UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ANAMOD — Módulo de Controle para o Sistema de Aquisição e Análise de Sinais Bioelétricos SAASBIO III

MÁRCIO ROSA DA SILVA

FLORIANÓPOLIS

1998

MÁRCIO ROSA DA SILVA

ANAMOD — Módulo de Controle para o Sistema de Aquisição e Análise de Sinais Bioelétricos SAASBIO III

DISSERTAÇÃO APRESENTADA AO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

FLORIANÓPOLIS

MARÇO 1998

MÁRCIO ROSA DA SILVA

ANAMOD — Módulo de Controle para o Sistema de Aquisição e Análise de Sinais Bioelétricos SAASBIO III

Esta dissertação foi julgada adequada para a obtenção do título de Mestre, especialidade em Engenharia Elétrica e aprovada em sua forma final pelo Programa de Pós-graduação em engenharia Elétrica

Prof. FERNANDO MENDES DE AZEVEDO, Dr.
Orientador

Prof. JOSÉ MARINO NETO, D.Sc.
Co-orientador

Prof. ADROALDO RAIZER, Dr. INPG.
Coordenador da Pós-graduação Eng. Elétrica

Banca Examinadora:

Prof. FERNANDO MENDES DE AZEVEDO, Dr.
Presidente

Prof. RENATO GARCIA OJEDA, D.Sc.

Prof. RAIMES MORAES, PhD.

Prof. JEFFERSON L. B. MARQUES, PhD.

Aos meus pais.

AGRADECIMENTOS

Aos meus pais,

pelo incentivo e apoio em todas as horas.

À minha noiva,

pela compreensão e paciência.

Aos Professores e amigos Fernando e Marino,

pela valiosa orientação neste trabalho.

Ao colega John Wisbeck,

pela ajuda para descobrir os "mistérios" do *Windows*.

Ao colega Marco Aurélio

pela ajuda com o *hardware* do SAASBIO.

Ao colega Sílvio,

que foi a primeira "cobaia" do *software*, informando erros e ajudando na melhoria do programa.

Aos colegas Adriano, Pedro e Ciro,

pelas conversas e discussões a respeito do trabalho.

Aos demais colegas do GPEB.

Ao CNPq,

pelo auxílio financeiro.

E, sobretudo, a Deus,

que me deu forças para que este trabalho pudesse ser concluído.

Sumário

Lista de Figuras

Lista de Tabelas

ANAMOD — Módulo de Controle para o Sistema de Aquisição e Análise de Sinais Bioelétricos SAASBIO III

Resumo

O presente trabalho tem por objetivo a implementação do *software* que servirá de base para o sistema **SAASBIO III** (Sistema de Aquisição e Análise de Sinais Bioelétricos III). O projeto **SAASBIO III** é o resultado da evolução dos sistemas **SAASBIO** e **SAASBIO II** que o antecederam. Este *software* base do sistema é chamado de **ANAMOD** e foi totalmente desenvolvido em ambiente *Windows* 95, de modo a tornar a interface com o usuário final mais amigável. O **ANAMOD** é encarregado de controlar o *hardware* do sistema e fazer a aquisição dos sinais bioelétricos de interesse. Também é função do *software* servir como suporte a outros módulos que sejam desenvolvidos para a ampliação do sistema **SAASBIO III**, como rotinas de tratamento de sinais, ou até mesmo novos módulos de *hardware*. Para tanto o sistema foi totalmente desenvolvido em uma linguagem orientada a objetos (C++), que facilita a inclusão de novos módulos.

Palavras-chave: *software*, sistemas computadorizados, Engenharia Biomédica, aquisição de dados.

ANAMOD — Módulo de Controle para o Sistema de Aquisição e Análise de Sinais Bioelétricos SAASBIO III

Abstract

The purpose of this work was the implementation of the software that is the basis to the **SAASBIO III** system (System for Acquisition and Analysis of Biomedical Signals III). The **SAASBIO III** project is an improved version of **SAASBIO** and **SAASBIO II** systems. This software is called **ANAMOD** and was totally developed for Windows 95, making the user interface more friendly. **ANAMOD** controls the system hardware and the acquisition of biomedical signals. The software may hold other modules that are going to be developed to increase the **SAASBIO III** system functions, like signal processing and other hardware modules that may be added to the system. To allow this, the system was developed in a object oriented language (C++), making easier such modification.

Key-words: software, computer systems, Biomedical Engineering, data aquisition.

1. Introdução

A digitalização de sinais trouxe várias vantagens para a sua análise. Entre estas vantagens podemos citar a facilidade de armazenamento dos sinais em meio permanente com possibilidade de recuperá-los mais tarde.

Uma vantagem do armazenamento dos dados em meio permanente óptico ou magnético é que os dados adquiridos podem ser facilmente reproduzidos e, com isso, diminui a quantidade de papel armazenado (no caso de sistemas que geram seus resultados desta forma — muito comum em sistemas analógicos). Uma vez tendo os dados armazenados, pode-se realizar diversas análises com estes sinais, não só a partir de seus aspectos visuais, mas também filtragens, integração, análises de Fourier, etc. de uma forma mais fácil que em sistemas analógicos, permitindo, inclusive, análises mais sofisticadas, usando ferramentas como *Wavelet*.

Um problema decorrente da digitalização dos sinais é o erro de quantização. Este erro ocorre porque os sinais, quando são adquiridos, são transformados em valores com um número finito de *bits*. Desta forma, o existe uma quantidade finita de valores que podem ser representados. Por exemplo, se considerarmos que o número binário 00000001 equivale a 1 na base decimal e 00000010 equivale a 2, então não é possível representar valores intermediários como 1,5

e 1,7. Por mais que se aproxime a diferença entre um valor binário e outro, sempre existirão valores que não poderão ser representados.

1.1 Motivação

Com o objetivo de propiciar a aquisição e análise de dados de EEG, ECG, EMG e EOG, foi desenvolvido pelo Grupo de Pesquisas em Engenharia Biomédica (GPEB) da Universidade Federal de Santa Catarina (UFSC), o Sistema **SAASBIO** (Sistema de Aquisição e Análise de Sinais Bioelétricos (Coimbra, 1994)), que consiste de um sistema de aquisição que usa um polígrafo ligado via comunicação serial a um microcomputador IBM-PC compatível para a aquisição, armazenamento, processamento e análise de sinais bioelétricos visando o estudo da evolução dos sistemas de regulação da atividade elétrica do EEG e de suas relações com o sono.

Os projetos **SAASBIO** (Coimbra, 1994) e **SAASBIO II** (Rodrigues, 1997) fazem a aquisição de sinais bioelétricos, digitalizando e armazenando, permitindo ainda alguma análise dos sinais (de forma *off-line* no **SAASBIO**). Devido ao vasto espectro de manipulações de sinais úteis para a pesquisa e a clínica, ainda podem ser adicionados diversos recursos a estes projetos de modo a permitir uma análise mais apurada dos sinais. Entre eles podemos citar o uso de técnicas como *Wavelets* ou técnicas de inteligência artificial para reconhecimento automático de padrões.

Neste sistema existem dois módulos de *software*: um para realizar a aquisição dos sinais e armazená-los em disco e outro para a análise *off-line* dos sinais. No sistema **SAASBIO** não existe nenhum tipo de análise *on-line*. O sistema **SAASBIO** está sendo utilizado no Laboratório de Neurofisiologia I do Departamento de Ciências Biológicas da UFSC desde 1993 e tem servido para muitos trabalhos de pesquisa que

resultaram em uma série de publicações. (André, 1997; Bertemes, 1997; Bruno, 1996; Coimbra, 1994a; Coimbra, 1994b; Coimbra, 1994c; Coimbra, 1995; Dario 1993, Dario, 1996; Lopes, 1992; Ribeiro, 1995; Toazza 1998b).

Com a utilização do **SAASBIO**, foram detectadas algumas necessidades e deficiências do sistema, entre elas a necessidade de uso de outros *softwares* para a análise estatística do sinal. Outro ponto a melhorar no sistema é o acréscimo de rotinas para análise *on-line* do sinal, como o uso de alguns filtros. Como melhorias ao sistema, ainda podemos citar a utilização de um *hardware* específico, mais preciso que o polígrafo utilizado e, se possível, mais portátil que este.

Com o intuito de atender a estas necessidades do projeto **SAASBIO**, foi desenvolvido no mesmo grupo (GPEB), o sistema **SAASBIO II** (Rodrigues, 1996a; Rodrigues, 1996b; Rodrigues, 1997) que acrescentou inúmeras melhorias ao sistema original (conforme será citado no capítulo sobre o SAASBIO).

No entanto, apesar das melhorias acrescentadas pelo projeto **SAASBIO II**, este ainda precisava de algumas modificações, entre elas o desenvolvimento do *software* de controle em ambiente *Windows*, além de algumas modificações no *hardware* do sistema de modo a permitir a inclusão de novos módulos de *hardware*. Com isto, foi necessário o desenvolvimento de um novo projeto, o **SAASBIO III** que tem por objetivo suprir estas necessidades. O *software* desenvolvido para o sistema **SAASBIO III** é o objeto deste trabalho e foi denominado **ANAMOD**, que será o módulo de controle para o sistema, sendo também encarregado da aquisição dos sinais bioelétricos que serão estudados.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo deste trabalho é desenvolver um *software* que realize o interfaceamento entre o SAASBIO III e um computador pessoal padrão IBM-PC para a aquisição de dados de EEG, ECG, EMG e EOG. Este *software* deve ser desenvolvido de modo a possibilitar expansões com o acréscimo de outras rotinas para utilização com *hardware* adicional e rotinas de tratamento de sinais, para facilitar a interpretação dos dados adquiridos.

1.2.2 Objetivos Específicos

O *software* deve apresentar uma interface que facilite a interação com o usuário final, baseada no sistema *Windows*, uma vez que esta plataforma é bem conhecida por usuários de computadores pessoais.

Este *software* deve ser desenvolvido de modo a facilitar a inclusão de novos módulos, permitindo a expansão do mesmo. Para esta finalidade utilizar-se-á uma linguagem orientada a objetos, que permite o tratamento do *software* como blocos de código em vez de um bloco único.

Deverá também ser feito um estudo a respeito de alguns sinais biomédicos que serão adquiridos pelo **SAASBIO III** e um estudo de procedimentos de análise do sinal de EEG, porque, embora o sistema **SAASBIO** se destine também a outros tipos de sinal, o objetivo inicial deste sistema é tratar de sinais de EEG. Este estudo será feito visando expansões no sistema.

1.3 Organização do Trabalho

Para atender os objetivos mencionados anteriormente, esta dissertação está estruturada da forma descrita a seguir.

No capítulo 2 é feita uma descrição de alguns sinais bioelétricos e é dada uma ênfase à análise de sinais de EEG, já que estes serão objeto de pesquisas utilizando o **SAASBIO** num primeiro momento.

O capítulo 3 descreve o sistema **SAASBIO III**, que é o sistema no qual o sistema **ANAMOD** se encaixa, sendo o principal módulo de *software* do sistema, realizando a aquisição dos sinais e controlando o *hardware* do sistema.

No capítulo 4 é feita uma descrição do sistema **ANAMOD**, sendo introduzido alguns conceitos importantes como programação orientada a objeto. Neste capítulo também é feita uma breve descrição do funcionamento do sistema *Windows* 95, plataforma na qual o sistema **ANAMOD** foi desenvolvido. Esta análise permite um entendimento melhor do sistema, descrevendo as vantagens e dificuldades de trabalhar-se com este sistema (*Windows* 95).

Finalmente, no capítulo 5 são feitas algumas conclusões a respeito do trabalho desenvolvido, apontando algumas dificuldades e as soluções utilizadas e por fim são feitas algumas sugestões de trabalhos futuros, que servirão para melhorar o desempenho do sistema **ANAMOD** e ampliar as funcionalidades do sistema **SAASBIO III**.

2. Sinais Bioelétricos

Neste capítulo será feita uma descrição de alguns sinais bioelétricos de interesse no projeto **SAASBIO**.

Inicialmente, será feita uma descrição geral dos sinais, passando a seguir para uma análise mais detalhada do sinal de EEG, que será o sinal a que o sistema **SAASBIO** se destinará inicialmente.

2.1 Eletroencefalograma (EEG)

O registro da atividade elétrica do cérebro é conhecido como eletroencefalograma (EEG) e é amplamente utilizado para propósitos clínicos e de pesquisa. Muitos métodos têm sido desenvolvidos para o estudo do funcionamento das várias partes do cérebro através do EEG.

Três maneiras de registrar a atividade elétrica do cérebro são usualmente utilizados. O primeiro — registro em **Profundidade** — consiste na inserção de eletrodos no tecido neural do cérebro. No segundo — o **eletrocortigrama** — colocam-se eletrodos na superfície exposta do cérebro e o terceiro, mais utilizado, é feito com técnica não-invasiva, utilizando-se eletrodos de superfície colocados no escalpo.

Neste último procedimento são colocados em várias localizações padronizadas no couro cabeludo. São empregados de 6 a 32 canais, usualmente de 8 a 16 eletrodos e o registro é o resultado da diferença de potencial entre dois eletrodos quaisquer entre os colocados (Cohen, 1983). A figura 2.1 mostra eletrodos de 8 canais em EEG normal de um indivíduo em repouso.

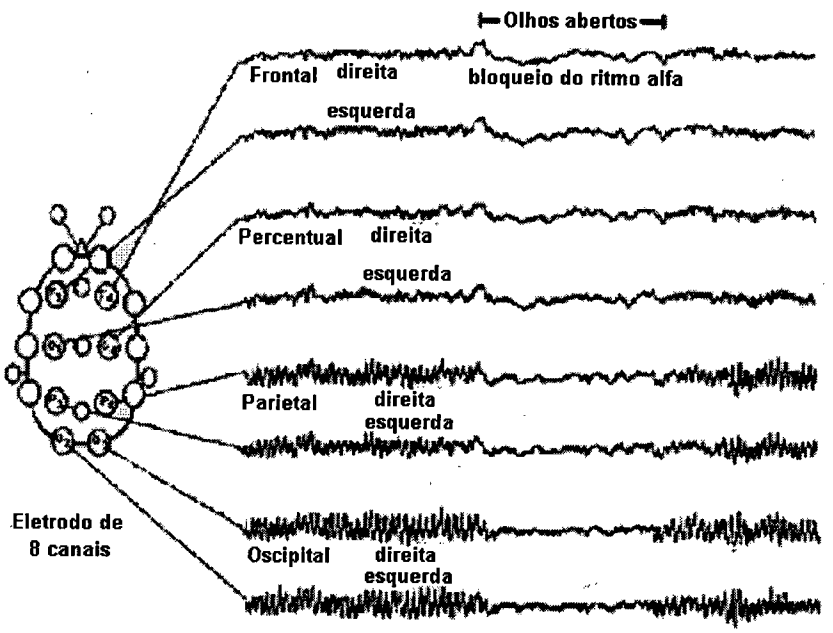

FIGURA 2.1 – EEG normal de um indivíduo, desperto e em repouso (Berne, 1990).

O estudo da atividade elétrica do cérebro é dividido em dois modos:

- o registro da **atividade espontânea** do cérebro, a qual é resultado do campo elétrico gerado pelo cérebro sem atividades/eventos específicos que a determine mas que estão relacionadas aos diferentes estados de atividade cerebral;
- os **potenciais evocados** (EP) que são potenciais gerados pelo cérebro como resultado de um estímulo (tal como um estímulo visual ou auditivo, etc.) (Ary et al., 1981; Silva Jr., 1998).

A análise de freqüência do EEG tem sido usada como ferramenta de processamento com o objetivo de auxiliar o diagnostico de epilepsia, doenças mentais, disfunções psiquiátricas, desordens no sono entre outros. A faixa de freqüência no EEG de escalpo é desde DC a 100 Hz, com a maior potência distribuída entre 0,5 e 60 Hz. As amplitudes neste caso são de 2 a 100 μV, para um EEG normal, chegando a 300 μV em descargas epileptiformes. A densidade espectral de potência do EEG varia muito com o local de registro, idade e estado físico-emocional do paciente.

O espectro de freqüência do EEG normal tem sido subdividido em bandas:

- **Faixa Alfa** — esta é a porção do espectro que ocupa a faixa de 8 a 13 Hz. Estes tipos de ritmos são comuns em pacientes normais e são registrados quando o paciente está acordado, com os olhos fechados, sob condições de relaxamento;
- **Faixa Beta** — ocupa de 13 a 22 Hz. Os ritmos **Beta** são registrados em pacientes adultos normais principalmente na região pré-central mas aparece também em outras regiões. O ritmo tem sido dividido em dois: **Beta I** e **Beta II**. O ritmo **Beta I** está presente durante intensa ativação do sistema nervoso central, enquanto **Beta II** é diminuído por tal ativação.
- **Faixa Teta** — neste caso as freqüências vão de 4 a 8 Hz. Alguns transientes de componentes de atividade teta têm sido encontradas em pacientes adultos normais. A atividade teta ocorre principalmente mas áreas central e temporal e é mais comum em crianças.
- **Faixa Delta** — é a parte do espectro cuja faixa de freqüências é de 0,5 a 4 Hz. As ondas delta aparecem em crianças, em pacientes em estado de sono

profundo e em portadores de algumas doenças cerebrais. Em adultos no estado de alerta, o aparecimento destas ondas pode indicar anormalidades.

A figura 2.2 ilustra estes diferentes tipos de ondas.

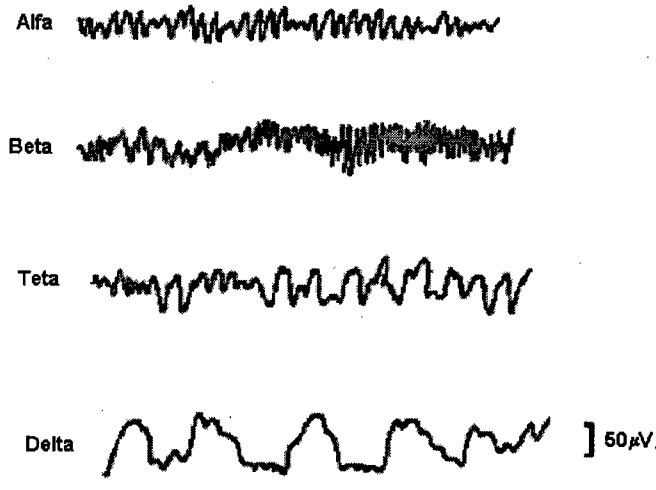

FIGURA 2.2 – Ritmos do Eletroencefalograma (Webster, 1992).

A análise no domínio do tempo é também usada no processamento do EEG para detectar pequenas ondas e estados do sono. O sono é um processo dinâmico que consiste de vários estágios e o EEG representa potenciais externos gerados pela atividade neuronal do córtex e quanto mais sincronizada for essa atividade, maior será a amplitude do sinal registrado.

A primeira modificação detectada pelo EEG, com o início da sonolência, é uma diminuição global da amplitude das ondas Alfa e Beta.

No primeiro estágio do sono, aparecem as **ondas Teta** que são uma mistura de baixas freqüências. No segundo estágio, surtos de alta freqüência, chamados **fusos do sono**, estão presentes, junto a ondas lentas, de ocorrência ocasional (ondas Delta). As ondas Delta ficam mais freqüentes no terceiro estágio do sono, onde aparecem também os **Complexos-K**. Estes complexos, a maioria facilmente provocada por estímulos auditivos, consiste de um ou dois pequenos picos de tensão (100 a

200 μV) em ondas de baixa freqüência, às vezes acompanhados por um curto episódio de atividade de 12 a 14 Hz. No quarto estágio, o de sono profundo, o EEG é dominado pela dificuldade de acordar, pela redução do tônus muscular, pela diminuição das freqüências cardíaca e respiratória e da pressão arterial e pode indicar estados de coma (Moffet, 1993). A figura 2.3 mostra os estados do sono de um paciente normal.

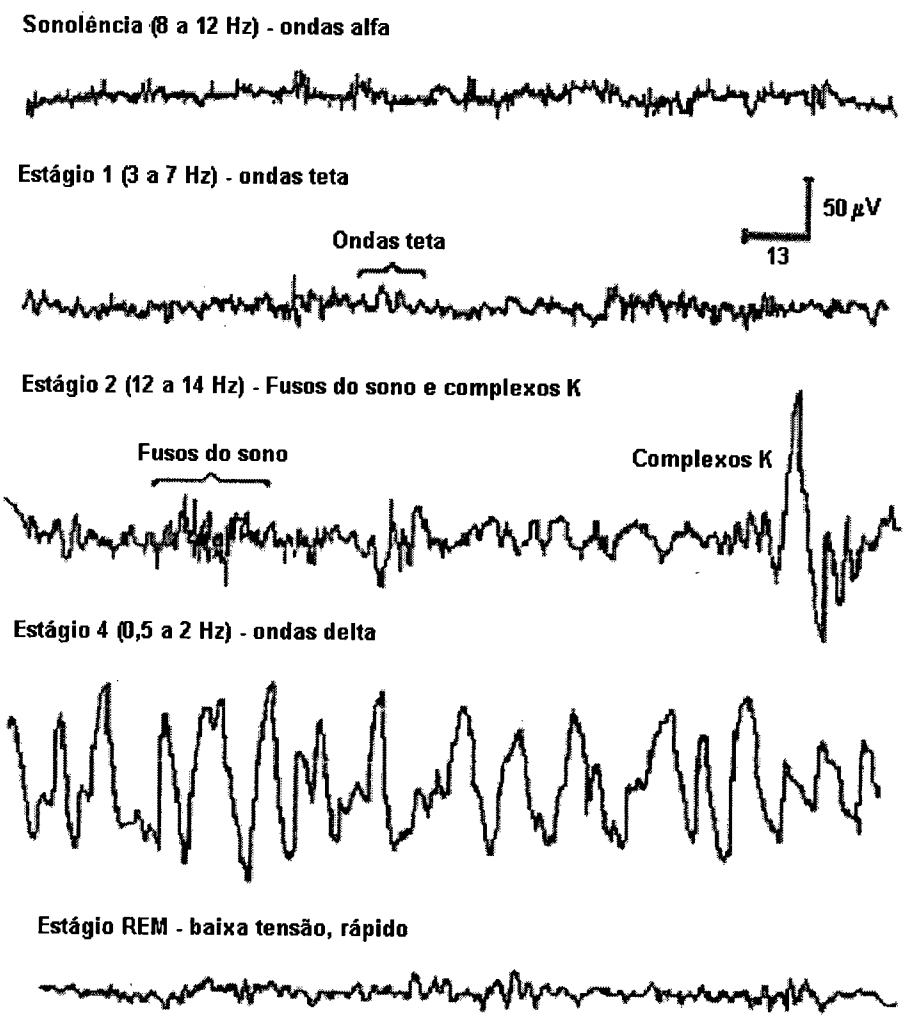

FIGURA 2.3 – Registros de EEG humano durante os estágios do sono (Berne, 1990).

Durante a fase do sono de movimentos rápidos do olho (a fase **REM**: *Rapid Eye Movement*) o EEG assemelha-se ao que é registrado em um paciente acordado. As freqüências cardíaca e respiratória e a pressão arterial são aumentadas. Apesar da ocorrência dos movimentos oculares, o tônus muscular fica ausente nos músculos antigravitacionais (ex. pescoço). Pessoas na fase REM do sono são muito

difíceis de serem acordadas propositalmente e são mais propensas a acordar espontaneamente.

