LCD
nop                       ; não faz nada
nop
nop
nop
nop
bsf portc,RE,0            ; habilita o LCD
nop
nop
nop
nop
nop
nop
bcf portc,RE,0            ; desabilita o LCD
movlw 0x11                ; WREG = 11h
call atraso1              ; chama a rotina de atraso, para permitir que o LCD

movf cont_1,0,0           ; WREG = cont_1
movwf cont_3,0            ; cont_3 = W
movlw 0xF0                ; W = F0h
andwf cont_3,1,0          ; cont_3 = W AND cont_3
swapf cont_3,1,0          ; SWAP cont_3, rotaciona os nibbles de cont_3
movlw 0x30                ; W = 30h
addwf cont_3,0,0          ; W = W + cont_3
movwf lcdtemp,0           ; lcdtemp = W
call lcdescv              ; chama a rotina que escreve no LCD

movf cont_1,0,0           ; W = cont_1
movwf cont_3,0            ; cont_3 = W
movlw 0x0F                ; W = 0Fh
andwf cont_3,1,0          ; cont_3 = W AND cont_3
movlw 0x30                ; W = 30h
addwf cont_3,0,0          ; W = W + cont_3
movwf lcdtemp,0           ; lcdtemp = W
call lcdescv              ; chama a rotina que escreve no LCD

movf cont_2,0,0           ; W = cont_2
movwf cont_3,0            ; cont_3 = W
movlw 0xF0                ; W = F0h
andwf cont_3,1,0          ; cont_3 = W AND cont_3
swapf cont_3,1,0          ; SWAP cont_3, rotaciona os nibbles de cont_3
movlw 0x30                ; W = 30h
addwf cont_3,0,0          ; W = W + cont_3
movwf lcdtemp,0           ; lcdtemp = W
call lcdescv              ; chama a rotina que escreve no LCD

movf cont_2,0,0           ; W = cont_2
movwf cont_3,0            ; cont_3 = W
movlw 0x0F                ; W = 0Fh
andwf cont_3,1,0          ; cont_3 = W AND cont_3
movlw 0x30                ; W = 30h
addwf cont_3,0,0          ; W = W + cont_3
movwf lcdtemp,0           ; lcdtemp = W
call lcdescv              ; chama a rotina que escreve no LCD
return                    ; retorna da rotina
; fim da rotina de msgcont
```


unesp

```
; Mensagem Inicial
msginicial                  ; escreve mensagem inicial no LCD
 movlw 0x4c                 ; move valor para W, correnponde a "L"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x65                 ; move valor para W, correnponde a "e"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6e                 ; move valor para W, correnponde a "n"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x64                 ; move valor para W, correnponde a "d"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ;  escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6f                 ; move valor para W, correnponde a "o"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x20                 ; move valor para W, correnponde a " "
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x53                 ; move valor para W, correnponde a "S"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x65                 ; move valor para W, correnponde a "e"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6e                 ; move valor para W, correnponde a "n"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x73                 ; move valor para W, correnponde a "s"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6f                 ; move valor para W, correnponde a "o"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x72                 ; move valor para W, correnponde a "r"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 return                     ; retorna da subrotina
; fim da msg inicial "Lendo Sensor Aguarde".

; Mensagem de Apresentação
msgapres                    ; escreve mensagem inicial no LCD
 movlw 0x43                 ; move valor para W, correnponde a "C"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6f                 ; move valor para W, correnponde a "o"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6e                 ; move valor para W, correnponde a "n"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x74                 ; move valor para W, correnponde a "t"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ;  escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x61                 ; move valor para W, correnponde a "a"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x67                 ; move valor para W, correnponde a "g"
```


unesp

```
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x65                 ; move valor para W, correnponde a "e"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x6d                 ; move valor para W, correnponde a "m"
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 movlw 0x3d                 ; move valor para W, correnponde a "="
 movwf lcdtemp,0            ; move valor para a variável do LCD, para ser
 call lcdescv               ; escrita no LCD. Chama a rotina de escrever no LCD
 return                     ; retorna da subrotina
; fim da msg inicial "Contagem=".

; Apaga o LCD
apaga
 clrf trisc,0               ; Coloca os pinos da porta B com saída
 clrf trisd,0               ; Coloca os pinos da porta D com saída
 bcf portc,RS,0             ; Bit RS da porta B = 0
 bcf portc,RW,0             ; Bit RW da porta B = 0
 movlw 0x01                 ; WREG = 01h
 movwf portd,0              ; porta D = WREG
 nop                        ; não faz nada
 nop
 nop
 nop
 nop
 bsf portc,RE,0             ; habilita o LCD
 nop
 nop
 nop
 nop
 nop
 nop
 bcf portc,RE,0             ; desabilita o LCD
 movlw 0x11                 ; WREG = 11h
 call atraso1               ; chama a rtoina de atraso, para um tempo de 15ms
 return                     ; retorna da rotina lcdinicio
; fim da rotina de apagar o LCD

; Escreve no LCD
lcdescv                     ; Escreve no LCD
 clrf trisd,0               ; direciona porta d como saída
 clrf trisc,0               ; coloca a porta a como saida
 movf lcdtemp,0,0           ; carrega W com o valor de LCDtemp, banco 0
 movwf portd,0              ; coloca o valor W para a porta D
 bsf portc,RS,0             ; habilita a escrita do dado
 bcf portc,RW,0             ; habilita o LCD
 nop
 nop
 nop
 nop
 nop
 bsf portc,RE,0             ; habilita o LCD
 nop
 nop
 nop
 nop
 nop
 nop
```


unesp

```
 bcf portc,RE,0              ; termina a escrita
 movlw 0x11                  ; WREG = 11h
 call atraso1                ; chama a rotina de atrso, dar um atros de 15 ms
 return                      ; retorna da rotina
; fim da rotina lcdescv

; Rotina de ligar o LCD
lcdliga                      ; roitna de ligar o LDC
 bcf portc,RW,0              ; zera o bit RW
 bcf portc,RS,0               ; zera o bit RS
 movlw 0x0F                  ; WREG = 0Fh
 movwf portd,0               ; porta D = WREG
 nop
 nop
 nop
 nop
 nop
 bsf portc,RE,0              ; habilita o LCD
 nop
 nop
 nop
 nop
 nop
 nop
 bcf portc,RE,0              ; desabilita o LCD
 movlw 0x11                  ; WREG = 11h
 call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms
 return
; fim da rotina de ligar o LCD

; Rotina de desligar o LCD
lcddesl
 movlw 0x0a                  ; WREG = 0Ah
 movwf portd,0               ; atribui o valor de W para a porta d
 bcf portc,RW,0              ; zera o bit RW
 bcf portc,RS,0              ; zera o bit RS
 nop
 nop
 nop
 nop
 nop
 bsf portc,RE,0              ; habilita o LCD
 call atraso2                ; chama a rotina de atraso, e guarda registros
 bcf portc,RE,0              ; desabilita o LCD
 return                      ; retorna da subrotina
; fim da rotina de desligar o lcd

; Inicializa o LCD (bytes de programa do LCD)
lcdinicio                    ; inicializa o lcd, em 8bits
                             ; Envia o valor 38h - 1a vez
 clrf trisd,0                ; coloca a porta d como saida
 clrf trisc,0                ; coloca a porta a como saida
 movlw 0x38                  ; WREG = 38h
 movwf portd,0               ; atribui o valor de W para a porta d
 bcf portc,RS,0              ; desabilita a escrita do dado
 bcf portc,RW,0              ; desabilita o LCD
 nop                         ; não faz nada
 nop
 nop
 nop
```


unesp

```
nop
bsf portc,RE,0              ; habilita o LCD
nop                         ; não faz nada
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
bcf portc,RE,0              ; desabilita o LCD
movlw 0x25                  ; WREG = 25h
call atraso1                ; chama a rotina de atraso, para permitir que o LCD

bsf portc,RE,0              ; habilita o LCD
nop                         ; não faz nada
nop
nop
nop
nop
nop
bcf portc,RE,0              ; desabilita o LCD
movlw 0x25                  ; atribui o valor para W
call atraso1                ; chama a rotina de atraso, para LCD processar

bsf portc,RE,0              ; habilita o LCD
nop                         ; não faz nada
nop
nop
nop
nop
nop
bcf portc,RE,0              ; desabilita o LCD
movlw 0x25                  ; WREG = 25h
call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms

movlw 0x08                  ; WREG = 08h
movwf portd,0               ; atribui o valor de W para a porta d
bsf portc,RE,0              ; habilita o LCD
nop                         ; não faz nada
nop
nop
nop
nop
nop
bcf portc,RE,0              ; desabilita o LCD

movlw 0x11                  ; WREG = 11h
call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms
movlw 0x01                  ; WREG = 01h
movwf portd,0               ; porta D = WREG
bsf portc,RE,0              ; habilita o LCD
nop                         ; não faz nada
nop
nop
nop
```


unesp

```
 nop
 nop
 bcf portc,RE,0              ; desabilita o LCD

 movlw 0x11                  ; WREG = 11h
 call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms
 movlw 0x06                  ; WREG = 06h
 movwf portd,0               ; porta D = WREG
 bsf portc,RE,0              ; habilita o LCD
 movlw 0x02                  ; WREG = 02h
 call atraso2                ; chama a rotina de atraso, e guarda registros
 bcf portc,RE,0              ; desabilita o LCD

 movlw 0x11                  ; WREG = 11h
 call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms
 movlw 0x14                  ; WREG = 14h
 movwf portd,0               ; porta D = WREG
 bsf portc,RE,0              ; habilita o LCD
 nop                         ; não faz nada
 nop
 nop
 nop
 nop
 nop
 bcf portc,RE,0              ; desabilita o LCD
 movlw 0x11                  ; WREG = 11h
 call atraso1                ; chama a rtoina de atraso, para um tempo de 15ms
 return                      ; retorna da rotina lcdinicio
; fim da rotina lcdinicio


; Rotinas de atraso do Microcontrolador
; Atraso 1
atraso1                      ; subrotina que realiza um atraso, o tempo é
                             ; definido pelo valor que o programa solicitante
                             ; atribui ao W.
 movwf atras_1,0             ; O valor de W é transferido para as 3 variáveis,
 movwf atras_2,0             ; no banco 00
 movwf atras_3,0             ;
rotin1
 decfsz atras_1,1,0          ; decrementa a variável atras_1
 goto rotin1                 ; volta para rotin1
 movwf atras_1,0             ; W está inalterado até agora, assim, carrega o
 decfsz atras_2,1,0          ; valor inicial para a variável. Aqui, decrementa a
                             ; variável.
 goto rotin1                 ; Volta para rotin1.
 movwf atras_2,0             ; duas variáveis com o valor inicial.
 decfsz atras_3,1,0          ; Decrementa a variável.
 goto rotin1                 ; Volta para rotin1.
 return 1                    ; Fim da rotina atraso
; fim da rotina atraso 1


; Atraso 2
atraso2                      ; subrotina que realiza um atraso, o tempo é
                             ; definido pelo valor que o programa solicitante
                             ; atribui ao W.
 movwf atras_1,0             ; O valor de W é transferido para as 3 variáveis,
 movwf atras_2,0             ; no banco 00
rotin2
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 decfsz atras_1,1,0          ; decrementa a variável atras_1
 goto rotin2                 ; volta para rotin1
 movwf atras_1,0             ; W está inalterado até agora, assim, carrega o
 decfsz atras_2,1,0          ; valor inicial para a variável. Aqui, decrementa a
                             ; variável.
 goto rotin2                 ; Volta para rotin1.
 return 1                    ; Fim da rotina atraso
; fim da rotina atraso 2

; Média
media                        ; calcula a média de 64 números, isto é, média entre
                             ; as últimas 64 leituras do se sensor.
 movlb 0x01                  ; indica banco de memoria 1
transferencia                ; inicia a tranferencias das variáveis, de forma a
                             ; descartar a medida mais antiga, usando para isso
                             ; um contador para indicar qual a variável a ser
                             ; trocada.


leit_0                       ; valor da leitura para m_0
 movlw 0x00                  ; move 0h para W
 cpfseq contmed,1
 goto leit_1
 movf leitL,0,1
 movwf m_0L,1                ; leitL => m_0L
 movf leitH,0,1
 movwf m_0H,1                ; leitH => m_0H
 incf contmed,1,1            ; incrementa o contador da média contmed
 goto somas_1

leit_1                       ; valor da leitura para m_1
 movlw 0x01                  ; move 1h para W
 cpfseq contmed,1
 goto leit_2
 movf leitL,0,1
 movwf m_1L,1                ; leitL => m_1L
 movf leitH,0,1
 movwf m_1H,1                ; leitH => m_1H
 incf contmed,1,1            ; incrementa o contador da média contmed
 goto somas_1

leit_2                       ; valor da leitura para m_2
 movlw 0x02                  ; move 2h para W
 cpfseq contmed,1
 goto leit_3
 movf leitL,0,1
 movwf m_2L,1                ; leitL => m_2L
 movf leitH,0,1
 movwf m_2H,1                ; leitH => m_2H
 incf contmed,1,1            ; incrementa o contador da média contmed
 goto somas_1

leit_3                       ; valor da leitura para m_3
 movlw 0x03                  ; move 3h para W
 cpfseq contmed,1
 goto leit_4
 movf leitL,0,1
 movwf m_3L,1                ; leitL => m_3L
 movf leitH,0,1
 movwf m_3H,1                ; leitH => m_3H
```


unesp

```
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_4                     ; valor da leitura para m_4
 movlw 0x04                ; move 3h para W
 cpfseq contmed,1
 goto leit_5
 movf leitL,0,1
 movwf m_4L,1              ; leitL => m_4L
 movf leitH,0,1
 movwf m_4H,1              ; leitH => m_4H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_5                     ; valor da leitura para m_5
 movlw 0x05                ; move 5h para W
 cpfseq contmed,1
 goto leit_6
 movf leitL,0,1
 movwf m_5L,1              ; leitL => m_5L
 movf leitH,0,1
 movwf m_5H,1              ; leitH => m_5H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_6                     ; valor da leitura para m_6
 movlw 0x06                ; move 6h para W
 cpfseq contmed,1
 goto leit_7
 movf leitL,0,1
 movwf m_6L,1              ; leitL => m_6L
 movf leitH,0,1
 movwf m_6H,1              ; leitH => m_6H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_7                     ; valor da leitura para m_7
 movlw 0x07                ; move 7h para W
 cpfseq contmed,1
 goto leit_8
 movf leitL,0,1
 movwf m_7L,1              ; leitL => m_7L
 movf leitH,0,1
 movwf m_7H,1              ; leitH => m_7H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_8                     ; valor da leitura para m_8
 movlw 0x08                ; move 8h para W
 cpfseq contmed,1
 goto leit_9
 movf leitL,0,1
 movwf m_8L,1              ; leitL => m_8L
 movf leitH,0,1
 movwf m_8H,1              ; leitH => m_8H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_9                     ; valor da leitura para m_9
 movlw 0x09                ; move 9h para W
```


unesp

```
 cpfseq contmed,1
 goto leit_10
 movf leitL,0,1
 movwf m_9L,1              ; leitL => m_9L
 movf leitH,0,1
 movwf m_9H,1              ; leitH => m_9H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_10                    ; valor da leitura para m_10
 movlw 0x0A                ; move Ah para W
 cpfseq contmed,1
 goto leit_11
 movf leitL,0,1
 movwf m_10L,1             ; leitL => m_10L
 movf leitH,0,1
 movwf m_10H,1             ; leitH => m_10H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_11                    ; valor da leitura para m_11
 movlw 0x0B                ; move Bh para W
 cpfseq contmed,1
 goto leit_12
 movf leitL,0,1
 movwf m_11L,1             ; leitL => m_11L
 movf leitH,0,1
 movwf m_11H,1             ; leitH => m_11H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_12                    ; valor da leitura para m_12
 movlw 0x0C                ; move Ch para W
 cpfseq contmed,1
 goto leit_13
 movf leitL,0,1
 movwf m_12L,1             ; leitL => m_12L
 movf leitH,0,1
 movwf m_12H,1             ; leitH => m_12H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_13                    ; valor da leitura para m_13
 movlw 0x0D                ; move Dh para W
 cpfseq contmed,1
 goto leit_14
 movf leitL,0,1
 movwf m_13L,1             ; leitL => m_13L
 movf leitH,0,1
 movwf m_13H,1             ; leitH => m_13H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_14                    ; valor da leitura para m_14
 movlw 0x0E                ; move Eh para W
 cpfseq contmed,1
 goto leit_15
 movf leitL,0,1
 movwf m_14L,1             ; leitL => m_14L
 movf leitH,0,1
```


unesp

