UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO E VALIDAÇÃO DE UM HARDWARE EMULADOR DE SINAIS BIOLÓGICOS

Hudson Capanema Zaidan

Janeiro

- 2014 –

IMPLEMENTAÇÃO E VALIDAÇÃO DE UM HARDWARE EMULADOR DE SINAIS BIOLÓGICOS

Hudson Capanema Zaidan

Texto da dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciências.

Prof. Dr. Adriano O. Andrade, PhD	Prof. Dr. Edgard Afonso Lamounier Junior, PhD
Orientador	Coordenador do Curso de Pós Graduação

Janeiro

- 2014 –

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO E VALIDAÇÃO DE UM HARDWARE EMULADOR DE SINAIS BIOLÓGICOS

Hudson Capanema Zaidan

Texto da dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciências.

Banca Examinadora:

Prof. Dr. Adriano de Oliveira Andrade – Orientador (UFU)
Prof. Dr. Adriano Alves Pereira – (UFU)
Prof. Dr. César Ferreira Amorim – (UNICID)

RESUMO

Vários estudos na área da saúde exigem a aquisição e análise de sinais biológicos. A etapa de aquisição destes sinais depende de protocolos, equipamentos e técnicas específicas, e é parte obrigatória do cronograma de diversas pesquisas. Por este motivo, é interessante que se consiga otimizá-la. O principal objetivo do presente trabalho foi o desenvolvimeto de um hardware emulador de sinais biológicos para o ensino e aprendizagem na área de engenharia biomédica, e auxiliar as novas tecnologias provenientes da utilização de sinais biológicos. Neste contexto, o desenvolvimento de equipamentos capazes de emular e simular sinais biológicos se torna relevante, pois, com eles, pode-se subtrair a etapa de aquisição, economizando tempo e auxiliando os estudos voltados para o seu processamento. Estes dispositivos disponibilizam sinais biológicos em sua forma analógica, que previamente estavam armazenados digitalmente em uma base de dados, emulando-os, ou, simplesmente simulando estes sinais quando os mesmos são gerados por programas computacionais. Uma aplicação importante para os emuladores e simuladores desta natureza, é avaliar a exatidão, precisão e desempenho dos sistemas que processam sinais em tempo real. Assim, este estudo contempla uma revisão da literatura que descreve a importância do estudo envolvendo os sinais biológicos e propõe uma arquitetura e organização de um emulador/simulador denominado BioSim, a fim de melhorar as técnicas para o estudo destes sinais, bem como avaliar e calibrar equipamentos que os utilizam. O BioSim corresponde a um sistema de quatro canais, que consiste de módulos de processamento, MAPLE (ARM Cortex M3) e Arduino MEGA (ATMEGA 2560), com uma capacidade de armazenamento de 4 *Giga Bytes* por canal, através de um cartão micro SD (*SanDisk*), e uma interface homem-máquina, que compreende uma tela de cristal líquido e botões para navegação. Para avaliar o equipamento desenvolvido foram considerados sinais simulados por um programa computacional além da emulação de um banco de dados de diversos sinais biológicos coletados em estudos anteriores pelo nosso grupo de pesquisa do Laboratório de Engenharia Biomédica da Universidade Federal de Uberlândia – BioLab. Os resultados sugerem que o dispositivo é capaz de simular e emular sinais biológicos em tempo real, amostrados até 20 kHz, com uma resolução de 12 bits. Isto permitirá a otimização do processo de aquisição dos sinais biológicos, reduzindo o tempo para a realização de pesquisas na área da Engenharia Biomédica.

Palavras-chave: Emulador, Sinais Biológicos, Instrumentação Biomédica.

ABSTRACT

Several studies in healthcare require the acquisition and analysis of biological signals. The stage of acquisition of these signals depends on protocols, equipment and specific techniques, and is a compulsory part of the schedule of several studies. For this reason, it is interesting to be able to optimize it. The main objective of this work is the development in an emulator hardware of biological signals for teaching and learning in area of biomedical engineering and aid to new technologies arising from the use of biological signals. In this context, the development of equipment capable of emulating and simulating biological signals becomes relevant because, with them, you can subtract the acquisition step, saving time and at aiding aimed studies its processing. These devices provide a biological signal in analog form that was previously stored digitally in a database emulating them, or simply simulating these signals when they are generated by computer programs. An important application for the emulators and simulators of this kind, is to evaluate the accuracy, precision and performance of the systems processing signals in real time. Thus, this study includes a literature review that describes the importance of the study of biological signals and proposes an architecture and organization of an emulator / simulator called BioSim in order to improve techniques for the study of these signals as well as to evaluate and calibrate equipment that use these signals. The BioSim corresponds to a four-channel system consisting of processing modules, MAPLE (ARM Cortex M3) and Arduino MEGA (ATMEGA 2560) , with a storage capacity of 4 Giga Bytes per channel , through a micro SD card (SanDisk), and a man-machine interface comprising a liquid crystal display and buttons for navigation. To evaluate the developed equipment were considered signs simulated by a computer program, in emulation of database a several biological signals collected in previous studies by our research group of the Biomedical Engineering Laboratory of the Federal University of Uberlândia - BioLab. The results suggest that the device is able to simulate and emulate biological signals in real time, sampled to 20 kHz, with a resolution of 12 bits. This will allow the optimization of the acquisition process of the biological signals, reducing the time for carrying out research in area of the Biomedical Engineering.

Keywords : Emulator, Biological Signals, Biomedical Instrumentation.

GLOSSÁRIO

Emulador: Dispositivo ou programa que reproduz um sistema, valendo-se de um outro sistema distinto, ou seja, um *software* ou *hardware* que permite que uma máquina se comporte igual a outra.

Simulador: Dispositivo que imita um processo ou operação do mundo real. Ou seja, é o estudo do comportamento de sistemas reais através do exercício de modelos.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

ANEXOS

Capítulo 1

Introdução

Muitos órgãos no corpo humano, como o coração, cérebro, músculos e olhos manifestam suas funções através de atividade elétrica (GEDDES e BAKER, 1989).O coração, produz um sinal chamado de eletrocardigrama (ECG). O cérebro, produz um sinal chamado eletroencefalograma (EEG). A atividade muscular produz o eletromiograma (EMG), e assim por diante.

As mensurações destes e de outros sinais biológicos podem mostrar informações vitais relativas à funções normais e patológicas dos órgãos. Por exemplo, batidas cardíacas anormais ou arritmias podem ser rapidamente diagnosticadas por um ECG. Neurologistas interpretam sinais EEG para identificar eventos de epilepsia. Sinais EMG podem auxiliar na avaliação da função muscular (THAKOR, 1999).

Os biopotenciais são originados da atividade elétrica no nível celular. O potencial de ação elétrico, através da membrana celular, é o resultado de diferentes concentrações iônicas existentes dentro e fora da célula. (THAKOR, 1999).

O gradiente de concentração eletroquímico através da membrana semipermeável tem como resultado o potencial de Nerst. A membrana celular separa altas concentrações de íons de potássio e baixas concentrações de íons de sódio dentro da célula, e exatamente o oposto do fora da célula. (THAKOR, 1999).

Essa diferença de concentrações iônicas através da membrana celular produz o potencial de repouso. Algumas das células do corpo são excitáveis e produzem o que é chamado de potencial de ação, o que é resultado de um rápido fluxo de íons através da membrana celular em resposta a uma estimulação elétrica. A excitação das células gera correntes que se manifestam como potenciais no corpo (THAKOR, 1999).

Com relação à eletromiografia, as fibras musculares geram atividade sempre que os músculos estiverem em contração. A EMG possui um problema constante, que é o registro de artefatos alheios ao sinal, que podem ser filtrados para reduzir artefatos e interferências, isso

pode ser feito com a padronização da largura da banda para acima de 20 Hz (THAKOR, 1999).

Em várias linhas de pesquisa da Engenharia Biomédica e na Medicina, os sinais biológicos são utilizados para se determinar padrões normais e patológicos, ou ainda para modelar a fisiologia de sistemas biológicos (AKAY, 1994) e (CHALLIS; KITNEY, 1990).

As técnicas que utilizam processamento digital de sinais vêm ganhando espaço em diversas aplicações, como: processamento de voz, áudio, imagem e vídeo, comunicações, automação e controle de processos, robótica, visão computacional, sismologia, meteorologia, finanças, economia, sistemas embarcados, instrumentação, reconhecimento e identificação de padrões, sistemas especialistas, navegação aeroespacial, guerra eletrônica, agricultura de precisão, biomecânica, química, medicina, biologia e todas as áreas em que é possível utilizar um computador para processar informações.

No âmbito da Engenharia Biomédica também é possível observar o grande avanço tecnológico resultante dos equipamentos digitais. Exemplos disso são os modernos eletrocardiógrafos digitais, os equipamentos de tomografia computadorizada e os scanners de ressonância magnética nuclear (ROCHA; CARVALHO; NASCIMENTO, 2008).

Tendo em vista a grande importância da utilização dos sinais biológicos e a relativa complexidade em captá-los, deve-se focar em métodos que possam otimizar tempo e qualidade desses sinais. O uso de um simulador de sinais biológicos possui muitas vantagens, como a economia de tempo e a ausência das dificuldades e dos riscos em fixar e remover eletrodos superficiais como também inserir e remover eletrodos invasivos (KARTHIK, 2006).

Karthik (2006) reporta em seu estudo sobre um emulador de sinais ECG que o simulador pode possibilitar o estudo de sinais normais e patológicos, sem a utilização de todo o aparato e até mesmo dos voluntários. Isso otimiza tempo, dinheiro e processos burocráticos, como avaliações por comitês de ética, que podem inviabilizar a pesquisa pelo tempo de espera.

Outro problema enfrentado na aquisição de sinais biológicos é o tamanho da amostra. Sinais EMG são geralmente digitalizados com amostras que variam entre 1 e 20 kHz e com uma precisão entre 12 e 16 bits por amostra. Assim o armazenamento e, principalmente, a transmissão desses dados são dificultados de acordo com Merletti e Parker (2004).

Com relação a essa questão, conforme os autores Berger, Nascimento, Rocha e Carvalho (2007) que afirmavam no estudo "*A New Wavelet-Based Algorithm for*

Compression of Emg Signals" existem métodos de compressão do sinal EMG, que podem auxiliar em seu armazenamento e transmissão.

Apesar de apresentar muitos aspectos vantajosos, poucos são os trabalhos encontrados na literatura que abordam algum tema relativo à simuladores de sinais. O trabalho de Farias (2010), apresenta o desenvolvimento de um sistema embarcado que simula sinais ECG através da implementação das Séries de Fourier que permite a análise e o estudo de sinais ECG normais e anormais.

O sistema deste autor é embarcado em um chip programável (SOPC – *System on a Programmable Chip*) e foi prototipado com uma placa contendo periféricos e uma pastilha *Field-Programmable Gate Array* (FPGA); a arquitetura do *soft-core* foi configurada em sistema operacional compacto e módulos de *software* foram desenvolvidos, portados e validados sobre essa plataforma.

Outra aplicação de dispositivos emuladores está no auxílio ao desenvolvimento do Processador de Coleta de Dados (PROCOD III) utilizado no Sistema Brasileiro de Coleta de Dados (SBCD) de acordo com Cisotto e Ferreira (2011).

Com o crescimento do SBCD, foi necessário o desenvolvimento de uma nova versão de seu processador de dados. Este demandou um equipamento de testes que substituísse com vantagens as ferramentas comerciais tradicionais gerando sinais que representassem com precisão o tráfego de sinais do SBCD (CISOTTO; FERREIRA, 2011).

Os autores também afirmam que a solução encontrada para o desenvolvimento deste equipamento de testes foi um simulador e emulador de sinais, denominado UTV III (Unidade de Teste e Validação), que trabalha conectado diretamente a uma entrada do processador, podendo ser utilizado facilmente durante as fases de desenvolvimento operacional. A utilização deste emulador representou um ganho de tempo e de qualidade no desenvolvimento do processador.

De acordo com Cisotto e Ferreira (2011) o Simulador/Emulador UTV III, possui uma arquitetura desenvolvida para operar em um computador do tipo PC, utilizando o sistema operacional Windows XP ou equivalente. Ele é composto de módulos de software desenvolvidos em C++, e módulos em hardware implementados em componentes do tipo FPGA, conectados ao PC via interface PCI (Padrão de *Slots*).

De uma forma simples e didática, os autores dividiram o equipamento em quatro subsistemas com as seguintes funções: simulação, emulação, comunicação com o usuário e processamento de testes.

Eles ainda alegam que a utilização deste Simulador/Emulador facilitou muito a aplicação dos testes em todas as fases de desenvolvimento, mas um dos principais benefícios do sistema foram os ganhos de qualidade no processo.

Um outro sistema de simulação de sinais foi desenvolvido pela empresa Fluke Biomedical, que possui uma linha de simuladores denominados ProSim. Tais simuladores são capazes de simular sinais de ritmos sinusais normais, marcapasso, arritmia, teste de desempenho de ECG, ECG fetal e materno, pressão sanguínea, respiração, temperatura e débito cardíaco. Estes equipamentos são utilizados na manutenção preventiva, promovendo a aferição de equipamentos que monitoram os pacientes.