2.1.1 Epilepsia

O conceito de epilepsia foi e permanece sendo um tema gerador de controvérsias no meio médico. Por muitos séculos aceitou-se como "epiléptico" apenas o paciente que apresentasse crises de perda de consciência acompanhadas de convulsões generalizadas. Com a verificação de que alguns pacientes apresentavam episódios de obnubilação da consciência ou **ausências** (Lennox et al., 1949), alternadamente com as crises convulsivas generalizadas, este conceito foi se alargando para contemplar as várias síndromes epilépticas distintas já identificadas.

Entretanto, o primeiro cientista a desmistificar o mecanismo das epilepsias foi o inglês Hughlings Jackson em 1870 (Ives et al., 1993 e Niedermeyer et al., 1993), ao reconhecer como evento comum a todas as formas desta doença a ocorrência de descargas excessivas e desordenadas dos neurônios cerebrais, independentemente das diferentes manifestações clínicas e peculiariedades funcionais das áreas e formações nervosas do cérebro, postas em ação pelas descargas elétricas, características de cada tipo de epilepsia (Lennox et al., 1949). Curiosamente, apenas recentemente as conclusões de Jackson foram aceitas e devidamente reconhecidas.

Muitos estudos e muita polêmica seguiram o trabalho de Jackson, tanto acerca da etiologia (estudo das origens, das causas da doença), como da semiologia (estudo e descrição dos sinais da doença, sintomatologia) da epilepsia. Surgiram inúmeros conceitos e múltiplas classificações para as epilepsias, tais como **grande-mal** e **pequeno-mal**, Epilepsia Sintomática ou Idiopática, Epilepsia Psicomotora ou equivalentes epilépticos, Hipsarritmia, Disrritmia Cerebral Paroxística, etc. Muitas

destas idéias foram contestadas e postas por terra (Bittencourt et al., 1993), especialmente por Penfield e Jesper (Passold et al., 1993), do Instituto Neurológico de Montreal, nas décadas de 40 e 50, os quais reformularam algumas destas definições e ressuscitaram as idéias de Jackson. Alguns destes conceitos e classificações, porém, persistem até hoje, em algumas escolas, mas a concepção atualmente admitida é a que afirma que a epilepsia é "uma condição crônica, ou um grupo de doenças que têm em comum a ocorrência de crises epilépticas" (Grözinger et al., 1995).

2.2 ELETROCARDIOGRAMA (ECG)

O sinal de eletrocardiograma (ECG) é o registro da atividade elétrica do coração. A atividade mecânica da função cardíaca está ligada à atividade elétrica. O ECG é uma importante ferramenta para o avaliação da função cardíaca.

O ciclo cardíaco inicia-se com um estímulo elétrico gerado no nó sino-atrial (SA), localizado no átrio direito, que é um conjunto de células especializadas que determinam o ritmo de batimento do coração. Este estímulo elétrico do nó SA propaga-se pelo átrio causando, causando sua contração e gerando a onda **P** no ECG.

Os impulsos movem-se ao longo das fibras dentro do átrio, alcançando o nó átrio-ventricular (AV) que propicia a rápida transmissão de impulsos entre a base do átrio e o ápice do ventrículo. O tempo de condução átrio-ventricular é da ordem de 120 a 220 ms. Um sistema especial de condução denominado **Feixe de His** e as **Células de Purkinje** transferem os impulsos às partes inferiores dos ventrículos.

A contração dos ventrículos compreende a ação de bombeamento do coração e a geração do complexo **QRS** no ECG. Após aproximadamente 150 ms os ventrículos repolarizam, causando a onda **T** no ECG.

A repolarização do átrio raramente é vista no ECG. Em casos raros nos quais isto ocorre, aparece entre as ondas **P** e **Q** sendo denominado de onda **TA**. Uma onda adicional, a onda **U**, é às vezes registrada após a onda **T**. Acredita-se que sua causa seja a repolarização dos músculos papilares ventriculares.

O ritmo cardíaco, ou a **taxa de batimento**, é um processo aleatório e é usualmente medido no intervalo **R–R**. Durante o sono esta taxa é baixa (**braquicardia**) e acelera-se (**taquicardia**) durante exercícios, estresse emocional ou febre. Distúrbios nos ritmos, denominados arritmias, podem surgir sob condições normais.

Às vezes, uma porção do miocárdio descarrega-se independentemente, causando um batimento cardíaco anormal, que é conhecido como **contração pré-ventricular** (PVC). Quando as descargas independentes continuam, o coração entra em um estado de **fibrilação** atrial ou ventricular.

A figura 2.4 apresenta registros de um ECG e um eletrograma do Feixe de His.

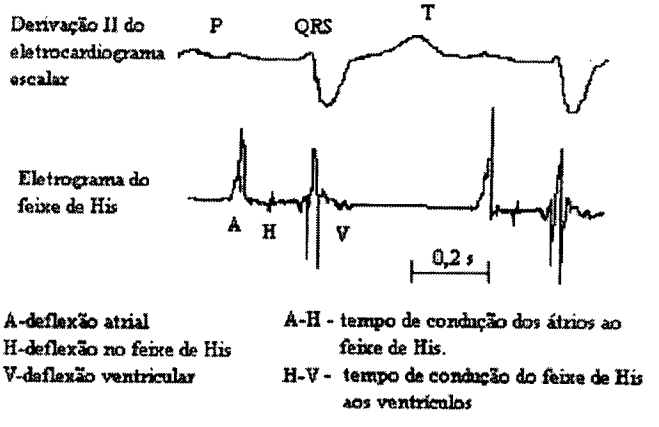

FIGURA 2.4 – Eletrocardiograma e Eletrograma do Feixe de His (Berne, 1990).

O ECG convencional consiste do complexo **PQRST** com amplitudes de alguns mV e é usualmente processado numa banda de freqüência de 0,05 a 100 Hz. O

primeiro passo no processamento do ECG é a identificação da onda **R**. Isto é feito de maneira a sincronizar os consecutivos complexos e para a análise do intervalo **R–R** do ritmo cardíaco. O problema torna-se severo quando as condições de ativação muscular e ruídos de outras fontes tendem a obscurecer o complexo **QRS**. A análise do intervalo **R–R** é uma importante parte do monitoramento de pacientes e muitos métodos têm sido empregados para o processamento automático do ECG para monitoração, compressão e classificação (Cohen, 1983).

2.3 ELETROMIOGRAMA (EMG)

A eletromiografia é o registro do potencial elétrico gerado por um músculo. A atividade do músculo pode ser monitorada através de eletrodos de superfície colocados sobre a pele do indivíduo. O sinal recebido produz a informação a respeito da atividade elétrica total associada com a contração muscular. Para um diagnóstico mais detalhado necessita-se de eletrodos de agulha concêntricos que são inseridos na pele, dentro do músculo.

2.3.1 Eletromiografia de uma única fibra (SFEMG)

O sinal recebido é conhecido como potencial de ação da unidade motora (MUAP). Uma alta resolução pode ser conseguida com o uso de microeletrodos através dos quais registram-se os potenciais de ação de uma única fibra muscular.

Os potenciais de ação de uma única fibra muscular têm a duração de cerca de 1 ms com amplitudes de poucos mV. A banda de freqüência usada para o processamento do SFEMG é de 500 Hz a 10 kHz. Embora o SFEMG contenha baixas freqüências é aconselhável cortar as bandas baixas a fim de que as contribuições de

fibras distantes possam ser minimizadas. Este corte pode ser feito através de filtros de freqüência.

O SFEMG é usado clinicamente para detectar disfunções neuromusculares tal como a miastenia grave.

2.3.2 Potencial de Ação de uma Unidade Motora (MUAP)

O complexo constituído de células nervosas, fibras nervosas, junções neuromusculares e fibras musculares é chamado de **unidade motora**. Os potenciais de ação registrados deste complexo são medidos através de um eletrodo de agulha concêntrico e são conhecidos por potenciais de ação da unidade motora (MUAP, MUP).

A figura 2.5 ilustra o registro de MUAP.

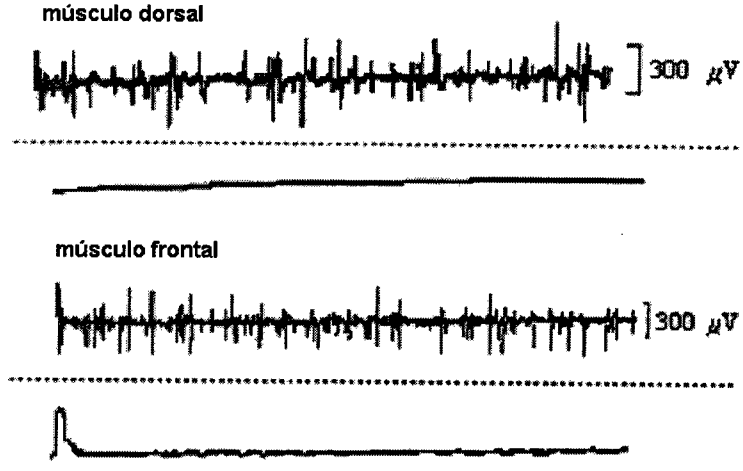

FIGURA 2.5 – Potencial de Ação de uma Unidade Motora.

O MUAP é subdividido em duas componentes. Uma ponta, tal como um espigão, é gerada por cerca de 2 a 12 fibras com uma forma e amplitude que dependem da sincronicidade entre as fibras dentro da unidade motora (MU).

As partes iniciais e terminais são geradas por fibras mais distantes da mesma unidade motora. A duração do MUAP é aproximadamente 2 a 10 ms, com amplitudes na faixa de 100 μV a 2 mV. A banda de freqüência é 5 Hz a 10 kHz.

Os MUAP são usados clinicamente para detectar miopatias, lesões neurogênicas e outras desordens neuromusculares.

2.3.3 Eletromiografia de Superfície (SEMG)

A aquisição não-invasiva do EMG através de eletrodos de superfície é um método conveniente mas que produz somente uma informação grosseira sobre o músculo em investigação. As amplitudes do SEMG dependem do músculo e dos eletrodos. A análise do espectro de freqüências do EMG é utilizada para caracterização de fadigas musculares.

Uma faixa de amplitudes entre 50 μV a 5 mV é normal. A banda de freqüência registrada para o músculo esquelético é de 2 a 500 Hz (para a musculatura lisa a banda é de 0,01 a 1 Hz). A densidade espectral de potência do EMG é sempre estimada para várias aplicações clínicas tais como tremores patológicos e análise de fadiga muscular.

2.4 Eletrooculograma (EOG)

O EOG é o registro do potencial córneo-retinal. Este potencial tem sido usado para medir a posição do olho tanto para pesquisa do sono como para uso clínico, diagnosticando habilidade de leitura e fadiga visual (Webster, 1992).

O EOG, ao contrário de outros potenciais, necessita de um amplificador DC e o sinal é medido por pares de eletrodos de superfície à direita e à esquerda bem

como acima e abaixo dos olhos. Os níveis de amplitude estão na faixa de 10 μV a 5 mV com uma faixa de freqüência de DC a 100 Hz.

O EOG fornece uma medida da função retinal que depende da integridade do pigmento da camada epitelial da retina. Quando esta camada encontra-se sadia, o potencial estabelecido entre as partes dianteira e traseira do olho responde a mudanças de luminosidade (Liebman, 1987).

A figura 2.6 mostra um eletro-oculograma condiderado normal.

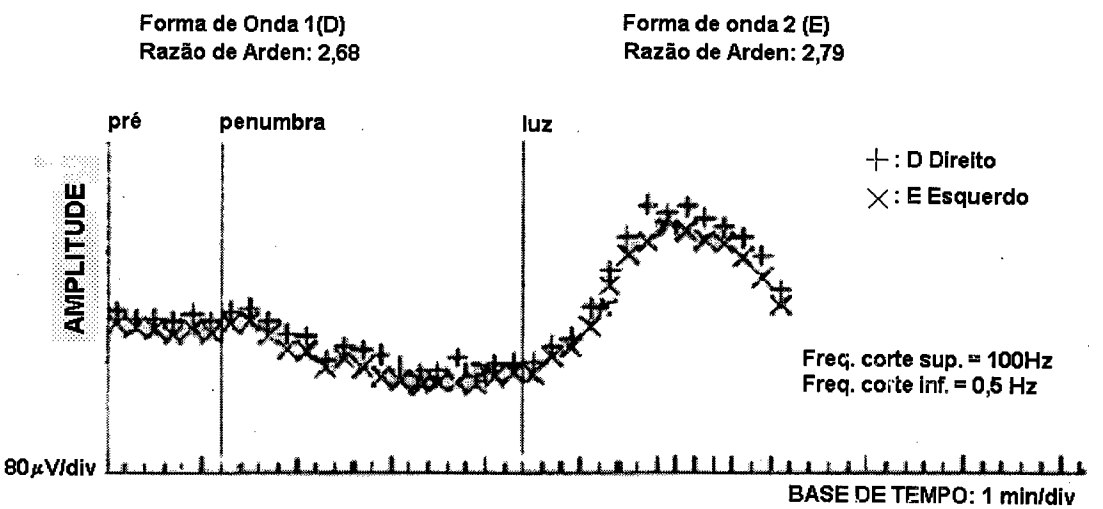

FIGURA 2.6 – Eletro-oculograma de um paciente normal (Marshall, 1993).

O potencial de repouso em um olho é estimado através da medida da tensão induzida utilizando-se um par de eletrodos de Ag-AgCl no olho. Estes eletrodos são necessários para prevenção de deriva. Este potencial declina, durante a adaptação à penumbra, a um valor mínimo e chega a um valor máximo durante a adaptação à luz antes de atingir o estado de equilíbrio.

A razão entre a amplitude máxima, sob condições de adaptação à luz, e a amplitude mínima sob condições de adaptação à penumbra, é chamada de **Razão de**

Arden. A Razão de Arden é a medida padrão para o diagnóstico de EOG (Marshall, 1993).

2.5 ANÁLISE DOS SINAIS

Em linhas gerais, a análise dos sinais por inspeção visual dos registros depende, basicamente, do reconhecimento de padrões comuns e sua associação com condições normais ou patológicas na rotina clínica; em trabalhos de pesquisa trata-se de associar os padrões a eventos comportamentais e/ou às influências do meio externo. Em muitos casos, é interessante que o equipamento utilizado na aquisição dos dados apresente os mesmos de forma *on-line* a medida que os armazena.

As análises visuais de sinais bioelétricos baseiam-se em critérios muitas vezes subjetivos. As variações dos sinais e seus padrões de atividade são geralmente descritos de forma verbal e publicações relacionadas ao tema vêm acompanhadas de trechos de gráfico de sinal para exemplificação. A parametrização de sinais, ou seja, a tradução em números de alguns aspectos dos sinais, não só constitui uma forma de modo geral mais objetiva de descrever o sinal como também proporciona mais e, por vezes, melhores dimensões para análise.

2.5.1 Análise de Freqüência

Sinais bioelétricos resultam de processos que ocorrem no domínio do tempo. Entretanto é, por vezes, mais conveniente e eficaz analisar estes sinais no domínio da freqüência. Isto ocorre tanto no caso de sinais determinísticos quanto estocásticos, onde seja importante, por exemplo, analisar padrões rítmicos de atividade.

2.5.2 Filtros de Freqüência

Alguns padrões de atividade de interesse em sinais bioelétricos possuem bandas de freqüência bem definidas. Entretanto podem aparecer, nos registros, contaminações por outros sinais e até ruídos externos ao sistema biológico em estudo.

Filtros de freqüência ressaltam a atividade alvo reduzindo perturbações externas cujas componentes de freqüência estejam fora da banda passante. Aplicações importantes aparecem, por exemplo, na eliminação de interferência da rede elétrica, ressaltamento de características de sinais polissonográficos para análise automática de registros, entre outras.

2.5.3 Análises de Período-Amplitude

Análises aperiódicas de formas de ondas ou análises de período-amplitude são métodos de análise de processos estocásticos utilizados para caracterizar cada onda ou evento elétrico de um sinal contínuo. Estas técnicas têm sido apontadas como eficazes na qualificação do efeito de drogas sobre o sistema nervoso central (SNC).

Trata-se na realidade de identificar seletivamente eventos elétricos contidos entre dois extremos relativos (vales ou picos) ou cruzamentos por zero (ou linha de base) do sinal.

2.6 Análise do Sinal de EEG

2.6.1 Introdução

A interpretação do sinal de EEG é feita rotineiramente através da inspeção visual dos registros armazenados e depende do reconhecimento de padrões

comuns e sua associação com condições normais ou com alguma patologia. Esta metodologia é utilizada pois um especialista é capaz de reconhecer os diferentes padrões e devido à ausência de *software* que realize tal tarefa de forma eficiente.

Infelizmente, estes métodos tradicionais apresentam algumas desvantagens, entre elas o fato de que diferentes disfunções podem apresentar sinais similares, dificultando a interpretação visual.

Existem duas formas de análise de sinais de EEG. A primeira delas inclui técnicas de medida e análise, podendo, por sua vez, ser subdividida em técnicas de análise quantitativa (para descrever o EEG numericamente em vez de verbalmente, objetivando aumentar a confiabilidade) e técnicas para representar o EEG em outro domínio (através de transformadas) para que determinadas características sejam evidenciadas em relação a outras; e outros métodos de visualização além de plotagem em papel. Estas técnicas podem ser utilizadas em conjunto ou separadamente e podem ser desenvolvidas para medir características que não podem ser identificadas visualmente — por exemplo, para medida de pequenas diferenças de tempo entre atividades similares em diferentes eletrodos. A segunda forma de análise engloba o controle das condições em que a aquisição foi realizada.

2.6.2 Medidas de Amplitude

Uma das características óbvias do EEG é a amplitude das flutuações existindo diversos métodos de análise. Durante a inspeção visual é avaliada a amplitude pico-a-pico do sinal de EEG, sendo que não é necessário medir todas as ondas. Um valor representativo normalmente é apropriado. É provável que a média das amplitudes de diversas ondas avaliadas por inspeção visual sejam maiores que a média verdadeira, a menos que uma regra de medida real seja utilizada.

Uma medida exata da média da amplitude pico-a-pico necessitaria que fossem medidas cada onda e que fosse calculada sua média. Para fazer isto manualmente é necessário um conjunto de regras para que não haja ambigüidade com relação a quais ondas incluir e quais não. Este método é ilustrado na figura 2.7(b) para o sinal mostrado em (a). A figura 2.7(c) mostra um método alternativo proposto por Walter e Yeager (1956).

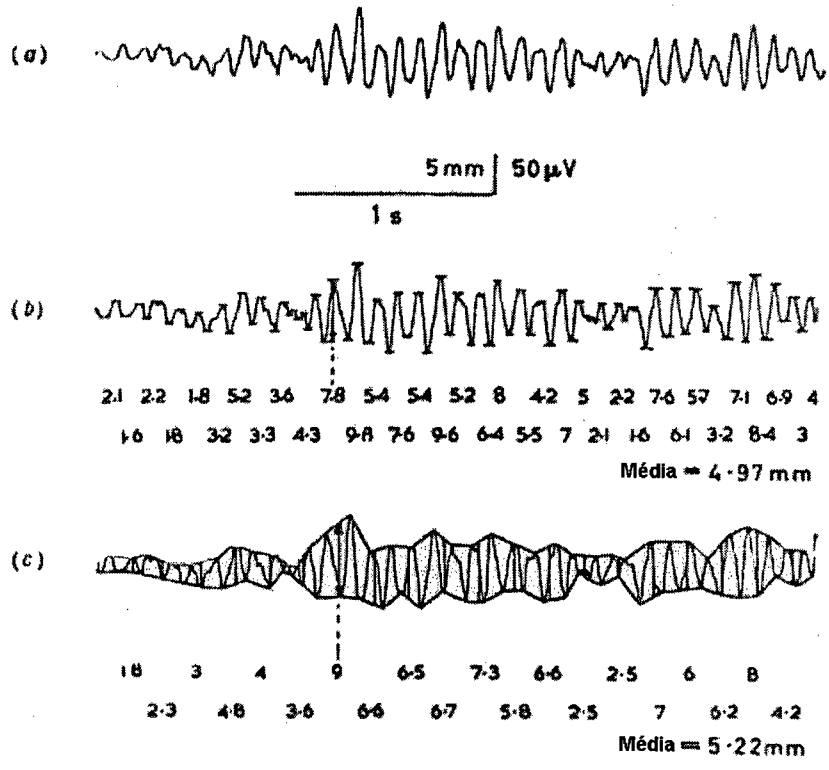

FIGURA 2.7 – Método alternativo usado por Walter e Yeager em 1956 (Cooper et al., 1974).
(a) Sinal de EEG. (b) Medida da amplitude média pico-a-pico. (c) Medida de amplitude de Walter e Yeager.

Outra medida de amplitude que é mais facilmente feita automaticamente é o desvio médio de amplitude. Com o uso de amplificadores AC, os registros de EEG oscilam em torno do zero e a média das deflexões positivas e negativas é zero. Entretanto, o desvio médio não é zero e pode ser medido. Primeiramente o sinal é retificado, tornando as deflexões negativas do sinal positivas. Desta forma todas as flutuações do sinal ficam de um só lado da linha de base. Isto é mostrado na figura

2.8(b) para o sinal de EEG mostrado em (a). O desvio médio é igual à área sob o sinal retificado e a linha de base dividido pela duração do sinal. A área é proporcional ao desvio médio e pode ser medida através da integração do sinal como mostra a figura 2.8(c). A amplitude deste gráfico, a qualquer momento, é proporcional à área sob o sinal original na figura 2.8(b) a partir da hora que a integração iniciou. Se o integrador for preparado para reiniciar de zero cada vez que a saída atingir um valor pré-determinado, então o número de reinicializações pode ser facilmente contado manualmente ou automaticamente. Isto é mostrado em (d) na figura. O número de reinicializações num tempo dado é proporcional à integral e, portanto, ao desvio médio.

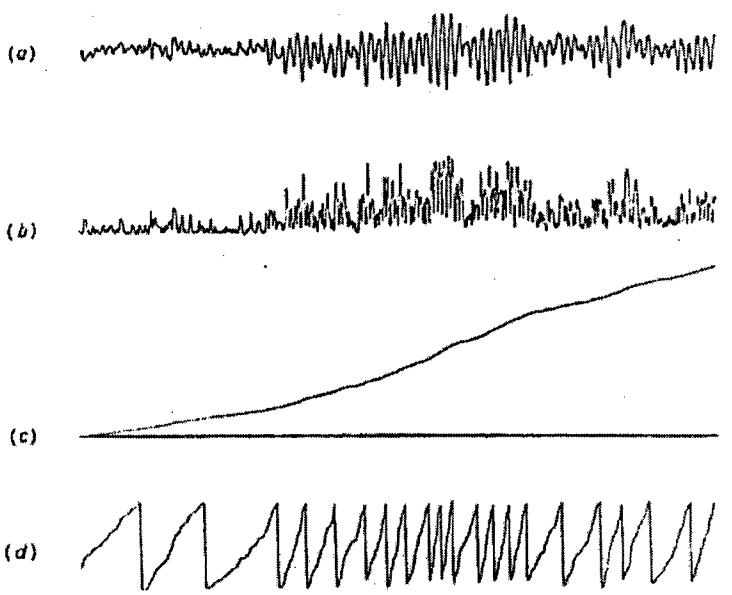

FIGURA 2.8 – Desvio Médio de Amplitude (Cooper et al., 1974).
(a) Sinal de EEG. (b) Sinal (a) retificado. (c) Sinal (b) integrado. (d) Sinal (b) integrado com reinicialização.