```
 movwf m_14H,1              ; leitH => m_14H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_15                     ; valor da leitura para m_15
 movlw 0x0F                 ; move Fh para W
 cpfseq contmed,1
 goto leit_16
 movf leitL,0,1
 movwf m_15L,1              ; leitL => m_15L
 movf leitH,0,1
 movwf m_15H,1              ; leitH => m_15H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_16                     ; valor da leitura para m_16
 movlw 0x10                 ; move 10h para W
 cpfseq contmed,1
 goto leit_17
 movf leitL,0,1
 movwf m_16L,1              ; leitL => m_16L
 movf leitH,0,1
 movwf m_16H,1              ; leitH => m_16H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_17                     ; valor da leitura para m_16
 movlw 0x11                 ; move 11h para W
 cpfseq contmed,1
 goto leit_18
 movf leitL,0,1
 movwf m_17L,1              ; leitL => m_17L
 movf leitH,0,1
 movwf m_17H,1              ; leitH => m_17H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_18                     ; valor da leitura para m_18
 movlw 0x12                 ; move 12h para W
 cpfseq contmed,1
 goto leit_19
 movf leitL,0,1
 movwf m_18L,1              ; leitL => m_18L
 movf leitH,0,1
 movwf m_18H,1              ; leitH => m_18H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_19                     ; valor da leitura para m_19
 movlw 0x13                 ; move 13h para W
 cpfseq contmed,1
 goto leit_20
 movf leitL,0,1
 movwf m_19L,1              ; leitL => m_19L
 movf leitH,0,1
 movwf m_19H,1              ; leitH => m_19H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_20                     ; valor da leitura para m_20
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 movlw 0x14              ; move 14h para W
 cpfseq contmed,1
 goto leit_21
 movf leitL,0,1
 movwf m_20L,1           ; leitL => m_20L
 movf leitH,0,1
 movwf m_20H,1           ; leitH => m_20H
 incf contmed,1,1        ; incrementa o contador da média contmed
 goto somas_1

leit_21                  ; valor da leitura para m_21
 movlw 0x15              ; move 15h para W
 cpfseq contmed,1
 goto leit_22
 movf leitL,0,1
 movwf m_21L,1           ; leitL => m_21L
 movf leitH,0,1
 movwf m_21H,1           ; leitH => m_21H
 incf contmed,1,1        ; incrementa o contador da média contmed
 goto somas_1

leit_22                  ; valor da leitura para m_22
 movlw 0x16              ; move 16h para W
 cpfseq contmed,1
 goto leit_23
 movf leitL,0,1
 movwf m_22L,1           ; leitL => m_22L
 movf leitH,0,1
 movwf m_22H,1           ; leitH => m_22H
 incf contmed,1,1        ; incrementa o contador da média contmed
 goto somas_1

leit_23                  ; valor da leitura para m_23
 movlw 0x17              ; move 17h para W
 cpfseq contmed,1
 goto leit_24
 movf leitL,0,1
 movwf m_23L,1           ; leitL => m_23L
 movf leitH,0,1
 movwf m_23H,1           ; leitH => m_23H
 incf contmed,1,1        ; incrementa o contador da média contmed
 goto somas_1

leit_24                  ; valor da leitura para m_24
 movlw 0x18              ; move 18h para W
 cpfseq contmed,1
 goto leit_25
 movf leitL,0,1
 movwf m_24L,1           ; leitL => m_24L
 movf leitH,0,1
 movwf m_24H,1           ; leitH => m_24H
 incf contmed,1,1        ; incrementa o contador da média contmed
 goto somas_1

leit_25                  ; valor da leitura para m_25
 movlw 0x19              ; move 19h para W
 cpfseq contmed,1
 goto leit_26
 movf leitL,0,1
 movwf m_25L,1           ; leitL => m_25L
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 movf leitH,0,1
 movwf m_25H,1              ; leitH => m_25H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_26                     ; valor da leitura para m_26
 movlw 0x1A                 ; move 1Ah para W
 cpfseq contmed,1
 goto leit_27
 movf leitL,0,1
 movwf m_26L,1              ; leitL => m_26L
 movf leitH,0,1
 movwf m_26H,1              ; leitH => m_26H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_27                     ; valor da leitura para m_27
 movlw 0x1B                 ; move 1Bh para W
 cpfseq contmed,1
 goto leit_28
 movf leitL,0,1
 movwf m_27L,1              ; leitL => m_27L
 movf leitH,0,1
 movwf m_27H,1              ; leitH => m_27H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_28                     ; valor da leitura para m_28
 movlw 0x1C                 ; move 1Ch para W
 cpfseq contmed,1
 goto leit_29
 movf leitL,0,1
 movwf m_28L,1              ; leitL => m_28L
 movf leitH,0,1
 movwf m_28H,1              ; leitH => m_28H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_29                     ; valor da leitura para m_29
 movlw 0x1D                 ; move 1Dh para W
 cpfseq contmed,1
 goto leit_30
 movf leitL,0,1
 movwf m_29L,1              ; leitL => m_29L
 movf leitH,0,1
 movwf m_29H,1              ; leitH => m_29H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_30                     ; valor da leitura para m_30
 movlw 0x1E                 ; move 1Eh para W
 cpfseq contmed,1
 goto leit_31
 movf leitL,0,1
 movwf m_30L,1              ; leitL => m_30L
 movf leitH,0,1
 movwf m_30H,1              ; leitH => m_30H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1
```


unesp

```
leit_31                 ; valor da leitura para m_31
 movlw 0x1F             ; move 1Fh para W
 cpfseq contmed,1
 goto leit_32
 movf leitL,0,1
 movwf m_31L,1          ; leitL => m_31L
 movf leitH,0,1
 movwf m_31H,1          ; leitH => m_31H
 incf contmed,1,1       ; incrementa o contador da média contmed
 goto somas_1

leit_32                 ; valor da leitura para m_32
 movlw 0x20             ; move 20h para W
 cpfseq contmed,1
 goto leit_33
 movf leitL,0,1
 movwf m_32L,1          ; leitL => m_32L
 movf leitH,0,1
 movwf m_32H,1          ; leitH => m_32H
 incf contmed,1,1       ; incrementa o contador da média contmed
 goto somas_1

leit_33                 ; valor da leitura para m_32
 movlw 0x21             ; move 21h para W
 cpfseq contmed,1
 goto leit_34
 movf leitL,0,1
 movwf m_33L,1          ; leitL => m_33L
 movf leitH,0,1
 movwf m_33H,1          ; leitH => m_33H
 incf contmed,1,1       ; incrementa o contador da média contmed
 goto somas_1

leit_34                 ; valor da leitura para m_34
 movlw 0x22             ; move 22h para W
 cpfseq contmed,1
 goto leit_35
 movf leitL,0,1
 movwf m_34L,1          ; leitL => m_34L
 movf leitH,0,1
 movwf m_34H,1          ; leitH => m_34H
 incf contmed,1,1       ; incrementa o contador da média contmed
 goto somas_1

 movlb 0x01             ; define o banco 1 com o de trabalho

leit_35                 ; valor da leitura para m_35
 movlw 0x23             ; move 23h para W
 cpfseq contmed,1
 goto leit_36
 movf leitL,0,1
 movwf m_35L,1          ; leitL => m_35L
 movf leitH,0,1
 movwf m_35H,1          ; leitH => m_35H
 incf contmed,1,1       ; incrementa o contador da média contmed
 goto somas_1

leit_36                 ; valor da leitura para m_36
 movlw 0x24             ; move 24h para W
 cpfseq contmed,1
```


unesp

```
 goto leit_37
 movf leitL,0,1
 movwf m_36L,1              ; leitL => m_36L
 movf leitH,0,1
 movwf m_36H,1              ; leitH => m_36H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_37                     ; valor da leitura para m_37
 movlw 0x25                 ; move 25h para W
 cpfseq contmed,1
 goto leit_38
 movf leitL,0,1
 movwf m_37L,1              ; leitL => m_37L
 movf leitH,0,1
 movwf m_37H,1              ; leitH => m_37H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_38                     ; valor da leitura para m_38
 movlw 0x26                 ; move 26h para W
 cpfseq contmed,1
 goto leit_39
 movf leitL,0,1
 movwf m_38L,1              ; leitL => m_38L
 movf leitH,0,1
 movwf m_38H,1              ; leitH => m_38H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_39                     ; valor da leitura para m_39
 movlw 0x27                 ; move 27h para W
 cpfseq contmed,1
 goto leit_40
 movf leitL,0,1
 movwf m_39L,1              ; leitL => m_39L
 movf leitH,0,1
 movwf m_39H,1              ; leitH => m_39H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_40                     ; valor da leitura para m_40
 movlw 0x28                 ; move 28h para W
 cpfseq contmed,1
 goto leit_41
 movf leitL,0,1
 movwf m_40L,1              ; leitL => m_40L
 movf leitH,0,1
 movwf m_40H,1              ; leitH => m_40H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_41                     ; valor da leitura para m_41
 movlw 0x29                 ; move 29h para W
 cpfseq contmed,1
 goto leit_42
 movf leitL,0,1
 movwf m_41L,1              ; leitL => m_41L
 movf leitH,0,1
 movwf m_41H,1              ; leitH => m_41H
```


unesp

```
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_42                    ; valor da leitura para m_42
 movlw 0x2A                ; move 2Ah para W
 cpfseq contmed,1
 goto leit_43
 movf leitL,0,1
 movwf m_42L,1             ; leitL => m_42L
 movf leitH,0,1
 movwf m_42H,1             ; leitH => m_42H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_43                    ; valor da leitura para m_43
 movlw 0x2B                ; move 2Bh para W
 cpfseq contmed,1
 goto leit_44
 movf leitL,0,1
 movwf m_43L,1             ; leitL => m_43L
 movf leitH,0,1
 movwf m_43H,1             ; leitH => m_43H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_44                    ; valor da leitura para m_44
 movlw 0x2C                ; move 2Ch para W
 cpfseq contmed,1
 goto leit_45
 movf leitL,0,1
 movwf m_44L,1             ; leitL => m_44L
 movf leitH,0,1
 movwf m_44H,1             ; leitH => m_44H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_45                    ; valor da leitura para m_45
 movlw 0x2D                ; move 2Dh para W
 cpfseq contmed,1
 goto leit_46
 movf leitL,0,1
 movwf m_45L,1             ; leitL => m_45L
 movf leitH,0,1
 movwf m_45H,1             ; leitH => m_45H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_46                    ; valor da leitura para m_46
 movlw 0x2E                ; move 2Eh para W
 cpfseq contmed,1
 goto leit_47
 movf leitL,0,1
 movwf m_46L,1             ; leitL => m_46L
 movf leitH,0,1
 movwf m_46H,1             ; leitH => m_46H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_47                    ; valor da leitura para m_47
 movlw 0x2F                ; move 2Fh para W
```


unesp

```
 cpfseq contmed,1
 goto leit_48
 movf leitL,0,1
 movwf m_47L,1              ; leitL => m_47L
 movf leitH,0,1
 movwf m_47H,1              ; leitH => m_47H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_48                     ; valor da leitura para m_48
 movlw 0x30                 ; move 30h para W
 cpfseq contmed,1
 goto leit_49
 movf leitL,0,1
 movwf m_48L,1              ; leitL => m_48L
 movf leitH,0,1
 movwf m_48H,1              ; leitH => m_48H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_49                     ; valor da leitura para m_49
 movlw 0x31                 ; move 31h para W
 cpfseq contmed,1
 goto leit_50
 movf leitL,0,1
 movwf m_49L,1              ; leitL => m_49L
 movf leitH,0,1
 movwf m_49H,1              ; leitH => m_49H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_50                     ; valor da leitura para m_50
 movlw 0x32                 ; move 32h para W
 cpfseq contmed,1
 goto leit_51
 movf leitL,0,1
 movwf m_50L,1              ; leitL => m_50L
 movf leitH,0,1
 movwf m_50H,1              ; leitH => m_50H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_51                     ; valor da leitura para m_51
 movlw 0x33                 ; move 33h para W
 cpfseq contmed,1
 goto leit_52
 movf leitL,0,1
 movwf m_51L,1              ; leitL => m_51L
 movf leitH,0,1
 movwf m_51H,1              ; leitH => m_51H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_52                     ; valor da leitura para m_52
 movlw 0x34                 ; move 34h para W
 cpfseq contmed,1
 goto leit_53
 movf leitL,0,1
 movwf m_52L,1              ; leitL => m_52L
 movf leitH,0,1
```


unesp

```
 movwf m_52H,1              ; leitH => m_52H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_53                     ; valor da leitura para m_53
 movlw 0x35                 ; move 35h para W
 cpfseq contmed,1
 goto leit_54
 movf leitL,0,1
 movwf m_53L,1              ; leitL => m_53L
 movf leitH,0,1
 movwf m_53H,1              ; leitH => m_53H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_54                     ; valor da leitura para m_54
 movlw 0x36                 ; move 36h para W
 cpfseq contmed,1
 goto leit_55
 movf leitL,0,1
 movwf m_54L,1              ; leitL => m_54L
 movf leitH,0,1
 movwf m_54H,1              ; leitH => m_54H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_55                     ; valor da leitura para m_55
 movlw 0x37                 ; move 37h para W
 cpfseq contmed,1
 goto leit_56
 movf leitL,0,1
 movwf m_55L,1              ; leitL => m_55L
 movf leitH,0,1
 movwf m_55H,1              ; leitH => m_55H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_56                     ; valor da leitura para m_56
 movlw 0x38                 ; move 38h para W
 cpfseq contmed,1
 goto leit_57
 movf leitL,0,1
 movwf m_56L,1              ; leitL => m_56L
 movf leitH,0,1
 movwf m_56H,1              ; leitH => m_56H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_57                     ; valor da leitura para m_57
 movlw 0x39                 ; move 39h para W
 cpfseq contmed,1
 goto leit_58
 movf leitL,0,1
 movwf m_57L,1              ; leitL => m_57L
 movf leitH,0,1
 movwf m_57H,1              ; leitH => m_57H
 incf contmed,1,1           ; incrementa o contador da média contmed
 goto somas_1

leit_58                     ; valor da leitura para m_58
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 movlw 0x3A                ; move 3Ah para W
 cpfseq contmed,1
 goto leit_59
 movf leitL,0,1
 movwf m_58L,1             ; leitL => m_58L
 movf leitH,0,1
 movwf m_58H,1             ; leitH => m_58H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_59                    ; valor da leitura para m_59
 movlw 0x3B                ; move 3Bh para W
 cpfseq contmed,1
 goto leit_60
 movf leitL,0,1
 movwf m_59L,1             ; leitL => m_59L
 movf leitH,0,1
 movwf m_59H,1             ; leitH => m_59H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_60                    ; valor da leitura para m_60
 movlw 0x3C                ; move 3Ch para W
 cpfseq contmed,1
 goto leit_61
 movf leitL,0,1
 movwf m_60L,1             ; leitL => m_60L
 movf leitH,0,1
 movwf m_60H,1             ; leitH => m_60H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_61                    ; valor da leitura para m_61
 movlw 0x3D                ; move 3Dh para W
 cpfseq contmed,1
 goto leit_62
 movf leitL,0,1
 movwf m_61L,1             ; leitL => m_61L
 movf leitH,0,1
 movwf m_61H,1             ; leitH => m_61H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_62                    ; valor da leitura para m_62
 movlw 0x3E                ; move 3Eh para W
 cpfseq contmed,1
 goto leit_63
 movf leitL,0,1
 movwf m_62L,1             ; leitL => m_62L
 movf leitH,0,1
 movwf m_62H,1             ; leitH => m_62H
 incf contmed,1,1          ; incrementa o contador da média contmed
 goto somas_1

leit_63                    ; valor da leitura para m_63
 movf leitL,0,1
 movwf m_63L,1             ; leitL => m_63L
 movf leitH,0,1
 movwf m_63H,1             ; leitH => m_63H
 clrf contmed,1            ; zera o contador das leituras
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
; fim das transferencias

somas_1                    ; as primeiras somas da média
                           ; aqui vamos somar as variáveis, usando o bit de
                           ; Carry, do registro Status

 movlb 0x01                ; indica banco de memoria 1

; som_0 = m_0 + m_1
 bcf status,c,0            ; zera o bit C de Status
 movf m_0L,0,1             ; W = m_0L
 movwf som_0L,1            ; som_0L = m_0L
 movf m_1L,0,1             ; W = m_1L
 addwf som_0L,1,1          ; som_0L = m_0L + m_1L
 movf m_0H,0,1             ; W = m_0H
 movwf som_0H,1            ; som_0H = m_0H
 movf m_1H,0,1             ; W = m_1H
 addwfc som_0H,1,1         ; som_0H = m_0H + m_1H + C