A partir de todas as informações supracitadas, é possível concluir que um emulador de sinais biológicos é uma ferramenta importante para profissionais da saúde e pesquisadores, devido à sua utilidade em diversas frentes, tais como: no ensino e aprendizagem sobre sinais biológicos normais e patológicos e nas pesquisas sobre estes sinais, principalmente na etapa de coleta e aquisição.

A utilização do aparelho emulando bancos de dados préexistentes eliminará problemas com o recrutamento de voluntários repetidamente, preparação do protocolo e dos experimentos, possíveis necessidades de refazer o procedimento devido a ruídos e outros artefatos que possam gerar problemas para a análise dos dados e toda a burocracia necessária para que o procedimento seja aprovado por comitês de ética. Outra função é aferir equipamentos e promover testes de novas tecnologias da área de Engenharia Biomédica.

Este estudo também se justifica pois um equipamento capaz de simular sinais biológicos pode ser uma ferramenta que auxilia na redução de custos e de tempo, tendo em vista que os sinais a serem analizados não precisam ser coletados *in loco*.

Para dados utilizados em pesquisas científicas, os emuladores evitam processos demorados e burocráticos envolvendo comitês de éticas e recrutamento de pacientes, otimizando, inclusive, o tempo que seria gasto com preparação dos eletrodos e toda a coleta de dados.

O processo de aprendizagem de análise de sinais biológicos também é mais dinâmico com a utilização de um emulador de sinais, pois os alunos podem, a qualquer momento, obter os dados desejados para a realização de seus estudos.

Todos os benefícios listados acima, esbarram na barreira do alto custo de um emulador de sinais. A proposta do presente trabalho é justamente o desenvolvimento de um emulador de sinais de baixo custo, mais acessível para instituições de ensino e pesquisa, não excluindo

clínicas e hospitais que necessitam deste tipo de instrumento para o desenvolvimento de suas atividades.

1.1 Objetivos

O objetivo geral do presente trabalho é a concepção de um hardware emulador de sinais biológicos que utiliza um banco de dados de sinais já coletados, para o ensino e aprendizagem na área biomédica, bem como a substituição da etapa da coleta destes sinais nas pesquisas e no auxílio às novas tecnologias provenientes da utilização de sinais biológicos.

Este objetivo geral se desmembra em quatro objetivos específicos:

- Especificar um sistema para conversão de arquivos contendo sinais digitais em sinais analógicos;
- Projetar e desenvolver um sistema computacional, capaz de converter arquivos gerados por equipamentos de coleta de sinais biológicos em arquivos apropriados para emulação em um equipamento específico.
- Projetar e desenvolver um equipamento específico, intitulado BioSim, capaz de gerar sinais analógicos provenientes de arquivos contendo sinais biológicos digitas.
- Realizar simulações para validar o *software* e o *hardware* desenvolvidos.

1.2 Contribuições desta dissertação

A principal contribuição deste trabalho é o desenvolvimento de um protótipo funcional (hardware/software) para emulação e simulação de sinais biológicos em tempo real.

1.3 Estrutura da dissertação

A estratégia escolhida para atingir os objetivos descritos acima, está relatada em cada um dos capítulos deste trabalho. Estruturalmente, este texto está dividido da seguinte forma:

- Capítulo 1: Introdução, justificativa, objetivo geral e objetivos específicos e estrutura do trabalho;
- Capítulo 2: apresentação das bases téoricas e funcionamento de equipamentos que coletam sinais biológicos;

- Capítulo 3: Especificação e desenvolvimento de um aplicativo capaz de ler bancos de dados de sinais biológicos e convertê-los em sinais adequados para o funcionamento do emulador BioSim;
- Capítulo 4: Especificação e construção de um equipamento emulador/simulador de sinais biológicos;
- Capítulo 5: Resultados;
- Capítulo 6: Conclusões finais relevantes desse trabalho e sugestões para trabalhos futuros que podem ser realizados a partir do mesmo.

Capítulo 2

Aquisição de Sinais Biológicos

Todos os organismos, desde os unicelulares até os mais complexos, produzem de alguma forma, sinais de origem biológica. Estes sinais podem ser elétricos, mecânicos ou químicos (BEMMEL; MUSEN, 1997).

O ponto de interesse desta pesquisa são os sinais biológicos de origem elétrica, como a despolarização da célula cardíaca (sinal cardíaco), a despolarização da célula neuronal (sinal encefálico), despolarização da célula muscular (sinal muscular) e a despolarização da célula muscular ocular (sinal ocular) de acordo com Bemmel e Musen (1997).

Para estudar os sinais biológicos, eles precisam ser coletados e processados. Esta aquisição depende de um protocolo bem elaborado e sua realização de forma fidedígna, o que demanda muito esforço e tempo.

As pesquisas de Giorgi (2013) sobre epilepsia e sinais eletroencefalográficos (EEG) ilustram a dificuldade na aquisição de sinais biológicos. Em seu protocolo o autor utilizou 963 voluntários adultos em experimentos demorados, no período de 2003 até 2010.

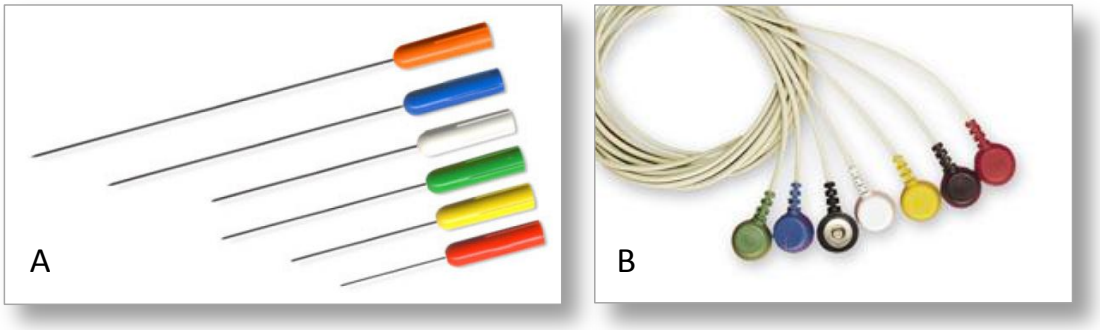
A demora e a dificuldade na realização destas pesquisas não se relacionam apenas com o processo de aquisição dos sinais em si, mas, com vários outros processos, como a elaboração de protocolos, dentro de normas que atendam às regras de segurança e ética, e toda a burocracia envolvida em pesquisas com seres humanos.

Um bom exemplo da complexidade dos sinais biológicos é o eletromiográfico, pois é afetado pelas propriedades anatômicas e fisiológicas da musculatura, pelo esquema de controle do sistema nervoso periférico, assim como pelos instrumentos utilizados para detectá-lo (BERNARDES; ANDRADE; MIOTTO E SÁ, 2007).

A detecção de sinais EMG é efetuada por eletrodos que variam de acordo com o tipo de músculo escolhido. Para músculos situados superficialmente no corpo, são utilizados eletrodos de superfície (não invasivos), para músculos profundos são utilizados eletrodos intramusculares (invasivos). (BERNARDES; ANDRADE; MIOTTO E SÁ, 2007).

Alguns exemplos de eletrodos podem ser visualizados na figura a seguir.

Figura 1: Eletrodos Invasivos (A) e Eletrodos de Superfície (B)



Fonte: A – Aplicação Prática da Fisiologia do Exercício (2013); B – spes medica Brasil (2013).

Para que eletrodos invasivos ou de superfície façam parte da coleta de dados, eles devem estar acoplados a um equipamento, no caso dos sinais EMG, um eletromiógrafo como ilustrado na figura abaixo.

Figura 2: Eletromiógrafo MyosystemBr1 PXX.

Fonte: Datahominis (2013)

A utilização de todas as técnicas, protocolos e equipamentos e de todo o trabalho de coleta, se fundamentam no processamento e análise destes sinais. A análise de dados pode servir para vários propósitos, sendo que um dos mais relevantes é a determinação de

parâmetros que permitam a construção de modelos. Por outro lado, a verificação da adequação destes modelos aos dados reais é fundamental para a validação dos mesmos. (PINTO, 2009).

Estudos envolvendo a aquisição de bio-sinais em animais, como o de Eykern (2001) também servem de modelo para expressar a complexidade destes estudos (figura 3).

Figura 3: Posicionamento de um eletrodo invasivo para EMG veterinário.

Fonte: The University of Tennessee, College of Veterinary Medicine (2012)

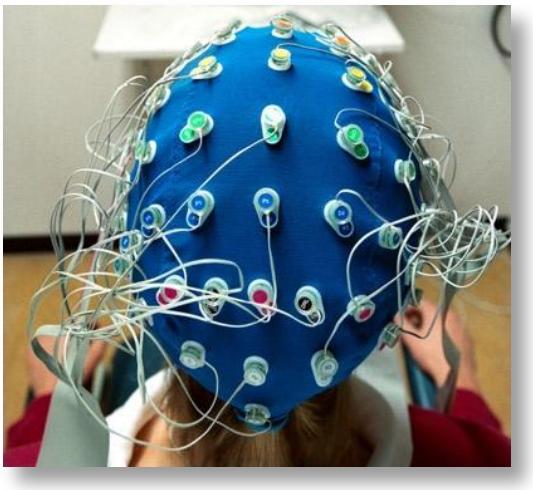
As pesquisas envolvendo eletromiografia na medicina veterinária, possuem uma etapa de coleta ainda mais difícil, pois os eletrodos são mais trabalhosos de serem posicionados e a execução voluntária de ações por parte dos animais, são solicitadas com dificuldade (EYKERN, 2001).

Outra modalidade de sinal biológico é o EEG, que é a atividade elétrica espontânea medida no couro cabeludo ou no cérebro. A amplitude de um sinal EEG é de cerca de 100 microvolts quando medido no couro cabeludo e de 1 a 2 milivolts quando medido na superfície do cérebro (RIEGER, 2006).

A frequência de um sinal cerebral em atividade espontânea, pode variar de menos de 1 Hz a 50 Hz. O EEG também possui componentes que surgem em resposta a um estímulo, que pode ser elétrico, auditivo, visual, entre outros. Estes componentes são denominados potenciais evocados (RIEGER, 2006).

O processo de coleta destes sinais é realizado através de eletrodos aplicados no couro cabeludo, na superfície encefálica, ou até mesmo dentro da substância encefálica. A Figura 4 mostra eletrodos de superfície aplicados no couro cabeludo para a execução de exames.

Figura 4: Eletrodos de Superfície para a realização de um eletroencefalograma.



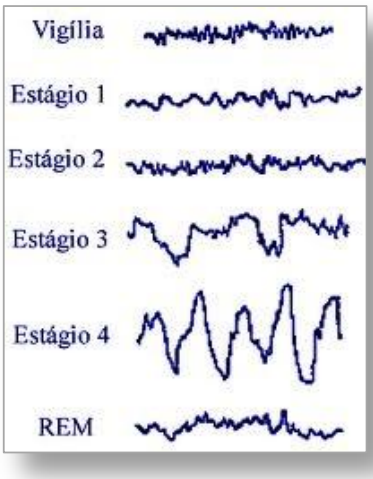
Fonte: Secretaria de Saúde do Estado da Bahia (2010)

Com a tecnologia existente para a realização de um EEG pode-se examinar um único neurônio, utilizando microeletrodos que conseguem ser inseridos apenas na célula de interesse (RIEGER, 2006).

O estudo do sinal encefálico é utilizado para o diagnóstico de síndromes epilépticas, avaliação de coma, morte encefálica, intoxicações, encefalites, síndromes demenciais, crises não epilépticas e distúrbios metabólicos. Sem contar as inúmeras aplicações utilizando potenciais evocados (RIEGER, 2006).

Algumas características deste sinal podem ser observadas na figura 5, que ilustra um estudo a respeito das fases do sono e suas respectivas formas de onda.

Figura 5: Registros eletroencefalográficos das fases do sono.



Fonte: Biolabor Medicina Diagnóstica (2011)

Outros sinais biológicos de origem elétrica são os sinais cardíacos e os sinais oculares. O processo de aquisição destes sinais são similares ao EMG e ao EEG, envolvendo protocolos, equipamentos e técnicas específicas para esta função.

Em todas as pesquisas na área biomédica envolvendo sinais biológicos, a etapa de coleta é parte obrigatória do cronograma, e é interessante que se consiga otimizá-la. Para tanto, surgiu a necessidade de se construir um equipamento capaz de emular estes sinais, economizando o tempo e o trabalho despendido na aquisição dos dados para estudo.