Outra medida de amplitude é a variância ou amplitude média quadrática. Quando um sinal é elevado ao quadrado — isto é, a amplitude a cada instante é multiplicada pelo seu próprio valor — os valores negativos tornam-se positivos e um valor médio pode ser medido da mesma forma que para o sinal retificado. O efeito de elevar ao quadrado o sinal da figura 2.8(a) é mostrado na figura 2.9; o resultado, em

cada caso, é proporcional à sua amplitude média quadrática. Quando o sinal tem uma amplitude média nula — como no caso de sinais de EEG amplificados por amplificadores AC — este resultado é chamado de variância do sinal. Se a média não é zero, os desvios de amplitude da média é que são elevados ao quadrado e somados para encontrar a variância. A raiz quadrada da variância — a raiz média quadrática ou valor RMS do sinal — é uma medida padrão de sinais sinusoidais.

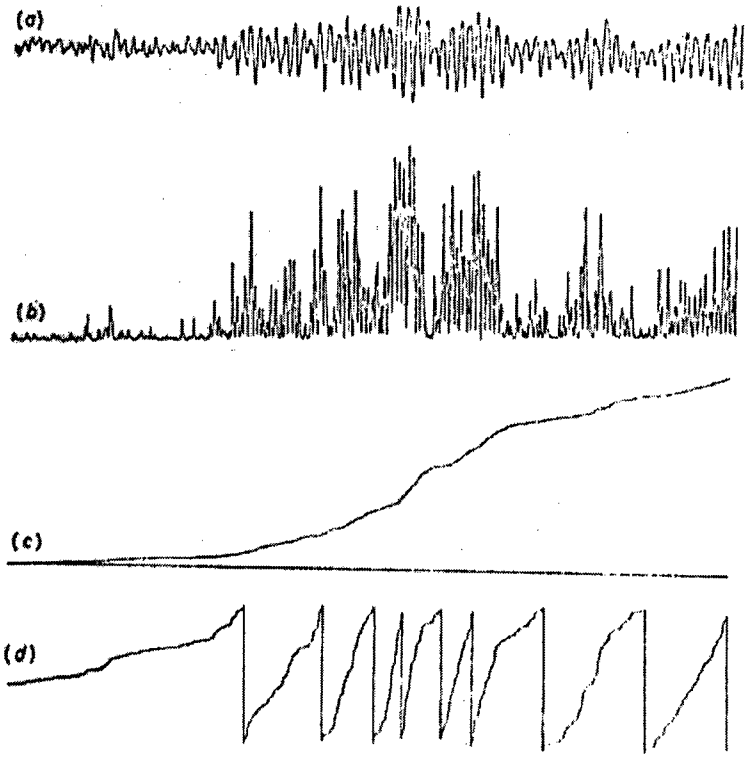

FIGURA 2.9 – Variância ou Amplitude Média Quadrática (Cooper et al., 1974).
(a) Sinal de EEG. (b) Sinal (a) elevado ao quadrado. (c) Sinal em (b) integrado. (d) Integrado com reinicialização.

Uma maneira de perceber-se as características das flutuações de amplitude de sinais de EEG é usando uma medida conhecida como função densidade de probabilidade de amplitude. Ela consiste de um gráfico mostrando a probabilidade de ocorrência de um valor de amplitude particular. Ela possui uma escala de amplitude no eixo x uma escala de probabilidade no eixo y. Uma onda quadrada periódica possui apenas dois valores de amplitude, cujos valores de x são $+a$ e $-a$ e apresentam uma

probabilidade (em y) de 0,5 (figura 2.10(a)). A distribuição de uma onda senoidal pura apresenta a forma de **U** simétrico ao eixo y (figura 2.10(b)).

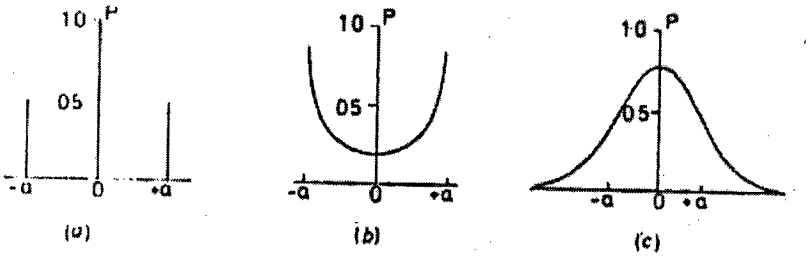

FIGURA 2.10 – Densidade de probabilidade de amplitude (Cooper et al., 1974).
(a) uma onda quadrada, (b) uma onda senoidal e (c) um sinal de EEG.

No caso de sinais de EEG a função apresenta a forma de uma **Gaussiana** com média zero (figura 2.10(c)) na maioria das vezes.

2.6.3 Análise de Freqüência — Base Teórica

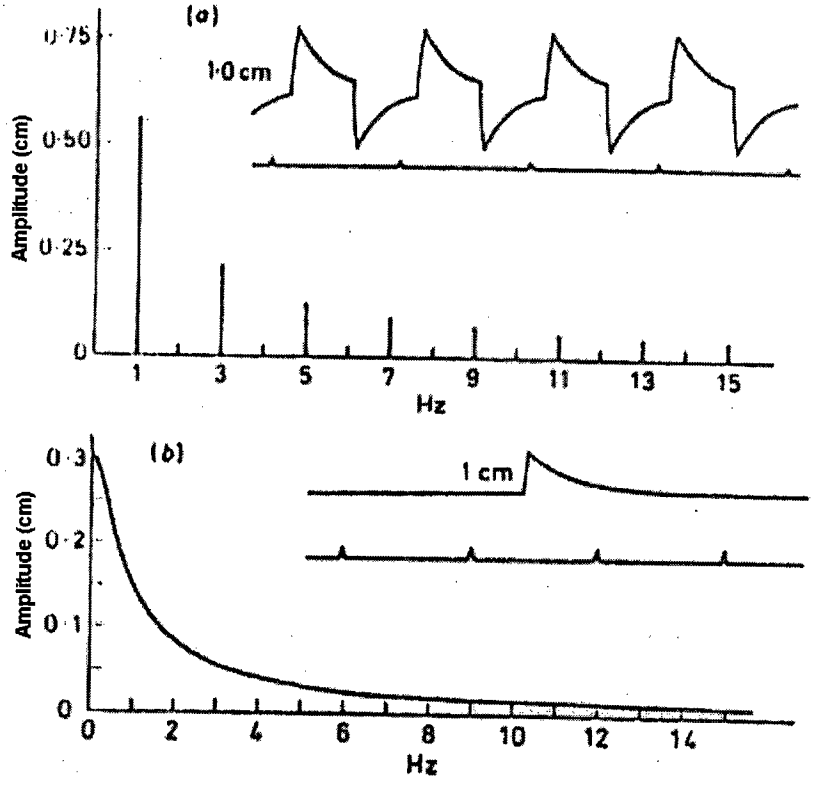

FIGURA 2.11 – Espectro da análise da Série de Fourier (Cooper et al., 1974).
(a) Espectro da análise da Série de Fourier de uma onda quadrada de calibração de 1Hz registrada com uma constante de tempo de 0,3s. (b) Espectro da análise da Integral de Fourier de um pulso de calibração.

Padrões rítmicos como os apresentados no sinal de EEG podem ser simulados pela adição de componentes de ondas sinusoidais. O processo inverso de separar um sinal em suas componentes senoidais é conhecido como análise espectral ou análise de freqüência. O resultado é que o sinal original, cuja amplitude é função do tempo, é convertido em um espectro de amplitudes. Este espectro pode ser contínuo (figura 2.11(b)) ou discreto (figura 2.11(a)).

Existem técnicas matemáticas para expressar qualquer forma de onda em termos de componentes sinusoidais de diferentes freqüências, mesmo que estes não apresentem características rítmicas. Isto é mostrado na figura 2.12.

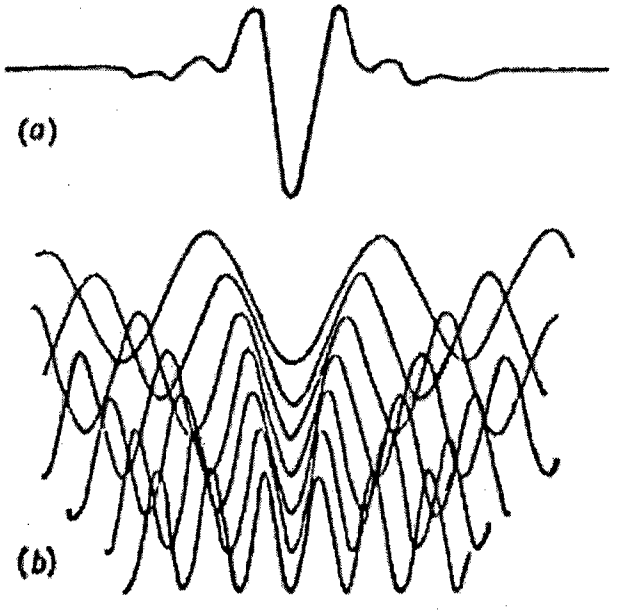

FIGURA 2.12 – Representação de um sinal em componentes sinusoidais (Cooper et al., 1974).
(a) Um pulso transiente. (b) Componentes sinusoidais de (a).

Na teoria de análise espectral é conveniente considerar-se três casos. O primeiro é a análise de sinais que podem ser representados pela soma de uma série de senoides cujas freqüências são harmonicamente relacionadas com a freqüência de repetição do sinal. As componentes apresentam amplitudes determinadas pelo padrão do sinal. Este método de análise é chamado de **análise por série de Fourier** e é mostrado

Estas alternativas são ilustradas na figura 2.14 que mostra uma onda senoidal e sua segunda harmônica somadas em três diferentes pontos de início (ou fase) da segunda harmônica. Isto resulta em três diferentes ondas complexas.

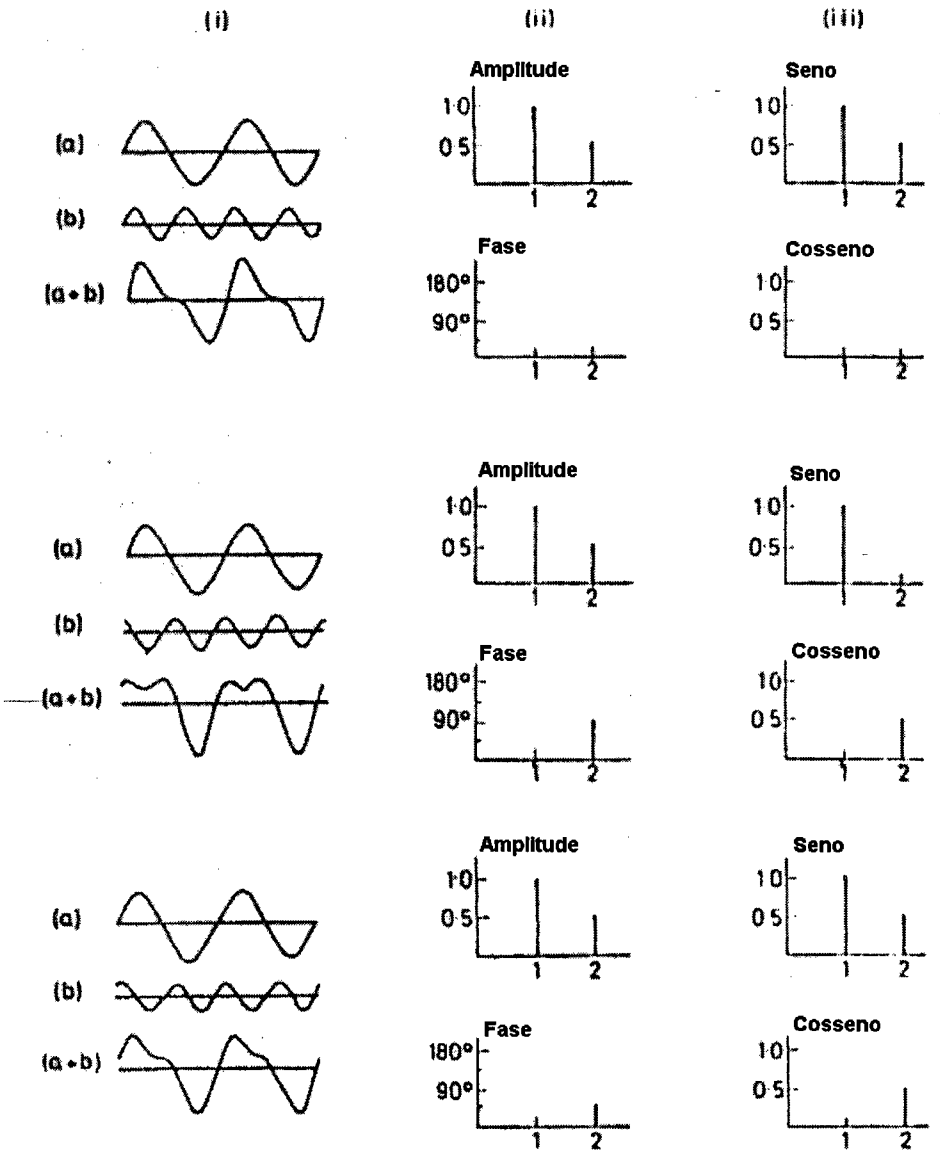

FIGURA 2.14 – Onda senoidal e sua segunda harmônica somadas em três diferentes pontos de início (ou fase) da segunda harmônica (Cooper et al., 1974).
Em (i) as ondas senoidais *a* e *b* são adicionadas para gerar a onda mais complexa *a+b*. A fase de *b* é diferente nos três casos. O espectro de amplitude e fase são mostrados em (ii), e os correspondentes espectros seno e coseno em (iii). Os números na escala de freqüência representam harmônicas.

2.6.4 Correlação

BASE TEÓRICA

O grau de similaridade entre dois sinais, tal como aqueles originários de regiões homólogas do cérebro, pode ser medido usando o coeficiente de correlação. A computação deste coeficiente é ilustrada na figura 2.15 para quatro casos separados. Em (a) os sinais são dois padrões similares de ondas senoidais; em (b) são dois padrões não-similares de ondas senoidais; em (c) temos dois sinais de EEG registrados simultaneamente de duas partes da cabeça e em (d) temos sinais de EEG registrados de duas fontes diferentes. Os pares de sinais em cada caso são mostrados nos dois primeiros traços das figuras. O terceiro traço mostra o resultado da multiplicação dos dois sinais. O valor médio deste produto (a integral dividida pelo tempo de integração) é chamado de covariância do correspondente par de sinais contanto que o valor médio de cada sinal seja zero. Como fica aparente na figura, a covariância é a medida da similaridade de dois sinais, mas ela depende também das respectivas amplitudes dos sinais. Desta forma, dois sinais podem possuir uma alta covariância por eles serem muito similares ou porque eles, apesar de serem pouco similares, apresentam grandes amplitudes. Para fazer a covariância independente da amplitude dos sinais, estes devem ser divididos pela raiz quadrada do produto da variância dos dois sinais. Esta covariância normalizada é denominada de coeficiente de correlação.

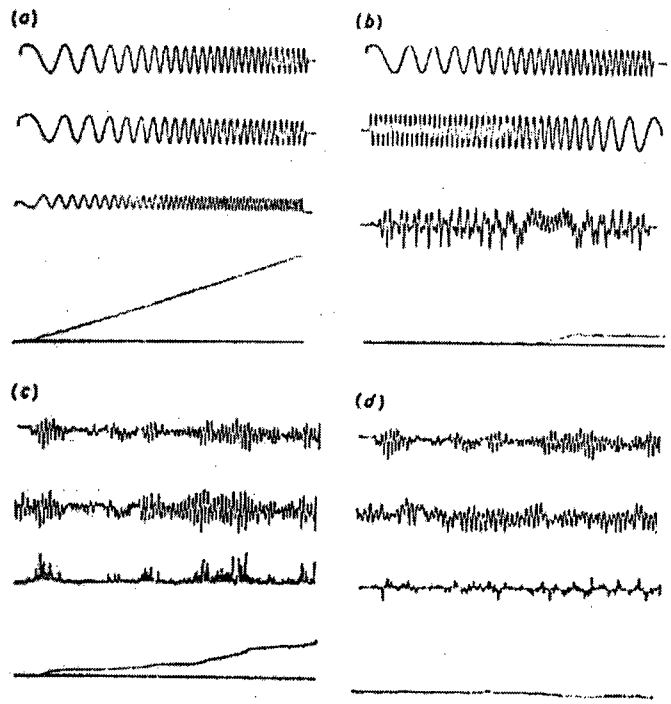

FIGURA 2.15 – Gráficos de covariância de sinais similares e não-similares (Cooper et al., 1974).

FUNÇÃO DE CORRELAÇÃO CRUZADA

Sinais com padrões similares freqüentemente ocorrem em diferentes partes de um sistema em instantes diferentes. Um exemplo de EEG é a atividade **alfa** que ocorre em instantes diferentes em posições diferentes do escalpo. O coeficiente de correlação entre dois sinais será máximo quando um dos sinais puder ser disposto no tempo com relação ao outro de forma que eles ocorram simultaneamente. Se a diferença de tempo não for conhecida ela pode ser calculada através do cálculo do coeficiente de correlação para um número de valores deslocados no tempo entre os dois sinais. A relação resultante entre o coeficiente de correlação e seu deslocamento no tempo é a **função de correlação**.

Quando o objetivo da função de correlação for apenas o de encontrar a diferença de tempo entre dois sinais, não é necessário normalizar o coeficiente de

correlação porque a covariância cruzada resultante terá um valor máximo quando a diferença de tempo entre os dois sinais for cancelada.

Se existir uma forte componente rítmica em cada traço as ondas entrarão e sairão de fase, a medida que uma for disposta no tempo em relação à outra. A função de covariância mostrará então uma onda cossenoidal na mesma freqüência do rítmo nas ondas primárias. Se — como no caso do EEG — a atividade não for exatamente periódica, o casamento entre os dois traços diminuirá. Conseqüentemente, as amplitudes da covariância também diminuirão. Se os dois sinais tiverem padrões de ruído similares e não apresentarem ritmicidade, o não-casamento aumentará rapidamente a medida que um for colocado em relação ao outro e a covariância irá rapidamente para zero.

Alguns exemplos de função de covariância cruzada são mostrados na figura 2.16(a), (b) e (c). Em cada um dos exemplos (i) mostra o registro de uma onda de calibração do equipamento e uma amostra de dois sinais sendo comparados — a duração do sinal é 40 s; (ii) mostra a função de covariância cruzada e (iii) mostra a função de auto-covariância para o segundo sinal em cada caso (que será explicada na próxima seção).

Em (a) a função de covariância cruzada apresenta uma onda cosenoidal cuja amplitude decresce exponencialmente nos dois lados do seu máximo. O decrescimento lento da amplitude mostra que existe uma componente de freqüência dominante compartilhada pelos dois sinais. O pico ocorre em um valor positivo de tempo, mostrando que um sinal está atrasado em relação ao outro.

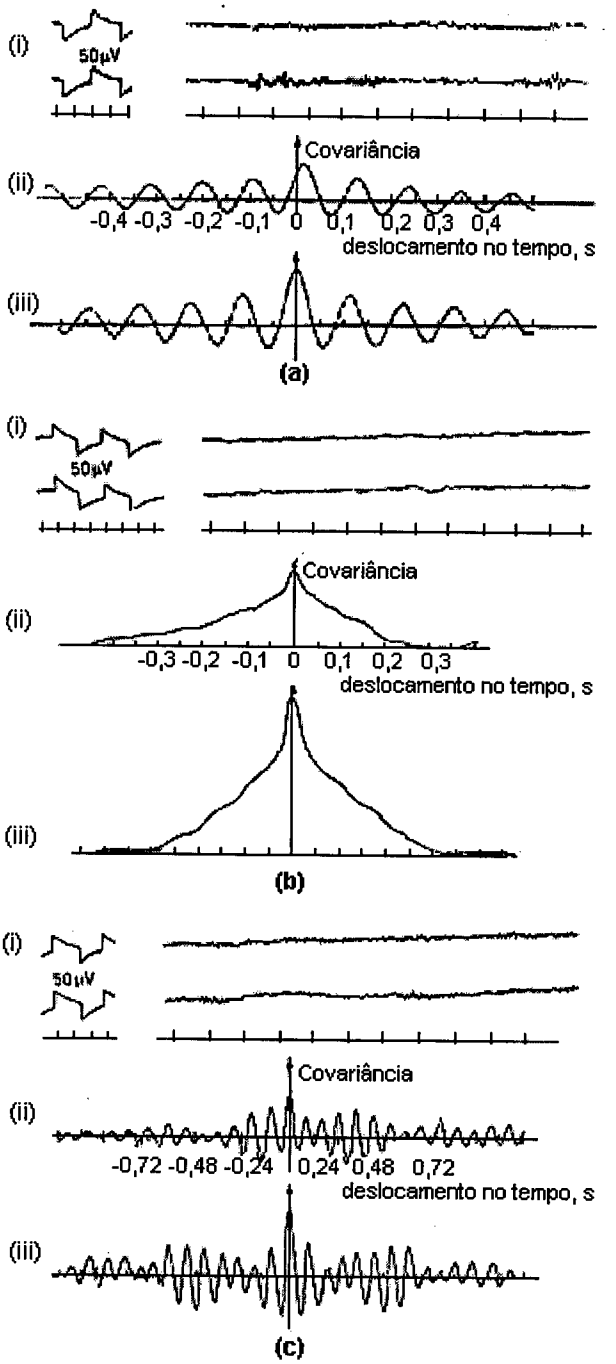

FIGURA 2.16 – Exemplos de (i) onda de calibração, (ii) funções de covariância cruzada e (iii) funções de auto-covariância (Cooper et al., 1974).

A função de covariância cruzada mostrada na figura 2.16(b) apresenta um decaimento mais rápido com uma componente oscilatória desprezível. Nenhum dos sinais primários apresenta muita ritmicidade e quase não existe atividade compartilhada pelos dois sinais.

Na figura 2.16(c) um caso mais complexo é mostrado. A função de covariância apresenta diversos componentes e um efeito de batimento é aparente, sugerindo que duas componentes de freqüências muito próximas são comuns aos dois sinais. Este último exemplo ilustra a dificuldade de interpretação de tais funções porque, se existe um atraso de tempo no pico da função de covariância, não está claro a que componente ele se refere.

FUNÇÃO DE AUTO-CORRELAÇÃO

Se os dois sinais utilizados para computar a correlação cruzada ou função de covariância forem o mesmo sinal (isto é, o sinal é correlacionado com ele mesmo deslocado no tempo), o resultado é chamado **função de auto-correlação**. Ela é utilizada principalmente para detectar periodicidades no sinal.