; som_1 = m_2 + m_3
 bcf status,c,0            ; zera o bit C de Status
 movf m_2L,0,1             ; W = m_2L
 movwf som_1L,1            ; som_1L = m_2L
 movf m_3L,0,1             ; W = m_3L
 addwf som_1L,1,1          ; som_1L = m_2L + m_3L
 movf m_2H,0,1             ; W = m_2H
 movwf som_1H,1            ; som_1H = m_4H
 movf m_3H,0,1             ; W = m_3H
 addwfc som_1H,1,1         ; som_1H = m_2H + m_3H + C

; som_2 = m_4 + m_5
 bcf status,c,0            ; zera o bit C de Status
 movf m_4L,0,1             ; W = m_4L
 movwf som_2L,1            ; som_2L = m_4L
 movf m_5L,0,1             ; W = m_5L
 addwf som_2L,1,1          ; som_2L = m_4L + m_5L
 movf m_4H,0,1             ; W = m_4H
 movwf som_2H,1            ; som_2H = m_4H
 movf m_5H,0,1             ; W = m_5H
 addwfc som_2H,1,1         ; som_2H = m_4H + m_5H + C

; som_3 = m_6 + m_7
 bcf status,c,0            ; zera o bit C de Status
 movf m_6L,0,1             ; W = m_6L
 movwf som_3L,1            ; som_3L = m_6L
 movf m_7L,0,1             ; W = m_7L
 addwf som_3L,1,1          ; som_3L = m_6L + m_7L
 movf m_6H,0,1             ; W = m_6H
 movwf som_3H,1            ; som_3H = m_6H
 movf m_7H,0,1             ; W = m_7H
 addwfc som_3H,1,1         ; som_3H = m_6H + m_7H + C

; som_4 = m_8 + m_9
 bcf status,c,0            ; zera o bit C de Status
 movf m_8L,0,1             ; W = m_8L
 movwf som_4L,1            ; som_4L = m_8L
 movf m_9L,0,1             ; W = m_17L
 addwf som_4L,1,1          ; som_4L = m_8L + m_9L
 movf m_8H,0,1             ; W = m_16H
```


unesp

```
 movwf som_4H,1              ; som_4H = m_8H
 movf m_9H,0,1              ; W = m_17H
 addwfc som_4H,1,1           ; som_4H = m_8H + m_9H + C

; som_5 = m_10 + m_11
 bcf status,c,0              ; zera o bit C de Status
 movf m_10L,0,1              ; W = m_10L
 movwf som_5L,1              ; som_5L = m_10L
 movf m_11L,0,1              ; W = m_11L
 addwf som_5L,1,1            ; som_5L = m_10L + m_11L
 movf m_10H,0,1              ; W = m_10H
 movwf som_5H,1              ; som_5H = m_10H
 movf m_11H,0,1              ; W = m_11H
 addwfc som_5H,1,1           ; som_5H = m_10H + m_11H + C

; som_6 = m_12 + m_13
 bcf status,c,0              ; zera o bit C de Status
 movf m_12L,0,1              ; W = m_12L
 movwf som_6L,1              ; som_6L = m_12L
 movf m_13L,0,1              ; W = m_13L
 addwf som_6L,1,1            ; som_6L = m_12L + m_13L
 movf m_12H,0,1              ; W = m_12H
 movwf som_6H,1              ; som_6H = m_12H
 movf m_13H,0,1              ; W = m_13H
 addwfc som_6H,1,1           ; som_6H = m_12H + m_13H + C

; som_7 = m_14 + m_15
 bcf status,c,0              ; zera o bit C de Status
 movf m_14L,0,1              ; W = m_14L
 movwf som_7L,1              ; som_7L = m_14L
 movf m_15L,0,1              ; W = m_15L
 addwf som_7L,1,1            ; som_7L = m_14L + m_15L
 movf m_14H,0,1              ; W = m_14H
 movwf som_7H,1              ; som_7H = m_14H
 movf m_15H,0,1              ; W = m_15H
 addwfc som_7H,1,1           ; som_7H = m_14H + m_15H + C

; som_8 = m_16 + m_17
 bcf status,c,0              ; zera o bit C de Status
 movf m_16L,0,1              ; W = m_16L
 movwf som_8L,1              ; som_8L = m_16L
 movf m_17L,0,1              ; W = m_17L
 addwf som_8L,1,1            ; som_8L = m_16L + m_17L
 movf m_16H,0,1              ; W = m_16H
 movwf som_8H,1              ; som_8H = m_16H
 movf m_17H,0,1              ; W = m_17H
 addwfc som_8H,1,1           ; som_8H = m_16H + m_17H + C

; som_9 = m_18 + m_19
 bcf status,c,0              ; zera o bit C de Status
 movf m_18L,0,1              ; W = m_18L
 movwf som_9L,1              ; som_9L = m_18L
 movf m_19L,0,1              ; W = m_19L
 addwf som_9L,1,1            ; som_9L = m_18L + m_19L
 movf m_18H,0,1              ; W = m_18H
 movwf som_9H,1              ; som_9H = m_18H
 movf m_19H,0,1              ; W = m_19H
 addwfc som_9H,1,1           ; som_9H = m_18H + m_19H + C

; som_10 = m_20 + m_21
```


unesp

```
 bcf status,c,0              ; zera o bit C de Status
 movf m_20L,0,1              ; W = m_20L
 movwf som_10L,1             ; som_10L = m_20L
 movf m_21L,0,1              ; W = m_21L
 addwf som_10L,1,1           ; som_10L = m_20L + m_21L
 movf m_20H,0,1              ; W = m_20H
 movwf som_10H,1             ; som_10H = m_20H
 movf m_21H,0,1              ; W = m_21H
 addwfc som_10H,1,1          ; som_10H = m_20H + m_21H + C

; som_11 = m_22 + m_23
 bcf status,c,0              ; zera o bit C de Status
 movf m_22L,0,1              ; W = m_22L
 movwf som_11L,1             ; som_11L = m_22L
 movf m_23L,0,1              ; W = m_23L
 addwf som_11L,1,1           ; som_11L = m_22L + m_23L
 movf m_22H,0,1              ; W = m_22H
 movwf som_11H,1             ; som_11H = m_22H
 movf m_23H,0,1              ; W = m_23H
 addwfc som_11H,1,1          ; som_11H = m_22H + m_23H + C

; som_12 = m_24 + m_25
 bcf status,c,0              ; zera o bit C de Status
 movf m_24L,0,1              ; W = m_24L
 movwf som_12L,1             ; som_12L = m_24L
 movf m_25L,0,1              ; W = m_25L
 addwf som_12L,1,1           ; som_12L = m_24L + m_25L
 movf m_24H,0,1              ; W = m_24H
 movwf som_12H,1             ; som_12H = m_24H
 movf m_25H,0,1              ; W = m_25H
 addwfc som_12H,1,1          ; som_12H = m_24H + m_25H + C

; som_13 = m_26 + m_27
 bcf status,c,0              ; zera o bit C de Status
 movf m_26L,0,1              ; W = m_26L
 movwf som_13L,1             ; som_13L = m_26L
 movf m_27L,0,1              ; W = m_27L
 addwf som_13L,1,1           ; som_13L = m_26L + m_27L
 movf m_26H,0,1              ; W = m_26H
 movwf som_13H,1             ; som_13H = m_26H
 movf m_27H,0,1              ; W = m_27H
 addwfc som_13H,1,1          ; som_13H = m_26H + m_27H + C

; som_14 = m_28 + m_29
 bcf status,c,0              ; zera o bit C de Status
 movf m_28L,0,1              ; W = m_28L
 movwf som_14L,1             ; som_14L = m_28L
 movf m_29L,0,1              ; W = m_29L
 addwf som_14L,1,1           ; som_14L = m_28L + m_29L
 movf m_28H,0,1              ; W = m_28H
 movwf som_14H,1             ; som_14H = m_28H
 movf m_29H,0,1              ; W = m_29H
 addwfc som_14H,1,1          ; som_14H = m_28H + m_29H + C

; som_15 = m_30 + m_31
 bcf status,c,0              ; zera o bit C de Status
 movf m_30L,0,1              ; W = m_30L
 movwf som_15L,1             ; som_15L = m_30L
 movf m_31L,0,1              ; W = m_31L
 addwf som_15L,1,1           ; som_15L = m_30L + m_31L
```




```
 movf m_30H,0,1          ; W = m_30H
 movwf som_15H,1         ; som_15H = m_30H
 movf m_31H,0,1          ; W = m_31H
 addwfc som_15H,1,1      ; som_15H = m_30H + m_31H + C

; som_16 = m_32 + m_33
 bcf status,c,0          ; zera o bit C de Status
 movf m_32L,0,1          ; W = m_32L
 movwf som_16L,1         ; som_16L = m_32L
 movf m_33L,0,1          ; W = m_33L
 addwf som_16L,1,1       ; som_16L = m_32L + m_33L
 movf m_32H,0,1          ; W = m_32H
 movwf som_16H,1         ; som_16H = m_32H
 movf m_33H,0,1          ; W = m_33H
 addwfc som_16H,1,1      ; som_16H = m_32H + m_33H + C

; som_17 = m_34 + m_35
 bcf status,c,0          ; zera o bit C de Status
 movf m_34L,0,1          ; W = m_34L
 movwf som_17L,1         ; som_17L = m_34L
 movf m_35L,0,1          ; W = m_35L
 addwf som_17L,1,1       ; som_17L = m_34L + m_35L
 movf m_34H,0,1          ; W = m_34H
 movwf som_17H,1         ; som_17H = m_34H
 movf m_35H,0,1          ; W = m_35H
 addwfc som_17H,1,1      ; som_17H = m_34H + m_35H + C

; som_18 = m_36 + m_37
 bcf status,c,0          ; zera o bit C de Status
 movf m_36L,0,1          ; W = m_36L
 movwf som_18L,1         ; som_18L = m_36L
 movf m_37L,0,1          ; W = m_37L
 addwf som_18L,1,1       ; som_18L = m_36L + m_37L
 movf m_36H,0,1          ; W = m_36H
 movwf som_18H,1         ; som_18H = m_36H
 movf m_37H,0,1          ; W = m_37H
 addwfc som_18H,1,1      ; som_18H = m_36H + m_37H + C

; som_19 = m_38 + m_39
 bcf status,c,0          ; zera o bit C de Status
 movf m_38L,0,1          ; W = m_38L
 movwf som_19L,1         ; som_19L = m_38L
 movf m_39L,0,1          ; W = m_39L
 addwf som_19L,1,1       ; som_19L = m_38L + m_39L
 movf m_38H,0,1          ; W = m_38H
 movwf som_19H,1         ; som_19H = m_38H
 movf m_39H,0,1          ; W = m_39H
 addwfc som_19H,1,1      ; som_19H = m_38H + m_39H + C

; som_20 = m_40 + m_41
 bcf status,c,0          ; zera o bit C de Status
 movf m_40L,0,1          ; W = m_40L
 movwf som_20L,1         ; som_20L = m_40L
 movf m_41L,0,1          ; W = m_41L
 addwf som_20L,1,1       ; som_20L = m_40L + m_41L
 movf m_40H,0,1          ; W = m_40H
 movwf som_20H,1         ; som_20H = m_40H
 movf m_41H,0,1          ; W = m_41H
 addwfc som_20H,1,1      ; som_20H = m_40H + m_41H + C
```


unesp

```
; som_21 = m_42 + m_43
 bcf status,c,0             ; zera o bit C de Status
 movf m_42L,0,1             ; W = m_42L
 movwf som_21L,1            ; som_21L = m_42L
 movf m_43L,0,1             ; W = m_43L
 addwf som_21L,1,1          ; som_21L = m_42L + m_43L
 movf m_42H,0,1             ; W = m_42H
 movwf som_21H,1            ; som_21H = m_42H
 movf m_43H,0,1             ; W = m_43H
 addwfc som_21H,1,1         ; som_21H = m_42H + m_43H + C

; som_22 = m_44 + m_45
 bcf status,c,0             ; zera o bit C de Status
 movf m_44L,0,1             ; W = m_44L
 movwf som_22L,1            ; som_22L = m_44L
 movf m_45L,0,1             ; W = m_45L
 addwf som_22L,1,1          ; som_22L = m_44L + m_45L
 movf m_44H,0,1             ; W = m_44H
 movwf som_22H,1            ; som_22H = m_44H
 movf m_45H,0,1             ; W = m_45H
 addwfc som_22H,1,1         ; som_22H = m_44H + m_45H + C

; som_23 = m_46 + m_47
 bcf status,c,0             ; zera o bit C de Status
 movf m_46L,0,1             ; W = m_46L
 movwf som_23L,1            ; som_23L = m_46L
 movf m_47L,0,1             ; W = m_47L
 addwf som_23L,1,1          ; som_23L = m_46L + m_47L
 movf m_46H,0,1             ; W = m_46H
 movwf som_23H,1            ; som_23H = m_46H
 movf m_47H,0,1             ; W = m_47H
 addwfc som_23H,1,1         ; som_23H = m_46H + m_47H + C

; som_24 = m_48 + m_49
 bcf status,c,0             ; zera o bit C de Status
 movf m_48L,0,1             ; W = m_48L
 movwf som_24L,1            ; som_24L = m_48L
 movf m_49L,0,1             ; W = m_49L
 addwf som_24L,1,1          ; som_24L = m_48L + m_49L
 movf m_48H,0,1             ; W = m_40H
 movwf som_24H,1            ; som_24H = m_48H
 movf m_49H,0,1             ; W = m_49H
 addwfc som_24H,1,1         ; som_24H = m_48H + m_49H + C

; som_25 = m_50 + m_51
 bcf status,c,0             ; zera o bit C de Status
 movf m_50L,0,1             ; W = m_50L
 movwf som_25L,1            ; som_25L = m_50L
 movf m_51L,0,1             ; W = m_51L
 addwf som_25L,1,1          ; som_25L = m_50L + m_51L
 movf m_50H,0,1             ; W = m_50H
 movwf som_25H,1            ; som_25H = m_50H
 movf m_51H,0,1             ; W = m_51H
 addwfc som_25H,1,1         ; som_25H = m_50H + m_51H + C

; som_26 = m_52 + m_53
 bcf status,c,0             ; zera o bit C de Status
 movf m_52L,0,1             ; W = m_52L
 movwf som_26L,1            ; som_26L = m_52L
 movf m_53L,0,1             ; W = m_53L
```


unesp

```
 addwf som_26L,1,1          ; som_26L = m_52L + m_53L
 movf m_52H,0,1             ; W = m_52H
 movwf som_26H,1            ; som_26H = m_52H
 movf m_53H,0,1             ; W = m_53H
 addwfc som_26H,1,1         ; som_26H = m_52H + m_53H + C

; som_27 = m_54 + m_55
 bcf status,c,0             ; zera o bit C de Status
 movf m_54L,0,1             ; W = m_54L
 movwf som_27L,1            ; som_27L = m_54L
 movf m_55L,0,1             ; W = m_55L
 addwf som_27L,1,1          ; som_27L = m_54L + m_55L
 movf m_54H,0,1             ; W = m_54H
 movwf som_27H,1            ; som_27H = m_54H
 movf m_55H,0,1             ; W = m_55H
 addwfc som_27H,1,1         ; som_27H = m_54H + m_55H + C

; som_28 = m_56 + m_57
 bcf status,c,0             ; zera o bit C de Status
 movf m_56L,0,1             ; W = m_56L
 movwf som_28L,1            ; som_28L = m_56L
 movf m_57L,0,1             ; W = m_57L
 addwf som_28L,1,1          ; som_28L = m_56L + m_57L
 movf m_56H,0,1             ; W = m_56H
 movwf som_28H,1            ; som_28H = m_56H
 movf m_57H,0,1             ; W = m_57H
 addwfc som_28H,1,1         ; som_28H = m_56H + m_57H + C

; som_29 = m_58 + m_59
 bcf status,c,0             ; zera o bit C de Status
 movf m_58L,0,1             ; W = m_58L
 movwf som_29L,1            ; som_29L = m_58L
 movf m_59L,0,1             ; W = m_59L
 addwf som_29L,1,1          ; som_29L = m_58L + m_59L
 movf m_58H,0,1             ; W = m_58H
 movwf som_29H,1            ; som_29H = m_58H
 movf m_59H,0,1             ; W = m_59H
 addwfc som_29H,1,1         ; som_29H = m_58H + m_59H + C

; som_30 = m_60 + m_61
 bcf status,c,0             ; zera o bit C de Status
 movf m_60L,0,1             ; W = m_60L
 movwf som_30L,1            ; som_30L = m_60L
 movf m_61L,0,1             ; W = m_61L
 addwf som_30L,1,1          ; som_30L = m_60L + m_61L
 movf m_60H,0,1             ; W = m_60H
 movwf som_30H,1            ; som_30H = m_60H
 movf m_61H,0,1             ; W = m_61H
 addwfc som_30H,1,1         ; som_30H = m_60H + m_61H + C

; som_31 = m_62 + m_63
 bcf status,c,0             ; zera o bit C de Status
 movf m_62L,0,1             ; W = m_62L
 movwf som_31L,1            ; som_31L = m_62L
 movf m_63L,0,1             ; W = m_63L
 addwf som_31L,1,1          ; som_31L = m_62L + m_63L
 movf m_62H,0,1             ; W = m_62H
 movwf som_31H,1            ; som_31H = m_62H
 movf m_63H,0,1             ; W = m_63H
 addwfc som_31H,1,1         ; som_31H = m_62H + m_63H + C
```


unesp

```
; fim das primeiras somas. Os resultados estão armazenados em som_0L, som_0H,
; ..., som_15L e som_15H, tendo em vista q se trata de númeors de 16Bits.