Capítulo 3

Software BioSim

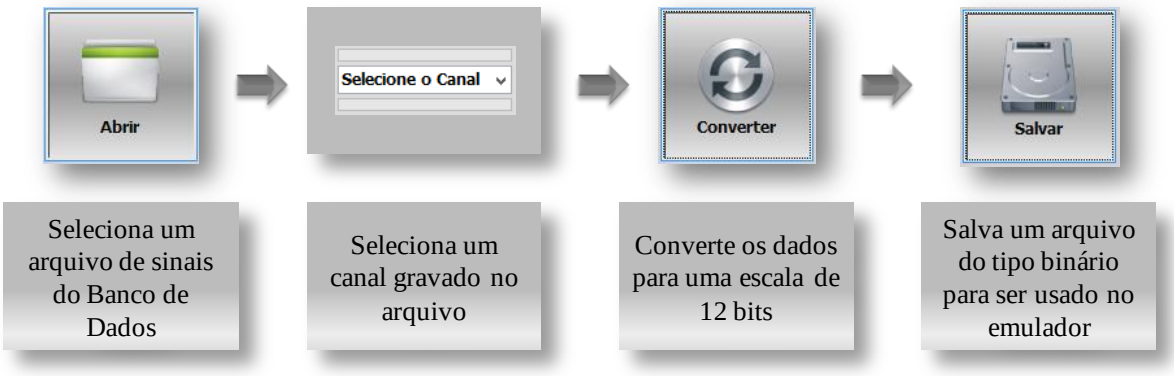
3.1 – Proposta de um sistema computacional para conversão de sinais biológicos

Para a construção deste sistema computacional a linguagem de programação utilizada foi o C#, uma ferramenta de programação do pacote de desenvolvimento da Microsoft, o Visual Studio 2010 Express.

Os computadores que utilizam processadores com velocidade acima de 2.5 gigahertz e memória RAM acima de 2 gigabytes, com sistema operacional Windows XP, ou superior, suportam perfeitamente o programa, e promovem a conversão dos dados sem maiores dificuldades.

O programa desenvolvido levou em consideração a facilidade de operação, dispensando menus complicados, tornando a conversão dos arquivos uma tarefa simples. Basicamente, o programa funciona de acordo com o fluxograma ilustrado na Figura 6.

Figura 6: Fluxograma simplificado do s*oftware* BioSim.

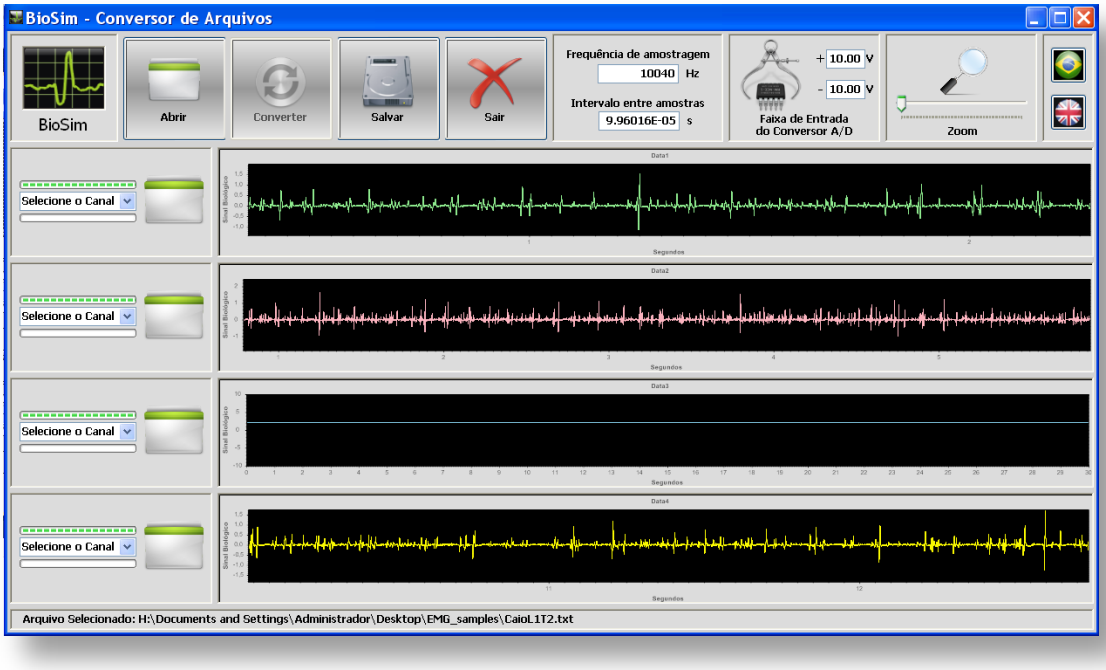


Fonte: Próprio autor (2013).

3.2 – Interface do programa

Na tela de apresentação do programa responsável pela conversão dos arquivos (Figura 7), é possível visualizar todas as informações necessárias para utilização do *software* dispostas através de botões, tais como: Abrir, Converter, Salvar e Sair; além de informações, como a frequência de amostragem, o intervalo entre as amostras e a faixa de entrada do conversor analógico/digital (A/D), que gerou o sinal inicialmente.

Figura 7: Tela de apresentação do *software* BioSim.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A faixa de entrada do conversor A/D pode ser regulada de acordo com o equipamento responsável pela leitura que deu origem aos dados.

Com um alcance entre +10 Volts e -10 Volts, esta regulagem dimensiona a amplitude da forma de onda que será exibida no Gráfico contido no programa, e posteriormente gerada nas saídas analógicas do equipamento BioSim.

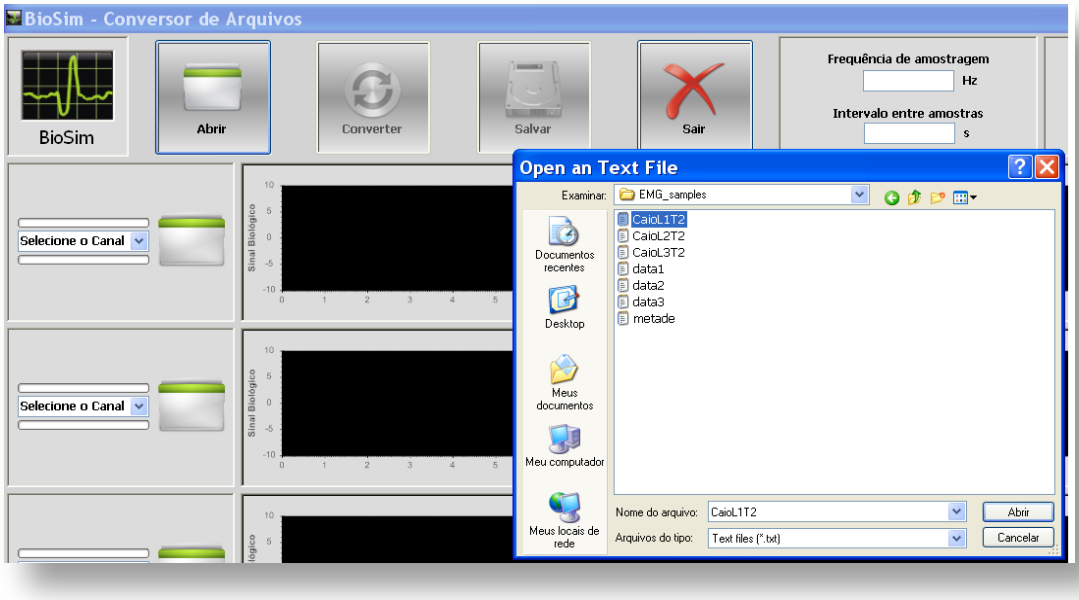
O programa disponibiliza uma ferramenta de *zoom*, para facilitar a visualização do sinal nos Gráficos, além de contar com um botão de seleção de idiomas, podendo optar pelo inglês ou o português. Assim que o usuário estiver com a interface do programa na tela do computador, e os dados dos sinais biológicos em mãos, os passos para a obtenção e conversão

dos arquivos, se restringem apenas aos botões, Abrir, Converter e Salvar. Para finalizar a interface e encerrar a conversão deve-se utilizar o botão Sair.

3.2.1 – Abertura dos arquivos para a conversão

Assim que o usuário clicar no botão Abrir, o *software* mostrará uma janela de navegação que o permitirá escolher o arquivo para conversão como pode ser visualizado na Figura 8.

Figura 8: Tela que permite a escolha do arquivo para a conversão dos dados.

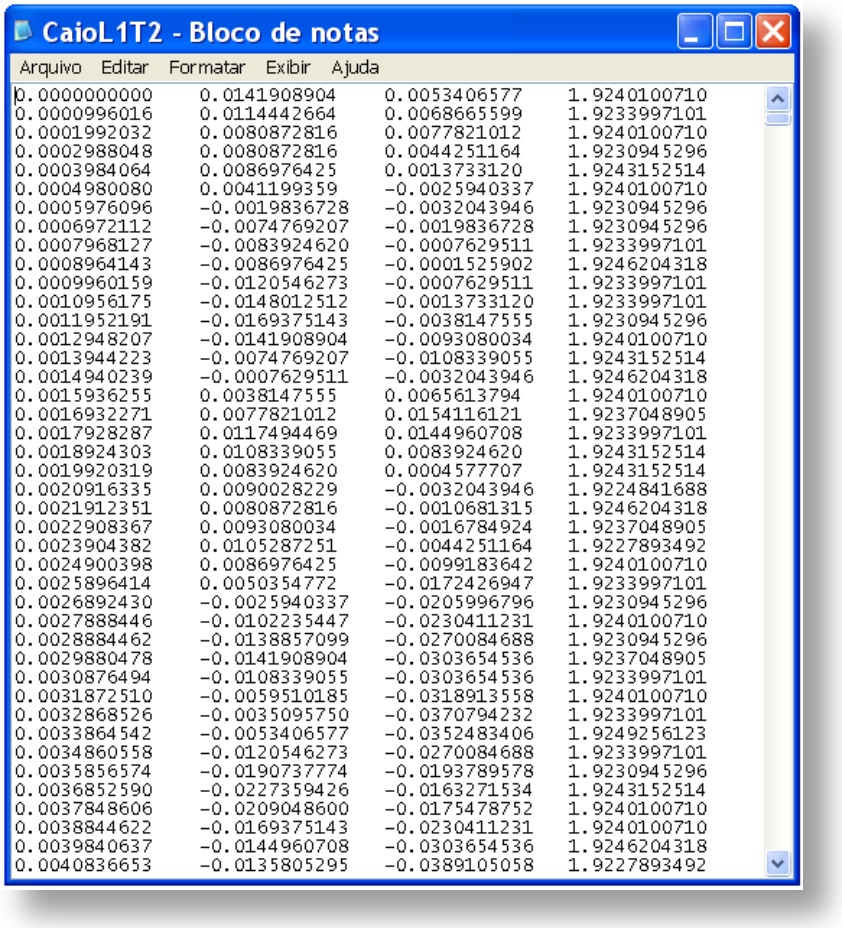


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O arquivo deve estar no formato de texto “.txt” para que o programa consiga efetuar a leitura. Os arquivos provenientes de equipamentos que promovem aquisição de dados biológicos normalmente disponibilizam as informações em colunas, sendo, a primeira coluna o intervalo de tempo entre as amostras, que é obrigatória para o funcionamento correto do BioSim, e as demais colunas, as amostras coletadas pelos canais do equipamento.

Uma amostra de um banco de dados de sinais eletromiográficos coletados a partir de um eletromiógrafo, com intervalo entre amostras de 99,6016 microsegundos e três canais de aquisição distintos, com precisão de 12 digitos, pode ser observado na Figura 9.

Figura 9: Arquivo contendo informações sobre sinais musculares (eletromiográficos).



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Este arquivo conta com um total de 301.201 amostras em cada coluna, armazenado 30 segundos de leitura de um sinal muscular.

Assim que um arquivo como este, for escolhido, o *software* promove sua leitura e identifica como estão dispostas as informações, no caso do arquivo da Figura 9, o programa estabelece a primeira coluna como o intervalo de tempo entre as amostras, e calcula desta forma a frequência de amostragem.

As colunas subsequentes informam quantos canais estão à disposição, além do número de amostras de cada canal e a intensidade (amplitude) do seu sinal.

O programa foi desenvolvido para conseguir identificar até 50 canais diferentes e um único arquivo.

Um sinal sonoro e o aparecimento de uma seta ao lado da imagem de uma pasta de arquivos indica que a abertura do arquivo e a identificação de todas as suas informações necessárias para a conversão foram lidas com sucesso.

O *software* apresenta uma opção para seleção de canais, onde o usuário escolhe qual canal deseja emular, e libera o botão Converter para que se possa dar início à conversão, conforme ilustra a Figura 10.