A função de auto-correlação sempre apresenta seu máximo valor ($+1$) em $t=0$ e é simétrica. A função de auto-covariância também apresenta seu valor máximo para $t=0$, onde ela é igual à variância do sinal. A forma destas ondas é determinada pela composição de freqüência e grau de previsibilidade do sinal. Elas normalmente apresentam certas flutuações oscilatórias decrescendo exponencialmente em amplitude a medida que o atraso aumenta. As oscilações refletem as componentes de freqüência do sinal primário. O decaimento exponencial indica o grau de randomicidade presente.

Funções de auto-covariância são mostradas nos exemplos (iii) da figura 2.16(a), (b) e (c).

2.6.5 Outros Métodos de Processamento de Sinais de EEG

INTRODUÇÃO

Métodos para a descrição quantitativa de sinais de EEG foram classificados por Hjorth (1970) em dois tipos: '(a) para definir quantidades para caracterização geral da amplitude e padrões de tempo, ex.: a aplicação de métodos estatístico-matemáticos que também incluem considerações de freqüência; (b) para descrever padrões individuais ou para detectar padrões específicos que podem ser definidos antecipadamente'.

ANÁLISE DE HJORTH

Hjorth desenvolveu em 1970 um método para análise de sinais de EEG. Três parâmetros de um sinal de EEG são medidos de épocas sucessivas de um a vários segundos de duração. Dois dos três parâmetros são obtidos a partir da primeira e segunda derivadas das flutuações de amplitude do sinal. Os três parâmetros usados por Hjorth são os seguintes. O primeiro é chamado **atividade** e é a variância das flutuações de amplitude no período. O segundo é chamado **mobilidade**. Este é obtido dividindo-se a variância calculada a partir da derivada primeira do sinal pela variância do sinal original (atividade) e extraindo a raiz quadrada do resultado. Isto resulta na medida da freqüência média do sinal. O terceiro parâmetro é a **complexidade**, às vezes chamado de fator do sinal e é a razão entre a mobilidade da derivada primeira do sinal e a mobilidade do sinal.

Uma onda senoidal apresenta uma complexidade igual a 1. Outras formas apresentam valores maiores, crescentes com sua complexidade.

Hjorth mostra que estes três parâmetros descrevem a forma do sinal e são matematicamente relacionados com o espectro de potência do sinal. Eles podem ser medidos facilmente por métodos analógicos ou digitais e isto pode ser feito *on-line*. Um exemplo de parâmetros Hjorth é apresentado na figura 2.17.

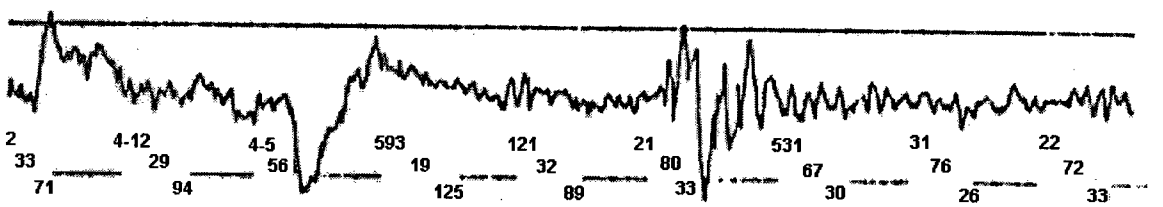

FIGURA 2.17 – Exemplo de análise de Hjorth (Cooper et al., 1974).
Exemplo de análise de Hjorth de um sinal de EEG mostrando *on-line* os parâmetros Hjorth ao final de cada período. Os números são os valores da variância, mobilidade e complexidade, mostrados de cima para baixo.

ANÁLISE DE AUTO-REGRESSÃO

Sinais como o EEG são imprevisíveis; por causa disso, estes sinais devem ser representados por parâmetros estatísticos tais como a média e a variância do espectro de freqüência de um período para outro. Não importa quanto se conhece das flutuações na amplitude do sinal em um determinado instante de tempo, o padrão de flutuação não pode ser previsto deste instante em diante. Outra característica deste tipo de sinal é que as medidas não são constantes de período para período, mesmo que mantidas as condições de aquisição, estado da fonte, etc. Como estas características são uma função do processo de geração do sinal, é mais apropriado derivar um modelo que produza o sinal em vez de descrever o próprio sinal. A saída de tal gerador deve apresentar as mesmas características estatísticas do sinal, embora a forma de onda gerada pelo modelo ser consideravelmente diferente do sinal do qual o modelo é derivado.

Dois modelos são comumente utilizados (Hurwicz, 1962). O primeiro é chamado aditivo porque assume que as randomicidades do sinal resultam da adição de

valores de amplitude randômicos a uma flutuação periódica. O segundo é chamado **auto-regressão**. O valor de amplitude a qualquer instante de tempo tem dois componentes: um é correlacionado com todos os valores passados, o outro é uma perturbação randômica adicionada. Embora este componente randômico adicionado em um instante particular seja independente daqueles adicionados em outros instantes de tempo, ele é correlacionado com os valores passados. Como resultado disso, o sinal tem uma função de auto-correlação periódica decaindo exponencialmente.

3. O SAASBIO III

Visando suprir as necessidades do projeto **SAASBIO** (Coimbra, 1994a), foi desenvolvido no Grupo de Pesquisas em Engenharia Biomédica o **SAASBIO II** (Rodrigues, 1996a; Rodrigues, 1996b; Rodrigues, 1997).

Neste sistema foi acrescentado um novo *hardware* que é composto de um bloco de aquisição, um bloco de processamento interligado ao computador, sendo também utilizado como uma base onde outros módulos podem ser conectados.

Durante a implementação do **SAASBIO II** foi desenvolvido um *software* para utilizar o equipamento. Este *software* foi desenvolvido em ambiente MS-DOS, com uma interface gráfica que apresenta os dados adquiridos de forma *on-line* e possui alguns recursos de filtragem. Para facilitar a interação do produto com o usuário final, o *software* precisa ser implementado em plataforma *Windows* (Holzner, 1994; Kruglinski, 1994; Yao, 1995), uma vez que este tipo de interface é muito mais intuitiva (devido aos diversos *softwares* desenvolvidos e a padronização de procedimentos adotada pelos desenvolvedores nesta plataforma).

O **SAASBIO II** possui boa precisão na aquisição dos dados mas carece de recursos para análise destes dados adquiridos, como filtros flexíveis, FFT e integração entre outros.

Outra característica a ser acrescentada ao **SAASBIO II** é a capacidade de análise estatística dos sinais adquiridos, para eliminar a necessidade de utilização de outros *softwares* específicos para este fim, com isso reduzindo o tempo necessário para a análise dos dados.

A partir dos problemas e necessidades encontrados no projeto **SAASBIO II** foram traçados os objetivos e metas a serem atingidos e, a partir destes, foi iniciado um novo projeto o **SAASBIO III**, ainda não concluído.

Primeiramente será feita uma breve descrição do projeto **SAASBIO II** de modo a facilitar o entendimento do novo projeto.

3.1 SAASBIO II

O projeto **SAASBIO II** (Rodrigues, 1997) foi desenvolvido pelo **Grupo de Pesquisas em Engenharia Biomédica** (**GPEB**) do Departamento de Engenharia Elétrica da Universidade Federal de Santa Catarina (UFSC) juntamente com o **Laboratório de Neurofisiologia I** (**LNI**) do **Centro de Ciências Biológicas** (**CCB**) da mesma universidade. Ele teve por objetivo suprir algumas necessidades e deficiências do projeto **SAASBIO**, desenvolvido pelo mesmo grupo.

Para facilitar o entendimento do *software* desenvolvido é necessário que se conheça algumas características do *hardware* que será utilizado. Uma vez que o hardware do SAASBIO III é basicamente o mesmo do SAASBIO II com algumas modificações, será feita uma rápida descrição deste para um entendimento mais global do sistema. Para uma explicação mais detalhada do SAASBIO II consulte Rodrigues (1997).

3.1.1 *Hardware* do SAASBIO II

O *Hardware* do SAASBIO II pode ser decomposto em dois módulos principais: módulo de aquisição e módulo de controle.

3.1.1.1 Módulo de Aquisição

A figura 3.1 apresenta um diagrama de blocos da etapa de aquisição do SAASBIO II.

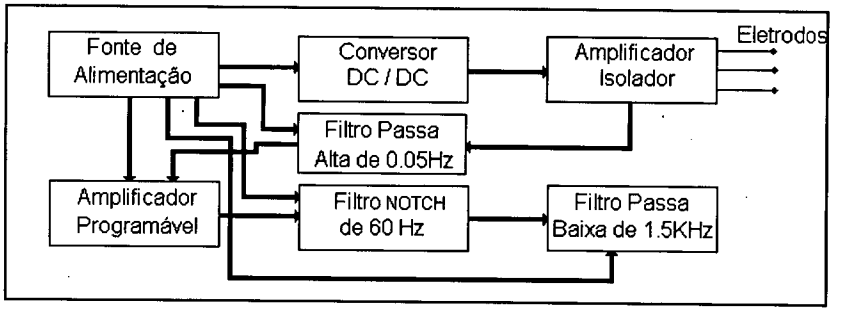

FIGURA 3.1 – Diagrama de blocos da etapa de aquisição (Rodrigues, 1997).

Para aumentar a segurança do paciente foram utilizados um transformador isolador e conversores DC–DC, pois desta forma o paciente fica isolado do circuito, evitando assim o choque elétrico.

AMPLIFICADOR ISOLADOR

Para aplicações médico-hospitalares é necessária a utilização de amplificadores isoladores para isolar o paciente do equipamento, evitando choque elétrico. A figura 3.2 mostra uma comparação entre um amplificador isolador e um amplificador de instrumentação.

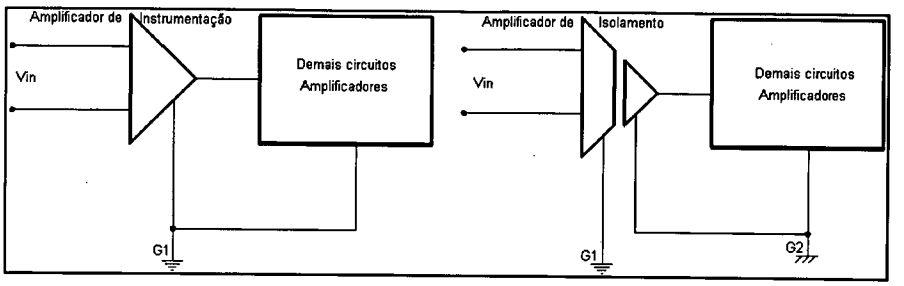

FIGURA 3.2 – Diferença entre ligações de terra entre amplificadores de Instrumentação e amplificadores com isolamento.

Nela é possível verificar que um amplificador isolador, diferentemente de um amplificador não-isolador, possui dois pinos de terra (**G1** e **G2**). Desta forma, existe um circuito fechado do lado do paciente e outro do lado do equipamento. Assim, eventuais problemas no equipamento, como curto circuitos e correntes de fuga, não afetarão o paciente. A isolação é feita comumente através de opto-isoladores (que utilizam fotodiodos e fototransistores para o isolamento) ou através de transformadores isolados.

AMPLIFICADOR PROGRAMÁVEL (PGA)

Devido à baixa amplitude dos sinais bioelétricos, torna-se necessária uma elevada amplificação dos mesmos de modo a podermos fazer a aquisição destes sinais. Estes sinais, porém, não apresentam a mesma amplitude, o que torna necessário um ganho ajustável. Na tabela 3.1 temos alguns valores de potenciais com suas respectivas amplitudes.

TABELA 3.1 – Níveis de tensão de alguns potenciais bioelétricos.

Sinal	Nível de Tensão
Potencial de ECG	50μV a 5mV
Potencial de EEG	2 a 300μV (escalpo)
Potencial de EMG	20μV a 10 mV
Potencial de EOG	10μV a 4mV

Para permitir o controle de amplificação pelo *software*, foi utilizado um amplificador com ganho programável digitalmente. Este amplificador foi composto pelos amplificadores de ganho programável **PGA202** e **PGA203** colocados em cascata (Rodrigues, 1997).

O **PGA202** apresenta ganhos de 1, 10, 100 e 1000 enquanto o **PGA203** apresenta ganhos de 1, 2, 4 e 8. Como os PGAs foram colocados em cascata, seus ganhos são multiplicados, resultando nos ganhos mostrados na tabela 3.2.

TABELA 3.2 – Ganhos do amplificador de ganho programável.

PGA202	PGA203	TOTAL
1	1	1
1	2	2
1	4	4
1	8	8
10	1	10
10	2	20
10	4	40
10	8	80
100	1	100
100	2	200
100	4	300
100	8	800
1000	1	1000
1000	2	2000
1000	4	4000
1000	8	8000

FILTRO PASSA ALTA DE 0,05HZ

Devido ao *offset* do amplificador isolador foi necessário a inserção de um filtro passivo passa-alta de 0,05Hz na saída do amplificador de isolamento. Esse tipo de filtro tornou-se necessário devido à altos ganhos proporcionados pelos amplificadores programáveis. Pois um nível muito pequeno de tensão DC na saída do amplificador isolador resulta na saturação dos amplificadores programáveis. Este problema pode ser

contornado com o ajuste manual da tensão de *offset* do amplificador isolador, porém esses ajustes são inviáveis para o SAASBIO II.

Um inconveniente causado pelo filtro passa alta, é a ocorrência de um atraso de tempo na eliminação do *offset* (de alguns segundos), na resposta do sinal de saída devido ao tempo de descarga do capacitor que compõe o filtro. Este atraso ocorre quando o amplificador é ligado ou logo após a modificação da escala de ganho dos PGAs, para amplificar um sinal de amplitude maior do que a anterior. Portanto, deve-se tomar o cuidado de desconsiderar os primeiros instantes dos registros.

FILTROS NOTCH DE 60 HZ

Existe a necessidade de inclusão de filtros Notch de 60 Hz para eliminar o ruído gerado pela rede. Este ruído pode afetar o sinal adquirido e causar distorções pois pode confundir-se com algumas harmônicas do sinal, dificultando a análise deste.

FILTRO PASSA BAIXA DE 1,5 KHZ

Ao final da cadeia de aquisição foi implementado um filtro ativo de segunda ordem para eliminar interferências de alta freqüência no sinal e para evitar possíveis erros na cadeia de conversão A/D.

O valor de 1,5 kHz para freqüência de corte do filtro foi adotado com base em um sinal de EMG, pois a faixa de interesse neste sinal pode chegar, no máximo, a 1 kHz. Como a freqüência do sinal miográfico é a maior freqüência a ser adquirida pelo SAASBIO II, um filtro de 1.5 kHz não afetará nenhum outro sinal bioelétrico, pois sua freqüência é bem maior que a dos demais potenciais.

3.1.1.2 Módulo de Controle

A etapa de controle, segundo módulo do SAASBIO II, responsável pela multiplexação e conversão dos sinais analógicos e controle da etapa de comunicação com um micro tipo IBM-PC, pode ser decomposta como mostra a figura 3.3.

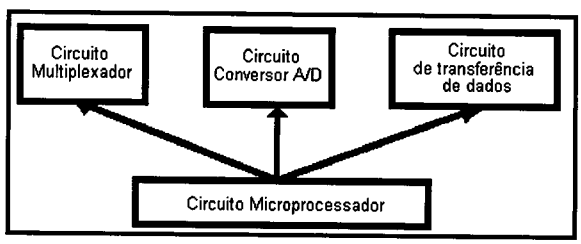

FIGURA 3.3 – Diagrama em blocos da etapa de controle do SASBIO II (Rodrigues, 1997).

MULTIPLEXADORES

Em sistemas de aquisição, onde existem muitos sinais de entrada, torna-se inconveniente utilizar um conversor A/D para cada canal, salvo no caso de se lidar com freqüências muito elevadas. Como os sinais são de origem biológica, onde a freqüência máxima está situada em torno de, no máximo, alguns kHz, a utilização de multiplexadores pode ser uma boa escolha, possibilitando a redução de custos do projeto.

Conversor A/D

O SAASBIO II realiza uma conversão com 16 canais analógicos. Por esse motivo, foi escolhido um conversor com taxa de conversão alta e interface de dados compatível com circuitos digitais e microprocessadores, o ADS 774 da Burr-Brown. Este conversor trabalha em modo bipolar, conversão para 12 bits ou 8 bits, por rotina de aproximações sucessivas, com tempo máximo de conversão de 8 μs, clock e

sistema de referência internos e saída paralela através de *buffers* de terceiro estado (*tri-state*) compatíveis com tecnologia CMOS.

Torna-se evidente que a etapa de conversão A/D dos sinais bioelétricos deve ser a mais rápida possível. Para isso necessita-se de uma boa taxa de amostragem (Jaeger, 1982). A aplicação e a forma de uso que são utilizadas nos conversores A/D determinam a amostragem necessária de nosso sistema. A figura 3.4, ilustra uma forma de determinarmos a taxa de amostragem, pela maior largura de banda do canal, o número de canais de dados e o número de amostras por ciclo.

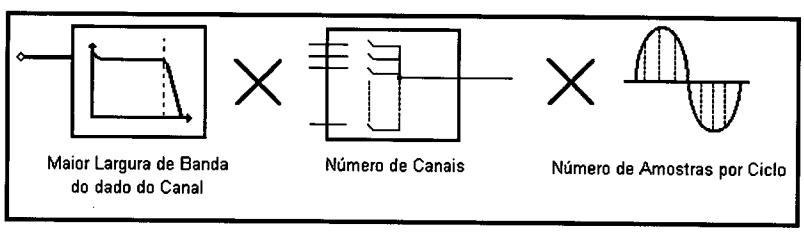

FIGURA 3.4 – Determinação da mínima taxa de amostragem do sistema (Rodrigues, 1997).

A partir da figura 3.4, temos que para determinarmos a taxa de amostragem devemos multiplicar a maior banda do canal, ou seja, a maior banda que possua todas as freqüências que o sinal apresenta e que são de interesse; o número de canais e o número de amostras por ciclo. Pelo critério de Nyquist, deve haver, pelo menos, duas amostras por ciclo de amostragem. Multiplicando-se este número de amostras pela maior banda do sinal, tem-se a taxa de amostragem por canal. Finalmente, multiplicando-se este valor pelo número de canais, tem-se a taxa desejada.

Por exemplo, para trabalhar com sinais de até 500 Hz e 8 canais, precisa-se, no mínimo, de:

$$500\ \text{Hz} \times 8\ \text{canais} \times 2\ \text{amostras} = 8000\ \text{amostras/s}$$

Havendo disponibilidade, é interessante utilizar mais do que duas amostras por segundo, pois assim haverá uma maior resolução do sinal amostrado, melhorando assim a morfologia do sinal.

O MICROCONTROLADOR

A meta do sistema de aquisição de dados bioelétricos é a criação de um sistema totalmente modular de pequeno porte e que possa ser conectado a um computador por um médico ou profissional da área médica. Por este fato, tornou-se necessário a criação de um sistema que não dependesse de instalação de placas internas aos microcomputadores, para facilitar sua implementação pelo usuário. Uma maneira de tornar mais ágil o processo de aquisição e amostragem dos sinais bioelétricos é trabalhar com um microcontrolador para monitorar as atividades de conversão e armazenamento temporário de dados.

Um microcontrolador da família Intel (80C31) foi utilizado para realizar as tarefas de comunicações e controle da etapa de aquisição. Este microcontrolador foi escolhido para o processo de gerenciamento do SAASBIO II devido a sua ótima relação custo/benefício. Existem limitações com esse microcontrolador se comparados a sucedâneos recentes. No entanto, pouco adianta utilizar controladores mais poderosos se apenas uma fração de seus potenciais é utilizada. Microcontroladores mais poderosos já possuem conversores A/D internamente, dispensando o uso de um conversor externo. Porém, estes tipos de conversores são muito lentos, com tempos de conversão que são bastante inferiores a 8 μs, tornando inviável a sua utilização nesta aplicação conforme concebida até aqui.

A função do microcontrolador 80C31 é gerenciar o processo de conversão A/D dos sinais provenientes da placa de aquisição, através de controle dos

multiplexadores e do conversor ADS 774. O microcontrolador possui também como função a comunicação com o IBM-PC. Essa comunicação se divide em duas etapas. A primeira é quando, através do *software*, inicializa-se o sistema. Nesta etapa o microcontrolador recebe informações do *software* sobre o canal a ser ativado, o ganho, a inserção ou não de filtros de 60Hz e freqüência de aquisição dos sinais analógicos. Após o microcontrolador receber essas informações, estas são passadas para a etapa de aquisição, onde as informações são armazenadas, e a placa é configurada. A segunda etapa de comunicação ocorre quando o microcontrolador envia os dados digitalizados pelo conversor A/D para o microcomputador. Nessa etapa, faz-se necessária uma comunicação à uma velocidade bem superior à anterior, pelo fato de se transmitir um volume de dados maior.

Na figura 3.5, é apresentado o esquema de controle feito através do microcontrolador 80C31, desde a etapa de comunicação até a etapa de programação da placa de aquisição.

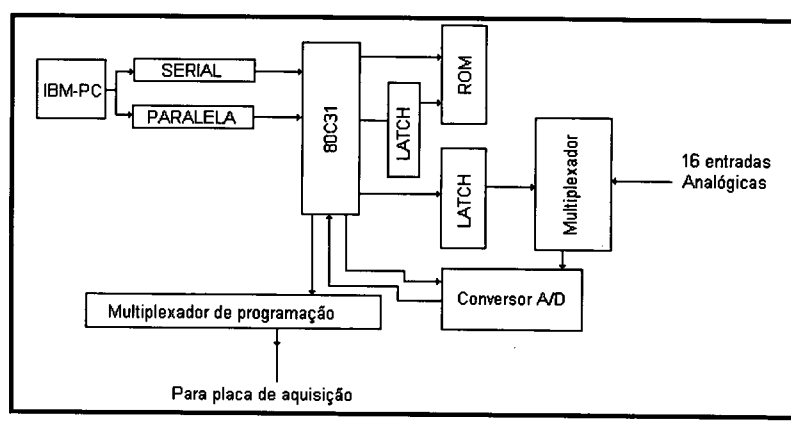

FIGURA 3.5 – Diagrama em blocos de funções de controle exercida pelo microcontrolador 80C31 (Rodrigues, 1997).

3.1.2 *Software* do SAASBIO II

O *software* do SAASBIO II foi desenvolvido utilizando a linguagem C, pois ela apresenta uma maior versatilidade para o interfaceamento com *hardware* externo e apresenta um alto desempenho para cálculos e impressões gráficas.

Este *software* é dividido em dois grandes módulos: o módulo de **gerenciamento** (situado no IBM-PC) e o módulo do ***firmware*** (situado na EPROM).

3.1.2.1 Módulo de Gerenciamento

O módulo de Gerenciamento tem a finalidade de controlar as atividades realizadas através do *mouse* e teclado, bem como o controle da etapa de aquisição e da programação do *hardware* do SAASBIO, através do auxílio de interrupções (Holzner, 1988).

A figura 3.6 apresenta uma tela do módulo de gerenciamento do SAASBIO II.