; divisão de som_0, som_1, som_2 e som_3 por 2 (som_X / 2)
                          ; esta operação é realizada com a anulação de um
                          ; bit menos significativo do valor de 16 bits.
                          ; A retirada de cada bit menos significativo, é o
                          ; mesmo que dividir por 2.

 movlb 0x01               ; indica banco de memoria 1

; som_0 / 2
 movlw 0xfe               ; W = 1111 1110
 andwf som_0L,1,1         ; W AND som_0L => som_0L
 rrncf som_0L,1,1         ; rotate som_0L para a direita
 rrncf som_0H,1,1         ; rotate W para a direita
 movf som_0H,0,1          ; W = som_0H
 andlw 0x80               ; W AND 1000 000 => W
 addwf som_0L,1,1         ; W + som_0L, na verdade, com as operações
                          ; anteriores, essa operação coloca os dois bit's
                          ; menos significativos de som_0H nos dois bits mais
                          ; significativos de som_0L
 movlw 0x7f               ; W = 0111 1111
 andwf som_0H,1,1         ; W AND som_0H, esta operação anula os dois bits
                          ; mais significativos, que forma rotacionados em
                          ; operações anteriores, e assim, termina-se a
                          ; divisão, pois, este dois bits já foram
                          ; transfereidos para som_0L.
; fim da divisão de som_0, os valores continuam armazenados em som_0L e som_0H

; som_1 / 2
 movlw 0xfe               ; W = 1111 1110
 andwf som_1L,1,1         ; W AND som_1L => som_1L
 rrncf som_1L,1,1         ; rotate som_L para a direita
 rrncf som_1H,1,1         ; rotate W para a direita
 movf som_1H,0,1          ; W = som_1H
 andlw 0x80               ; W AND 1000 000 => W
 addwf som_1L,1,1         ; W + som_1L, na verdade, com as operações
                          ; anteriores, essa operação coloca os dois bit's
                          ; menos significativos de som_1H nos dois bits mais
                          ; significativos de som_1L
 movlw 0x7f               ; W = 0111 1111
 andwf som_1H,1,1         ; W AND som_1H, esta operação anula os dois bits
                          ; mais significativos, que forma rotacionados em
                          ; operações anteriores, e assim, termina-se a
                          ; divisão, pois, este dois bits já foram
                          ; transfereidos para som_1L.
; fim da divisão de som_1, os valores continuam armazenados em som_1L e som_1H

; som_2 / 2
 movlw 0xfe               ; W = 1111 1110
 andwf som_2L,1,1         ; W AND som_2L => som_2L
 rrncf som_2L,1,1         ; rotate som_2L para a direita
 rrncf som_2H,1,1         ; rotate W para a direita
 movf som_2H,0,1          ; W = som_2H
 andlw 0x80               ; W AND 1000 000 => W
 addwf som_2L,1,1         ; W + som_2L, na verdade, com as operações
                          ; anteriores, essa operação coloca os dois bit's
```


unesp

```
                        ; menos significativos de som_2H nos dois bits mais
                        ; significativos de som_2L
 movlw 0x7f             ; W = 0111 1111
 andwf som_2H,1,1       ; W AND som_2H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_2L.
; fim da divisão de som_2, os valores continuam armazenados em som_2L e som_2H

; som_3 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_3L,1,1       ; W AND som_3L => som_3L
 rrncf som_3L,1,1       ; rotate som_3L para a direita
 rrncf som_3H,1,1       ; rotate W para a direita
 movf som_3H,0,1        ; W = som_3H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_3L,1,1       ; W + som_3L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_3H nos dois bits mais
                        ; significativos de som_3L
 movlw 0x7f             ; W = 0111 1111
 andwf som_3H,1,1       ; W AND som_3H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_3L.
; fim da divisão de som_3, os valores continuam armazenados em som_3L e som_3H

; som_4 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_4L,1,1       ; W AND som_4L => som_4L
 rrncf som_4L,1,1       ; rotate som_4L para a direita
 rrncf som_4H,1,1       ; rotate W para a direita
 movf som_4H,0,1        ; W = som_4H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_4L,1,1       ; W + som_4L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_4H nos dois bits mais
                        ; significativos de som_4L
 movlw 0x7f             ; W = 0111 1111
 andwf som_4H,1,1       ; W AND som_4H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_4L.
; fim da divisão de som_4, os valores continuam armazenados em som_4L e som_4H

; som_5 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_5L,1,1       ; W AND som_5L => som_5L
 rrncf som_5L,1,1       ; rotate som_5L para a direita
 rrncf som_5H,1,1       ; rotate W para a direita
 movf som_5H,0,1        ; W = som_5H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_5L,1,1       ; W + som_5L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_5H nos dois bits mais
                        ; significativos de som_5L
 movlw 0x7f             ; W = 0111 1111
```


unesp

```
 andwf som_5H,1,1           ; W AND som_5H, esta operação anula os dois bits
                            ; mais significativos, que forma rotacionados em
                            ; operações anteriores, e assim, termina-se a
                            ; divisão, pois, este dois bits já foram
                            ; transfereidos para som_5L.
; fim da divisão de som_5, os valores continuam armazenados em som_5L e som_5H

; som_6 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf som_6L,1,1           ; W AND som_6L => som_6L
 rrncf som_6L,1,1           ; rotate som_6L para a direita
 rrncf som_6H,1,1           ; rotate W para a direita
 movf som_6H,0,1            ; W = som_6H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf som_6L,1,1           ; W + som_6L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de som_6H nos dois bits mais
                            ; significativos de som_6L
 movlw 0x7f                 ; W = 0111 1111
 andwf som_6H,1,1           ; W AND som_6H, esta operação anula os dois bits
                            ; mais significativos, que forma rotacionados em
                            ; operações anteriores, e assim, termina-se a
                            ; divisão, pois, este dois bits já foram
                            ; transfereidos para som_6L.
; fim da divisão de som_6, os valores continuam armazenados em som_6L e som_6H

; som_7 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf som_7L,1,1           ; W AND som_7L => som_7L
 rrncf som_7L,1,1           ; rotate som_7L para a direita
 rrncf som_7H,1,1           ; rotate W para a direita
 movf som_7H,0,1            ; W = som_7H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf som_7L,1,1           ; W + som_7L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de som_7H nos dois bits mais
                            ; significativos de som_7L
 movlw 0x7f                 ; W = 0111 1111
 andwf som_7H,1,1           ; W AND som_7H, esta operação anula os dois bits
                            ; mais significativos, que forma rotacionados em
                            ; operações anteriores, e assim, termina-se a
                            ; divisão, pois, este dois bits já foram
                            ; transfereidos para som_7L.
; fim da divisão de som_7, os valores continuam armazenados em som_7L e som_7H

; som_8 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf som_8L,1,1           ; W AND som_8L => som_8L
 rrncf som_8L,1,1           ; rotate som_8L para a direita
 rrncf som_8H,1,1           ; rotate W para a direita
 movf som_8H,0,1            ; W = som_8H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf som_8L,1,1           ; W + som_8L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de som_8H nos dois bits mais
                            ; significativos de som_8L
 movlw 0x7f                 ; W = 0111 1111
 andwf som_8H,1,1           ; W AND som_8H, esta operação anula os dois bits
                            ; mais significativos, que forma rotacionados em
                            ; operações anteriores, e assim, termina-se a
```


unesp

```
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_8L.
; fim da divisão de som_8, os valores continuam armazenados em som_8L e som_8H

; som_9 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_9L,1,1       ; W AND som_9L => som_9L
 rrncf som_9L,1,1       ; rotate som_9L para a direita
 rrncf som_9H,1,1       ; rotate W para a direita
 movf som_9H,0,1        ; W = som_9H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_9L,1,1       ; W + som_9L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_9H nos dois bits mais
                        ; significativos de som_9L
 movlw 0x7f             ; W = 0111 1111
 andwf som_9H,1,1       ; W AND som_9H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_9L.
; fim da divisão de som_9, os valores continuam armazenados em som_9L e som_9H

; som_10 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_10L,1,1      ; W AND som_10L => som_10L
 rrncf som_10L,1,1      ; rotate som_10L para a direita
 rrncf som_10H,1,1      ; rotate W para a direita
 movf som_10H,0,1       ; W = som_10H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_10L,1,1      ; W + som_10L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_10H nos dois bits mais
                        ; significativos de som_10L
 movlw 0x7f             ; W = 0111 1111
 andwf som_10H,1,1      ; W AND som_10H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_10L.
; fim da divisão de som_10, os valores continuam armazenados em som_10L e
; som_10H

; som_11 / 2
 movlw 0xfe             ; W = 1111 1110
 andwf som_11L,1,1      ; W AND som_11L => som_11L
 rrncf som_11L,1,1      ; rotate som_11L para a direita
 rrncf som_11H,1,1      ; rotate W para a direita
 movf som_11H,0,1       ; W = som_11H
 andlw 0x80             ; W AND 1000 000 => W
 addwf som_11L,1,1      ; W + som_11L, na verdade, com as operações
                        ; anteriores, essa operação coloca os dois bit's
                        ; menos significativos de som_11H nos dois bits mais
                        ; significativos de som_11L
 movlw 0x7f             ; W = 0111 1111
 andwf som_11H,1,1      ; W AND som_11H, esta operação anula os dois bits
                        ; mais significativos, que forma rotacionados em
                        ; operações anteriores, e assim, termina-se a
                        ; divisão, pois, este dois bits já foram
                        ; transfereidos para som_11L.
```


unesp

```
; fim da divisão de som_11, os valores continuam armazenados em som_11L e
; som_11H

; som_12 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_12L,1,1         ; W AND som_12L => som_12L
 rrncf som_12L,1,1         ; rotate som_12L para a direita
 rrncf som_12H,1,1         ; rotate W para a direita
 movf som_12H,0,1          ; W = som_12H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_12L,1,1         ; W + som_12L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_12H nos dois bits mais
                           ; significativos de som_12L
 movlw 0x7f                ; W = 0111 1111
 andwf som_12H,1,1         ; W AND som_12H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_12L.
; fim da divisão de som_12, os valores continuam armazenados em som_12L e
; som_12H

; som_13 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_13L,1,1         ; W AND som_13L => som_13L
 rrncf som_13L,1,1         ; rotate som_13L para a direita
 rrncf som_13H,1,1          ; rotate W para a direita
 movf som_13H,0,1          ; W = som_13H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_13L,1,1         ; W + som_13L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_13H nos dois bits mais
                           ; significativos de som_13L
 movlw 0x7f                ; W = 0111 1111
 andwf som_13H,1,1         ; W AND som_13H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_13L.
; fim da divisão de som_13, os valores continuam armazenados em som_13L e
; som_13H

; som_14 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_14L,1,1         ; W AND som_14L => som_14L
 rrncf som_14L,1,1         ; rotate som_14L para a direita
 rrncf som_14H,1,1         ; rotate W para a direita
 movf som_14H,0,1          ; W = som_14H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_14L,1,1         ; W + som_14L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_14H nos dois bits mais
                           ; significativos de som_14L
 movlw 0x7f                ; W = 0111 1111
 andwf som_14H,1,1         ; W AND som_14H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_14L.
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
; fim da divisão de som_14, os valores continuam armazenados em som_14L e
; som_14H

; som_15 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf som_15L,1,1           ; W AND som_15L => som_15L
 rrncf som_15L,1,1           ; rotate som_15L para a direita
 rrncf som_15H,1,1           ; rotate W para a direita
 movf som_15H,0,1            ; W = som_15H
 andlw 0x80                  ; W AND 1000 000 => W
 addwf som_15L,1,1           ; W + som_15L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de som_15H nos dois bits mais
                             ; significativos de som_15L
 movlw 0x7f                  ; W = 0111 1111
 andwf som_15H,1,1           ; W AND som_15H, esta operação anula os dois bits
                             ; mais significativos, que forma rotacionados em
                             ; operações anteriores, e assim, termina-se a
                             ; divisão, pois, este dois bits já foram
                             ; transfereidos para som_15L.
; fim da divisão de som_15, os valores continuam armazenados em som_15L e
; som_15H

; som_16 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf som_16L,1,1           ; W AND som_16L => som_16L
 rrncf som_16L,1,1           ; rotate som_16L para a direita
 rrncf som_16H,1,1           ; rotate W para a direita
 movf som_16H,0,1            ; W = som_16H
 andlw 0x80                  ; W AND 1000 000 => W
 addwf som_16L,1,1           ; W + som_16L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de som_16H nos dois bits mais
                             ; significativos de som_16L
 movlw 0x7f                  ; W = 0111 1111
 andwf som_16H,1,1           ; W AND som_16H, esta operação anula os dois bits
                             ; mais significativos, que forma rotacionados em
                             ; operações anteriores, e assim, termina-se a
                             ; divisão, pois, este dois bits já foram
                             ; transfereidos para som_16L.
; fim da divisão de som_16, os valores continuam armazenados em som_16L e
; som_16H

; som_17 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf som_17L,1,1           ; W AND som_17L => som_17L
 rrncf som_17L,1,1           ; rotate som_17L para a direita
 rrncf som_17H,1,1           ; rotate W para a direita
 movf som_17H,0,1            ; W = som_17H
 andlw 0x80                  ; W AND 1000 000 => W
 addwf som_17L,1,1           ; W + som_17L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de som_17H nos dois bits mais
                             ; significativos de som_17L
 movlw 0x7f                  ; W = 0111 1111
 andwf som_17H,1,1           ; W AND som_17H, esta operação anula os dois bits
                             ; mais significativos, que forma rotacionados em
                             ; operações anteriores, e assim, termina-se a
                             ; divisão, pois, este dois bits já foram
                             ; transfereidos para som_17L.
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
; fim da divisão de som_17, os valores continuam armazenados em som_17L e
; som_17H

; som_18 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_18L,1,1         ; W AND som_18L => som_18L
 rrncf som_18L,1,1         ; rotate som_18L para a direita
 rrncf som_18H,1,1         ; rotate W para a direita
 movf som_18H,0,1          ; W = som_18H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_18L,1,1         ; W + som_18L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_18H nos dois bits mais
                           ; significativos de som_18L
 movlw 0x7f                ; W = 0111 1111
 andwf som_18H,1,1         ; W AND som_18H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_18L.
; fim da divisão de som_18, os valores continuam armazenados em som_18L e
; som_18H

; som_19 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_19L,1,1         ; W AND som_19L => som_19L
 rrncf som_19L,1,1         ; rotate som_19L para a direita
 rrncf som_19H,1,1         ; rotate W para a direita
 movf som_19H,0,1          ; W = som_19H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_19L,1,1         ; W + som_19L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_19H nos dois bits mais
                           ; significativos de som_19L
 movlw 0x7f                ; W = 0111 1111
 andwf som_19H,1,1         ; W AND som_19H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_19L.
; fim da divisão de som_19, os valores continuam armazenados em som_19L e
; som_19H

; som_20 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_20L,1,1         ; W AND som_20L => som_20L
 rrncf som_20L,1,1         ; rotate som_20L para a direita
 rrncf som_20H,1,1         ; rotate W para a direita
 movf som_20H,0,1          ; W = som_20H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_20L,1,1         ; W + som_20L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_20H nos dois bits mais
                           ; significativos de som_20L
 movlw 0x7f                ; W = 0111 1111
 andwf som_20H,1,1         ; W AND som_20H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_20L.
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
; fim da divisão de som_20, os valores continuam armazenados em som_20L e
; som_20H