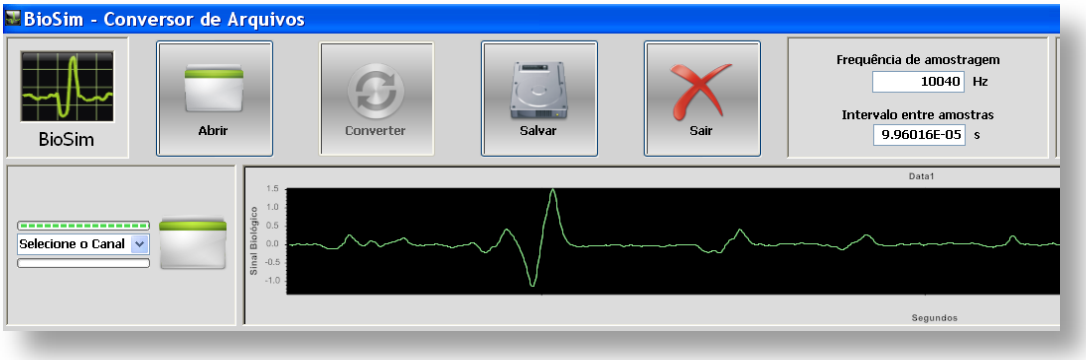
Figura 10: Tela que permite a seleção de canais para conversão.

Fonte: Sistema BioSim desenvolvido pelo autor (2013).

3.2.2 – Conversão dos canais selecionados no arquivo.

O usuário, depois de ter selecionado o canal desejado, basta clicar no botão converter, e o programa automaticamente converterá os dados lidos do sinal biológico contido no arquivo ".txt", para uma escala de 12 bits, com valores variando entre 0 e 4095, que será utilizada para plotar o Gráfico, conforme mostra a Figura 11.

Figura 11: Conversão de arquivos.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Estes mesmos dados, após convertidos, serão salvos em um arquivo binário, e gravados em cartões de memória microSD, que serão lidos mais tarde pelo *hardware* BioSim e finalmente voltarão a apresentar sua forma analógica.

3.2.3 – Salvando os dados que serão usados no *hardware*

Após a conversão do arquivo, o usuário deverá clicar no botão Salvar. Dessa forma, o programa irá gerar um arquivo binário ".bin" nomeado de acordo com um dos quatro canais dispostos na tela do programa: data1.bin para o canal 1, data2.bin para o canal 2, data3.bin para o canal 3 e data4.bin para o canal 4.

Este arquivo deverá ser salvo em um cartão de memória de no máximo 4 Gb de capacidade, para então ser utilizado pelo *hardware*.

3.2.4 – Campos que facilitam a utilização por parte do usuário.

Para melhorar a interação com o usuário, o software dispõe de algumas informações como a frequência de amostragem e o intervalo entre as amostras, ilustrados na Figura 12. Conta também com um campo onde o usuário informa a faixa de entrada do conversor A/D utilizado na coleta original, o que permite a confecção de um Gráfico com as mesmas características de amplitude.

Figura 12: Informações e ferramentas do BioSim.

Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O programa disponibiliza ferramentas de zoom, ampliando o sinal de acordo com o desejo do usuário, e de mudança de idiomas, que alterna entre o inglês e o português, conforme mostra a figura acima.

Capítulo 4

Hardware BioSim

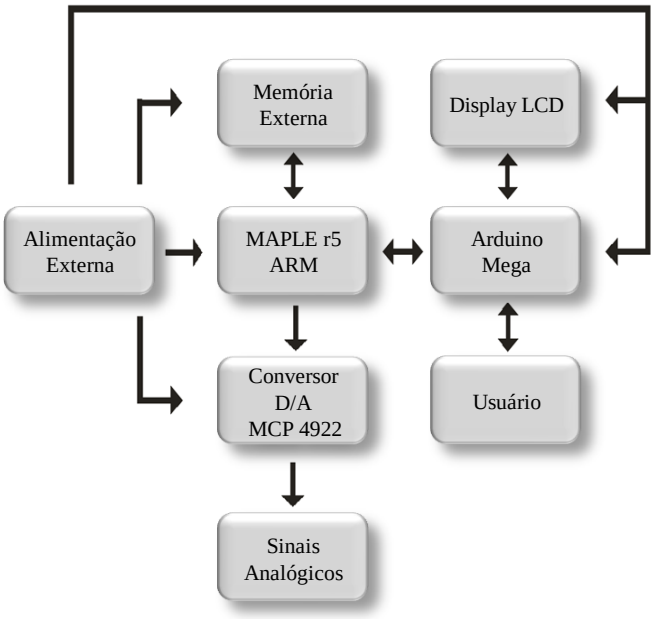
4.1 – Projeto do Sistema

O *hardware* proposto é capaz de receber um arquivo binário, que contém as informações convertidas e gravadas pelo *software* BioSim, referente à coleta de sinais biológicos em uma determinada taxa de amostragem.

Ao receber e ler este arquivo de uma memória externa, o equipamento faz a leitura destes sinais digitais e os envia a um conversor digital/analógico (DAC) de 12 bits, que se encarrega da conversão dos sinais digitais em sinais analógicos, que podem então, ser observados e analisados.

A figura 13 mostra o diagrama de blocos com a arquitetura de apenas um canal do *hardware* emulador/simulador de sinais biológicos.

Figura 13: Fluxograma de Funcionamento de um canal do equipamento BioSim.



Fonte: Próprio autor (2013).

A alimentação do sistema é fornecida por uma fonte de corrente contínua, com tensão e corrente adequadas ao funcionamento de todos os componentes envolvidos no projeto, neste caso, uma fonte de tensão de 7,5 Volts e capacidade para fornecer até 1,5 Ampères.

Este sistema lê arquivos de texto no formato binário, armazenados em uma memória externa (cartão micro SD), em que estão gravados dados que representam a coleta de sinais biológicos em uma determinada taxa de amostragem.

A interação do usuário com o Emulador/Simulador de Sinais Biológicos – BioSim, é realizada por meio de uma interface homem-máquina composta de um display de cristal líquido e botões para navegação, controlados por um microcontrolador.

Com os botões de navegação, pode-se escolher um canal para emulação dos dados, assim como determinar a taxa de amostragem, sendo que neste projeto é possível emular até quatro canais simultaneamente com frequências que podem variar de 10 Hz até 20 kHz.

A interface também possui um botão ON/OFF (liga/desliga), compartimentos para inserção dos cartões micro SD, teclado numérico para escolha das frequências, botão de *LOOP* (opção para deixar o sinal emulando indefinidamente), botão ENTER (define a frequência desejada), botão ESC (volta para a tela inicial), botão PLAY (inicia a emulação), botão PAUSE (pausa a emulação), botão STOP (para a emulação).

Existem ainda, botões nomeados com as letras A (emular canais), B (ajustes) e C (ajuda), responsáveis pelos comandos da tela de navegação do *hardware*. O Botão A, promove a emulação com a frequência de 10.040 Hz (valor *default* do equipamento).

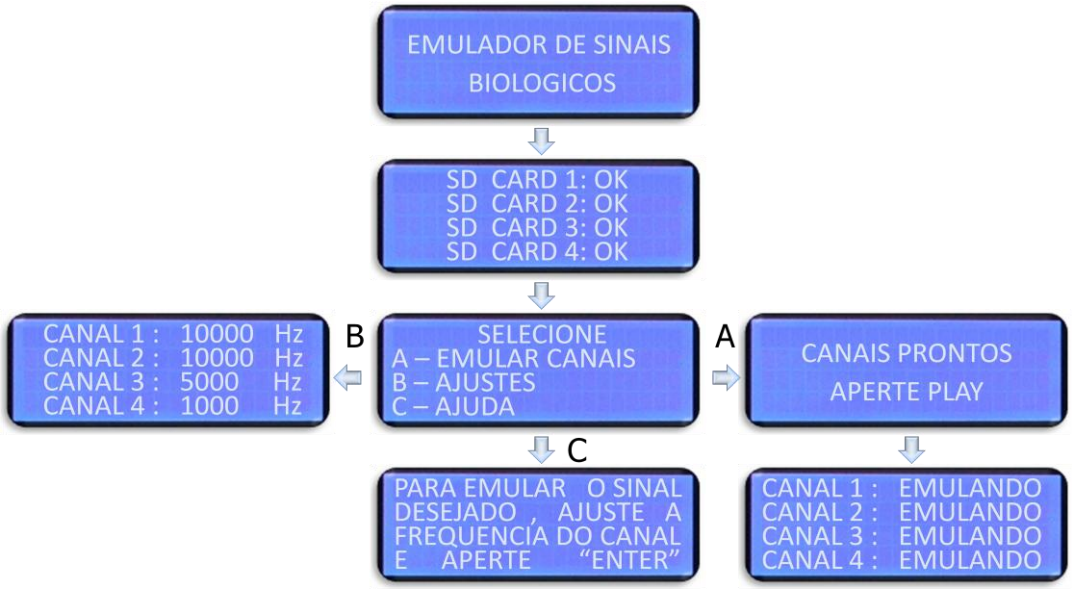
O valor da frequência *default* de 10.040 Hz, foi escolhido por conveniência do programador, já que no princípio do estudo, o emulador seria utilizado apenas para sinais eletromiográficos, pois o grupo de pesquisas do BioLab possui um grande banco de dados destes sinais coletados especificamente nesta frequência.

Neste estudo, dois modelos de microcontroladores foram escolhidos e utilizados para maximizar a eficiência do *hardware*, um microcontrolador ATmega 2560 em uma placa denominada Arduino Mega, e o microcontrolador ARM Cortex M3 em uma placa MAPLE.

Para realizar a conversão dos dados digitais contidos nos cartões de memória em sinais analógicos, foi utilizado um conversor digital/analógico (DAC) de 12 bits, denominado MCP4922.

A Figura 14 ilustra as mensagens no display do emulador de sinais biológicos, com uma sequência de telas que tem como objetivo deixar a interação homem-máquina o mais simples possível.

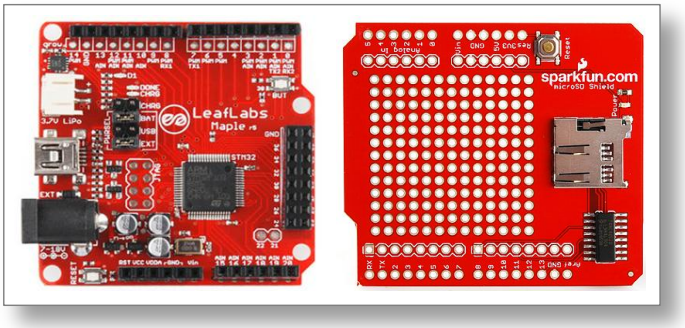
Figura 14: Ilustração da sequência de mensagens do *display* de cristal líquido.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Para processar os dados dos sinais biológicos de cada canal separadamente, foi escolhida uma placa MAPLE (ARM Stm 32), figura 15, que conta com um poderoso microcontrolador ARM Cortex M3, com velocidade de *clock* de 72MHz e 32 bits o que garante uma emulação em tempo real.

Figura 15: Placa MAPLE r5.

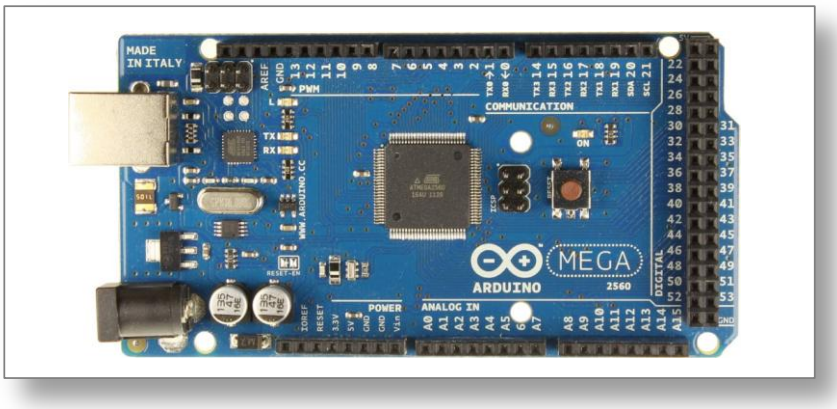


Fonte: Leaflabs (2012)

A conveniência de se utilizar as placas MAPLE e Arduino é que elas possuem todos os componentes necessários ao suporte do microcontrolador, podendo ser conectada a um cabo USB convencional que além de energizá-las é usado para a gravação de seus "*firmwares*" de acordo com Leaflabs e Arduino (2012).

Para promover a interação entre os microprocessadores ARM e a interface homem-máquina foi escolhido uma placa Arduino Mega (Figura 16).

Figura 16: Placa Arduino Mega.



Fonte: Arduino (2012).

A placa Arduino Mega é a responsável por controlar o display de cristal líquido e os botões de navegação, além de gerenciar os canais do emulador, fazendo com que eles sejam acionados de forma individual ou conjunta.