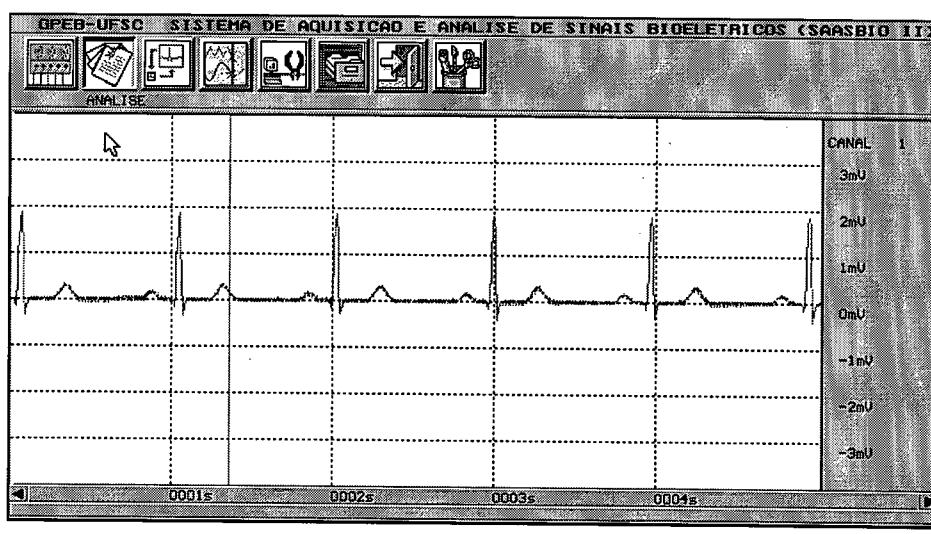

FIGURA 3.6 – Tela do SAASBIO II (Rodrigues, 1997).

Apesar da aparência semelhante aos *softwares* desenvolvidos sob plataforma *Windows*, todo o *software* do SAASBIO II foi desenvolvido sob plataforma DOS.

O *software* do SAASBIO II permite a aquisição de 16 canais, mas apresenta na tela um máximo de 8 canais (devido à resolução da tela e tempo de plotagem). Independente do número de canais plotados na tela, todos os canais adquiridos são armazenados em disco para posterior análise.

A partir do módulo de gerenciamento (e por intermédio do *firmware*) é possível controlar o *hardware* do SAASBIO II. É possível selecionar os canais desejados, ajustar o ganho dos PGAs, selecionar os filtros NOTCH.

No SAASBIO II a taxa de aquisição é igual para todos os canais, a fim de obter o mesmo número de pontos em todos os registros e facilitar o processo de correlação entre os sinais (Nilson, 1993).

A interface do PC com o *hardware* externo é feita através da interface paralela do microcomputador. Esta escolha se deveu ao fato da interface serial apresentar baixas taxas de transmissão enquanto a paralela apresenta taxas mais elevadas. Isto se torna necessário pois, no pior caso, precisa-se de uma taxa de 32000 *bytes* por segundo. Este caso é o de fazer-se aquisição nos 16 canais a uma taxa de 2000 amostras por segundo, pois a freqüência máxima que será adquirida é 1000 Hz e, pelo critério de Nyquist (Garret, 1981), a freqüência de amostragem necessária deve ser duas vezes a freqüência máxima para evitar o efeito de *aliasing*.

A figura 3.7 nos apresenta um esquema que ilustra o princípio do erro de *aliasing*. Onde uma onda senoidal de uma determinada freqüência se transforma em outra (após o processo de amostragem) de freqüência totalmente diferente da original.

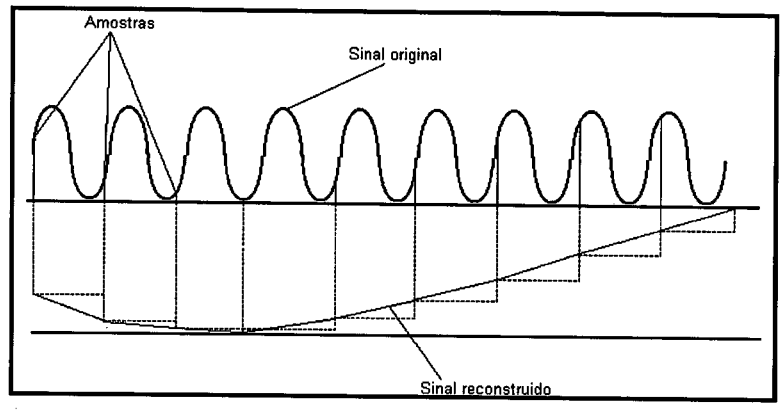

FIGURA 3.7 – Erro de *aliasing* para taxas de amostragem menores que duas vezes a freqüência do sinal amostrado (Rodrigues, 1997).

Após o sinal ter sido armazenado em disco é possível recuperá-lo para processá-lo. Por exemplo, a utilização de filtros de *hardware*, principalmente filtros NOTCH de 60Hz, afeta alguns componentes de freqüências que são importantes em determinados sinais, como o ECG de alta resolução (Marques, 1996). Através desse módulo, procedimentos para eliminação de interferências de 60Hz podem ser implementados via *software* nos sinais armazenados (Dotsinsky et al., 1996), eliminando os filtros de *hardware* e suas distorções em componentes de interesse no sinal.

Para fazer a aquisição e mostrar simultaneamente os dados adquiridos na tela do computador foi utilizado um processo de interrupção. Enquanto a rotina principal se preocupa em plotar os dados na tela e gerenciar o programa, existe uma rotina chamada por uma interrupção via *timer* que faz a comunicação com o *hardware* externo e armazena os dados no disco.

3.1.2.2 O Firmware

O *firmware* é o responsável pelo gerenciamento das atividades do microcontrolador e dos periféricos do módulo de controle do *hardware* do SAASBIO II.

As rotinas que constituem esse módulo foram desenvolvidas em linguagem C e compiladas através do compilador Franklin (Franklin 8051, 1993; Franklin A51, 1993; Franklin C51, 1993), proporcionando uma rápida interação com o *hardware*. Para mais detalhes a respeito do firmware, consulte Rodrigues, 1997.

3.2 O SAASBIO III

O projeto SAASBIO III vem para trazer melhorias ao projeto anterior, fornecendo uma interface mais amigável ao usuário final mantendo algumas características do SAASBIO II. Existem mudanças tanto de *hardware* quanto de *software*.

3.2.1 O *Hardware* do SAASBIO III

O *hardware* do SAASBIO III sofreu algumas modificações em relação ao anterior. A mais significativa delas para o escopo deste trabalho, foi a inclusão de dois *buffers* que são necessários devido a limitações do *Windows*, como será explicado mais detalhadamente no capítulo referente ao ANAMOD.

3.2.2 O *Software* do SAASBIO III

O novo *software* do SAASBIO III foi totalmente desenvolvido em ambiente *Windows* para melhorar ainda mais a interface com o usuário. Este ambiente (*Windows*) é muito mais intuitivo para o usuário, uma vez que grande parte das pessoas está acostumada a utilizar algum aplicativo que seja executado sobre esta plataforma, como processadores de texto e planilhas de cálculo. No sistema ANAMOD foi tomado o cuidado de adotar padrões já consagrados no ambiente *Windows* de modo a facilitar o uso deste *software*.

O sistema ANAMOD será explicado mais detalhadamente no capítulo seguinte, onde serão descritos alguns procedimentos na implementação do *software*.

4. O ANAMOD

O **ANAMOD** tem por objetivo dotar o **SAASBIO III** dos recursos necessários para a análise dos sinais biológicos adquiridos. Também deve ser acrescentado ao **SAASBIO III** a capacidade de comunicação com um computador IBM-PC compatível com uma interface amigável de modo a facilitar a interação com o usuário final.

O sistema **ANAMOD** é totalmente modular de modo a facilitar a inclusão de novos módulos de *software* para análise ou para controlar *hardware* adicional como os que estão sendo desenvolvidos nos projetos **IMPMOD** (módulo de medida de impedância) (Toazza, 1998a), **TELEMOD** (módulo de telemetria) (Bertemes, 1998) e **EVOMOD** (módulo de potenciais evocados) (Silva Jr., 1998), que estão sendo desenvolvidos em paralelo com o **ANAMOD**.

4.1 AMBIENTE DE DESENVOLVIMENTO DO *SOFTWARE*

O sistema **ANAMOD** está sendo implementado em plataforma *Windows* (Holzner, 1994; Kruglinski, 1994; Yao, 1995) de forma a suprir as necessidades do **SAASBIO III** relativas à facilidade de uso pelo usuário final.

A facilidade de uso pelo usuário final deve-se ao fato do *Windows* oferecer uma interface de usuário padrão. Um aplicativo "compatível com *Windows*" é o que — para um usuário de *Windows* experiente — se comporta de modo previsível, pelo menos no que diz respeito à interface do programa. Isto facilita o aprendizado de novos programas e facilita memorizar tarefas comuns, como editar, imprimir e obter ajuda.

Para ter uma idéia melhor de como essa padronização simplifica as coisas para o usuário, considere um comando simples — o de sair de um aplicativo. A Tabela 4.1 mostra alguns dos vários comandos de saída entre os aplicativos não-*Windows*.

TABELA 4.1 – Comandos de saída

Aplicativo	**Comando de Saída**
dBASE III	Digite "quit" no sinal de prontidão
Lotus 1-2-3 para MS-DOS	/W(orksheet) Q(uit) Y(es)
Editor UNIX "vi"	:q!
Tango	M(enu) F(ile) Q(uit)

Essas diferenças surgem devido aos estilos de interface de usuário e porque estes programas foram escritos sem a preocupação com padrões de interface.

Os aplicativos *Windows* compartilham um comando de saída comum, o que, para um usuário que saiba sair de um aplicativo, facilita saber como sair de qualquer outro aplicativo *Windows*.

4.1.1 Programação Orientada por Eventos

Os programas escritos para DOS normalmente são feitos de maneira seqüencial e orientado por procedimentos. Este tipo de programa tem início, meio e fim bem definidos. Por exemplo, um programa que faça o cadastro de clientes de uma empresa que exibe uma série de telas de entradas de dados para criação de um arquivo

em meio permanente (disco) implementado seqüencialmente é apresentado na figura 4.1.

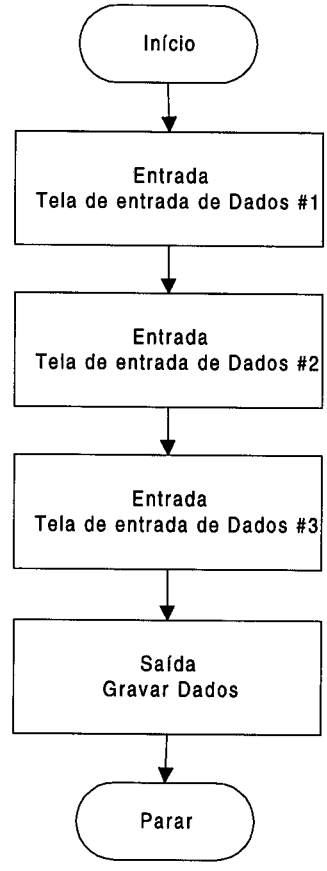

FIGURA 4.1 – Um programa orientado seqüencialmente.

Nesta figura a tela de entrada de dados #1 é para entrada dos dados pessoais do cliente, a tela #2 para dados da empresa e a tela #3 para outros dados. Cada tela de entrada deve ser preenchida corretamente, antes que se possa prosseguir, e as três telas devem ter informações corretas para serem salvas no arquivo.

À primeira vista este parece ser um modo razoável de processar o programa. Contudo existem limitações nessa abordagem, que são resultado direto desta orientação seqüencial.

Por exemplo, como o programa dita a seqüência de operação, o usuário não pode ir à segunda tela sem primeiro introduzir todas as informações da primeira tela.

Embora o programa garanta que todas as informações exigidas tenham sido introduzidas, ele não leva em conta as exceções do mundo real. Por exemplo, se estiverem sendo cadastrados vários clientes de uma mesma empresa, seria preciso percorrer todas as telas para cada cliente.

Por outro lado, um programa orientado por eventos permite ao usuário que introduza os dados na ordem que lhe parecer adequada. E esta é a principal diferença da abordagem seqüencial sobre a abordagem orientada a eventos. Enquanto na primeira, o programa tem o controle, na segunda quem comanda é o usuário. A figura 4.2 apresenta o mesmo programa adotando a programação por eventos.

Porém, esse é apenas um aspecto pelo qual um programa orientado por eventos difere de um seqüencial.

Um programa seqüencial é construído em um conjunto mal organizado de **modos**. Um modo é o estado de um programa no qual as ações do usuário são interpretadas de uma maneira específica e produzem um conjunto específico de resultados.

Um grande problema relacionado aos modos ocorre quando o usuário não pode ir de um modo para outro facilmente. Outro problema com os modos ocorre em programas que dependem de que o usuário lembre o modo corrente. Em vez disso, o programa deve oferecer pistas visuais para ajudar o usuário a identificar o modo corrente.

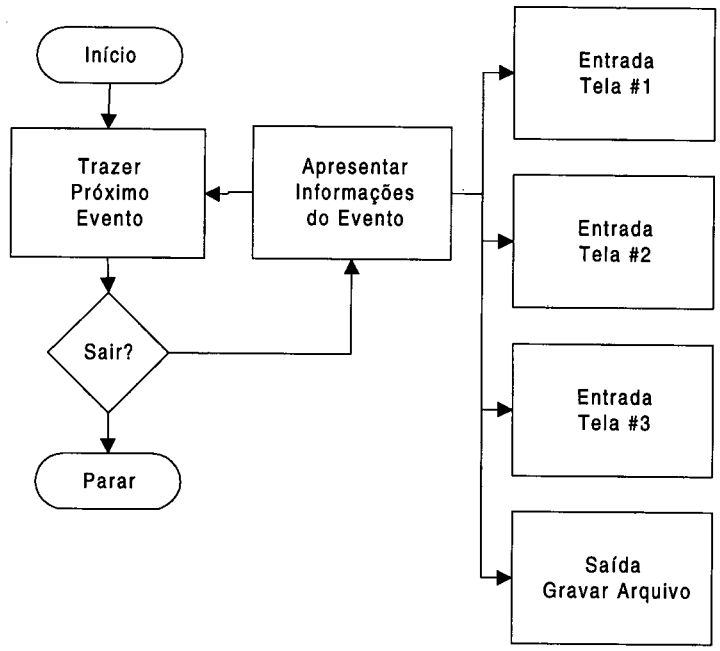

FIGURA 4.2 – Um programa orientado por eventos.

Do ponto de vista de programação, os programas modais são mais fáceis de serem implementados que os que não têm modo. O código que suporta cada modo pode ser escrito e depurado em relativo isolamento das outras partes do programa. Porém, o *Windows* facilita a criação de programas sem modo, pois toda a interação com o mundo exterior — todos os eventos — é direcionada a um programa assim. Todos os eventos de interesse geram uma mensagem.

As mensagens são informações sobre uma alteração na interface de usuário como uma janela sendo movida ou uma tecla sendo pressionada.

INTERFACE DE PROGRAMAÇÃO DE APLICATIVOS

Quando se fala em desenvolver um aplicativo para rodar em um sistema operacional, normalmente não há distinção entre o **sistema operacional** (**OS** — *Operational System*) e a **Interface de Programação de Aplicativo** (API — *Application Program Interface*). Uma API é um conjunto de regras para acessar um conjunto de

serviços. As regras incluem funções, tipos de dados e quaisquer considerações especiais sobre a obtenção dos serviços. Juntas, essas regras estabelecem um contrato entre os aplicativos que utilizam um serviço e o fornecedor do serviço.

Como alguns sistemas operacionais suportam várias APIs, é importante distinguir entre um sistema operacional e as APIs suportadas. A tabela 4.2 lista vários sistemas operacionais e as APIs que cada um deles suporta.

TABELA 4.2 – Sistemas operacionais e as APIs suportadas.

Sistema Operacional	**API(s) Suportada(S)**
MS-DOS	MS-DOS
Windows 3.1	MS-DOS, Win16
Windows 3.1 c/ Win32s	MS-DOS, Win16, Win32s
OS/2 2.0	MS-DOS, Win16, OS/2 caractere, OS/2 Presentation Manager
Windows NT / 95	MS-DOS, Win16, Win32, POSIX, OS/2 baseado em caractere

MENSAGENS E ESCALONAMENTO DO PROGRAMA

Há uma estreita correlação entre as mensagens e o escalonamento do processador, já que uma mensagem representa um pedido de tempo de processador. O que isto significa realmente depende da implementação específica do *Windows*. Todas as implementações do *Windows* são multitarefa, o que significa simplesmente que vários programas podem estar em execução simultaneamente. Mas existem dois mecanismos básicos usados para compartilhar o processador nos vários sistemas *Windows*.

O *Windows* 3.x usa um sistema de escalonamento **não-preemptivo**. Os programas *Windows* não são interrompidos pelo sistema operacional; em vez disso, cada programa interrompe sua própria operação voluntariamente para permitir que outros programas sejam executados. Esse sistema funciona, porque esses programas precisam de muitas mensagens e, assim, cada um deles verifica se o sistema operacional tem mensagens para eles. Desde que todos os programas sigam estas regras, esse sistema de multitarefa é tão eficiente quanto um outro sistema de escalonamento preemptivo.

Entretanto, um ponto fraco deste sistema é que algum programa pode não seguir as regras. Um programa poderia, por exemplo, tornar-se "ocupado" ou simplesmente parar. Quando a linha de código a seguir é executada em um programa *Windows*, por exemplo, a interface do usuário inteira pára:

```
while (1);
```

Esta linha de código aguarda até que a expressão entre parênteses seja falsa, ou seja, igual a zero. Evidentemente que uma situação como esta não aconteceria desta forma. O que pode acontecer é que seja utilizada uma expressão dentro dos parênteses que, por algum problema não previsto pelo programa, não mude, permanecendo sempre verdadeira. Naturalmente, isto torna o sistema inteiro vulnerável a um problema em qualquer programa.

Por compatibilidade, os programas do *Windows* NT e *Windows* 95 também são orientados por mensagens. Mas o mecanismo de envio de mensagens funciona junto a um sistema multitarefa mais forte, baseado em *threads* (enlaces). Um enlace é a unidade de escalonamento sob o *Windows* NT e o *Windows* 95. Às vezes, os enlaces são chamados de processos leves, pois possibilitam o escalonamento independentemente de um processo sem a sobrecarga a ele associada. Quando um programa roda sob o *Windows* NT ou *Windows* 95, um processo é criado e a ele é dado um enlace. Esse enlace é escalonado independentemente dos outros enlaces do sistema. Embora uma mensagem que espera por um enlace específico dê a ele uma maior prioridade, o sistema de escalonamento do *Windows* NT e do *Windows* 95 não pode ser perturbado por um "laço infinito" que faz o escalonador do *Windows* 3.x parar.

4.1.2 Saída Gráfica

Toda saída gerada pelo *Windows* é gráfica. Saída gráfica significa que figuras geométricas podem ser desenhadas — linhas, círculos, caixas, etc. Além disso, o próprio texto é tratado como um objeto gráfico. Isso facilita, por exemplo, misturar livremente texto e figuras geométricas. Paradoxalmente, enquanto os sistemas de saída gráfica facilitam a saída de formas geométricas, também tendem a tornar a saída de texto mais difícil.

As formas geométricas são mais fáceis, pois o programa não precisa calcular cada *pixel*. Pela simples chamada da rotina `Rectangle`, por exemplo, a GDI (*Graphical Device Independent* — Gráficos Independentes de Dispositivo) desenha um retângulo preenchido. A saída de texto é mais difícil, pois a orientação gráfica da GDI exige que se trate texto como um objeto gráfico. O texto é posicionado usando-se coordenadas de pixel no lugar de coordenadas de célula (utilizada em sistemas baseados em texto — cada caractere ocupa uma célula).

A GDI oferece gráficos independentes do dispositivo. Isto significa que um programa *Windows* pode desenhar em qualquer dispositivo usando o mesmo conjunto de funções. Por exemplo, a rotina `Rectangle` é chamada para desenhar retângulos no vídeo e também em impressoras. A GDI esforça-se para que, do ponto de vista do programa, todos os dispositivos pareçam semelhantes. Isso inclui os dispositivos que só sabem ligar e desligar *pixels* — como a placa de vídeo — bem como dispositivos que conseguem fazer desenhos complexos, como as impressoras *PostScript*. Cada dispositivo tem um controlador responsável por fazer o desenho real. Para dispositivos que precisam ajuda, a GDI oferece **simulações de *software*** que usam os recursos de baixo nível de um dispositivo para oferecer funcionalidade de alto nível.

4.1.3 Arquitetura do Sistema *Windows*

Os programas *Windows* utilizam três bibliotecas: KERNEL, USER e GDI. Estas bibliotecas aparecem com os nomes **KRNL286.EXE** ou **KRNL386.EXE**, **USER.EXE** e **GDI.EXE** no *Windows* 3.x. Já no *Windows* NT e *Windows* 95 os nomes são **KERNEL32.DLL**, **USER32.DLL** e **GDI32.DLL**. A figura 4.3 mostra essas bibliotecas junto com as partes principais do *Windows* 3.x. Um diagrama equivalente para o *Windows* NT e *Windows* 95 aparece na figura 4.4.

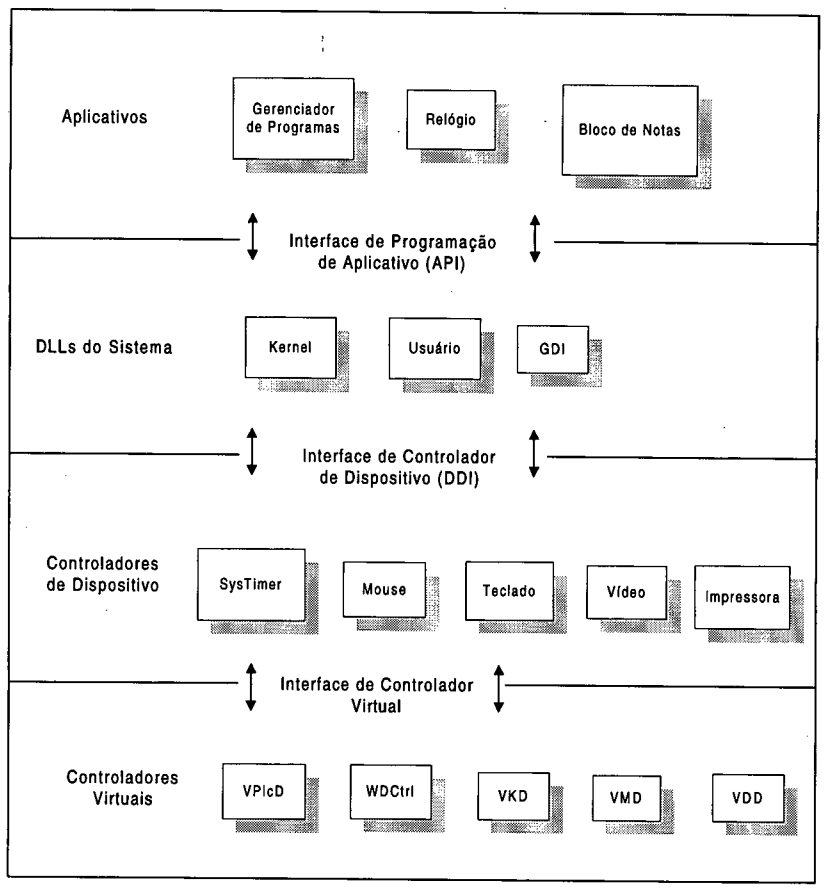

FIGURA 4.3 – Principais componentes do *Windows* 3.x.