; som_21 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_21L,1,1         ; W AND som_21L => som_21L
 rrncf som_21L,1,1         ; rotate som_21L para a direita
 rrncf som_21H,1,1         ; rotate W para a direita
 movf som_21H,0,1          ; W = som_21H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_21L,1,1         ; W + som_21L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_21H nos dois bits mais
                           ; significativos de som_21L
 movlw 0x7f                ; W = 0111 1111
 andwf som_21H,1,1         ; W AND som_21H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_21L.
; fim da divisão de som_21, os valores continuam armazenados em som_21L e
; som_21BH

; som_22 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_22L,1,1         ; W AND som_22L => som_22L
 rrncf som_22L,1,1         ; rotate som_22L para a direita
 rrncf som_22H,1,1         ; rotate W para a direita
 movf som_22H,0,1          ; W = som_22H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_22L,1,1         ; W + som_22L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_22H nos dois bits mais
                           ; significativos de som_22L
 movlw 0x7f                ; W = 0111 1111
 andwf som_22H,1,1         ; W AND som_22H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_22L.
; fim da divisão de som_22, os valores continuam armazenados em som_22L e
; som_22H

; som_23 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_23L,1,1         ; W AND som_23L => som_23L
 rrncf som_23L,1,1         ; rotate som_23L para a direita
 rrncf som_23H,1,1         ; rotate W para a direita
 movf som_23H,0,1          ; W = som_23H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_23L,1,1         ; W + som_23L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_23H nos dois bits mais
                           ; significativos de som_23L
 movlw 0x7f                ; W = 0111 1111
 andwf som_23H,1,1         ; W AND som_23H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_23L.
```


unesp

```
; fim da divisão de som_23, os valores continuam armazenados em som_23L e
; som_23H

; som_24 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_24L,1,1         ; W AND som_24L => som_24L
 rrncf som_24L,1,1         ; rotate som_24L para a direita
 rrncf som_24H,1,1         ; rotate W para a direita
 movf som_24H,0,1          ; W = som_24H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_24L,1,1         ; W + som_24L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_24H nos dois bits mais
                           ; significativos de som_24L
 movlw 0x7f                ; W = 0111 1111
 andwf som_24H,1,1         ; W AND som_24H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_24L.
; fim da divisão de som_24, os valores continuam armazenados em som_24L e
; som_24H

; som_25 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_25L,1,1         ; W AND som_25L => som_25L
 rrncf som_25L,1,1         ; rotate som_25L para a direita
 rrncf som_25H,1,1         ; rotate W para a direita
 movf som_25H,0,1          ; W = som_25H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_25L,1,1         ; W + som_25L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_25H nos dois bits mais
                           ; significativos de som_25L
 movlw 0x7f                ; W = 0111 1111
 andwf som_25H,1,1         ; W AND som_25H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_25L.
; fim da divisão de som_25, os valores continuam armazenados em som_25L e
; som_25H

; som_26 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_26L,1,1         ; W AND som_26L => som_26L
 rrncf som_26L,1,1         ; rotate som_26L para a direita
 rrncf som_26H,1,1         ; rotate W para a direita
 movf som_26H,0,1          ; W = som_26H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_26L,1,1         ; W + som_26L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_26H nos dois bits mais
                           ; significativos de som_26L
 movlw 0x7f                ; W = 0111 1111
 andwf som_26H,1,1         ; W AND som_26H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_26L.
```




```
; fim da divisão de som_26, os valores continuam armazenados em som_26L e
; som_26H

; som_27 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_27L,1,1         ; W AND som_27L => som_27L
 rrncf som_27L,1,1         ; rotate som_27L para a direita
 rrncf som_27H,1,1         ; rotate W para a direita
 movf som_27H,0,1          ; W = som_27H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_27L,1,1         ; W + som_27L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_27H nos dois bits mais
                           ; significativos de som_27L
 movlw 0x7f                ; W = 0111 1111
 andwf som_27H,1,1         ; W AND som_27H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_27L.
; fim da divisão de som_27, os valores continuam armazenados em som_27L e
; som_27H

; som_28 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_28L,1,1         ; W AND som_28L => som_28L
 rrncf som_28L,1,1         ; rotate som_28L para a direita
 rrncf som_28H,1,1         ; rotate W para a direita
 movf som_28H,0,1          ; W = som_28H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_28L,1,1         ; W + som_28L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_28H nos dois bits mais
                           ; significativos de som_28L
 movlw 0x7f                ; W = 0111 1111
 andwf som_28H,1,1         ; W AND som_28H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_28L.
; fim da divisão de som_28, os valores continuam armazenados em som_28L e
; som_28H

; som_29 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_29L,1,1         ; W AND som_29L => som_29L
 rrncf som_29L,1,1         ; rotate som_29L para a direita
 rrncf som_29H,1,1         ; rotate W para a direita
 movf som_29H,0,1          ; W = som_29H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_29L,1,1         ; W + som_29L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_29H nos dois bits mais
                           ; significativos de som_29L
 movlw 0x7f                ; W = 0111 1111
 andwf som_29H,1,1         ; W AND som_29H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_29L.
```


unesp

```
; fim da divisão de som_29, os valores continuam armazenados em som_29L e
; som_29H

; som_30 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_30L,1,1         ; W AND som_30L => som_30L
 rrncf som_30L,1,1         ; rotate som_30L para a direita
 rrncf som_30H,1,1         ; rotate W para a direita
 movf som_30H,0,1          ; W = som_30H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_30L,1,1         ; W + som_30L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_30H nos dois bits mais
                           ; significativos de som_30L
 movlw 0x7f                ; W = 0111 1111
 andwf som_30H,1,1         ; W AND som_30H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_30L.
; fim da divisão de som_30, os valores continuam armazenados em som_30L e
; som_30H

; som_31 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf som_31L,1,1         ; W AND som_31L => som_31L
 rrncf som_31L,1,1         ; rotate som_31L para a direita
 rrncf som_31H,1,1         ; rotate W para a direita
 movf som_31H,0,1          ; W = som_30H
 andlw 0x80                ; W AND 1000 000 => W
 addwf som_31L,1,1         ; W + som_31L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de som_31H nos dois bits mais
                           ; significativos de som_31L
 movlw 0x7f                ; W = 0111 1111
 andwf som_31H,1,1         ; W AND som_31H, esta operação anula os dois bits
                           ; mais significativos, que forma rotacionados em
                           ; operações anteriores, e assim, termina-se a
                           ; divisão, pois, este dois bits já foram
                           ; transfereidos para som_31L.
; fim da divisão de som_31, os valores continuam armazenados em som_31L e
; som_31H

; fim das divisões de som_X


; somas das primeiras médias (som_0,..., som_15)

somas2                     ; aqui vamos somar as variáveis, usando o bit de
                           ; Carry, do registro Status
; med_0 = som_0 + som_1
 bcf status,c,0            ; zera o bit C de Status
 movf som_0L,0,1           ; W = som_0L
 movwf med_0L,1            ; med_0L = som_0L
 movf som_1L,0,1           ; W = som_1L
 addwf med_0L,1,1          ; med_0L = som_0L + som_1L
 movf som_0H,0,1           ; W = som_0H
 movwf med_0H,1            ; med_0H = som_0H
 movf som_1H,0,1           ; W = som_1H
 addwfc med_0H,1,1         ; med_0H = som_0H + som_1H + C
```




```
; med_1 = som_2 + som_3
 bcf status,c,0              ; zera o bit C de Status
 movf som_2L,0,1             ; W = som_2L
 movwf med_1L,1              ; med_1L = som_2L
 movf som_3L,0,1             ; W = som_3L
 addwf med_1L,1,1            ; med_1L = som_2L + som_3L
 movf som_2H,0,1             ; W = som_0H
 movwf med_1H,1              ; med_1H = som_2H
 movf som_3H,0,1             ; W = som_3H
 addwfc med_1H,1,1           ; med_1H = som_2H + som_3H + C

; med_2 = som_4 + som_5
 bcf status,c,0              ; zera o bit C de Status
 movf som_4L,0,1             ; W = som_4L
 movwf med_2L,1              ; med_2L = som_4L
 movf som_5L,0,1             ; W = som_5L
 addwf med_2L,1,1            ; med_2L = som_4L + som_5L
 movf som_4H,0,1             ; W = som_4H
 movwf med_2H,1              ; med_2H = som_4H
 movf som_5H,0,1             ; W = som_5H
 addwfc med_2H,1,1           ; med_2H = som_4H + som_5H + C

; med_3 = som_6 + som_7
 bcf status,c,0              ; zera o bit C de Status
 movf som_6L,0,1             ; W = som_8L
 movwf med_3L,1              ; med_3L = som_8L
 movf som_7L,0,1             ; W = som_9L
 addwf med_3L,1,1            ; med_3L = som_8L + som_9L
 movf som_6H,0,1             ; W = som_8H
 movwf med_3H,1              ; med_3H = som_8H
 movf som_7H,0,1             ; W = som_9H
 addwfc med_3H,1,1           ; med_3H = som_8H + som_9H + C

; med_4 = som_8 + som_9
 bcf status,c,0              ; zera o bit C de Status
 movf som_8L,0,1             ; W = som_8L
 movwf med_4L,1              ; med_4L = som_8L
 movf som_9L,0,1             ; W = som_9L
 addwf med_4L,1,1            ; med_4L = som_8L + som_9L
 movf som_8H,0,1             ; W = som_8H
 movwf med_4H,1              ; med_4H = som_8H
 movf som_9H,0,1             ; W = som_9H
 addwfc med_4H,1,1           ; med_4H = som_8H + som_9H + C

; med_5 = som_10 + som_11
 bcf status,c,0              ; zera o bit C de Status
 movf som_10L,0,1            ; W = som_10L
 movwf med_5L,1              ; med_5L = som_10L
 movf som_11L,0,1            ; W = som_11L
 addwf med_5L,1,1            ; med_5L = som_10L + som_11L
 movf som_10H,0,1            ; W = som_10H
 movwf med_5H,1              ; med_5H = som_10H
 movf som_11H,0,1            ; W = som_11H
 addwfc med_5H,1,1           ; med_5H = som_10H + som_11H + C

; med_6 = som_12 + som_13
 bcf status,c,0              ; zera o bit C de Status
 movf som_12L,0,1            ; W = som_12L
 movwf med_6L,1              ; med_6L = som_12L
```


unesp

```
 movf som_13L,0,1           ; W = som_13L
 addwf med_6L,1,1           ; med_6L = som_12L + som_13L
 movf som_12H,0,1           ; W = som_12H
 movwf med_6H,1             ; med_6H = som_12H
 movf som_13H,0,1           ; W = som_13H
 addwfc med_6H,1,1          ; med_6H = som_12H + som_13H + C

; med_7 = som_14 + som_15
 bcf status,c,0             ; zera o bit C de Status
 movf som_14L,0,1           ; W = som_14L
 movwf med_7L,1             ; med_7L = som_14L
 movf som_15L,0,1           ; W = som_15L
 addwf med_7L,1,1           ; med_7L = som_14L + som_15L
 movf som_14H,0,1           ; W = som_14H
 movwf med_7H,1             ; med_7H = som_14H
 movf som_15H,0,1           ; W = som_15H
 addwfc med_7H,1,1          ; med_7H = som_14H + som_15H + C

; med_8 = som_16 + som_17
 bcf status,c,0             ; zera o bit C de Status
 movf som_16L,0,1           ; W = som_16L
 movwf med_8L,1             ; med_8L = som_16L
 movf som_17L,0,1           ; W = som_17L
 addwf med_8L,1,1           ; med_8L = som_16L + som_17L
 movf som_16H,0,1           ; W = som_16H
 movwf med_8H,1             ; med_8H = som_16H
 movf som_17H,0,1           ; W = som_17H
 addwfc med_8H,1,1          ; med_8H = som_16H + som_17H + C

; med_9 = som_18 + som_19
 bcf status,c,0             ; zera o bit C de Status
 movf som_18L,0,1           ; W = som_18L
 movwf med_9L,1             ; med_9L = som_18L
 movf som_19L,0,1           ; W = som_19L
 addwf med_9L,1,1           ; med_9L = som_18L + som_19L
 movf som_18H,0,1           ; W = som_18H
 movwf med_9H,1             ; med_9H = som_18H
 movf som_19H,0,1           ; W = som_19H
 addwfc med_9H,1,1          ; med_9H = som_18H + som_19H + C

; med_A = som_20 + som_21
 bcf status,c,0             ; zera o bit C de Status
 movf som_20L,0,1           ; W = som_20L
 movwf med_AL,1             ; med_AL = som_20L
 movf som_21L,0,1           ; W = som_21L
 addwf med_AL,1,1           ; med_AL = som_20L + som_21L
 movf som_20H,0,1           ; W = som_20H
 movwf med_AH,1             ; med_AH = som_20H
 movf som_21H,0,1           ; W = som_21H
 addwfc med_AH,1,1          ; med_AH = som_20H + som_21H + C

; med_B = som_22 + som_23
 bcf status,c,0             ; zera o bit C de Status
 movf som_22L,0,1           ; W = som_22L
 movwf med_BL,1             ; med_BL = som_22L
 movf som_23L,0,1           ; W = som_23L
 addwf med_BL,1,1           ; med_BL = som_22L + som_23L
 movf som_22H,0,1           ; W = som_22H
 movwf med_BH,1             ; med_BH = som_22H
 movf som_23H,0,1           ; W = som_23H
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 addwfc med_BH,1,1          ; med_BH = som_22H + som_23H + C

; med_C = som_24 + som_25
 bcf status,c,0             ; zera o bit C de Status
 movf som_24L,0,1           ; W = som_24L
 movwf med_CL,1             ; med_CL = som_24L
 movf som_25L,0,1           ; W = som_25L
 addwf med_CL,1,1           ; med_CL = som_24L + som_25L
 movf som_24H,0,1           ; W = som_24H
 movwf med_CH,1             ; med_CH = som_24H
 movf som_25H,0,1           ; W = som_25H
 addwfc med_CH,1,1          ; med_CH = som_24H + som_25H + C

; med_D = som_26 + som_27
 bcf status,c,0             ; zera o bit C de Status
 movf som_26L,0,1           ; W = som_26L
 movwf med_DL,1             ; med_DL = som_26L
 movf som_27L,0,1           ; W = som_27L
 addwf med_DL,1,1           ; med_DL = som_26L + som_27L
 movf som_26H,0,1           ; W = som_26H
 movwf med_DH,1             ; med_DH = som_26H
 movf som_27H,0,1           ; W = som_27H
 addwfc med_DH,1,1          ; med_DH = som_26H + som_27H + C

; med_E = som_28 + som_29
 bcf status,c,0             ; zera o bit C de Status
 movf som_28L,0,1           ; W = som_28L
 movwf med_EL,1             ; med_EL = som_28L
 movf som_29L,0,1           ; W = som_29L
 addwf med_EL,1,1           ; med_EL = som_28L + som_29L
 movf som_28H,0,1           ; W = som_28H
 movwf med_EH,1             ; med_EH = som_28H
 movf som_29H,0,1           ; W = som_29H
 addwfc med_EH,1,1          ; med_EH = som_28H + som_29H + C

; med_F = som_30 + som_31
 bcf status,c,0             ; zera o bit C de Status
 movf som_30L,0,1           ; W = som_30L
 movwf med_FL,1             ; med_FL = som_30L
 movf som_31L,0,1           ; W = som_31L
 addwf med_FL,1,1           ; med_FL = som_30L + som_31L
 movf som_30H,0,1           ; W = som_30H
 movwf med_FH,1             ; med_FH = som_30H
 movf som_31H,0,1           ; W = som_30H
 addwfc med_FH,1,1          ; med_FH = som_30H + som_31H + C

; fim das somas das médias


; divisão de med_1, med_2,..., med_4 por 2 (med_X / 2)
                            ; esta operação é realizada com a anulação de um
                            ; bit menos significativo do valor de 16 bits.
                            ; A retirada de cada bit menos significativo, é o
                            ; mesmo que dividir por 2.

 movlb 0x01                 ; indica banco de memoria 1

; med_0 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_0L,1,1           ; W AND med_0L => med_0L
```


unesp

```
 rrncf med_0L,1,1           ; rotate med_0L para a direita
 rrncf med_0H,1,1           ; rotate W para a direita
 movf med_0H,0,1            ; W = med_0H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_0L,1,1           ; W + med_0L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_0H nos dois bits mais
                            ; significativos de med_0L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_0H,1,1           ; W AND med_0H
; fim da divisão de med_0, os valores continuam armazenados em med_0L e med_0H

; med_1 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_1L,1,1           ; W AND med_1L => med_1L
 rrncf med_1L,1,1           ; rotate med_1L para a direita
 rrncf med_1H,1,1           ; rotate W para a direita
 movf med_1H,0,1            ; W = med_1H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_1L,1,1           ; W + med_1L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_1H nos dois bits mais
                            ; significativos de med_1L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_1H,1,1           ; W AND med_1H
; fim da divisão de med_1, os valores continuam armazenados em med_1L e med_1H