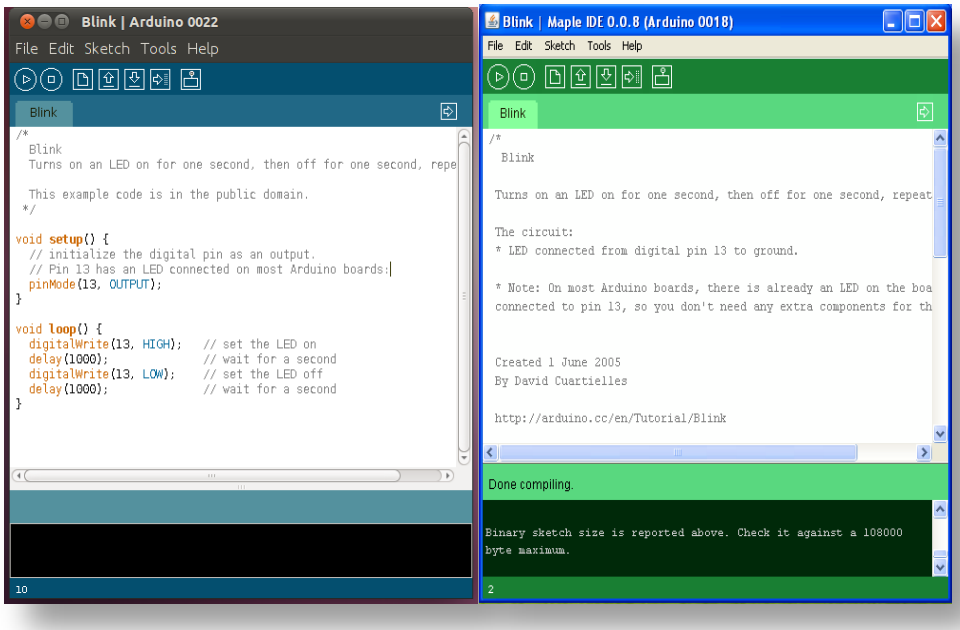
Estas funções só são possíveis graças à quantidade de portas disponíveis nesta placa, 54 no total, que podem ser configuradas como entradas e saídas. Além desta vantagem, esta placa possui quatro portas de comunicação serial, (ARDUINO, 2012). utilizadas para comunicação com as placas MAPLE.

A placa MAPLE compõe o canal do emulador. Quando é acionada através do Arduino Mega, realiza a leitura do cartão de memória e envia estes dados para um conversor de sinais digitais/analógicos (MCP4922). Este conversor, por sua vez, disponibiliza os sinais nas saídas do equipamento.

A escolha destas placas se deve à simplificação do processo ao se trabalhar com microcontroladores, alocando os detalhes complexos de programação e interligação de diversos componentes em um pacote simples. Tanto o *software* Arduino, como o *software* MAPLE, podem ser executados nas plataformas Windows, Macintosh OSX e Linux, além de serem *open source* e possuírem hardware extensível (ARDUINO, 2012).

As interfaces do *software* utilizado para a programação das duas placas são mostradas na Figura 17.

Figura 17: Interfaces do *software.*



Fonte: Leaflabs e Arduino (2012).

As bibliotecas utilizadas para a programação embarcada são similares para as duas placas, o que facilita a comunicação entre elas, conforme o Leaflabs e Arduino (2012).

Após interação do usuário e a definição de quais canais devem ser emulados o microcontrolador Arduino envia um comando para o microcontrolador ARM referente ao canal escolhido, que, por sua vez, lê os arquivos do cartão de memória e os envia para o conversor MCP4922 que promove a conversão destes dados em sinais analógicos.

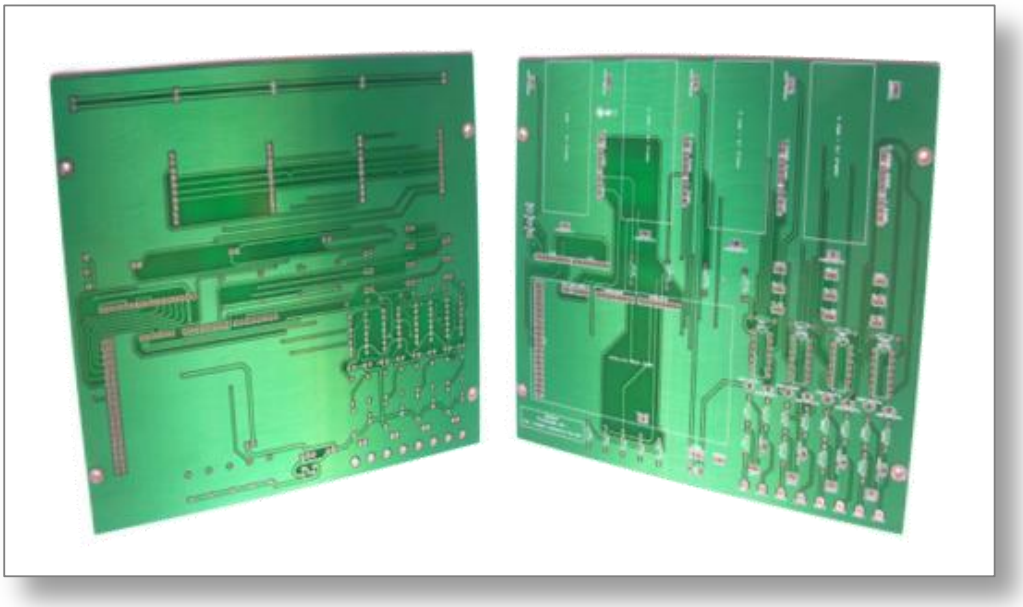
A partir deste ponto, é possível captar o sinal emulado disponível nas saídas do conversor e realizar suas atividades de interesse, como por exemplo, testes de instrumentação de coleta em tempo real.

4.2 – Construção do equipamento

Conforme citado anteriormente o equipamento foi construido a partir de quatro placas MAPLE para o processamento, uma placa Arduino MEGA e quatro conversores MCP4922, além dos teclados e do display de cristal líquido.

Para interligar todos estes componentes e garantir que a comunicação entre eles fosse perfeita, foi desenvolvida uma placa de circuito impresso como mostra a Figura 18.

Figura 18: Placa de circuito impresso.

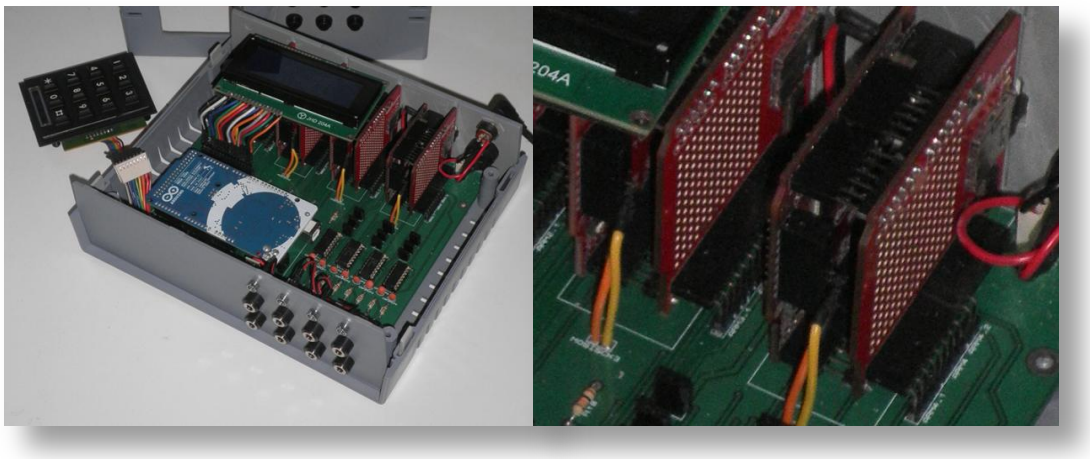


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Desde o começo do projeto, a exigência foi que o equipamento pudesse ser de fácil construção e manutenção, dessa forma a placa foi projetada para ser, essencialmente, um conector entre os diversos componentes, podendo ser montada e desmontada com facilidade.

A placa que abriga o cartão de memória, foi construída para se encaixar na placa de processamento MAPLE r5, conforme mostra o detalhe ampliado na Figura 19.

Figura 19: Shield do cartão micro SD.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O conjunto placa do cartão microSD e placa MAPLE, são conectados na placa de circuito impresso com os pinos de alimentação (power/ground), sinais digitais (entrada e

saída) e comunição SPI (Serial Peripheral Interface), devidamente posicionados para cada um dos quatro canais.

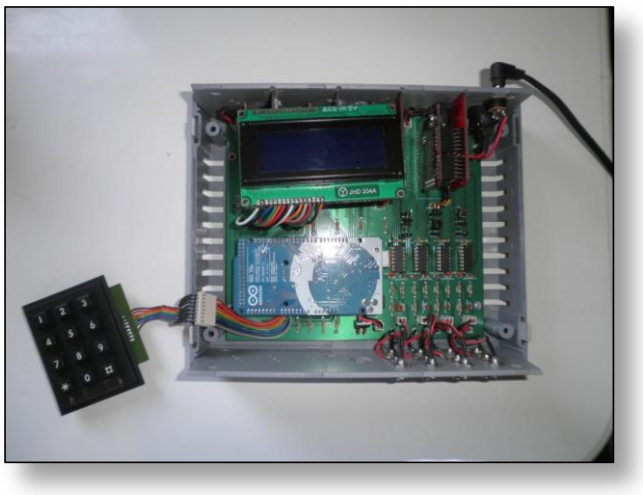
A placa Arduino Mega 2560 é encaixada em uma sequência de pinos, dispostos de tal forma que só possibilita uma forma de conexão conforme mostra a Figura 20.

Figura 20: Vista interna posterior do equipamento.

Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O display de cristal líquido, assim como o teclado numérico possuem suas conexões bem definidas, inclusive com cores guias, impossibilitando qualquer erro em suas ligações. Estes detalhes podem ser observados na Figura 21.

Figura 21: Vista superior evidenciando cabos e cores guias.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

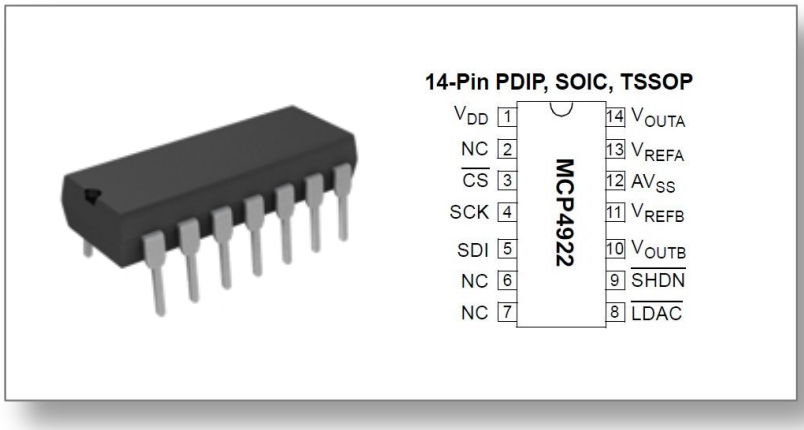
O dispositivo de conversão utilizado, foi o MCP4922, um conversor digital/analógico de 12 bits, que se orienta por uma referência de tensão externa (V_{REF}), o que significa que este

dispositivo converte uma escala numérica de 0 a 4095 em tensões que variam de 0 Volts a 5 Volts (V_{REF}).

Este circuito integrado oferece alta precisão e baixo consumo de energia. A comunicação é feita através de uma interface em série, utilizando protocolos SPI. Algumas de suas vantagens são a precisão elevada, o baixo nível de ruído e custo bastante acessível (MICROSHIP, 2013).

O encapsulamento e o *footprint* do dispositivo podem ser vistos na Figura 22.

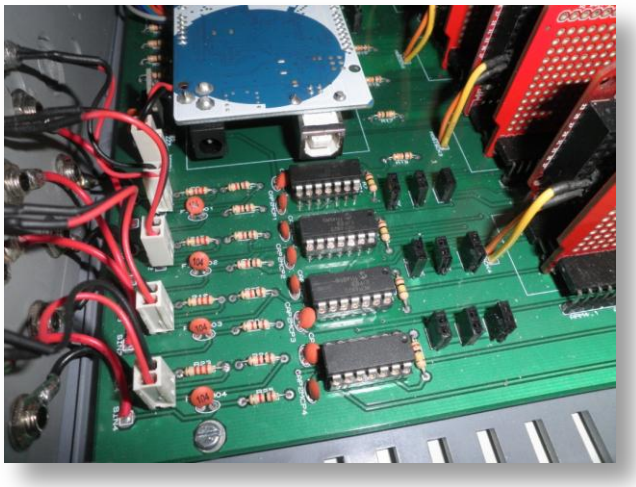
Figura 22: Encapsulamento e o *footprint* do conversor digital analógico.



Fonte: Microship (2013).

A disposição dos conversores MCP4922 na placa de circuito impresso é mostrada em detalhes na figura 23.

Figura 23: Disposição dos Conversores MCP4922



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Depois de finalizada a etapa de montagem, a Figura 24 mostra o *hardware* BioSim pronto para uso.

Figura 24: *Hardware* BioSim.

Fonte: Sistema BioSim desenvolvido pelo autor (2013).