Cada componente cria e gerencia um conjunto de objetos do sistema: KERNEL cria memória, USER — o componente de **interface com o usuário** — cria janelas e GDI — a interface de **Gráficos Independentes de Dispositivo** — cria canetas

e pincéis para desenho. Quando um componente cria um objeto, uma identificação numérica — conhecida como *handle* ou manipulador — é gerada. Para usar um objeto, é necessário passar este *handle* para a função da API adequada.

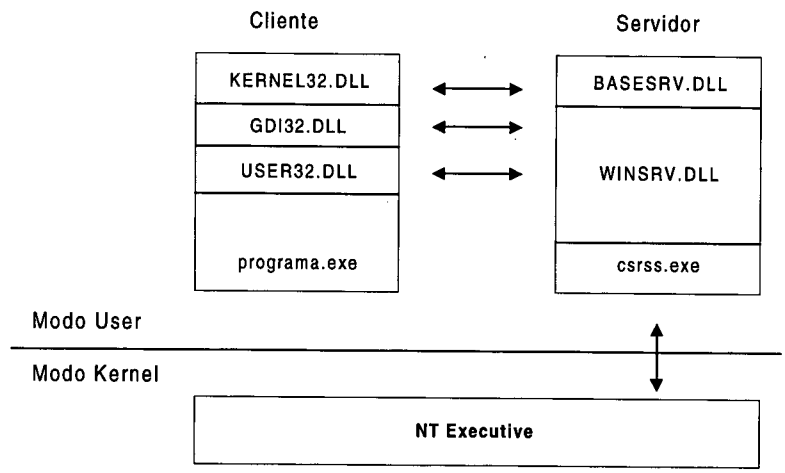

FIGURA 4.4 – Principais componentes do *Windows* NT e *Windows* 95.

KERNEL

O módulo KERNEL do *Windows* é responsável pelo suporte de baixo nível para E/S em arquivo, memória, carga de programa, ligações dinâmicas e tarefas tradicionalmente pertinentes a um sistema operacional. Embora o diagrama mostre três as bibliotecas de sistema em um mesmo nível, na realidade, a KERNEL suporta a USER e a GDI, quando elas precisam de seus serviços.

USER

O módulo USER trata da criação e do gerenciamento de objetos de interface de usuário, incluindo janelas, menus, caixas de diálogo, controles de caixa de diálogo, ponto de inserção, cursores e ícones. O escalonamento de aplicativos — algo que parece ser tarefa adequada ao KERNEL — também é de domínio do USER. Sob o

Windows 3.x, o escalonamento de aplicativos é *inteiramente* de domínio do USER. O motivo é que a atividade de *mouse* e teclado gera **mensagens**, e isso serve como a fatia de tempo do *Windows* 3.x. Sob o *Windows* NT e *Windows* 95, o USER compartilha essa tarefa com o NT *Executive*, que tem um escalonador preemptivo. Quando o USER recebe entrada do usuário — isto é, quando as mensagens são geradas — ele chama o *Executive* para auxiliar a prioridade do destinatário. Deste modo o USER32 executa uma tarefa no escalonamento de aplicativos desproporcional à importância normalmente dada à interface do usuário.

GDI

É a responsável por ligar os *pixels* na tela ou desenhar os pontos que serão impressos pela impressora do sistema. A GDI é chamada pelos aplicativos quando eles querem criar qualquer tipo de saída visual. Mesmo o USER, que gerencia a operação interna da interface do usuário, depende da GDI para desenhar janelas, menus, caixas de diálogo e todos os outros objetos.

4.2 LINGUAGEM DE PROGRAMAÇÃO

Foi escolhida a linguagem C++ (Montenegro et al., 1994) para desenvolvimento do sistema **ANAMOD** por diversas características apresentadas pela linguagem. Entre elas a modularidade, o alto desempenho, o acesso a rotinas de baixo nível e a portabilidade da linguagem.

MODULARIDADE

Devido ao fato de ser uma linguagem de programação orientada a objetos, o C++ possui as características necessárias para tornar o *software* modular,

possibilitando reaproveitamento de código para a implementação de novas rotinas a serem agregadas ao *software*.

Novas partes de código poderão ser acrescentadas ao código existente reutilizando as rotinas existentes através de recursos de linguagens orientadas a objetos como a herança (Montenegro et al., 1994).

Poderão, inclusive, ser criados outros programas, independentes do **ANAMOD**, e posteriormente serem incluídos no sistema de forma relativamente fácil e rápida (desde que estes programas também sejam desenvolvidos nesta linguagem).

ALTO DESEMPENHO (VELOCIDADE)

Devido ao elevado volume de cálculos que são realizados durante o processamento e a análise dos sinais, é necessária uma linguagem que possua um bom desempenho com alta velocidade nos cálculos. Além disso, durante a aquisição, o *software* deverá estar plotando os dados adquiridos na tela, armazená-los em disco e fazer algum processamento extra, como filtragem.

Devido às características da linguagem C (e da linguagem C++ — derivada da linguagem C), ela produz um código muito otimizado, com isto é possível tirar o máximo proveito do processamento, tornando esta linguagem ideal para programas que exijam alta velocidade. Devido à implementação estar sendo feita em ambiente *Windows*, um ambiente "orientado a objetos e eventos", opta-se pela linguagem C++ que apresenta as mesmas características da linguagem C, em um ambiente orientado a objetos.

ACESSO A ROTINAS DE "BAIXO NÍVEL" (CABRAL, 1989)

Uma característica interessante das linguagens C e C++ é a possibilidade de acesso a rotinas de acesso a *hardware* e o uso de ponteiros de uma maneira extremamente simples. O acesso a este tipo de estrutura é imprescindível no sistema **ANAMOD**, pois todos os recursos de memória necessários ao sistema serão alocados dinamicamente, ou seja, serão criados durante a execução do programa. Isto facilita a otimização da memória a ser utilizada pelo sistema, permitindo que os programas utilizem apenas os recursos do sistema enquanto estiverem necessitando deles, liberando os mesmos quando não forem mais necessários.

PORTABILIDADE C++

Outra característica interessante é a vasta implementação da linguagem em múltiplas plataformas (DOS, *Windows*, UNIX, etc...). Esta característica permite, caso haja interesse, uma mudança de plataforma apenas alterando a *interface* do sistema, sendo possível manter todo o código de processamento inalterado.

4.2.1 Compilador Utilizado

Uma vez escolhida a linguagem de programação (C++) tornou-se necessária a escolha do compilador a utilizar para a confecção do *software*. Este compilador deve ser capaz de gerar código utilizável na plataforma *Windows* 95, uma vez que esta foi a interface escolhida para o sistema.

Inicialmente, optou-se pela utilização do compilador *Visual C++ 4.0* da *Microsoft*, que era o compilador disponível para programação em ambiente *Windows*.

Com o decorrer do trabalho, foram surgindo várias dificuldades na utilização do *Visual C++*, a maioria delas no que se refere à criação da interface gráfica

do sistema **ANAMOD**. O desenvolvimento da interface do usuário é bastante trabalhosa no *Visual C++*. Para criar a interface básica existem alguns assistentes que auxiliam nesta tarefa. Entretanto, para a expansão da interface, é necessário um conhecimento mais aprofundado do *Windows* e do próprio compilador *Visual C++* pois apesar do nome *Visual* ele não apresenta uma forma facilitada de criação da interface como o *Visual Basic*, também da *Microsoft*, que é um aplicativo do tipo **RAD** (*Rapid Application Development* — Desenvolvimento Rápido de Aplicativo).

Apesar do compilador *Visual C++* apresentar uma documentação *on-line* bastante extensa, por diversas vezes fica difícil encontrar-se o que realmente se necessita. Somando-se a isto, existe o fato de que não havia ninguém trabalhando com este compilador no GPEB, não havendo portanto alguém próximo para discussões a respeito dos problemas e auxílio nas soluções.

Outro ponto a ser considerado é que não se encontrou nenhuma bibliografia a respeito de acesso direto a dispositivos do sistema, como portas paralelas e interrupções. A única maneira de se utilizar a porta paralela era a utilização de rotinas próprias para este fim existentes na API. O problema é que algumas destas rotinas são específicas para utilização da porta paralela com a impressora e as outras (relativas ao uso de portas paralelas tipo EPP — *Enhanced Parallel Port* — e ECP — *Enhanced Communications Port*) não são muito bem documentadas.

Durante o desenvolvimento do *software*, tomou-se conhecimento da existência do compilador *C++ Builder* da *Borland*. Este sim, uma aplicação do tipo **RAD**, que facilita muito a criação da interface do *software*. O *C++ Builder* apresenta uma interface de criação de aplicativos muito semelhante à do *Delphi* da *Borland* ou

Visual Basic da *Microsoft*, com a vantagem de ser um compilador C++, que apresenta um bom desempenho, como descrito na seção anterior.

Com este compilador, foi possível desenvolver-se rapidamente a interface do *software*, sendo dedicado mais tempo a outros problemas como a comunicação paralela e o uso de interrupções no *Windows* 95.

Para resolver estes problemas, foi utilizado um *driver* encontrado na *Internet* desenvolvido por *Victor Ishikeev* (*victor@ivi.ugatu.ac.ru*). Juntamente com este *driver* foi desenvolvido uma interface para utilização com o *Delphi* e *C++ Builder*. A partir desta interface, é possível utilizar os recursos do *driver* de forma simplificada, pois é criado um componente e as funções e variáveis do *driver* são acessados como se fossem membros do objeto criado (Montenegro et al., 1994).

4.3 IMPLEMENTAÇÃO DO SISTEMA

O **ANAMOD** apresenta uma interface bastante amigável proporcionando ao usuário um rápido aprendizado das funcionalidades do sistema. O programa possui menus com todas as funções e procedimentos disponíveis e barras de ferramentas com as funções e procedimentos mais utilizados de modo a agilizar o acesso a estes.

Entre as características do programa podemos citar também o uso de interface MDI (*multiple document interface* — interface de múltiplos documentos) que possibilita a abertura de mais de um documento para análise, podendo serem feitas comparações visuais entre os dados obtidos e outros já armazenados em meio permanente.

4.3.1 Descrição da Interface do *Software*

A tela de apresentação do programa **ANAMOD** é apresentada na figura 4.5.

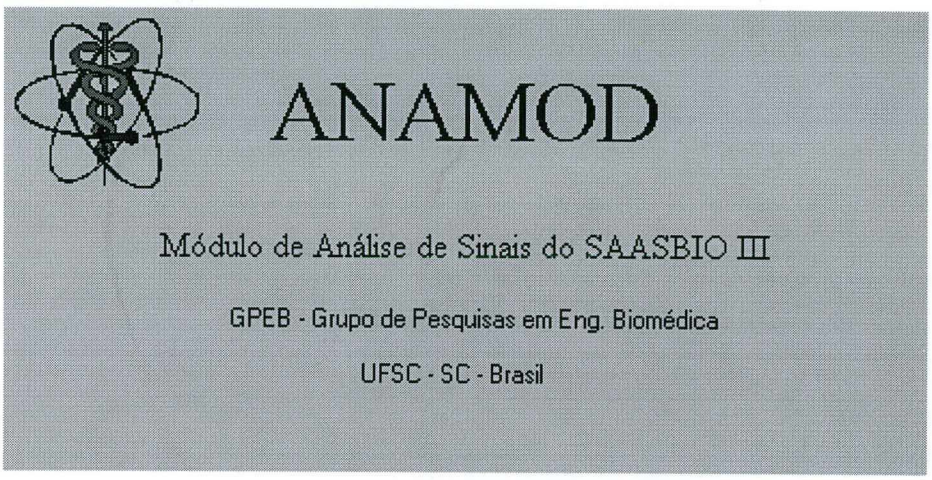

FIGURA 4.5 – Tela de apresentação do ANAMOD.

Após, é apresentada a tela principal do programa com uma janela de aquisição vazia para que possa ser feita uma aquisição. Não é obrigatório que o *software* inicie fazendo uma aquisição, mas como esta vai ser normalmente a primeira opção do usuário, é criada uma janela no início do programa. A figura 4.6 mostra a tela inicial do programa com a janela de aquisição.

Para iniciar uma aquisição primeiramente será necessário configurar o sistema. Para isto, basta clicar no botão **Configura** e será apresentada a caixa de diálogo da figura 4.7:

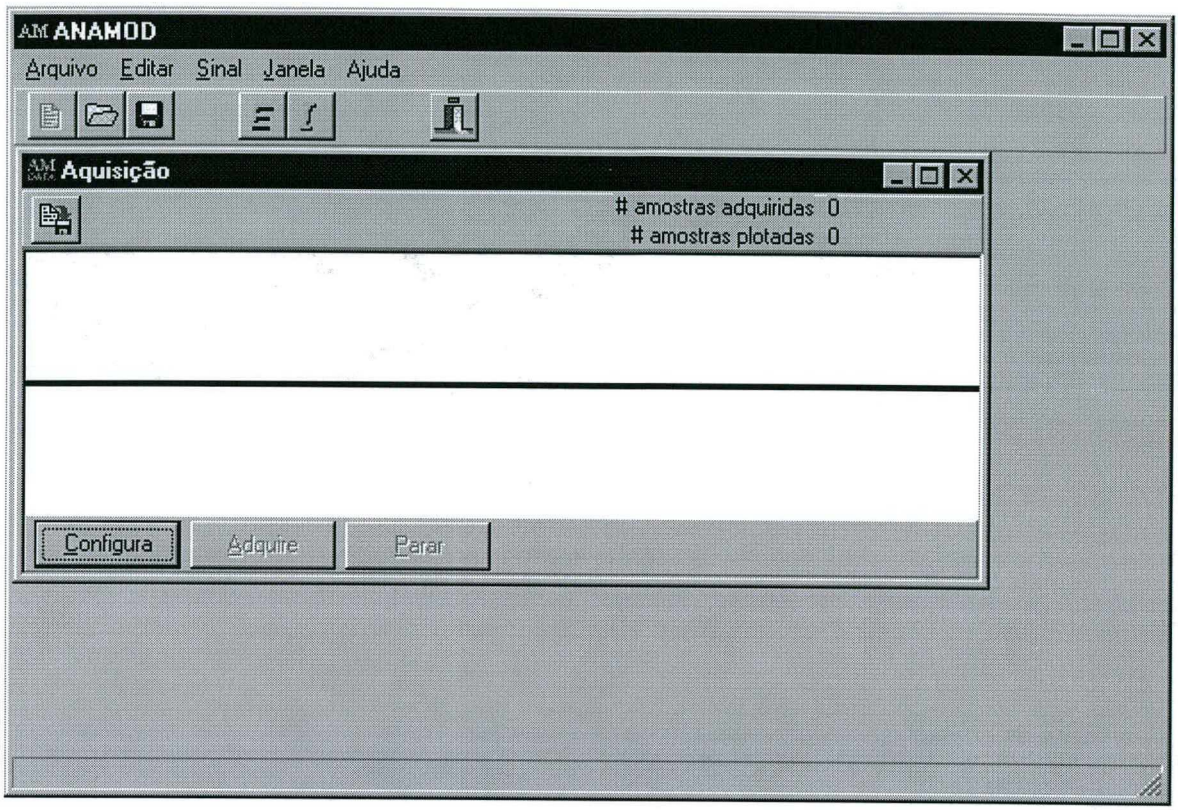

FIGURA 4.6 – Tela principal do **ANAMOD** com a janela de aquisição.

Nesta tela serão apresentados valores padrão para os itens listados. **Data** e **Hora** são obtidas do sistema, e tanto estes como os outros valores poderão ser modificados pelo usuário. **Arquivo** é o nome que será dado ao arquivo gerado na aquisição. **Data** e **Hora** são relativas à aquisição. **Número de Canais** define quantos canais serão utilizados na aquisição (quais deles será definido no próximo item). O **Número de Pontos** define o número máximo de pontos a ser adquirido. O usuário pode interromper a aquisição quando desejar antes disso, mas não será possível exceder este valor na aquisição. O motivo disso está relacionado a alocação de memória e será discutido mais adiante. **Número de *bits*** refere-se à resolução a ser adotada pelo conversor **A/D**. **VMáx** refere-se ao valor máximo de tensão utilizado na aquisição, ou seja, a tensão de fundo de escala para o **A/D**. Atenção, esta tensão é dada em milivolts.

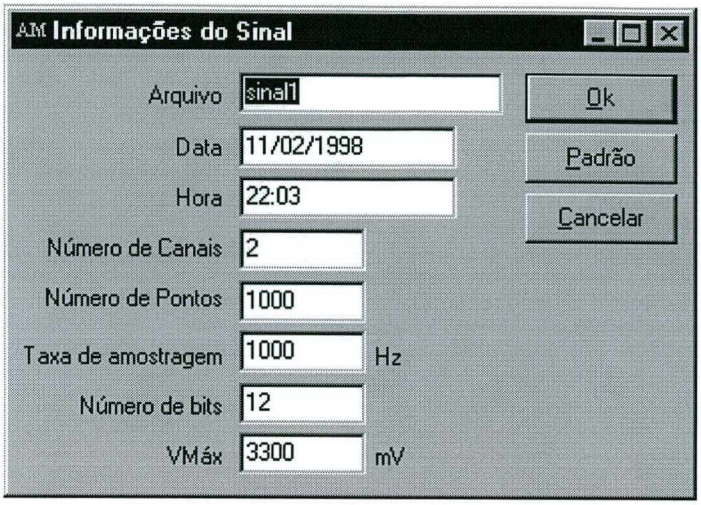

FIGURA 4.7 – Configuração do sistema **ANAMOD**.

A seguir é necessário configurar cada canal. Note que serão acrescentados mais botões à barra de ferramentas da janela de aquisição. Será um botão por canal definido na caixa de diálogo de configuração do sistema. Ao clicar-se nestes botões, aparece a caixa de diálogo ilustrada na figura 4.8. Note que neste momento já são criadas as linhas divisórias dos canais.

FIGURA 4.8 – Configuração dos canais (exemplo com o primeiro canal).

Nesta caixa de diálogo são apresentados valores padrão como na anterior. Pode-se modificar o número do canal (referente ao *hardware*), o nome que o usuário desejar, bem como o ganho do canal e a inclusão ou não do filtro *notch*.

Após configurar o sistema basta clicar em **Adquire** e será iniciada a aquisição. A aquisição só irá parar ao número de pontos adquiridos chegar ao valor configurado ou clicar-se no botão **Parar**. A figura 4.9 mostra um sinal sendo adquirido.

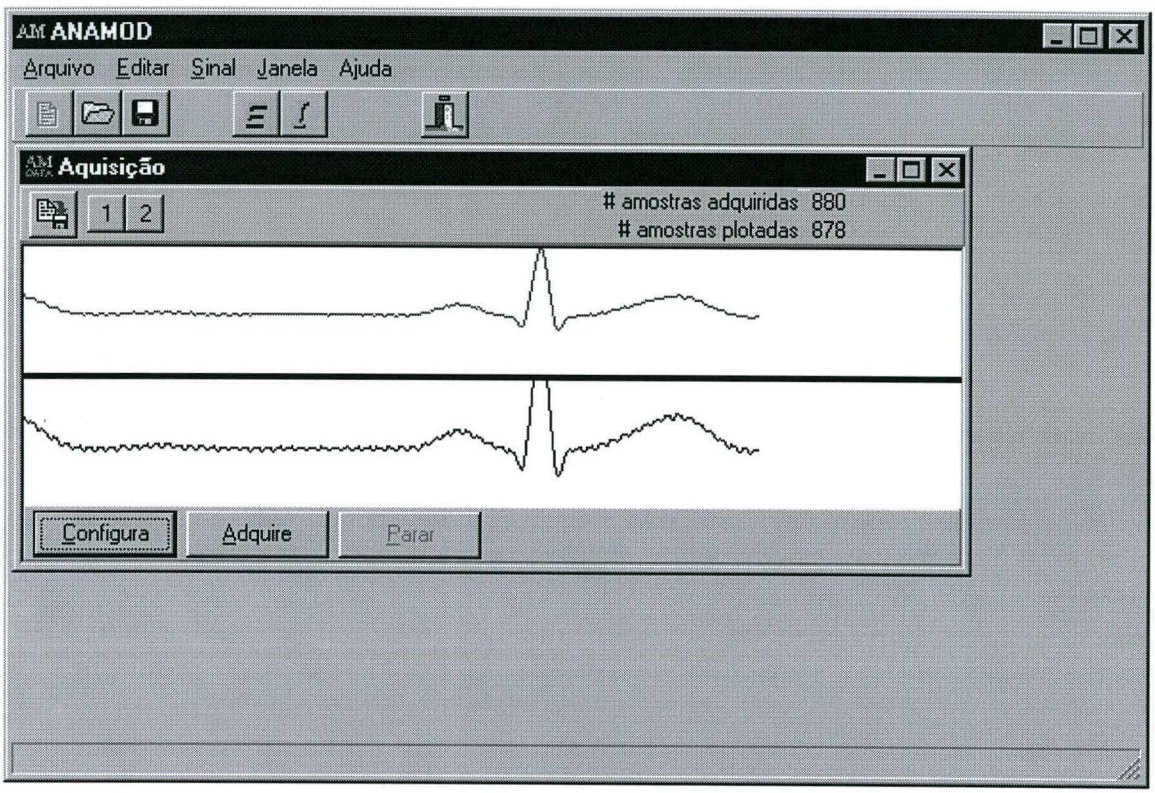

FIGURA 4.9 – Exemplo de sinal sendo adquirido pelo sistema **ANAMOD**.

Após o sinal ser adquirido, o mesmo pode ser armazenado em disco utilizando o botão salvar (ao lado dos botões de configuração de canal). O sinal será salvo em dois arquivos. Um com a extensão **inf**, conterá informações a respeito do canal (basicamente os dados das caixas de diálogo de configuração do sistema e dos canais). O outro arquivo possuirá a extensão **dta** e possuirá os dados do canal em formato texto, sendo um valor por linha alternando os canais. Por exemplo em dois canais a primeira linha terá o primeiro dado do primeiro canal, a segunda o primeiro dado do segundo canal, a terceira o segundo dado do primeiro canal e assim por diante. Foi escolhido o

formato texto e manter os dados separados da configuração para facilitar a troca de informações com outros *softwares* como planilhas de texto, entre outros.