; med_2 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_2L,1,1           ; W AND med_2L => med_2L
 rrncf med_2L,1,1           ; rotate med_2L para a direita
 rrncf med_2H,1,1           ; rotate W para a direita
 movf med_2H,0,1            ; W = med_2H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_2L,1,1           ; W + med_2L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_2H nos dois bits mais
                            ; significativos de med_2L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_2H,1,1           ; W AND med_2H
; fim da divisão de med_2, os valores continuam armazenados em med_2L e med_2H

; med_3 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_3L,1,1           ; W AND med_3L => med_3L
 rrncf med_3L,1,1           ; rotate med_3L para a direita
 rrncf med_3H,1,1           ; rotate W para a direita
 movf med_3H,0,1            ; W = med_3H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_3L,1,1           ; W + med_3L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_3H nos dois bits mais
                            ; significativos de med_3L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_3H,1,1           ; W AND med_3H
; fim da divisão de med_3, os valores continuam armazenados em med_3L e med_3H

; med_4 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_4L,1,1           ; W AND med_4L => med_4L
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 rrncf med_4L,1,1           ; rotate med_4L para a direita
 rrncf med_4H,1,1           ; rotate W para a direita
 movf med_4H,0,1            ; W = med_4H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_4L,1,1           ; W + med_4L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_4H nos dois bits mais
                            ; significativos de med_4L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_4H,1,1           ; W AND med_4H
; fim da divisão de med_4, os valores continuam armazenados em med_4L e med_4H

; med_5 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_5L,1,1           ; W AND med_5L => med_5L
 rrncf med_5L,1,1           ; rotate med_5L para a direita
 rrncf med_5H,1,1           ; rotate W para a direita
 movf med_5H,0,1            ; W = med_5H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_5L,1,1           ; W + med_5L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_5H nos dois bits mais
                            ; significativos de med_5L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_5H,1,1           ; W AND med_5H
; fim da divisão de med_5, os valores continuam armazenados em med_5L e med_5H

; med_6 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_6L,1,1           ; W AND med_6L => med_6L
 rrncf med_6L,1,1           ; rotate med_6L para a direita
 rrncf med_6H,1,1           ; rotate W para a direita
 movf med_6H,0,1            ; W = med_6H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_6L,1,1           ; W + med_6L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_6H nos dois bits mais
                            ; significativos de med_6L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_6H,1,1           ; W AND med_6H
; fim da divisão de med_6, os valores continuam armazenados em med_6L e med_6H

; med_7 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_7L,1,1           ; W AND med_7L => med_7L
 rrncf med_7L,1,1           ; rotate med_7L para a direita
 rrncf med_7H,1,1           ; rotate W para a direita
 movf med_7H,0,1            ; W = med_7H
 andlw 0x80                 ; W AND 1000 000 => W
 addwf med_7L,1,1           ; W + med_7L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_7H nos dois bits mais
                            ; significativos de med_7L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_7H,1,1           ; W AND med_7H
; fim da divisão de med_7, os valores continuam armazenados em med_7L e med_7H

; med_8 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_8L,1,1           ; W AND med_8L => med_7L
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 rrncf med_8L,1,1            ; rotate med_8L para a direita
 rrncf med_8H,1,1            ; rotate W para a direita
 movf med_8H,0,1             ; W = med_8H
 andlw 0x80                  ; W AND 1000 000 => W
 addwf med_8L,1,1            ; W + med_8L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de med_8H nos dois bits mais
                             ; significativos de med_8L
 movlw 0x7f                  ; W = 0111 1111
 andwf med_8H,1,1            ; W AND med_8H
; fim da divisão de med_8, os valores continuam armazenados em med_8L e med_8H

; med_9 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf med_9L,1,1            ; W AND med_9L => med_9L
 rrncf med_9L,1,1            ; rotate med_9L para a direita
 rrncf med_9H,1,1            ; rotate W para a direita
 movf med_9H,0,1             ; W = med_9H
 andlw 0x80                  ; W AND 1000 000 => W
 addwf med_9L,1,1            ; W + med_9L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de med_9H nos dois bits mais
                             ; significativos de med_9L
 movlw 0x7f                  ; W = 0111 1111
 andwf med_9H,1,1            ; W AND med_9H
; fim da divisão de med_9, os valores continuam armazenados em med_9L e med_9H

; med_A / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf med_AL,1,1            ; W AND med_AL => med_AL
 rrncf med_AL,1,1            ; rotate med_AL para a direita
 rrncf med_AH,1,1            ; rotate W para a direita
 movf med_AH,0,1             ; W = med_AH
 andlw 0x80                  ; W AND 1000 000 => W
 addwf med_AL,1,1            ; W + med_AL, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de med_AH nos dois bits mais
                             ; significativos de med_AL
 movlw 0x7f                  ; W = 0111 1111
 andwf med_AH,1,1            ; W AND med_AH
; fim da divisão de med_A, os valores continuam armazenados em med_AL e med_AH

; med_B / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf med_BL,1,1            ; W AND med_BL => med_BL
 rrncf med_BL,1,1            ; rotate med_BL para a direita
 rrncf med_BH,1,1            ; rotate W para a direita
 movf med_BH,0,1             ; W = med_BH
 andlw 0x80                  ; W AND 1000 000 => W
 addwf med_BL,1,1            ; W + med_BL, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de med_BH nos dois bits mais
                             ; significativos de med_BL
 movlw 0x7f                  ; W = 0111 1111
 andwf med_BH,1,1            ; W AND med_BH
; fim da divisão de med_B, os valores continuam armazenados em med_BL e med_BH

; med_C / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf med_CL,1,1            ; W AND med_CL => med_CL
```




```
 rrncf med_CL,1,1          ; rotate med_CL para a direita
 rrncf med_CH,1,1          ; rotate W para a direita
 movf med_CH,0,1           ; W = med_CH
 andlw 0x80                ; W AND 1000 000 => W
 addwf med_CL,1,1          ; W + med_CL, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_CH nos dois bits mais
                           ; significativos de med_CL
 movlw 0x7f                ; W = 0111 1111
 andwf med_CH,1,1          ; W AND med_CH
; fim da divisão de med_C, os valores continuam armazenados em med_CL e med_CH

; med_D / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_DL,1,1          ; W AND med_DL => med_DL
 rrncf med_DL,1,1          ; rotate med_DL para a direita
 rrncf med_DH,1,1          ; rotate W para a direita
 movf med_DH,0,1           ; W = med_DH
 andlw 0x80                ; W AND 1000 000 => W
 addwf med_DL,1,1          ; W + med_DL, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_DH nos dois bits mais
                           ; significativos de med_DL
 movlw 0x7f                ; W = 0111 1111
 andwf med_DH,1,1          ; W AND med_DH
; fim da divisão de med_D, os valores continuam armazenados em med_DL e med_DH

; med_E / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_EL,1,1          ; W AND med_EL => med_EL
 rrncf med_EL,1,1          ; rotate med_EL para a direita
 rrncf med_EH,1,1          ; rotate W para a direita
 movf med_EH,0,1           ; W = med_EH
 andlw 0x80                ; W AND 1000 000 => W
 addwf med_EL,1,1          ; W + med_EL, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_EH nos dois bits mais
                           ; significativos de med_EL
 movlw 0x7f                ; W = 0111 1111
 andwf med_EH,1,1          ; W AND med_EH
; fim da divisão de med_E, os valores continuam armazenados em med_EL e med_EH

; med_F / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_FL,1,1          ; W AND med_FL => med_FL
 rrncf med_FL,1,1          ; rotate med_FL para a direita
 rrncf med_FH,1,1          ; rotate W para a direita
 movf med_FH,0,1           ; W = med_FH
 andlw 0x80                ; W AND 1000 000 => W
 addwf med_FL,1,1          ; W + med_FL, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_FH nos dois bits mais
                           ; significativos de med_FL
 movlw 0x7f                ; W = 0111 1111
 andwf med_FH,1,1          ; W AND med_FH
; fim da divisão de med_F, os valores continuam armazenados em med_FL e med_FH

; fim das divisões das médias
```


unesp

```
; soma das médias (med_T0 = med_0 + med_1,..., med_T3 = med_6 + med_7)
; med_T0 = med_0 + med_1
 bcf status,c,0              ; zera o bit C de Status
 movf med_0L,0,1             ; W = med_0L
 movwf med_T0L,1             ; med_T0L = med_0L
 movf med_1L,0,1             ; W = med_0L
 addwf med_T0L,1,1           ; med_T0L = med_0L + med_1L
 movf med_0H,0,1             ; W = med_0H
 movwf med_T0H,1             ; med_T0H = med_0H
 movf med_1H,0,1             ; W = med_1H
 addwfc med_T0H,1,1          ; med_T0H = med_0H + med_1H + C

; med_T1 = med_2 + med_3
 bcf status,c,0              ; zera o bit C de Status
 movf med_2L,0,1             ; W = med_2L
 movwf med_T1L,1             ; med_T1L = med_2L
 movf med_3L,0,1             ; W = med_3L
 addwf med_T1L,1,1           ; med_T1L = med_2L + med_3L
 movf med_2H,0,1             ; W = med_2H
 movwf med_T1H,1             ; med_T1H = med_2H
 movf med_3H,0,1             ; W = med_3H
 addwfc med_T1H,1,1          ; med_T1H = med_2H + med_3H + C

; med_T2 = med_4 + med_5
 bcf status,c,0              ; zera o bit C de Status
 movf med_4L,0,1             ; W = med_4L
 movwf med_T2L,1             ; med_T2L = med_4L
 movf med_5L,0,1             ; W = med_5L
 addwf med_T2L,1,1           ; med_T2L = med_4L + med_5L
 movf med_4H,0,1             ; W = med_4H
 movwf med_T2H,1             ; med_T2H = med_4H
 movf med_5H,0,1             ; W = med_5H
 addwfc med_T2H,1,1          ; med_T2H = med_4H + med_5H + C

; med_T3 = med_6 + med_7
 bcf status,c,0              ; zera o bit C de Status
 movf med_6L,0,1             ; W = med_6L
 movwf med_T3L,1             ; med_T3L = med_6L
 movf med_7L,0,1             ; W = med_7L
 addwf med_T3L,1,1           ; med_T3L = med_6L + med_7L
 movf med_6H,0,1             ; W = med_6H
 movwf med_T3H,1             ; med_T3H = med_6H
 movf med_7H,0,1             ; W = med_7H
 addwfc med_T3H,1,1          ; med_T3H = med_6H + med_7H + C

; med_T4 = med_8 + med_9
 bcf status,c,0              ; zera o bit C de Status
 movf med_8L,0,1             ; W = med_8L
 movwf med_T4L,1             ; med_T4L = med_8L
 movf med_9L,0,1             ; W = med_9L
 addwf med_T4L,1,1           ; med_T4L = med_8L + med_9L
 movf med_8H,0,1             ; W = med_8H
 movwf med_T4H,1             ; med_T4H = med_8H
 movf med_9H,0,1             ; W = med_9H
 addwfc med_T4H,1,1          ; med_T4H = med_8H + med_9H + C

; med_T5 = med_A + med_B
 bcf status,c,0              ; zera o bit C de Status
 movf med_AL,0,1             ; W = med_AL
 movwf med_T5L,1             ; med_T5L = med_AL
```


unesp

```
 movf med_BL,0,1            ; W = med_BL
 addwf med_T5L,1,1          ; med_T5L = med_AL + med_BL
 movf med_AH,0,1            ; W = med_AH
 movwf med_T5H,1            ; med_T5H = med_AH
 movf med_BH,0,1            ; W = med_BH
 addwfc med_T5H,1,1         ; med_T5H = med_AH + med_BH + C

; med_T6 = med_C + med_D
 bcf status,c,0             ; zera o bit C de Status
 movf med_CL,0,1            ; W = med_CL
 movwf med_T6L,1            ; med_T6L = med_CL
 movf med_DL,0,1            ; W = med_DL
 addwf med_T6L,1,1          ; med_T6L = med_CL + med_DL
 movf med_CH,0,1            ; W = med_CH
 movwf med_T6H,1            ; med_T6H = med_CH
 movf med_DH,0,1            ; W = med_DH
 addwfc med_T6H,1,1         ; med_T6H = med_CH + med_DH + C

; med_T7 = med_E + med_F
 bcf status,c,0             ; zera o bit C de Status
 movf med_EL,0,1            ; W = med_EL
 movwf med_T7L,1            ; med_T7L = med_EL
 movf med_FL,0,1            ; W = med_FL
 addwf med_T7L,1,1          ; med_T7L = med_EL + med_FL
 movf med_EH,0,1            ; W = med_EH
 movwf med_T7H,1            ; med_T7H = med_EH
 movf med_FH,0,1            ; W = med_FH
 addwfc med_T7H,1,1         ; med_T7H = med_EH + med_FH + C

; fim das somas das médias


; divisão de med_TX
; med_T0 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_T0L,1,1          ; W AND med_T0L => med_T0L
 rrncf med_T0L,1,1          ; rotate med_T0L para a direita
 rrncf med_T0H,1,1          ; rotate W para a direita
 movf med_T0H,0,1           ; W = med_T0H
 andlw 0x80                 ; W AND 1000 000
 addwf med_T0L,1,1          ; W + med_T0L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_T0H nos dois bits mais
                            ; significativos de med_T0L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_T0H,1,1          ; W AND med_T0H

; med_T1 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_T1L,1,1          ; W AND med_T1L => med_T1L
 rrncf med_T1L,1,1          ; rotate med_T1L para a direita
 rrncf med_T1H,1,1          ; rotate W para a direita
 movf med_T1H,0,1           ; W = med_T1H
 andlw 0x80                 ; W AND 1000 000
 addwf med_T1L,1,1          ; W + med_T1L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_T1H nos dois bits mais
                            ; significativos de med_T1L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_T1H,1,1          ; W AND med_T1H
```


unesp

```
; med_T2 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_T2L,1,1         ; W AND med_T2L => med_T2L
 rrncf med_T2L,1,1         ; rotate med_T2L para a direita
 rrncf med_T2H,1,1         ; rotate W para a direita
 movf med_T2H,0,1          ; W = med_T2H
 andlw 0x80                ; W AND 1000 000
 addwf med_T2L,1,1         ; W + med_T2L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_T2H nos dois bits mais
                           ; significativos de med_T2L
 movlw 0x7f                ; W = 0111 1111
 andwf med_T2H,1,1         ; W AND med_T2H

; med_T3 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_T3L,1,1         ; W AND med_T3L => med_T3L
 rrncf med_T3L,1,1         ; rotate med_T3L para a direita
 rrncf med_T3H,1,1         ; rotate W para a direita
 movf med_T3H,0,1          ; W = med_T3H
 andlw 0x80                ; W AND 1000 000
 addwf med_T3L,1,1         ; W + med_T3L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_T3H nos dois bits mais
                           ; significativos de med_T3L
 movlw 0x7f                ; W = 0111 1111
 andwf med_T3H,1,1         ; W AND med_T3H

; med_T4 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_T4L,1,1         ; W AND med_T4L => med_T4L
 rrncf med_T4L,1,1         ; rotate med_T4L para a direita
 rrncf med_T4H,1,1         ; rotate W para a direita
 movf med_T4H,0,1          ; W = med_T4H
 andlw 0x80                ; W AND 1000 000
 addwf med_T4L,1,1         ; W + med_T4L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_T4H nos dois bits mais
                           ; significativos de med_T4L
 movlw 0x7f                ; W = 0111 1111
 andwf med_T4H,1,1         ; W AND med_T4H

; med_T5 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_T5L,1,1         ; W AND med_T5L => med_T5L
 rrncf med_T5L,1,1         ; rotate med_T5L para a direita
 rrncf med_T5H,1,1         ; rotate W para a direita
 movf med_T5H,0,1          ; W = med_T5H
 andlw 0x80                ; W AND 1000 000
 addwf med_T5L,1,1         ; W + med_T5L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de med_T5H nos dois bits mais
                           ; significativos de med_T5L
 movlw 0x7f                ; W = 0111 1111
 andwf med_T5H,1,1         ; W AND med_T5H

; med_T6 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf med_T6L,1,1         ; W AND med_T6L => med_T6L
```


unesp

```
 rrncf med_T6L,1,1          ; rotate med_T6L para a direita
 rrncf med_T6H,1,1          ; rotate W para a direita
 movf med_T6H,0,1           ; W = med_T6H
 andlw 0x80                 ; W AND 1000 000
 addwf med_T6L,1,1          ; W + med_T6L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_T6H nos dois bits mais
                            ; significativos de med_T6L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_T6H,1,1          ; W AND med_T6H