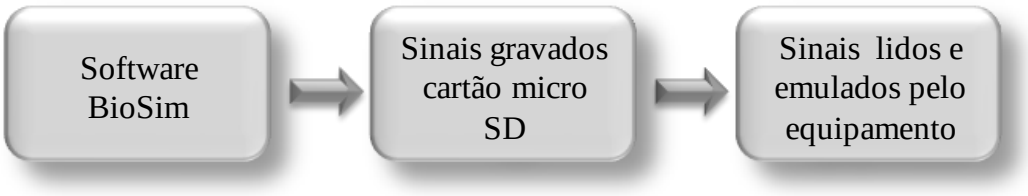
Capítulo 5

Resultados

No intuito de comprovar a eficácia do dispositivo e validar a qualidade dos sinais, foi realizado um processo de comparação entre os sinais reais, gerados via computador, e os sinais coletados na saída do equipamento de emulação.

O processo que vai desde o carregamento dos dados digitais até a saída do sinal analógico, nos canais do emulador, pode ser facilmente visualizado na Figura 25.

Figura 25: Fluxograma de funcionamento do equipamento.



Fonte: Próprio autor (2013).

Um conversor analógico/digital (A/D) de 12 bits, próprio para aquisição de dados, denominado USB-1208FS, da marca MC Measurement Computing (Figura 26), foi utilizado para coletar os sinais analógicos gerados pelo equipamento BioSim com a intenção de validar o sistema.

Figura 26 : Conversor A/D USB-1208FS

Fonte: MCCDAQ (2013).

Para que os gráficos pudessem ser comparados, foi necessário criar um programa MATLAB® da empresa MathWorks®, responsável por interligar o conversor ao computador, com a finalidade de capturar os dados do emulador e disponibilizá-los em gráficos de fácil visualização.

Este processo pode ser visualizado na Figura 27.

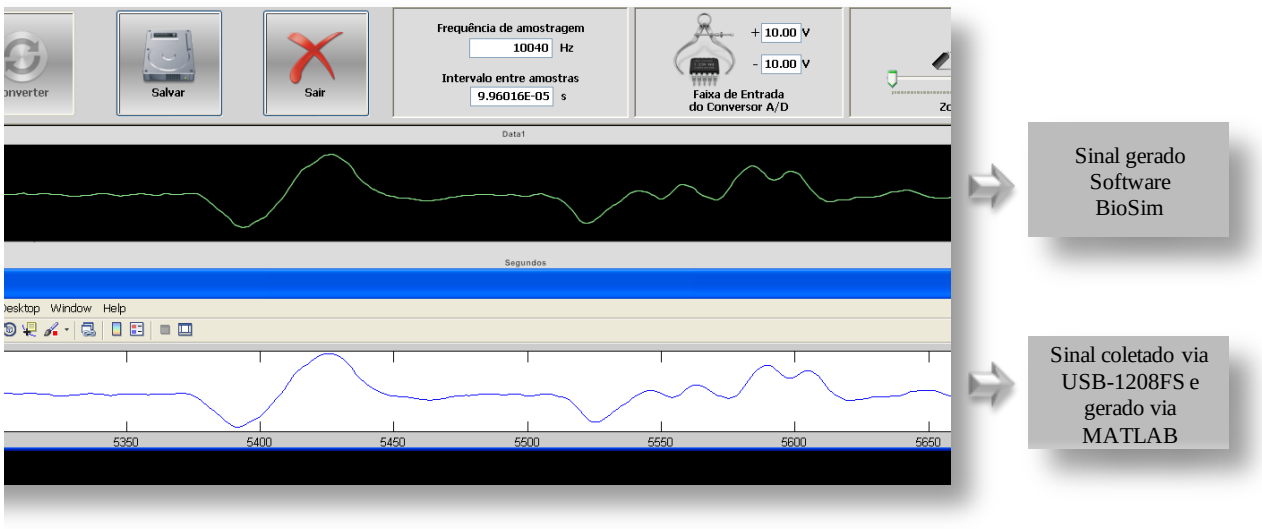
Figura 27: Processo de geração dos gráficos para análise.

Fonte: Próprio autor (2013).

Os gráficos gerados pelo programa MATLAB® foram comparados aos Gráficos fornecidos inicialmente pelo *software* BioSim.

Uma amostra visual desta comparação pode ser vista na Figura 28, onde o gráfico com fundo preto é o resultado gerado pelo programa computacional, e o gráfico com o fundo branco é o resultado obtido pelo programa MATLAB® interligado ao conversor A/D, conectado ao *hardware* BioSim.

Figura 28: Comparação entre resultados do *software* BioSim e MATLAB®.



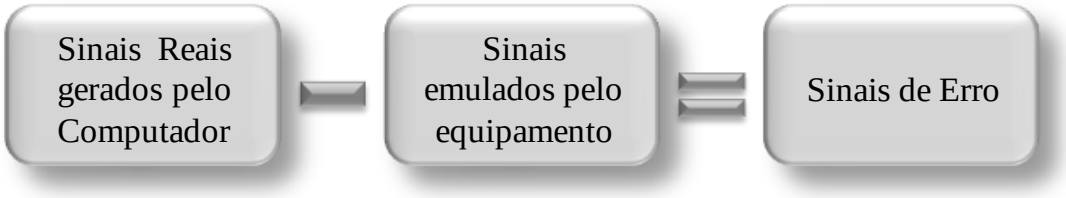
Fonte: Fonte: Sistema BioSim desenvolvido pelo autor (2013) e MATLAB (2012).

A Figura 28 ilustra os sinais gerados pelo programa computacional e os sinas obtidos do *hardware* BioSim.

Visualmente parece não existirem erros entre as amostras obtidas nos dois processos. Mas, estes erros existem, e descobrir sua amplitude é o fator fundamental para comprovar a eficácia do equipamento proposto.

Para tanto, as amostras reais, e as amostras geradas pelo equipamento, foram sobrepostas e comparadas, sendo que a diferença entre elas, nos dá a dimensão do sinal de erro. Abaixo é possível visualizar este processo através do fluxograma proposto (Figura 29).

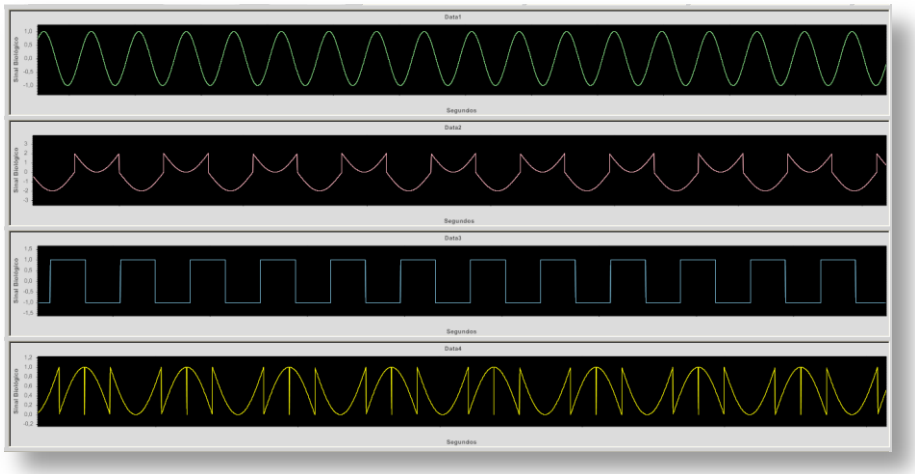
Figura 29: Comparação entre sinais e aquisição de sinais de erro.



Fonte: Próprio autor (2013).

Apesar da proposta do equipamento tratar de sinais biológicos, o BioSim é capaz de emular/simular qualquer forma de onda que o usuário desejar, desde que eles estejam na amplitude e até 5 volts e com frequência de amostragem de até 20kHz. Alguns exemplos são ilustrados na Figura 30.

Figura 30: Exemplos de formas de ondas.



Fonte: Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Os sinais utilizados no processo de validação foram escolhidos seguindo os seguintes critérios: sinais conhecidos contínuos (senóide), sinais conhecidos com interrupções brucas (onda quadrada) e sinais biológicos (sinais eletromiográficos).

Todos os sinais foram testados em diversas frequências de taxa de amostragem: 1 kHz, 5 kHz, 10 kHz, 15 kHz e 20 kHz.

Estas frequências foram escolhidas pois a proposta do equipamento emulador/simulador de sinais biológicos BioSim é conseguir reproduzir sinais amostrados entre 10 Hz e 20 kHz em tempo real, praticamente sem modificar a forma de onda dos sinais originais.

A seguir será detalhado, através de diversas figuras, como o equipamento se comporta reproduzindo analogicamente os sinais digitais contidos nos cartões de memória.

5.1 – Sinais biológicos

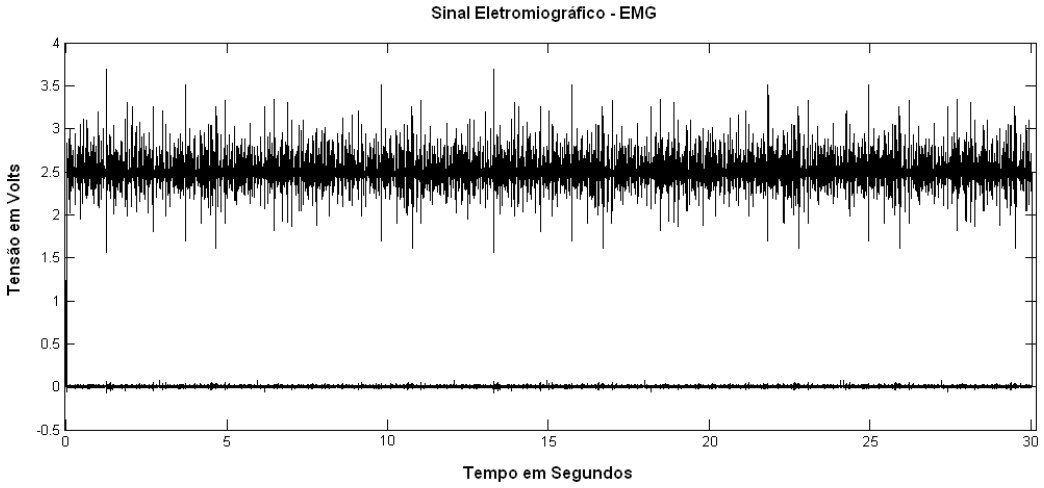
Buscando validar o equipamento com a utilização de sinais biológicos, foi escolhido um sinal eletromiográfico (EMG), que foi analisado em diversas frequências, e comparado com o sinal original, para a obtensão dos sinais de erro.

As frequências e amplitudes ficaram dentro dos limites de operação do aparelho BioSim. As frequências variam de 10 Hz a 20 kHz e a amplitude máxima é de 5 volts.

Após estas comparações, foi calculado o Desvio Padrão (DP), em todas as frequências sugeridas, para conferir uma análise estatística aos resultados obtidos pelo equipamento.

A Figura 31 ilustra uma sobreposição dos sinais originais e dos sinais gerados pelo equipamento em 10kHz, bem como sua diferença, o sinal de erro.

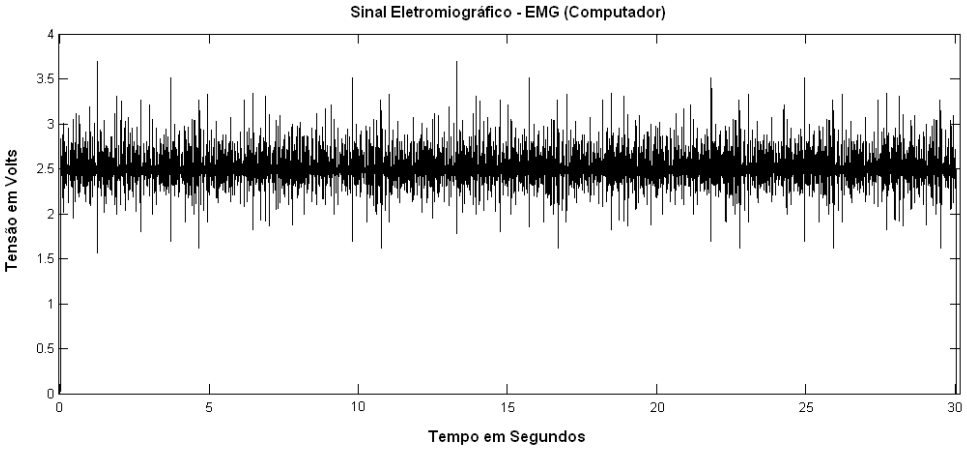
Figura 31: Sinal EMG emulado, sinal EMG original e sinal de erro.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Para visualisar com mais facilidade o que acontece com os sinais da Figura 31, foram geradas as Figuras 32, 33 e 34, com os mesmos sinais plotados separadamente.

Figura 32: Sinal eletromiográfico gerado pelo computador.