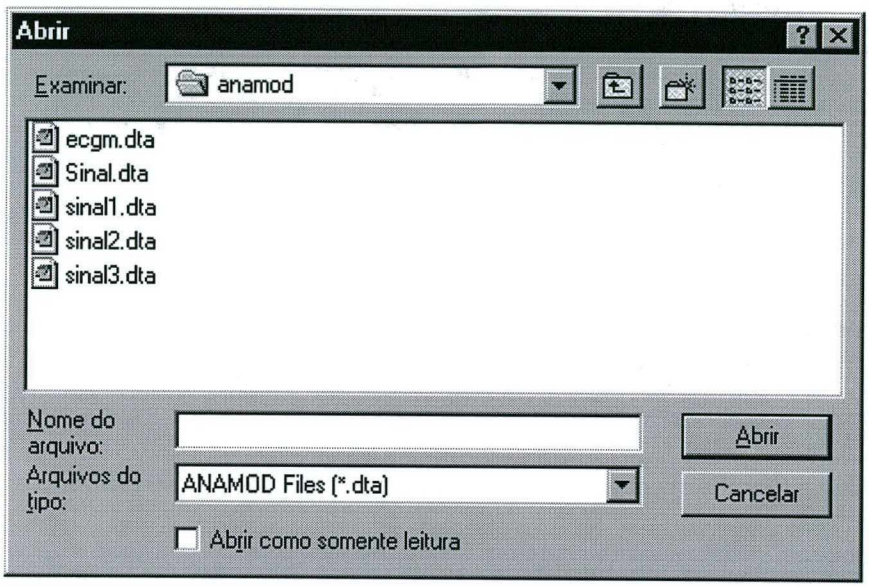

FIGURA 4.10 – Caixa de diálogo para selecionar arquivo.

Outra função disponível no **ANAMOD** é a de visualizar os arquivos salvos em aquisições anteriores. A figura 4.10 mostra a caixa de diálogo utilizada para abrir o arquivo e a figura 4.11 mostra um exemplo de sinal carregado pelo sistema. Observe que para a abertura do arquivo basta informar o arquivo com a extensão **dta**, o outro arquivo (extensão **inf**) é aberto automaticamente.

A rotina de visualização de arquivos permite uma manipulação dos dados melhor que a de aquisição. Nesta tela, além de serem plotados todos os pontos (na tela de aquisição isto não ocorre, como será explicado mais adiante) é possível ampliar o sinal na horizontal e vertical através de funções de *zoom*, utilizando-se as caixas marcadas com **Eixo X** e **Eixo Y**, onde será definido o fator de ampliação.

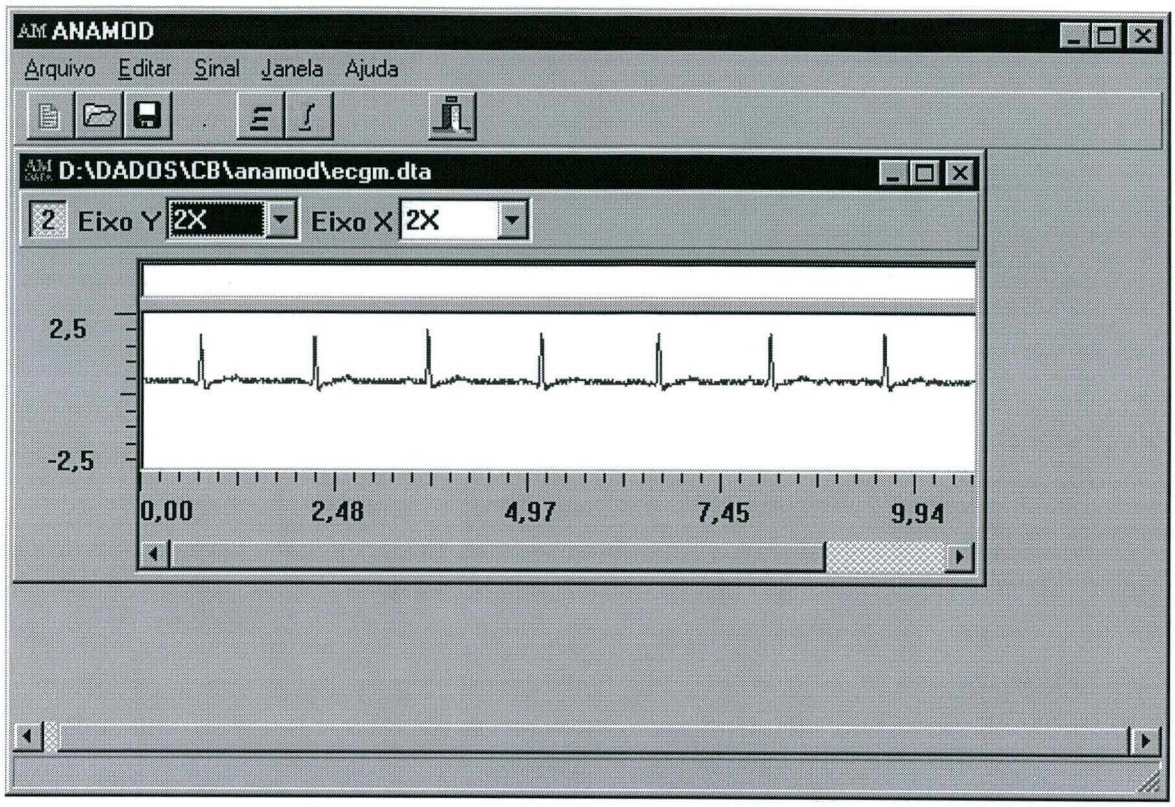

FIGURA 4.11 – Exemplo de tela do programa e janela com arquivo carregado.

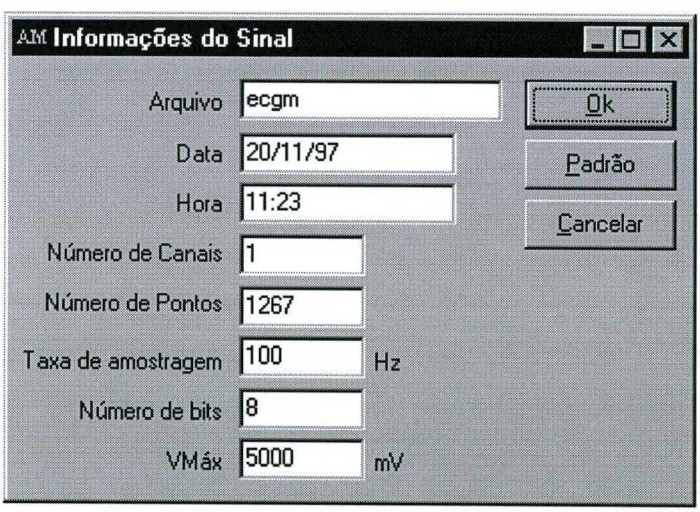

FIGURA 4.12 – Configuração do sinal de exemplo.

FIGURA 4.13 – Configuração do sinal de exemplo (continuação).

Note que para o sinal de exemplo foram carregadas as informações de configuração como mostram as figuras 4.12 e 4.13.

Devido ao tempo limitado e às dificuldades encontradas no desenvolvimento em ambiente *Windows* 95, dificuldades estas que serão descritas mais adiante, não foi possível a implementação das rotinas de tratamento de sinal, como filtros, transformadas entre outros. Mas tendo sido o sistema desenvolvido de forma modular, fica relativamente fácil a inclusão de novas rotinas no sistema posteriormente.

4.3.2 Descrição do *Software*

O *software* não apresenta muitas dificuldades de entendimento. As rotinas de implementação mais complicadas e que serão descritas em mais detalhes são a rotina de aquisição e a rotina de plotagem *on-line*. As demais rotinas serão comentadas rapidamente, pois são bem simples e os comentários colocados no programa e o próprio código são suficientes para seu entendimento.

ROTINA DE AQUISIÇÃO

Esta rotina foi implementada com o auxílio de um *driver*, desenvolvido por *Victor Ishikeev* (*victor@ivi.ugatu.ac.ru*), encontrado na *Internet*. Este *driver* permite que seja criada uma rotina para atender a uma determinada interrupção, que é definida com alguns parâmetros que são passados ao *driver*.

Depois de configurado o *driver*, e criada a rotina de atendimento à interrupção, basta habilitar o *driver* e ele automaticamente atenderá a interrupção.

Para o acesso à porta paralela foi utilizado um procedimento descrito em um artigo encontrado na internet sob o título: *"Use of a PC Printer Port for Control*

and Data Acquisition", de autoria de *Peter H. Anderson (pha@eng.morgan.edu)* do Departamento de Engenharia Elétrica da *Morgan State University*.

Este artigo descreve algumas técnicas para utilizar a porta paralela a partir de um programa DOS. Com algumas modificações foi possível fazer o acesso a porta paralela a partir do *Windows*.

Para o armazenamento dos dados de cada canal adquirido e de forma a manter o programa modular, foi criada uma classe chamada `TCanal` no sistema que é encarregada de armazenar todas as informações relativas a um canal, bem como os seus pontos. Na verdade, a estrutura de `TCanal` apenas guarda as informações relativas ao canal. Os pontos são armazenados em uma propriedade herdada de outra classe: `TSinal`. O sistema foi desenvolvido desta forma para que fiquem separados os conceitos de sinal e canal. Todas as rotinas que sejam aplicáveis a um sinal qualquer devem ser acrescentadas à classe `TSinal`, enquanto rotinas específicas para tratamento dos canais devem ser acrescentadas à `TCanal`. Isto foi feito para que haja um máximo reaproveitamento de código possível. Rotinas como filtros, transformadas e coisas do gênero são aplicáveis a sinais genéricos e não especificamente aos canais do **ANAMOD**. Desta forma, um outro *software* ou até mesmo um outro módulo do **ANAMOD** que não esteja trabalhando com canais de aquisição poderão utilizar as rotinas que se destinam a sinais genéricos, não tendo que carregar junto estruturas como número e nome de canal, ganhos, etc.

É preciso ficar bem claro, no entanto, que todas as funções pertencentes à `TSinal` são acessíveis em `TCanal`, já que o segundo e gerado com base no primeiro através de um processo de *herança* (como descrito em Montenegro et al., 1994), e é

através deste recurso que novas classes podem ser criadas em outros programas para reaproveitar os recursos de `TSinal`.

Na classe `TSinal`, encontra-se uma estrutura destinada a armazenar os pontos da aquisição. Para uma maior otimização da memória do micro, a memória é alocada dinamicamente, ou seja, existe uma estrutura do tipo ponteiro em `TSinal` e, quando necessário, é reservada uma área de memória com o tamanho definido pelo número de pontos máximo da aquisição. Daí vem o motivo pelo qual não é possível adquirir-se mais pontos do que o estipulado no início (durante a configuração do sistema). Caso o sistema seja interrompido antes do preenchimento desta área de memória, nada de errado acontece, apenas haverá uma área de memória reservada que não será utilizada. Já se a aquisição exceder o número de pontos previsto, os pontos serão armazenados em uma área que não foi reservada para eles. O resultado disso é imprevisível, mas normalmente representa um travamento do programa ou faz com que o programa pare e emita uma mensagem de erro.

ROTINA DE PLOTAGEM *ON-LINE*

A rotina de plotagem deve conseguir plotar o máximo de pontos adquiridos sem "travar" o *Windows*. "Plotar o máximo de pontos" refere-se ao fato de que para plotarmos um ponto na tela devemos utilizar a API do *Windows* ao invés de utilizar diretamente a placa de vídeo do *Windows*. Isto deve-se ao fato do *Windows* utilizar uma alocação de recursos virtual, como é explicado com mais detalhes na próxima seção. Com isso, dependendo da taxa de amostragem escolhida, não é possível plotar todos os pontos na tela, pois enquanto estiver sendo plotado um ponto, a rotina de interrupção já terá recebido mais pontos e não será possível sincronizar as duas rotinas.

Optou-se então por plotar os pontos de acordo com a disponibilidade do sistema, ou seja, plota-se um ponto; se enquanto este ponto estiver sendo plotado chegarem mais pontos, estes são armazenados na memória (para que possam ser armazenados em disco posteriormente) e são descartados da rotina de plotagem. Com isto, consegue-se obter um gráfico que se aproxima bastante do sinal adquirido a menos de eventuais espículas ou variações bruscas que possam haver entre um ponto plotado e outro, mas isto poderá ser visto após os pontos serem salvos e depois recuperados pela rotina de leitura de arquivo em disco.

Na *Internet* é possível encontrar alguns *drivers* para fazer acesso direto à placa de vídeo de forma a acelerar a plotagem de pontos na tela. Este tipo de rotina pode ser útil para acelerar a rotina de plotagem e pode ser tentada para aperfeiçoar o *software* em trabalhos futuros.

Para que possamos plotar o máximo de pontos sem trancar o *Windows* (colocando-o em um laço muito demorado) fica evidente que não podemos "prender" o programa principal na plotagem de pontos enquanto a rotina de aquisição se encarrega de adquirir os pontos e armazená-los na memória. Isto funcionaria muito bem em DOS, mas no *Windows* isto não é possível, pois se a rotina fosse implementada desta forma, não seria possível fazer mais nada no *Windows* enquanto a aquisição não terminasse, inclusive forçar a parada da aquisição. O motivo disso já foi discutido anteriormente e refere-se ao fato da mulitarefa no *Windows* ser baseada em mensagens. Apesar do *software* estar sendo implementado em *Windows* 95, que utiliza multitarefa preemptiva, isto não deve ser utilizado, pois o programa não ficará travado até o final da plotagem do gráfico como aconteceria no *Windows* 3.x, mas demoraria muito para o *Windows*

receber o controle e isto aconteceria por um curto instante de tempo, retomando em seguida o controle para o laço do programa.

Como uma primeira tentativa, optou-se por utilizar um *timer* do sistema para que a plotagem fosse feita de em intervalos igualmente espaçados. O problema disso é: como programar o *timer* para que o programa fique a maior parte do tempo plotando sem sobrepor as rotinas? Vale ressaltar que mudando a máquina (processador) ou simplesmente se houverem mais aplicativos abertos ao mesmo tempo, a carga do sistema é maior e estes tempos mudam.

A solução foi utilizar um evento chamado `OnIdle`, que é chamado pelo sistema *Windows* cada vez que o programa entra em estado de espera, ou seja, cada vez que o programa pára de fazer alguma coisa. Este evento foi associado a uma rotina que ficou encarregada de fazer a plotagem dos pontos. A partir daí o programa passou a se comportar como demonstra o fluxograma da figura 4.14.

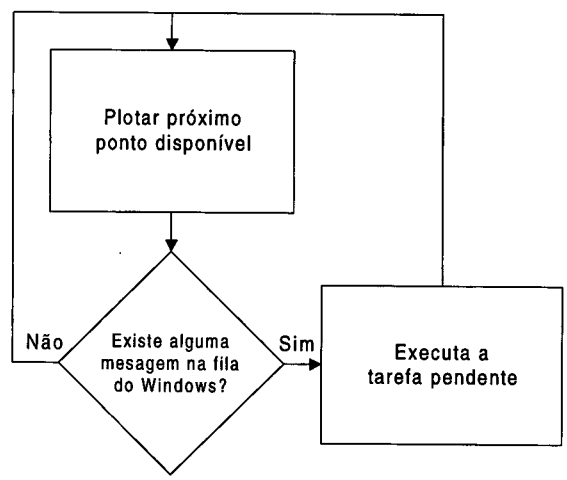

FIGURA 4.14 – Representação do funcionamento da rotina de plotagem *on-line*.

Desta forma, foi possível garantir que, independente da máquina ou de sua carga atual, o programa plotará o máximo de dados possível.

4.3.3 Outras Rotinas

A rotina de armazenamento dos dados em meio permanente utiliza o nome do arquivo definido durante a configuração do sistema e armazena os dados na forma de um arquivo texto. Foi escolhido este formato para permitir um intercâmbio de informações mais facilitado com outros *softwares* como planilhas de cálculo como o *Excel* ou de processamento de sinais como o *MatLab*. A interface com o *MatLab* permite que rotinas de processamento de sinais futuramente desenvolvidas possam ser testadas com sinais adquiridos ainda no ambiente do *MatLab* para que eventuais erros no algoritmo possam ser detectados, de forma que ao fazer a implementação em C++ o programador possa preocupar-se somente com a interface do *software* sem ter que localizar tais erros.

Outras rotinas como as utilizadas para o desenvolvimento da interface, tais como menus, botões e caixas de diálogo entre outros, foram desenvolvidas utilizando a abordagem de programação orientada a objetos, o que torna o código modular e bastante fácil de ser compreendido. Nas rotinas onde poderiam surgir dúvidas quanto à interpretação foram acrescentados comentários descrevendo mais detalhadamente o código. Foram acrescentados comentários também no início de cada sub-rotina que descrevem a utilização desta, quando necessário.

Uma classe criada para utilização no **ANAMOD** e que pode ser de grande utilidade para desenvolvimentos futuros é a classe `MCoord` que cria um sistema de coordenadas virtual para plotagem de gráficos. Por exemplo, no sistema **ANAMOD** para a plotagem dos gráficos dos sinais, foram criados sistemas de coordenadas que mudavam os limites da área de plotagem para valores condizentes com os do sinal a ser plotado. Assim, os limites da área a ser plotada passam, por exemplo, de 0 a 200 na

horizontal e 0 a 100 na vertical para 0 a 10 (segundos) na horizontal e -100 a +100 (μV) na vertical, conforme a figura 4.15. Note que além da mudança de escala, houve uma inversão na escala *y* para que os gráficos fossem plotados conforme o sistema de coordenadas cartesiano ao invés do sistema utilizado no *Windows*.

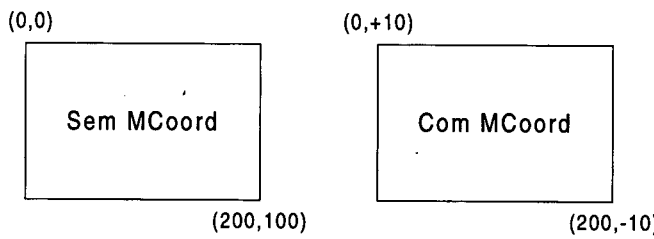

FIGURA 4.15 – Exemplo de utilização da classe MCoord.

O uso da classe `MCoord` é bastante simples. Basta que, ao criar um objeto desta classe ou fazendo uma chamada ao método `Init()` da classe, sejam passados os parâmetros com os limites da área a ser mapeada e os limites desejados. Após isto, basta utilizar os métodos `xx()` e `yy()` para ter o valor convertido para o sistema de coordenadas novo, passando como parâmetros para estes métodos os valores a serem convertidos.

4.4 DIFICULDADES ENCONTRADAS

O **ANAMOD** foi implementado em ambiente *Windows* com a intenção de melhorar a interface com o usuário, mas devido a este sistema (*Windows*) o **ANAMOD** ficou com algumas limitações.

Estas limitações devem-se ao fato do sistema *Windows* ser bastante "pesado", quer dizer, ele consome muito processamento por si só e com isso prejudica *softwares* que necessitem de um processamento elevado como o **ANAMOD**.

Outro problema refere-se ao fato do sistema *Windows* utilizar um sistema de alocação de recursos virtual, ou seja, os programas não têm acesso a periféricos do sistema diretamente. Isto torna o acesso a estes dispositivos mais lento, por outro lado — provavelmente o motivo pelo qual o *Windows* foi feito desta forma — torna transparente o acesso a estes dispositivos. Por exemplo, para acessar a impressora, utiliza-se rotinas da API do *Windows* e envia-se os comandos para a impressora através delas, ao invés de enviá-los diretamente à impressora. O mesmo ocorre com outros periféricos, como as portas paralela e serial. Para acessá-las diretamente torna-se necessária a utilização de *drivers* específicos para este fim. Para este tipo de acesso utilizou-se um *driver* encontrado na *Internet*, desenvolvido por *Victor Ishikeev*. Este *driver* permite o acesso diretamente à memória e acesso aos periféricos. Com isto tornou-se possível o acesso à porta paralela do micro e a utilização de processos de interrupção para realizar o processo de aquisição.

Este processo de alocação de recursos virtual também afeta a forma como as interrupções são atendidas pelo sistema. No caso do sistema **ANAMOD** o processo de aquisição é feito por meio de interrupção (método este que será explicado com mais detalhes adiante) e portanto é afetado por esta mudança neste processo.

No DOS, caso uma interrupção estivesse sendo atendida e ela fosse solicitada novamente, a rotina de atendimento à interrupção seria interrompida e recomeçaria do inicio.

Já no *Windows* isto não ocorre, pois existe uma "fila" de interrupções, ou seja, a cada nova interrupção é gerada uma mensagem no sistema e ela aguarda a sua vez de ser atendida. Isto pode parecer bom a princípio (e talvez em alguns casos seja realmente útil) mas no caso de um sistema de aquisição de dados, isto não é desejável,

uma vez que o período entre amostras ficará variável. É claro que este erro é relativamente pequeno e só seria considerável para taxas de amostragem elevadas, coisas que sob o *Windows* ficam bastante dificultadas pois o sistema leva cerca de 50 μs para atender um pedido de interrupção e, acrescentando-se o tempo da rotina de atendimento da interrupção, fica-se restrito a uma taxa de amostragem baixa.

Entretanto, este processo de atendimento de interrupções gerou um problema que levou um grande tempo para ser descoberto e corrigido. O que acontecia é que eventualmente o programa abortava, acusando um erro interno. Depois de procurar em vários trechos do programa e rastrear o programa inteiro, chegou-se à conclusão que o problema era devido à sobreposição das interrupções. O programa era configurado para adquirir, por exemplo, 100 pontos. Dependendo da taxa de amostragem e a carga do sistema *Windows*, acontecia uma sobreposição de interrupções, ou seja, uma nova interrupção chegava antes de terminada a anterior e entrava na fila. Daí, após serem adquiridos os 100 pontos ainda haviam 3 ou 4 pedidos de interrupção para serem atendidos, o que fazia com que o sistema entrasse novamente na rotina de interrupção e utilizasse mais 3 ou 4 posições de memória que não haviam sido reservadas para os pontos, derrubando o sistema. Isto foi resolvido com testes na rotina de interrupção e, caso ocorresse conflito de interrupções o sistema aborta a aquisição e emite uma mensagem de alerta para o usuário. Com este procedimento evita-se que dados sejam perdidos devido à falta de sincronização do sistema com o *hardware* e possibilita que o usuário salve os dados adquiridos até aquele momento.

Para a rotina de aquisição foi utilizado um processo de interrupção semelhante ao que era utilizado na versão para DOS do sistema (ver Rodrigues, 1997). Semelhante porque na versão DOS do sistema era utilizada a IRQ0 e a taxa de aquisição

era programada através do *chip* 8253 interno do micro. Este *chip* é um contador que gera uma interrupção que pode ser reconhecida e atendida pelo PC. O problema é que o *Windows* 95 utiliza esta mesma interrupção para fazer a "multitarefa" que permite com que mais de um programa seja executado em paralelo. Na verdade não é uma multitarefa real (daí o motivo das aspas), o que acontece é que o processador é compartilhado pelos diversos processos que estão em execução no sistema, entre eles programas iniciados pelo usuário e processos do sistema.

Tentou-se reprogramar esta interrupção para poder-se utilizá-la para a aquisição e no final da aquisição seria devolvido o controle para o *Windows*. Esta tentativa não obteve sucesso. Talvez exista uma maneira de fazer isto, mas devido ao tempo limitado para este trabalho, optou-se por uma solução que se não a melhor, pelo menos serviu para os propósitos deste trabalho.

A solução adotada foi acrescentar ao *hardware* uma 8253 que foi ligada ao PC pela porta paralela, que já estava sendo utilizada para a transferência de dados, e gerar a interrupção através da IRQ7 (porta paralela).