; med_T7 / 2
 movlw 0xfe                 ; W = 1111 1110
 andwf med_T7L,1,1          ; W AND med_T7L => med_T7L
 rrncf med_T7L,1,1          ; rotate med_T7L para a direita
 rrncf med_T7H,1,1          ; rotate W para a direita
 movf med_T7H,0,1           ; W = med_T7H
 andlw 0x80                 ; W AND 1000 000
 addwf med_T7L,1,1          ; W + med_T7L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de med_T7H nos dois bits mais
                            ; significativos de med_T7L
 movlw 0x7f                 ; W = 0111 1111
 andwf med_T7H,1,1          ; W AND med_T7H

; fim das divisões das médias med_TX's


; Mais somas de médias
; medi_0 = med_T0 + med_T1
 bcf status,c,0             ; zera o bit C de Status
 movf med_T0L,0,1           ; W = med_T0L
 movwf medi_0L,1            ; medi_0L = med_T0L
 movf med_T1L,0,1           ; W = med_T1L
 addwf medi_0L,1,1          ; medi_0L = med_T0L + med_T1L
 movf med_T0H,0,1           ; W = med_T0H
 movwf medi_0H,1            ; medi_0H = med_T0H
 movf med_T1H,0,1           ; W = med_T1H
 addwfc medi_0H,1,1         ; medi_0H = med_T0H + med_T1H + C

; medi_1 = med_T2 + med_T3
 bcf status,c,0             ; zera o bit C de Status
 movf med_T2L,0,1           ; W = med_T2L
 movwf medi_1L,1            ; medi_1L = med_T2L
 movf med_T3L,0,1           ; W = med_T3L
 addwf medi_1L,1,1          ; medi_1L = med_T2L + med_T3L
 movf med_T2H,0,1           ; W = med_T2H
 movwf medi_1H,1            ; medi_1H = med_T2H
 movf med_T3H,0,1           ; W = med_T3H
 addwfc medi_1H,1,1         ; medi_1H = med_T2H + med_T3H + C

; medi_2 = med_T4 + med_T5
 bcf status,c,0             ; zera o bit C de Status
 movf med_T4L,0,1           ; W = med_T4L
 movwf medi_2L,1            ; medi_2L = med_T4L
 movf med_T5L,0,1           ; W = med_T5L
 addwf medi_2L,1,1          ; medi_2L = med_T4L + med_T5L
 movf med_T4H,0,1           ; W = med_T4H
 movwf medi_2H,1            ; medi_2H = med_T4H
 movf med_T5H,0,1           ; W = med_T5H
```


unesp

```
 addwfc medi_2H,1,1          ; medi_2H = med_T4H + med_T5H + C

; medi_3 = med_T6 + med_T7
 bcf status,c,0              ; zera o bit C de Status
 movf med_T6L,0,1            ; W = med_T6L
 movwf medi_3L,1             ; medi_3L = med_T6L
 movf med_T7L,0,1            ; W = med_T7L
 addwf medi_3L,1,1           ; medi_3L = med_T6L + med_T7L
 movf med_T6H,0,1            ; W = med_T6H
 movwf medi_3H,1             ; medi_3H = med_T6H
 movf med_T7H,0,1            ; W = med_T7H
 addwfc medi_3H,1,1          ; medi_3H = med_T6H + med_T70H + C

; fim das somas das segundas médias

; Segunda divisão das médias
; medi_0 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf medi_0L,1,1           ; W AND medi_0L => medi_0L
 rrncf medi_0L,1,1           ; rotate medi_0L para a direita
 rrncf medi_0H,1,1           ; rotate W para a direita
 movf medi_0H,0,1            ; W = medi_0H
 andlw 0x80                  ; W AND 1000 000
 addwf medi_0L,1,1           ; W + medi_0L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de medi_0H nos dois bits mais
                             ; significativos de medi_0L
 movlw 0x7f                  ; W = 0111 1111
 andwf medi_0H,1,1           ; W AND medi_0H

; medi_1 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf medi_1L,1,1           ; W AND medi_1L => medi_1L
 rrncf medi_1L,1,1           ; rotate medi_1L para a direita
 rrncf medi_1H,1,1           ; rotate W para a direita
 movf medi_1H,0,1            ; W = medi_1H
 andlw 0x80                  ; W AND 1000 000
 addwf medi_1L,1,1           ; W + medi_1L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de medi_1H nos dois bits mais
                             ; significativos de medi_1L
 movlw 0x7f                  ; W = 0111 1111
 andwf medi_1H,1,1           ; W AND medi_1H

; medi_2 / 2
 movlw 0xfe                  ; W = 1111 1110
 andwf medi_2L,1,1           ; W AND medi_2L => medi_2L
 rrncf medi_2L,1,1           ; rotate medi_2L para a direita
 rrncf medi_2H,1,1           ; rotate W para a direita
 movf medi_2H,0,1            ; W = medi_2H
 andlw 0x80                  ; W AND 1000 000
 addwf medi_2L,1,1           ; W + medi_2L, na verdade, com as operações
                             ; anteriores, essa operação coloca os dois bit's
                             ; menos significativos de medi_2H nos dois bits mais
                             ; significativos de medi_2L
 movlw 0x7f                  ; W = 0111 1111
 andwf medi_2H,1,1           ; W AND medi_2H

; medi_3 / 2
 movlw 0xfe                  ; W = 1111 1110
```


unesp

```
 andwf medi_3L,1,1          ; W AND medi_3L => medi_3L
 rrncf medi_3L,1,1          ; rotate medi_3L para a direita
 rrncf medi_3H,1,1          ; rotate W para a direita
 movf medi_3H,0,1           ; W = medi_3H
 andlw 0x80                 ; W AND 1000 000
 addwf medi_3L,1,1          ; W + medi_3L, na verdade, com as operações
                            ; anteriores, essa operação coloca os dois bit's
                            ; menos significativos de medi_3H nos dois bits mais
                            ; significativos de medi_3L
 movlw 0x7f                 ; W = 0111 1111
 andwf medi_3H,1,1          ; W AND medi_3H

; fim das segundas divisões das segundas médias


; por falta de espaço no banco 1 da Ram, vamos passar os dados para o banco 2,
; usando o acumulador. Vamos usar o mesmo nome de variáveis
; tranferência da banco de memória
 movlb 0x01                 ; indica banco de memoria 1
 movf medi_0L,0,1           ; WREG = medi_0L
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_0L,1            ; medi_0L = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_0H,0,1           ; WREG = medi_0H
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_0H,1            ; medi_0H = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_1L,0,1           ; WREG = medi_1L
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_1L,1            ; medi_1L = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_1H,0,1           ; WREG = medi_1H
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_1H,1            ; medi_1H = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_2L,0,1           ; WREG = medi_2L
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_2L,1            ; medi_2L = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_2H,0,1           ; WREG = medi_2H
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_2H,1            ; medi_2H = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_3L,0,1           ; WREG = medi_3L
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_3L,1            ; medi_3L = WREG

 movlb 0x01                 ; indica banco de memoria 1
 movf medi_3H,0,1           ; WREG = medi_3H
 movlb 0x02                 ; indica o banco 2 da memória Ram
 movwf medi_3H,1            ; medi_3H = WREG

; fim das tranferências do banco 1 para o banco 2
```


unesp

```
 movlb 0x02                ; indica o banco 2 da memória Ram

; soma das médias (med_T = medi_T0 + medi_T1 + medi_T2 + medi_T3)
; media_0 = medi_0 + medi_1
 bcf status,c,0            ; zera o bit C de Status
 movf medi_0L,0,1          ; W = medi_0L
 movwf media_0L,1          ; media_0L = medi_0L
 movf medi_1L,0,1          ; W = medi_1L
 addwf media_0L,1,1        ; media_0L = medi_0L + medi_1L
 movf medi_0H,0,1          ; W = medi_0H
 movwf media_0H,1          ; media_0H = medi_0H
 movf medi_1H,0,1          ; W = medi_1H
 addwfc media_0H,1,1       ; media_0H = medi_0H + medi_1H + C

; media_1 = medi_2 + medi_3
 bcf status,c,0            ; zera o bit C de Status
 movf medi_2L,0,1          ; W = medi_2L
 movwf media_1L,1          ; media_1L = medi_2L
 movf medi_3L,0,1          ; W = medi_3L
 addwf media_1L,1,1        ; media_1L = medi_2L + medi_3L
 movf medi_2H,0,1          ; W = medi_2H
 movwf media_1H,1          ; media_1H = medi_2H
 movf medi_3H,0,1          ; W = medi_3H
 addwfc media_1H,1,1       ; media_1H = medi_2H + medi_3H + C

; fim das somas das terceira méidas

; Terceira divisão das médias
; media_0 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf media_0L,1,1        ; W AND media_0L => media_0L
 rrncf media_0L,1,1        ; rotate media_0L para a direita
 rrncf media_0H,1,1        ; rotate W para a direita
 movf media_0H,0,1         ; W = media_0H
 andlw 0x80                ; W AND 1000 000
 addwf media_0L,1,1        ; W + media_0L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de media_0H nos dois bits
                           ; mais significativos de media_0L
 movlw 0x7f                ; W = 0111 1111
 andwf media_0H,1,1        ; W AND media_0H

; medi_1 / 2
 movlw 0xfe                ; W = 1111 1110
 andwf media_1L,1,1        ; W AND media_1L => media_1L
 rrncf media_1L,1,1        ; rotate media_1L para a direita
 rrncf media_1H,1,1        ; rotate W para a direita
 movf media_1H,0,1         ; W = media_1H
 andlw 0x80                ; W AND 1000 000
 addwf media_1L,1,1        ; W + media_1L, na verdade, com as operações
                           ; anteriores, essa operação coloca os dois bit's
                           ; menos significativos de media_1H nos dois bits
                           ; mais significativos de media_1L
 movlw 0x7f                ; W = 0111 1111
 andwf media_1H,1,1        ; W AND media_1H

; fim da divisão das terceiras médias

; Soma da média total (med_T = media_0 + media_1)
```


cm 1 2 3 4 5 6 7 8 unesp 12 13 14 15 16 17 18

```
 bcf status,c,0              ; zera o bit C de Status
 movf media_0L,0,1           ; W = media_0L
 movwf med_TL,1              ; med_TL = media_0L
 movf media_1L,0,1           ; W = media_1L
 addwf med_TL,1,1            ; med_TL = media_1L + media_2L
 movf media_0H,0,1           ; W = media_0H
 movwf med_TH,1              ; med_TH = med_T0H
 movf media_1H,0,1           ; W = media_1H
 addwfc med_TH,1,1           ; med_TH = media_0H + media_1H + C
; fim da soma de med_T = media_0 + media_1

; temos agora que repassar esses dados para o banco 1 da ram
 movlb 0x02                  ; indica o banco de memória 2
 movf med_TL,0,1             ; WREG = med_TL
 movlb 0x01                  ; indica o banco de memória 1
 movwf med_TL,1              ; med_TL = WREG

 movlb 0x02                  ; indica o banco de memória 2
 movf med_TH,0,1             ; WREG = med_TH
 movlb 0x01                  ; indica o banco de memória 1
 movwf med_TH,1              ; med_TH = WREG

 return                      ; retorna da sub rotina
; fim da soma de med_T, os valores estão armazenados em med_TL e med_TH do
; banco 2 da Ram.

; fim da média, seu valor está armazenado em med_TL e med_TH e representa a
média
; proporcional das 64 últimas leituras.