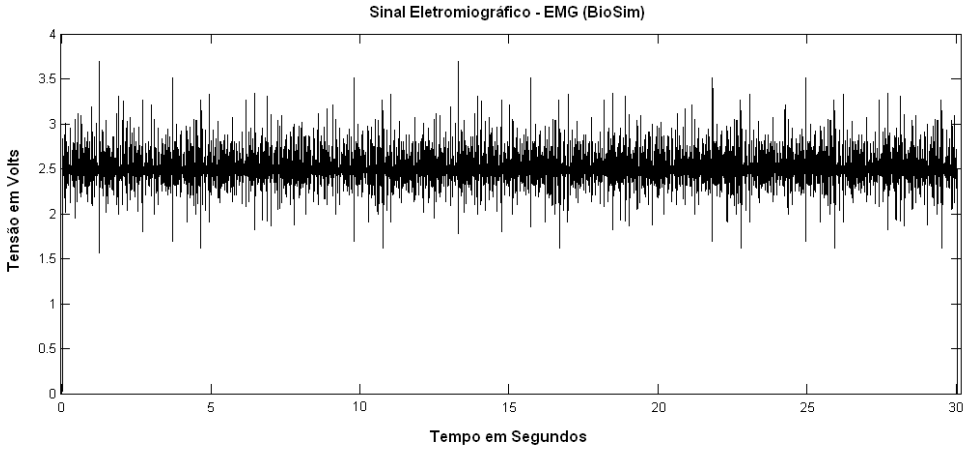


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A Figura 32 foi gerada a partir de um sinal real digital, convertido pelo computador. Este mesmo sinal, foi gravado em um cartão de memória e inserido no equipamento BioSim, com a finalidade de ser obtido analogicamente em um de seus canais.

A Figura 33 foi gerada com a coleta dos dados analógicos na saída do equipamento BioSim, feita pelo conversor USB-1208FS.

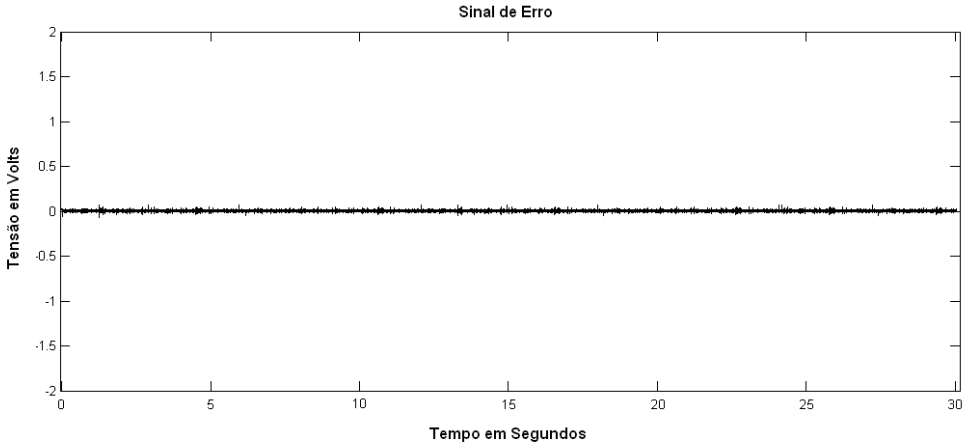
Figura 33: Sinal eletromiográfico gerado pelo equipamento BioSim.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Analisando visualmente as Figuras nesta resolução, é praticamente impossível notar erros entre os dois sinais. Para tanto, a Figura 34 ilustra o sinal de erro entre as amostras, que são as diferenças entre os sinais digitais fornecidos pelo computador e os sinais analógicos fornecidos pelo equipamento BioSim.

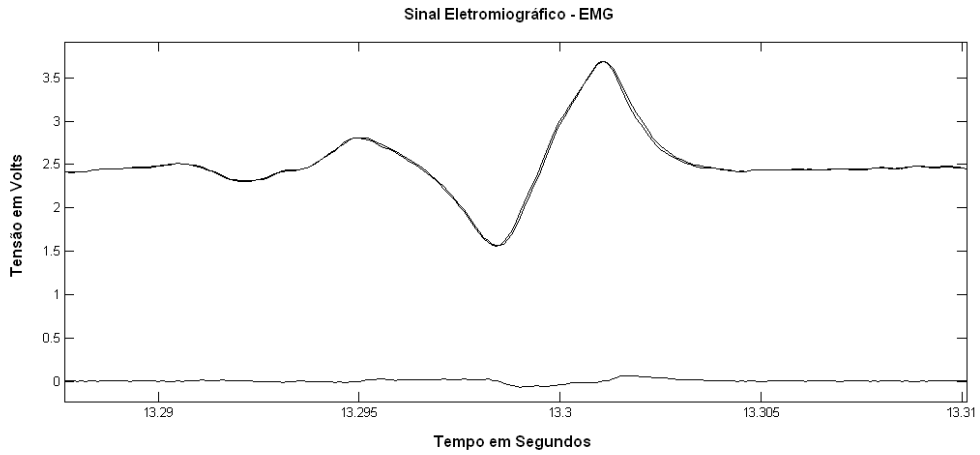
Figura 34: Sinal de erro gerado pela diferença entre o sinal emulado e o sinal original.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A Figura 35 ilustra as mesmas formas de onda vistas anteriormente, mas em uma resolução que se consegue evidenciar, nitidamente, as diferenças entre os sinais digitais e os sinais analógicos, bem como a variação do sinal de erro.

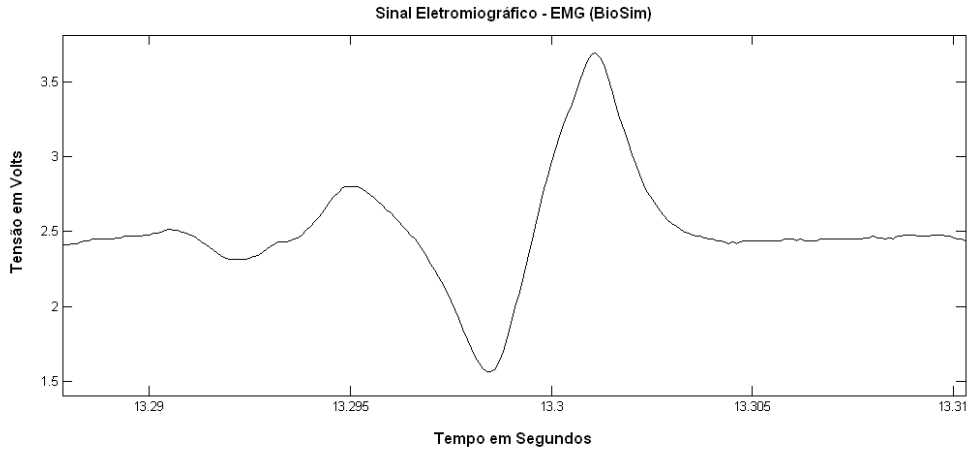
Figura 35: Sinal EMG emulado, sinal EMG original e sinal de erro (alta resolução).



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Na Figura 36 nota-se que o sinal eletromiográfico (analógico), gerado pelo equipamento BioSim, é significativamente similar ao sinal original (digital), gerado pelo computador.

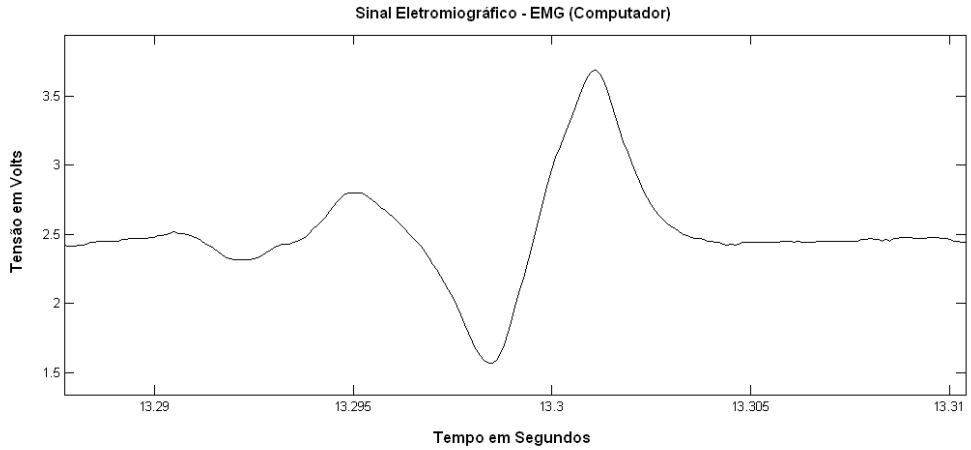
Figura 36: Sinal eletromiográfico gerado pelo equipamento BioSim (alta resolução).



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O sinal gerado pelo computador é ilustrado pela Figura 7, conforme pode ser verificado abaixo.

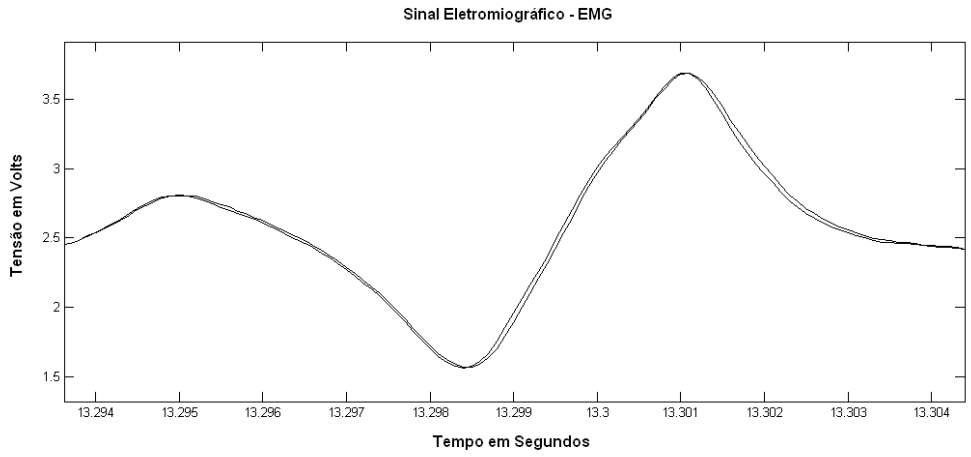
Figura 37: Sinal eletromiográfico gerado pelo computador (alta resolução).



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A Figura 38 mostra os dois sinais sobrepostos com uma resolução ainda maior. Nesta imagem é possível identificar as pequenas variações de amplitude entre os sinais.

Figura 38: Sinais eletromiográficos sobrepostos.

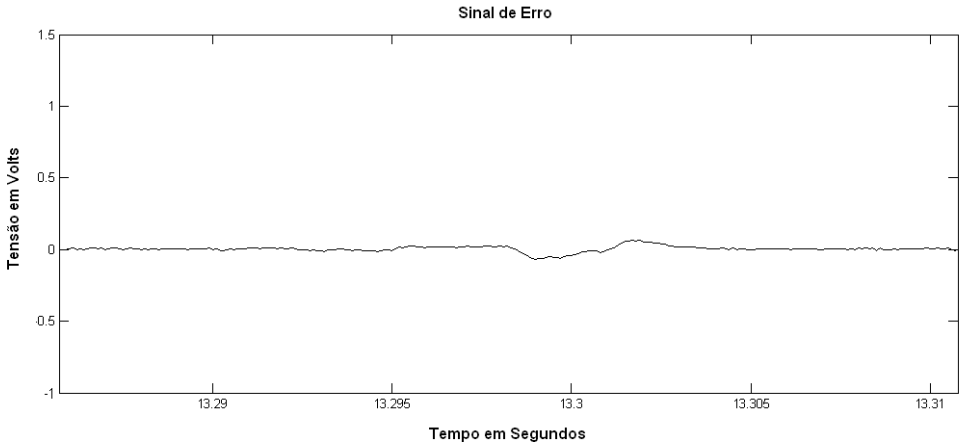


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

As pequenas variações identificadas nos sinais da Figura 38, podem ser expressas com o sinal de erro, gerado pela diferença entre eles. Este sinal conforme observa-se na Figura 39,

é sempre bem próximo do zero, o que reflete a qualidade do equipamento BioSim, na geração de sinais analógicos derivados de sinais digitais.