Com isto conseguiu-se fazer a aquisição, mas devido às limitações do *Windows* explicadas anteriormente relativas ao tempo de atendimento da interrupção, só se pode atingir uma taxa de aproximadamente 1000 Hz em um micro Pentium 166 MHz, o que é muito baixo, pois pretendia-se atingir pelo menos uma taxa de 2000 Hz para que, seguindo o critério de Nyquist, fosse possível adquirir sinais com até 1000 Hz. E no caso do sistema **EVOMOD** (que está sendo desenvolvido em paralelo com este trabalho), precisaria-se de taxas de até 8000 Hz. Devido a problemas com o tempo limitado, e tendo obtido um resultado inicial satisfatório, optou-se por deixar a taxa como estava e partir para outras implementações no *software*.

Após várias tentativas, conseguiu-se atingir uma taxa de 4000 Hz com o sistema **ANAMOD**. Isto foi conseguido ao diminuir-se o número de cores utilizado pelo sistema *Windows* de 256 para 16. A explicação para este fato é que as rotinas de atualização de vídeo estão entre as mais demoradas dentro do sistema *Windows*. Ao mudar-se o número de cores de 256 para 16, o sistema passou a necessitar de 4 *bits* para representar cada ponto na tela em oposição aos 8 *bits* utilizados anteriormente. Isto reduziu de 1 para ½ *byte* para armazenar o ponto na tela, reduzindo pela metade a memória de vídeo necessária para armazenar a tela e diminuindo consideravelmente o tempo de gerenciamento do vídeo.

Uma solução para melhorar o desempenho do sistema (que fica como sugestão para trabalhos futuros) seria a inclusão de *buffers* externos (no *hardware* do sistema) e fazer a transferência dos dados em blocos maiores. Por exemplo, poderiam ser colocados dois *buffers* de 8 kbytes no módulo de *hardware* e quando um destes *buffers* estivesse cheio o sistema começaria a colocar os dados no outro e geraria uma interrupção no PC que iria ler de uma só vez os 8 kbytes de dados. A vantagem desta abordagem estaria no fato de que o *Windows* seria interrompido de maneira mais espaçada, ou seja, haveria um tempo maior entre cada interrupção, deixando com isso mais tempo para o sistema realizar outras operações, uma vez que para a leitura de 8 kbytes de uma só vez não sobrecarregaria a rotina de interrupção.

Na verdade, esta implementação traria problemas com a rotina de plotagem *on-line*. O *Windows* devido à sua estrutura interna não permite que se use diretamente os recursos do sistema, como já foi explicado anteriormente. Esta restrição inclui também a placa de vídeo do micro. Ou seja, para plotar-se um ponto na tela, devemos informar ao *Windows* a localização do ponto (coordenadas), cor, etc., e deixar

que ele plote o ponto na tela. Isto, é claro, demanda tempo. Tempo perdido! A rotina de plotagem atual plota os pontos na medida do possível, conforme foi descrito, mas para a nova rotina, que receberá os dados em blocos de 8 kbytes, será necessário desenvolver uma rotina que possa fazer a plotagem dos dados de forma que estes fiquem igualmente espaçados na tela sem que a rotina perca o sincronismo com a rotina de aquisição.

5. Discussões e Conclusões

Os objetivos propostos para este trabalho foram mudando com o decorrer do mesmo. Inicialmente desejava-se ampliar o sistema **SAASBIO II** acrescentando rotinas de processamento de sinais e possibilitar a integração deste sistema aos sistemas desenvolvidos nos trabalhos **EVOMOD** (Silva Jr., 1998), **IMPMOD** (Toazza, 1998a) e **TELEMOD** (Bertemes, 1998).

Entretanto, durante o desenvolvimento do trabalho constatou-se algumas necessidades do sistema, como uma interface mais intuitiva e mais padronizada. Devido à grande popularidade do sistema *Windows* optou-se por desenvolver o *software* neste ambiente. Aliado a isto, houve a necessidade de alteração do *hardware* do sistema de modo a permitir as expansões que seriam feitas pelos projetos **EVOMOD**, **IMPMOD** e **TELEMOD**. A partir deste ponto, optou-se pela produção de um novo sistema, o **SAASBIO III**.

Para este novo sistema, tornou-se necessária a modificação do *hardware*, acrescentando novas funcionalidades a este e, conseqüentemente, uma mudança no

software do sistema que, além de implementar a nova interface (*Windows*), também deveria se ajustar ao novo *hardware*.

No desenvolvimento deste novo *software*, houveram muitos obstáculos — a maioria deles por conta do sistema *Windows* que utiliza os recursos do computador de forma muito diferente do que se está acostumado em ambiente DOS. Apesar disso, conseguiu-se implementar as rotinas que compõe a interface do sistema e a rotina de aquisição, que seriam as rotinas mais fundamentais para o prosseguimento do trabalho. A partir deste ponto é possível acrescentar novos módulos ao *software* para processamento de sinais de forma bastante simples.

Com relação ao desempenho do *software*, foi atingido um resultado satisfatório, sendo que a taxa máxima de aquisição ainda pode ser melhorada. Conseguiu-se atingir, inicialmente, 1000 Hz, o que é satisfatório para aquisições de sinais com freqüências de até 500 Hz. Muito disso deve-se ao fato do sistema *Windows* 95 ser um sistema muito "pesado", ou seja, ele sozinho consome muitos recursos do computador, sobrecarregando-o. Desta forma fica bastante dificultado o desenvolvimento de *softwares* que necessitem de muito processamento e velocidade do computador, como o caso do sistema **ANAMOD**.

Uma prova deste "peso" do sistema *Windows* é o fato de que ao mudar-se o número de cores que o sistema utiliza de 256 cores que era o utilizado originalmente para 16 cores, conseguiu-se que o sistema atingisse taxas de até 4000 Hz, conforme explicado no capítulo relativo ao sistema **ANAMOD**.

A interface do *software* ficou de acordo com as padronizações adotadas pelos programas desenvolvidos para o ambiente *Windows*, tornando o programa de mais fácil utilização pelo usuário.

Um dos problemas relacionados à implementação do sistema que era relacionado à dificuldade de acesso aos recursos do sistema foi resolvido com a utilização de um *driver* específico para este fim.

Outro problema era relativo à velocidade para plotar os dados *on-line*, de forma a aproveitar ao máximo os recursos do sistema, sem deixar o mesmo ocioso nem sobrecarregado foi solucionado através da implementação do código para plotagem do sinal em uma rotina de "espera" do sistema (`OnIdle`) como foi explicado no capítulo anterior.

5.1 TRABALHOS FUTUROS

Evidentemente o sistema **ANAMOD** não está completo e o objetivo deste trabalho era exatamente criar um *software* de maneira que fosse possível acrescentar-se novas funcionalidades ao sistema de forma a torná-lo cada vez mais robusto e versátil. Devido à forma de construção do *software* estas expansões ficam facilitadas.

Para que o sistema se torne funcional será necessário acrescentar algumas rotinas de filtragem e processamento dos sinais adquiridos.

Também será necessário a integração do sistema aos sistemas **EVOMOD**, **IMPMOD** e **TELEMOD**. Isto não foi feito até a conclusão deste trabalho porque os outros sistemas estavam sendo desenvolvidos em paralelo. Assim, com o término de todos os trabalhos, será possível a integração de todos os sistemas e a implementação dos procedimentos para o processamento digital de sinais bioelétricos criando-se assim o sistema **SAASBIO III**.

Com o uso do sistema, novas necessidades serão detectadas. Modificações deverão então ser introduzidas no sistema. Uma das primeiras modificações que poderá ser feita refere-se à otimização do sistema de plotagem do sinal *on-line* e aumento da taxa de amostragem máxima do sistema. Isto poderá ser conseguido, através das modificações sugeridas no capítulo anterior, na seção que se refere aos problemas encontrados na implementação do sistema, como o uso de *buffers* no *hardware* externo de forma a diminuir o número de interrupções geradas no *Windows* e, desta forma, "desafogar" o sistema, e também a utilização de outros *drivers* para acesso direto à placa de vídeo do computador "passando por cima" do *Windows* e assim acelerando a plotagem *on-line*.

Outra forma de melhoria no sistema poderia ser um estudo a respeito de rotinas de DMA (*Direct Memory Access* — Acesso Direto à Memória), disponíveis nas interfaces paralelas ECP (Enhanced Communications Port — Porta de Comunicações Melhorada) que vem nos microcomputadores Pentium e em alguns 486. Com isto seria possível mandar os dados diretamente para a memória, deixando o "pesado" sistema *Windows* aguardando a aquisição. Isto, evidentemente, terá um desempenho ainda maior juntamente com a inclusão dos *buffers* externos.

Espera-se que este trabalho possa ser continuado, pois o sistema ainda necessita de recursos que o tornem mais versátil e funcional. A idéia é que o sistema **ANAMOD,** depois de integrado com os sistemas **EVOMOD**, **IMPMOD** e **TELEMOD**, gerencie todos estes sistemas, sendo o módulo principal de controle do sistema **SAASBIO III**.

Referências Bibliográficas

ALMEIDA, M. A. F. **Filtragem Digital de Sinais Biomédicos**. Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, fevereiro de 1997.

ANDERSON, P. H. **Use of a PC Printer Port for Control and Data Acquisition**. [online] Disponível na internet via WWW. URL: http://et.nmsu.edu/~etti /fall96/computer/printer/printer.html. Arquivo capturado em 01 de abril de 1998.

ANDRÉ, E. S.; BRUNO-NETO, R.; PASCHOALINI, M.A; FREITAS, C.G.; MARINO-NETO, J. e HACKL, L.P.N. **Efeitos da escopolamina, mecamilamina e eserina sobre o eletroscilograma hipocampal de pombos**. In: Reunião Anual da Federação das Sociedades de Biologia Experimental, XII , 1997, Caxambu. *Anais...* São Paulo: FESBE, 74-75.

ARY, J. P., KLEIN, S. A., FENDER, D. K. **Location of Sources of Evoked Potentials**. In *Transactions on Biomedical Engineering* no. 28. IEEE, 1981, pp. 447–452.

BERNE, R. M., LEVY, M. N. **Fisiologia**. Guanabara Koogan S. A., Rio de Janeiro, 1990.

BERTEMES FILHO, P. **Uma Proposta de um Sistema Telamétrico para Registro de Potenciais Bioelétricos**. Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, 1998.

BERTEMES FILHO, P., AZEVEDO, F. M. AND MARINO-NETO, J. **Proposal of A Telemetry System For Bioelectrical Signals Monitoring**. In: XV Congreso Anualde la Sociedad Espanola de Ingenieria Biomedica, CASEIB '97, 1997, Valencia, Espanha. Anais...., 214 - 218.

BITTENCOURT, P. C. T. et al. **Disritmia Cerebral em um Hospital Universitário**. *J. Bras. Psiq.*, v. 42, n. 10, p 541–545, 1993.

BRUNO, R.; ANDRÉ, E. S.; FREITAS, C. G.; PASCHOALINI, M. A. & MARINO-NETO, J. (1996) **Análise qualitativa e quantitativa do eletrooscilograma hipocampal em diferentes estados comportamentais no pombo**. In: Reunião Anual da Federação das Sociedades de Biologia Experimental, XI , 1996, Caxambu. *Anais...* São Paulo: FESBE, 348-349.

BURR-BROWN IC DATA BOOK. **Linear Products**, USA : 1995.

BURR-BROWN. **Aplications Handbook**. USA : 1994.

BURR-BROWN. IC DATA BOOK. **Data Conversion Products**, USA : 1995.

BURR-BROWN.**The Handbook of Linear IC applications**. USA : 1987.

CABRAL, F. **Linguagem C e o PC-BIOS**, Editora Campus, São Paulo, 1989.

COHEN, A. **Biomedical Signal Processing**. In *Time and Frequency Domain Analysis*. CRC Press, Boca Raton, 1983.

COIMBRA, A. J. F. **Análise Computadorizada de Sinais Bioelétricos**. Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, 1994a.

COIMBRA, A. J. F.; D'ANGELO, G. G.; DE AZEVEDO, F.M.; MARINO-NETO, J. & BARRETO, J.M. (1994b) **Neural Networks in brain electrographic state analysis**. In: Reunião Anual da Federação das Sociedades de Biologia Experimental, IX, 1994a, Caxambu. *Anais...* São Paulo: FESBE, p.37.

COIMBRA, A. J. F.; D'ANGELO, G.G.; DE AZEVEDO, F.M.; MARINO-NETO, J. & BARRETO, J.M. (1994c). **Electrographic analysis of brain states using neural networks**. Physics in Medicine and Biology, vol. 39 a, p. 463, Bristol (UK): IO Publishing Ltd.

COIMBRA, A. J. F.; MARINO-NETO, J. ; LIMA, W.C. & FREITAS, C.G. (1994d). **Computer based system for electrographic analysis**. In: Reunião Anual da Federação das Sociedades de Biologia Experimental, IX, 1994, Caxambu. *Anais...* São Paulo: FESBE, p.37.

COIMBRA, A.J.F.; MARINO-NETO, J.; DE AZEVEDO, F.M. e BARRETO, J.M. (1995) **Brain electrographic state detection using combined unsupervised and supervised neural networks**. In: D.W. Pearson, N.C.Steele & R.F. Albrecht (eds.) "Artificial Neural Nets and Genetic Algorithms". Wien: Springer-Verlag, pp 76-79, 1995.

COOPER, R., OSSELTON, J. W., SHAW, J. C. **EEG Tecnology**. 2.ed. Ed. Butterworth, 1974.

DARIO, A. J. S.; LOPES, P. R. C.; FREITAS, C. G.; PASCHOALINI, M. A. e MARINO-NETO, J. (1996) . **Electrographic patterns of postprandial sleep after food deprivation or intraventricular adrenaline injections in pigeons.** Brain Res. Bull . 39(4): 249-254.

DARIO, A. J. S; SCUSSIATTO, E. A.; RIBEIRO, S. K.; MONTANHA, E. V. S.; FREITAS, C.G. & MARINO-NETO, J. (1993). **Modulação de eletrocorticograma no pombo. I. Efeitos do bloquieo central de receptores muscarínicos e nicotínicos.** In: Reunião Anual da Federação das Sociedades de Biologia Experimental, VIII, 1993, Caxambu. *Anais...* São Paulo: FESBE, p.35.

DOTSINSKY, J. A.; DASKALOV, J. K. Accuraty of 50 Hz interference subtraction from an eletrocardiogram. **Med. & Biol. Eng. & computing**, 34 : 489-494, 1996.

FRANKLIN 8051, **Standard Utilities User's Guide**, USA, 1993.

FRANKLIN A51, **8051 Macro Assembler**, USA, 1993. V. 4.6.

FRANKLIN C51, **8051 Optimizing C Cross Compiler**, USA, 1993, V.3.4.

GARRET, P. **Analog I/O Design Acquisition : Conversion : Recovery**. Virginia : Reston Publishing Company Inc., 1981.

GRÖZINGER, M., RÖSCHKE, J., KLÖPPEL, B. **Automatic Recognition of Rapid Eye Movement (REM) Sleep by Artificial Neural Networks**. J. Sleep Res. — European Sleep Research Society, v. 4, n. 1, p. 86–91, Fev., 1995.

GUYTON, A. C. **Tratado de Fisiologia Médica**. Guanabara Koogan, Rio de Janeiro, 1989.

HOLZNER, S. **Borland C++ Programaçao for Windows**. 3rd ed. São Paulo: Makron Books, 1994.

HOLZNER, S. **Linguagem Assembly Avançada para o IBM PC.** McGraw-Hill, 1988.

HURWICZ, L. **Basic Mathematical and Statistical Considerations in the Study of Rhythms and Near-rhythms**. Ann. N. Y. Acad. Sci., 1962.

IVES, J. R., MAINWARNING, N. R., SCHOMER, D. L. O. **Monitorização eletroencefalográfica**. In: C. A. Guerreiro e M. M. Guerreiro. *Epilepsia*, p. 39–46. São Paulo, 1993.

JAEGER, R. Tutorial: Analog Data Acquisition Techology: part II - Analog-to-Digital ,Conversation. **IEEE MICRO**, p. 46-56, Aug. 1982.

KRUGLINSKI, D. J. **Explorando Visual C++**. Rio de Janeiro: Campus, 1994.

LENNOX, W. G., DAVIS, J. P. **Clinical Correlates of the Fast and the Slow Spike-wave Eletroencephalogram**. Trans. Am. Neurol. Ass., v. 74, n.1, p. 194–197, 1949.

LIEBMAN P. A. **The Molecular Mechanism of Visual Excitation and Its Relaction to the Structure and Composition of Rod Outer Segment**. *Annu. Rev. Phisiol.*, 49 (1987).

LOPES, P. R. C.; DARIO, A. J. S. e MARINO-NETO, J. (1992). **Sono pós-prandial após injeção intracerebroventricular de adrenalina no pombo. Uma avaliação eletroencefalográfica e sua comparação com o padrão observado após jejuns prolongados**. In: Seminário Catarinense de Iniciação Científica, II, 1992, Florianópolis. *Anais*... Florianópolis: UFSC, p.186.

MARQUES, J. L. B. **Eletrocardiografia de Alta Resolução** : Metodologia e Aplicação clínica. Florianópolis, 1996. Trabalho Submetido como parte dos Requisitos ao Concurso Público para Professor Adjunto no Departamento de Engenharia Elétrica, Área de Engenharia Biomédica - Centro Tecnológico, Universidade Federal de Santa Catarina.

MARSHALL A. **Signal Processing — Medical Applications**. *In Research Evaluation Report of 1993* (Aalborg University, 1993), Departament of Medical Informatics and Image Analysis.

MOFFET, S. **Fisiologia Humana**. Guanabara Koogan S. A., Rio de Janeiro, 1993.

MONTENEGRO, F; PACHECO, Roberto. **Orientação a Objetos em C++**. Rio de Janeiro: Editora Ciência Moderna Ltda., 1994.

NIEDERMEYER, E., SILVA, F. L. **Electroencephalography — Basic Principles, Clinical Applications and Related Fields**. Williams and Wilkins, Maryland, EUA, 3a. edição, 1993.

NILSON, J; PANIZZA, M; HALLET, M. Principles of digital sampling of a phisiologic sinal **Electroencephalography and Clinical Neurophysiology**, 89: 349 - 358, 1993.

PASSOLD, F., OLIVEIRA, J. P., DIAS, J. S., OJEDA, R. G., LIMA, W. C. **Proposta e Avaliação de Procedimentos Anestésicos para Pacientes Críticos/Problemáticos Através de Sistema Especialista Híbrido**. In: X Congresso Chileno de Ingenieria Electrica, p. I–43/48, Valdivia, Chile, Nov., 1993.

RIBEIRO, S. K.; CAMPOS, R. O. P.;.; FREITAS, C. G.; DARIO, A. J. S. ; PASCHOALINI, M. A. & MARINO-NETO, J. (1995) **Modulação colinérgica do ECoG em pombos. Análises espectral e aperiódica dos efeitos da administração central de mecamilamina e de escopolamina no alerta provocado por estimulação sonora**. In: Seminário Catarinense de Iniciação Científica, V, 1995, Florianópolis, SC. Anais.... Florianópolis: UFSC, p. 380-381.

RODRIGUES, M. A. B. **Desenvolvimento de Um Sistema Virtual para Aquisição e Análise de Sinais Bioelétricos**. Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, fevereiro de 1997.

RODRIGUES, M. A. B.; Azevedo, F. M. de; Marino Neto, José. **Instrumento Virtual para Aquisição de Sinais Bioelétricos**. In: III Fórum Nacional de Ciência e Tecnologia em Saúde. Campos do Jordão - SP, Anais p. 257, 1996. v.1.

RODRIGUES, M. A. B.; Azevedo, F. M. de; Marino Neto, José. **Sistema Computadorizado para Análise de Sinais Eletrográficos**. In: III Simpósio Latino Americano de Ingeniería Biomédica. Bucaramanga, Colômbia, Oct, 1996. v.1.

SILVA JR, S. M. **Sistema Microcontrolado de Estimulação e Análise de Potenciais Evocados para Utilização com Eletroencefalografia Computadorizada**. Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, 1998.

TOAZZA, A. **Sistema Microcontrolado para Medição de Impedância Pele-eletrodo em Registradores Bioelétricos.** Dissertação de Mestrado em Engenharia Elétrica, área de concentração Engenharia Biomédica - Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, 1998a.

TOAZZA, A., AZEVEDO, FM AND MARINO-NETO, J **Microcontrolled System for Measuring Skin/Electrode Impedance in Bioelectrical Recordings**. In: Second IEEE INTERNATIONAL CARACAS CONFERENCE ON DEVICES, CIRCUITS AND SYSTEMS, 1998b, Caracas, Venezuela, Anais...., 96 - 100.

WALTER, R. D., YEAGER, C. L. **Visual Imagery and Electroencephalographic Changes**. *Eletroenceph. Clin. Neurophysiol.*, 1956.

WEBSTER, J. G. **Medical Instrumentation Application e Design**. 2 ed., Boston : Houghton Mifflin, 1992.

YAO, P. **Borland C++ 4.0 : Programação for Windows - Guia Oficial Borland**. São Paulo: Makron Books, 1995.

Glossário

alisamento	Erro causado pela etapa conversora. Ocorre quando o número de amostras é menor que duas vezes a freqüência do sinal que desejamos amostrar.
API	*Application Program Interface* — Interface de Programação de Aplicativo. É um conjunto de funções e tipos de dados para obtenção de serviços junto ao sistema *Windows*. Serve como interface para o programa e o sistema operacional.
chip	circuito eletrônico encapsulado.
DMA	*Direct Memory Access* — Acesso Direto à Memória.
DOS	*Disk Operating System* — Sistema Operacional de Disco.
driver	Programa que faz a interface entre o *Windows* e os periféricos do computador.
E/S	Entrada e saída.
ECP	*Enhanced Communications Port* — Porta de Comunicações Melhorada. Substitui, nos computadores Pentium, as antigas portas paralelas unidirecionais existente nos modelos anteriores (486, 386, 282 e XT).
EPROM	*Electrically Programmable Read Only Memory* —Memória de leitura programável eletricamente.
Filtro ***NOTCH***	Tipo de filtro que não permite a passagem do sinal em uma determinada freqüência.
firmware	*Software* armazenado em um chip; por exemplo, um programa armazenado em ROM.
GDI	Graphical Device Independent — Gráficos Independentes do Dispositivo. Responsável pelo desenho dos gráficos, na tela, impressora, ou outro dispositivo.
hardware	toda a parte física do sistema, incluindo componentes eletrônicos.
I/O	*Input/Output* — Entrada e saída.
IRQ	*Interrupt Request* — Requisição de Interrupção. Utilizada pelo sistema para interromper o processador para acessar algum periférico.
kbytes	kilo bytes.
microcontrolador	microprocessador com alguns recursos adicionais como contadores, conversores A/D, comunicação serial, entre outros.
OS	*Operational System* — Sistema Operacional.
PC	*Personal Computer* — Computador pessoal. Originalmente desenvolvido pela **IBM**.
pixel	Cada um dos pontos de uma tela gráfica.
RAD	*Rapid Application Development* — Desenvolvimento Rápido de Aplicativo
ROM	*Read Only Memory* — Memória somente de leitura.
software	Programa de computador.
Windows	Sistema operacional gráfico, baseado em janelas (daí o nome) desenvolvido pela Microsoft.