 end
; fim do programa
```

APÊNDICE 3

A3. ARTIGO SUBMETIDO E ACEITO EM CONGRESSO INTERNACIONAL

A3.1. Introdução

Este artigo foi submetido e aceito no II Congresso Latino-americano de Engenharia Biomédica, realizado em Havana-Cuba, em Maio de 2001.

Ele foi escrito quando com resultados parciais da pesquisa, onde não havia sido ainda utilizado o microcontrolador PIC 18, e o circuito com FPGA estava imcompleto, em relação aos circuitos atuais.

A3.2. O Artigo

DESENVOLVIMENTO DE UM SENSOR A CORDA VIBRANTE PARA APLICAÇÕES EM ENGENHARIA DE REABILITAÇÃO

Tony I. Silva*, Aparecido A. Carvalho, Debora D. Leister, Alexandre C. R. Silva

Departamento de Engenharia Elétrica de Ilha Solteira – UNESP
Caixa Postal 31, 15385-000 – Ilha Solteira (SP) - Brasil
*e-mail: tysilva@dee.feis.unesp.br

RESUMO

Foi implementado um sensor à corda vibrante e circuitos de condicionamento de sinais. Construído em estrutura de dura-alumínio, instrumentará um dinamômetro, objetivando-se medir a soma das forças exercidas pelos dedos das mãos. Trata-se de uma aplicação inovadora em Engenharia de Reabilitação. O artigo apresenta os conceitos fundamentais do sensor, descreve suas características construtivas e dois circuitos distintos de condicionamento de sinais, empregando microcontrolador e dispositivos lógicos programáveis (FPGA). O sensor apresentou resposta quase linear na faixa de 0 a 100 N, boa repetibilidade e resolução de 1 N. É robusto, compacto e seu tamanho é adequado para utilização em dinamômetros biomédicos.

Palavras chave: *Sensor à Corda Vibrante, Dinamômetro Biomédico, Engenharia de Reabilitação, FPGA, Microcontrolador.*

1. INTRODUÇÃO

Há muitas aplicações na Engenharia de Reabilitação nas quais se faz necessário o monitoramento de forças exercidas pelos membros superiores e inferiores dos pacientes, tais como os sistemas artificiais de reabilitação. A quantificação da real necessidade de estímulo artificial, que o paciente precisa, é feita através de sensores. Um sistema de controle relaciona as forças mensuradas com a intensidade de pulsos elétricos que serão aplicados a um paciente [1].

Um projeto em desenvolvimento em nosso laboratório consiste em instrumentar um dinamômetro biomédico com sensores à corda vibrante, objetivando a medição de forças exercidas pelos dedos de pacientes (Figura 1). A força a ser medida está compreendida na faixa de 0 a 100 N.

A técnica de corda vibrante é conhecida há, pelo menos, 100 anos, sendo utilizada no desenvolvimento de sensores de força, deslocamento, torque, pressão, com grande precisão e confiabilidade [2]. As vantagens do sensor à corda vibrante em relação ao extensômetro estão relacionadas a sua grande robustez, longa vida útil e facilidade de instalação. Por isto ele é largamente empregado em barragens, minas, escavações, estruturas prediais, entre outras.

O princípio de funcionamento e a teoria do sensor à corda vibrante são simples. A evolução tecnológica do sensor ocorreu em função das técnicas de construção, processo de fabricação e principalmente no sistema de excitação e leitura do sinal [3]. Uma grande inovação descrita por Feng *et. al.* [4] consistiu em conduzir o sinal do sensor a um sistema de controle por meio de fibra óptica, tornando o sensor praticamente imune a interferências eletromagnéticas. Pesquisas recentes tem empregado a técnica à corda vibrante no desenvolvimento de novos sensores (viscosímetros e densímetros) [5], com sensíveis modificações em relação aos primeiros dispositivos.

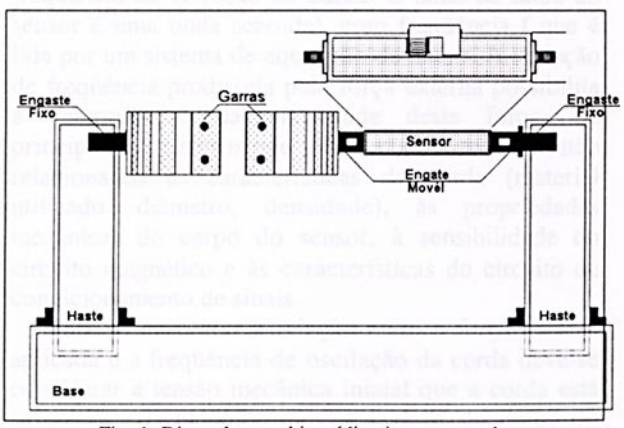


Fig. 1. Dinamômetro biomédico instrumentado com sensor à corda vibrante.

Na seção seguinte são descritos os detalhes de construção do sensor e dois circuitos de condicionamento de sinais implementados com um microcontrolador e com dispositivos lógicos programáveis.

Um microcontrolador é essencialmente um computador em um único chip, possuindo em sua estrutura interna circuitos de entrada/saída, a unidade de processamento, memória permanente e temporária, circuito de relógio e tempo, interfaces serial, entradas e saídas analógicas e conversores AD e DA (dependendo do modelo adotado), mas desprovido de monitores e discos rígidos. Não podem carregar

programas como jogos e utilitários, mas somente programas muitos específicos.

Os microcontroladores substituem com grande eficiência até mesmo circuitos inteiros e são extremamente flexíveis por possuírem funções que são facilmente modificadas via software.

Dispositivos lógicos programáveis (FPGA) são componentes formados por um enorme número de portas lógicas que podem ser interligadas através de software, podendo formar circuitos digitais específicos. Suas linguagens de programação mais conhecidas são a VHDL, a Verilog e a AHDL. A linguagem VHDL se tornou um padrão do IEEE (IEEE-1076) em 1987.

A utilização de FPGA se justifica por suas grandes vantagens. Com o VHDL pode-se realizar projetos em menor espaço de tempo, pois se elimina arranjos com componentes discretos, provendo uma descrição em alto-nível para projetar lógicas complexas, que requerem milhares de portas lógicas, além de suportar metodologia de projeto modular e vários níveis de hierarquia. O sistema FPGA utilizado neste trabalho foi o ambiente de projeto Max+Plus II da Altera, com linguagem AHDL. Com este sistema pode-se efetuar a descrição do circuito digital e sua simulação [6].

2. METODOLOGIA

O princípio de funcionamento de um sensor à corda vibrante fundamenta-se no comportamento de uma corda tensionada cuja freqüência natural de oscilação é dada pela Equação (1):

$$f = \frac{1}{2L}\left(\frac{T}{\rho}\right)^{1/2} \qquad (1)$$

sendo L é o comprimento da corda, ρ a sua massa por comprimento e T a força que a tenciona.

Com a variação de um dos três parâmetros (T, ρ ou L) haverá a mudança na freqüência f de oscilação. Esta alteração pode ser "sentida" por um circuito magnético instalado próximo da corda oscilante, conforme ilustra a Figura 2. O circuito magnético é constituído pela bobina sensora.

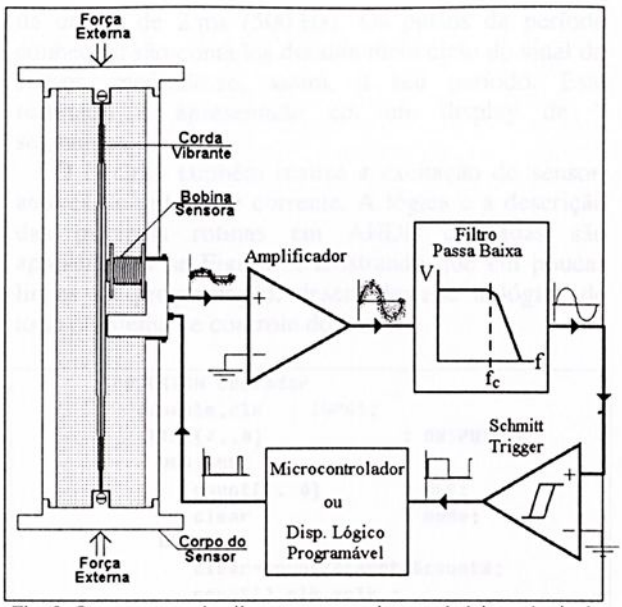


Fig. 2. O sensor a corda vibrante e o seu sistema de leitura de sinal. Ao lado de cada estágio está representada a forma de onda da saída do estágio. Note que a bobina sensora possui uma saída e uma entrada.

Nos sensores mais comuns, a tensão mecânica aplicada sobre a corda é a principal variável. Esta tensão pode ser alterada mediante uma força externa aplicada ao corpo do sensor. A bobina sensora é utilizada para iniciar a oscilação da corda, com um pulso elétrico, logo após, ela é usada para captar a freqüência de vibração da corda. O sinal de saída do sensor é uma onda senoidal, com freqüência f que é lida por um sistema de aquisição de dados. A variação de freqüência produzida pela força externa possibilita a determinação da magnitude desta força. Os principais parâmetros que influenciam a medida estão relacionados às características da corda (material utilizado, diâmetro, densidade), às propriedades mecânicas do corpo do sensor, à sensibilidade do circuito magnético e às características do circuito de condicionamento de sinais.

Para se encontrar a relação entre a força externa aplicada e a freqüência de oscilação da corda deve-se considerar a tensão mecânica inicial que a corda está submetida, os módulos de elasticidade do corpo do sensor e o da corda, o comprimento da corda e, evidentemente, a força externa. A Equação (2) mostra a equação que relaciona estas grandezas.

$$f = \frac{1}{2L}\sqrt{\frac{T_0 - \frac{k_1}{k_1 + k_2}F}{\rho}} \qquad (2)$$

sendo T_0 a tensão mecânica inicial a que a corda foi submetida, F a força externa aplicada ao sensor, k_1 e k_2 constantes da corda e do corpo. Estas constantes são calculadas pela Equação (3):

$$k = \frac{EA}{l} \quad (3)$$

sendo E é o módulo de elasticidade, A a área transversal e l o comprimento.

Neste projeto, construiu-se um sensor em dura alumínio, de formato cilíndrico de paredes de espessura de 1,4 mm, com duas tampas rosqueáveis. A corda é presa e tencionada por meio das tampas. Utilizou-se uma corda em aço inox de 0,2 mm de diâmetro, e um comprimento de 0,1 m. Todo este sistema permitiu uma freqüência natural de oscilação próxima de 600 Hz, podendo ser regulada por meio de parafusos e da tensão aplicada sobre a corda.

O amplificador e o filtro passa-baixa foram implementados com amplificadores operacionais (TL 082 e TL 084 respectivamente). O filtro é de oitava ordem. O estágio Schmitt Trigger foi realizado com o comparador LM 311.

O circuito de controle e leitura do sensor foi implementado de duas formas diferentes, uma utilizando o microcontrolador 8031 da Intel e a outra com o FPGA da Altera.

O sistema de controle desempenha duas funções essenciais: medir o período do sinal do sensor e enviar pulsos de corrente ao sensor, excitando a corda vibrante. Um número fixo de ciclos do sinal do sensor determina a freqüência dos pulsos de excitação.

Neste projeto o sensor possui um sinal de saída com freqüência aproximada de 600 Hz e o sinal de excitação com 6 Hz, com a geração de um pulso de excitação a cada 100 ciclos do sinal do sensor.

O microcontrolador 8031 utilizado está instalado em um módulo educativo da Degem Systems, modelo EB-153, onde estão disponíveis apenas alguns dos pinos e das funções do 8031.

O sinal do sensor funciona como uma fonte de interrupção, disparando um temporizador de 16 bits do microcontrolador e também desligando-o. Este sistema possui um clock externo de 12 MHz, mas trabalha internamente em 1 MHz, limitando a precisão/resolução a ser alcançada no temporizador. O resultado de cada contagem é apresentado em um display em formato hexadecimal. Com este resultado chega-se a freqüência de oscilação e à força externa aplicada.

Com um FPGA, utilizando um componente da família Max (7000S), consegue-se reduzir em uma única pastilha de sílicio, de alguns centímetros quadrados, todo o circuito digital, diminuindo a dimensão do sistema final e aumentando a facilidade de implementação do circuito.

A medição de freqüência foi feita através de portas lógicas. Um pulso positivo da ordem de 200 ns (5 MHz), proveniente de um oscilador, é aplicado a uma das entradas da porta AND, à outra entrada são aplicados os pulsos de período desconhecido, que são da ordem de 2 ms (500 Hz). Os pulsos de período conhecido são contados durante meio ciclo do sinal do sensor, medindo-se, assim, o seu período. Este resultado é apresentado em um display de 7 segmentos.

O circuito também realiza a excitação do sensor, através de pulsos de corrente. A lógica e a descrição das pricipais rotinas em AHDL utilizadas são apresentadas na Figura 3, mostrando que em poucas linhas de programação, desenvolveu-se a lógica de todo o sistema de controle do sensor.

```
SUBDESIGN contador
   (enable,clk  : INPUT;
    TIME[7..0]          : OUTPUT; )
    VARIABLE
       count[7..0]      : DFF;
       clear            : node;
    BEGIN
       clear=count7&count6&count3;
       count[].clk =clk ;
       time[]      =count[];
       count[].clrn =!clear;
       IF enable THEN
       count[]  = count[]+1;
       else
       count[]= count[];
       end IF ;
     end;
SUBDESIGN controlador
  ( relogio : INPUT;
    sensorSa    : OUTPUT; )
VARIABLE
 sensorOUT[6..0] : DFF;
BEGIN
   sensorOUT[].clk = relogio;
   sensorOUT[]  = sensorOUT[] + 1;
   IF  sensorOUT[] <= B"0110010" THEN
       sensorSA = B"1";
   else
      sensorSA = B"0";
   end IF;
end;
```

Fig. 3. Principias rotinas em AHDL que compõem a programação do dispositivo FPGA utilizado.

3. RESULTADOS E DISCUSSÃO

A calibração estática do sensor foi realizada aplicando-se forças, em ordem crescente e decrescente de magnitude, na faixa de 0 a 100 N, utilizando-se pesos conhecidos. A Figura 4 ilustra os resultados obtidos.

unesp

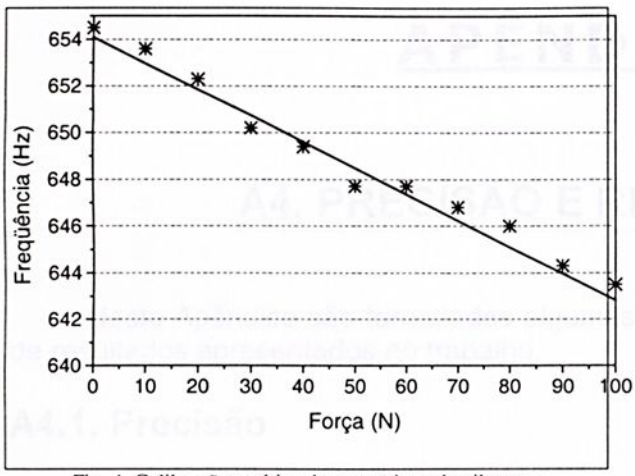


Fig. 4. Calibração estática do sensor à corda vibrante.

O sensor teve resposta quase linear. A curva traçada corresponde à resposta teórica do sensor. Observa-se que os valores experimentais obtidos acompanham a resposta teórica.

A resolução foi de 1 N, quando se utilizaram como elementos de leitura-aquisição os dispositivos lógicos programáveis e frequência de clock de 5 MHz.

Quando foi utilizado o microcontrolador a resolução foi de 4 N. Esta não foi satisfatória devido à baixa freqüência de trabalho do microcontrolador.

A histese observada nas medidas foi pequena.

5. CONCLUSÃO

O artigo descreveu o primeiro protótipo de um sensor à corda vibrante que apresentou boas características como funcionamento de acordo com a resposta teórica esperada, resolução de 1 N, que é adequada para um dinamômetro biomédico e pequena histerese.

Com a construção deste protótipo, iniciam-se aplicações inovadoras para os sensores à corda vibrante, na importante área da Engenharia de Reabilitação. Sendo um sensor robusto, de longa vida útil e de grande confiabilidade poderá ser utilizado como uma interessante alternativa em relação aos extensômetros.

O protótipo já está sendo objeto de algumas modificações que poderão aprimorar o seu funcionamento como, por exemplo, a construção de uma bobina sensora com maior sensibilidade e a implementação de um sistema de leitura-aquisição utilizando um microcontrolador com freqüência interna de trabalho de 10 MHz.

AGRADECIMENTOS

Os autores agradecem ao graduando Adriano Santos Cardoso e ao engenheiro Josivaldo Godoy da Silva por suas importantes contribuições. Agradecem também a Fundação de Amparo a Pesquisa do Estado de São Paulo/Fapesp.

REFERÊNCIAS

[1] M. K. Shigetomi, M. K. Fujita and A. A. F. Quevedo, "Estimulador Elétrico Neuromuscular Microcontrolado de Dois Canais", XVII Congresso Brasileiro de Engenharia Biomédica, Florianópoilis: Setembro/2000, vol. 1, pp. 688-692.

[2] R. Pallás-Areny and J. G. Webster., "Sensors and Signal Conditioning"; *John Willey & Sons,* Inc, 1991.

[3] C. L. Nachtigal, "Intrumentation and Control: Fundamentlas and Applications"; *Wiley-Interscience*, 1990.

[4] M. Q. Feng, H.Suzuki and I. Yokoi, "Development of Optical Sensing Systems for Smart Civil Infrastructure", *Smart Mater. Struct*, vol. 4, 1995, pp. A114-A120.

[5] J. R. Owers-Bradley et. al., "Damping of a Vibrating Wire in Spin Polarized Liquid ^{3}He-^{4}He Mixtures", *Journal of Low Temperature Physics*, vol. 110, nos. 1/2, 1998, pp. 321-326.

[6] MAX+PLUS II, Programmable Logic Development Sytem, AHDL, Altera Corporation: Sao Jose, Version 6.0, November/1995.

APÊNDICE 4

A4. PRECISÃO E REPETIBILIDADE

Neste Apêndice são formuladas algumas definições que foram utilizadas na obtenção de resultados apresentados no trabalho.

A4.1. Precisão

É a qualidade que caracteriza a capacidade de um instrumento de medida de dar a mesma leitura quando repetidademente se mede a mesma quantidade sob as mesmas condições (ambientaism, do operador, etc.), não importando a concordância ou discrepância entre o resultado e o valor medido.

Média:

$$\overline{X} = \frac{S_1 + ... + S_N}{N}$$

onde: $S_1,...,S_N$ são as diversas leituras de uma mesma aplicação de força.

Desvio padrão:

$$dp = \sqrt{\frac{\sum_{j=1}^{N} \left(S_j - \overline{X}\right)^2}{N}}$$

Provável erro:

$$PE(\%) = \frac{2dp}{\overline{X}} \cdot 100$$

Precisão:

$$p(\%) = 100\% - PE(\%)$$

A4.2. Repetibilidade

É a proximidade da concordância enter resultados sucessivos obtidos com o mesmo método, sob as mesmas condicões e em um curto intervalo de tempo.

A4.3. Histerese

A histerese é definida como a diferença entre os dois valores da saída correspondente à mesma entrada[02]. Considerando que um transdutor apresente resposta conforme a ilustrada na Figura A4.1, a histerese pode ser calculada pela expressão (A4.1):

unesp

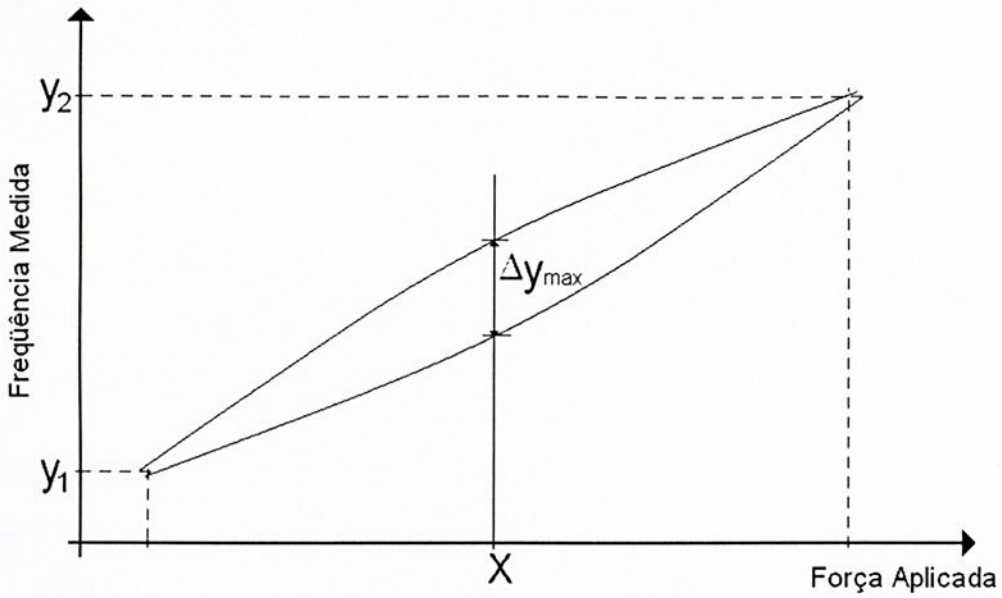


Figura A4.1 – Pontos a serem considerados no calculo da histerese.

$$\text{Histerese}(\%) = \frac{\Delta y_{max}}{y_2 - y_1} \cdot 100 \qquad \text{(A4.1)}$$

unesp

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Programa de Pós-Graduação em Engenharia Elétrica
Av. Brasil Centro, 56
15385-000 Ilha Solteira - SP
www.dee.feis.unesp.br