Figura 39: Sinal de erro

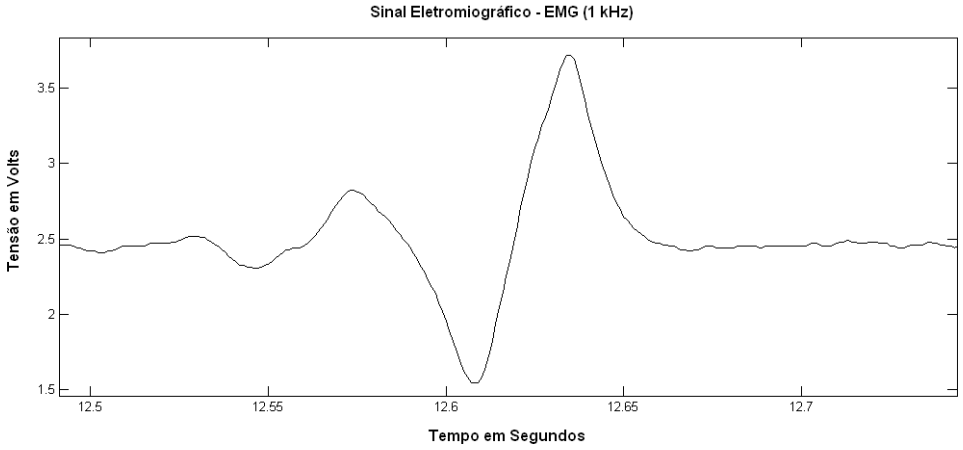


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

5.1.1 – Diferenças na frequência de aquisição do sinal

Os próximos gráficos apresentam o potencial de ação de unidades motoras, gerado em diferentes taxas de amostragem, e tem a intenção de comparar a resposta do equipamento BioSim, nas frequências de 1 kHz, 5 kHz, 10 kHz, 15 kHz e 20 kHz. A Figura 40, mostra um sinal plotado em 1 kHz, ilustrado abaixo:

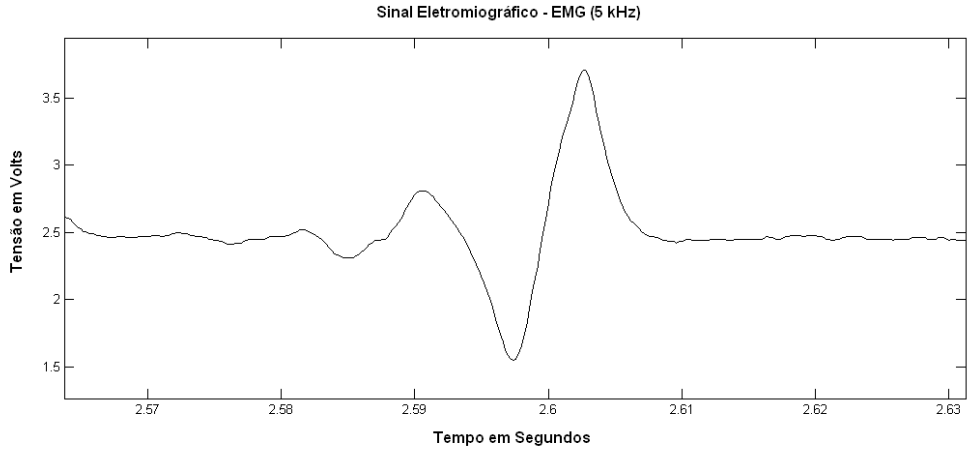
Figura 40: Sinal eletromiográfico formado por amostras em 1 kHz.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O sinal abaixo, conforme a Figura 41, foi gerado por pontos adquiridos em uma taxa de amostragem de 5 kHz.

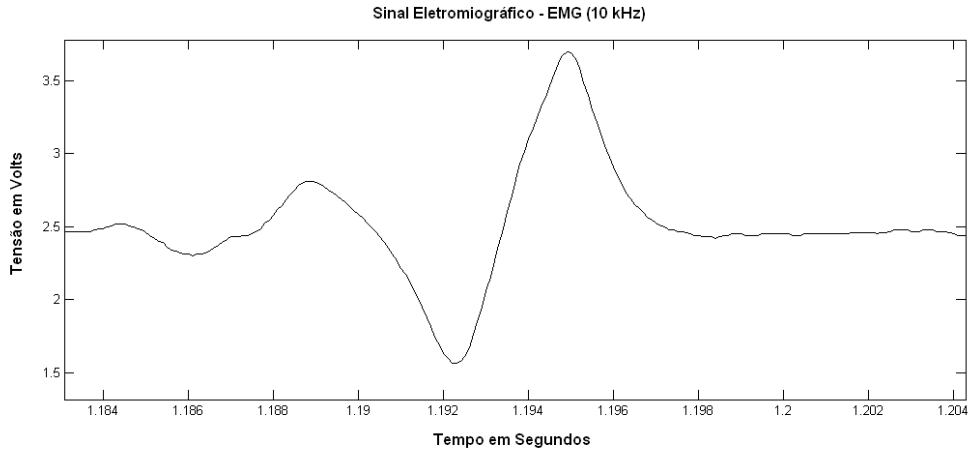
Figura 41: Sinal eletromiográfico formado por amostras em 5 kHz.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Mais uma vez, a mesma parte do sinal eletromiográfico, formado por amostras em 10 kHz, como nos mostra a Figura 42.

Figura 42: Sinal eletromiográfico formado por amostras em 10 kHz.



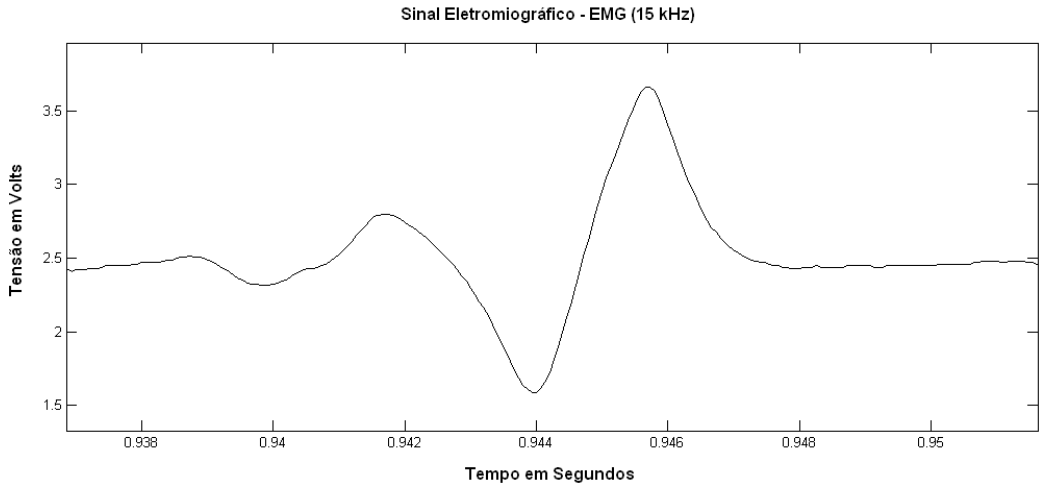
Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A pequena diferença visível entre uma onda e outra, são variações de resolução no momento em que aproximamos para melhor visualizar o gráfico. As formas de onda

apresentam variações mínimas, praticamente imperceptíveis entre uma frequência de amostragem e outra. A compreensão destas variações só é possível com uma análise estatística, apresentada na seção 5.4.

A Figura 43 apresenta a forma de onda gerada a uma taxa de amostragem de 15 kHz.

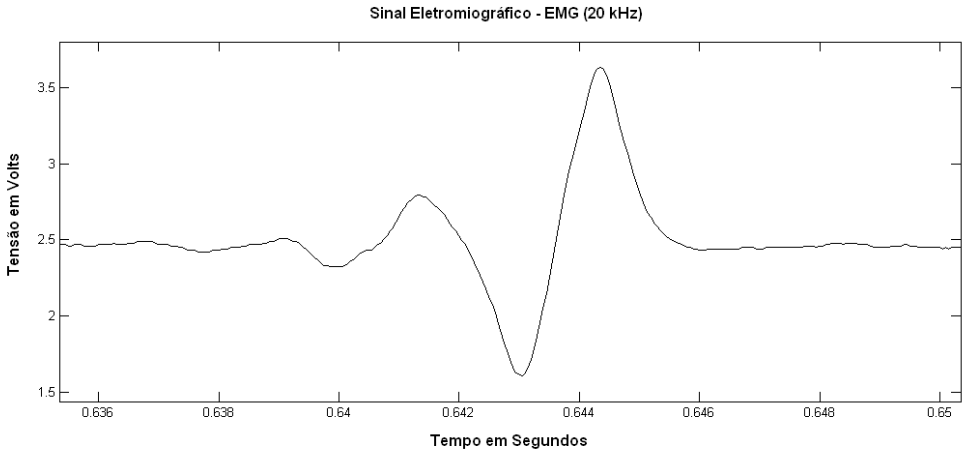
Figura 43: Sinal eletromiográfico formado por amostras em 15 kHz.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Finalmente, a Figura 44, ilustra a mesma forma de onda anterior, mas desta vez, com uma taxa de aquisição de amostras de 20 kHz, que é a taxa máxima permitida pelo equipamento.

Figura 44: Sinal eletromiográfico formado por amostras em 20kHz.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Acima desta taxa, os conversores MCP4922, começam a apresentar dificuldades para reconstruir os dados analógicos com precisão e o grau de distorção das ondas apresentadas aumenta.

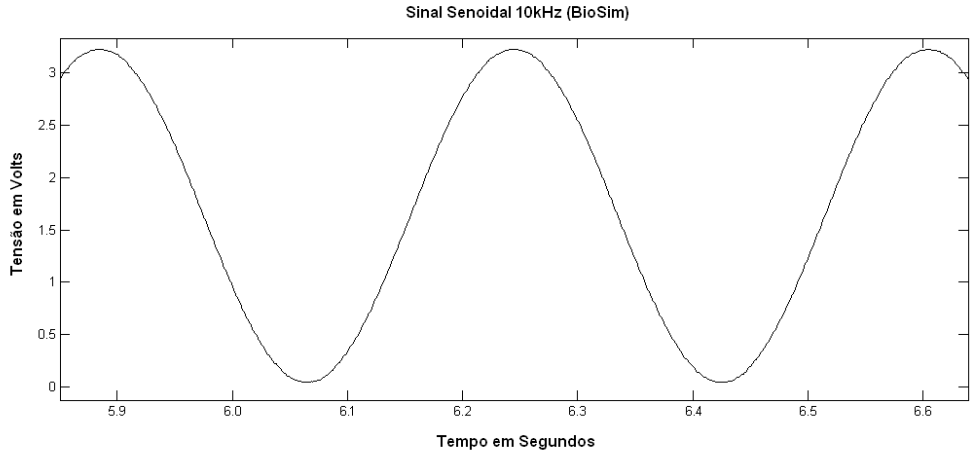
5.2 – Sinais Senoidais

Com a finalidade de analisar completamente a resposta do emulador de sinais biológicos, BioSim, foram geradas e reproduzidas formas de ondas conhecidas, como por exemplo a senoidal e a onda retangular.

Apesar de o equipamento ter sido desenvolvido para gerar sinais biológicos, ele é capaz de gerar qualquer tipo de onda, sinais biológicos ou não, e sua resposta na construção destes sinais é mostrada a seguir.

A Figura 45 apresenta uma onda senoidal com 3,3 volts de amplitude, amostrada em uma frequência de 10 kHz. Vale ressaltar, neste exemplo, que 10 kHz, não é a frequência da onda senoidal, e sim quantidade de amostras (pontos) por segundo, utilizadas pelo equipamento, para gerar este sinal.

Figura 45: Sinal senoidal de 10kHz



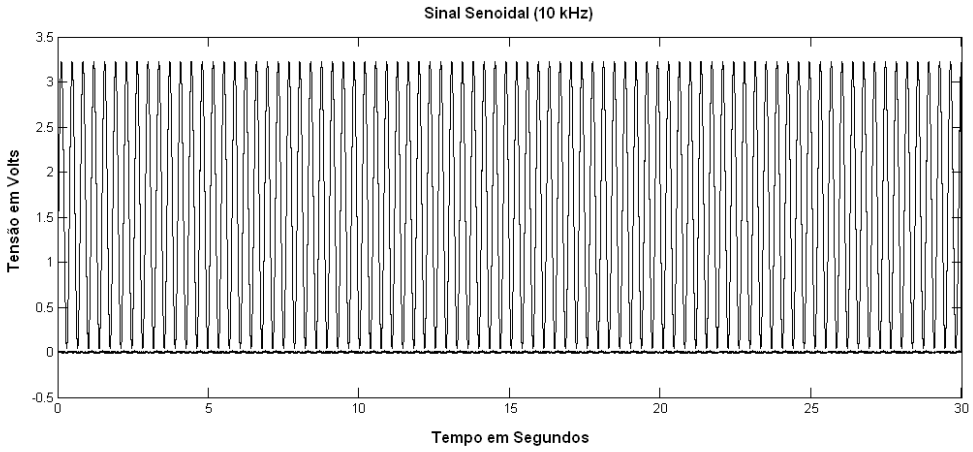
Fonte: Sistema BioSim desenvolvido pelo autor (2013).

De acordo com a Figura 45, percebe-se que o sinal coletado pelo conversor A/D USB-1208FS na saída do equipamento BioSim, deixa claro a qualidade da forma de onda, tanto em relação à estrutura do sinal, quanto em relação à amplitude do mesmo.

A Figura 46 ilustra o mesmo sinal da Figura 45, em uma perspectiva diferente, desta vez a Figura ilustra o sinal em um intervalo de trinta segundos.

Três Gráficos foram gerados simultaneamente e ilustrados na Figura 46 são respectivamente: o sinal gerado pelo computador, sobrepondo o sinal gerado pelo emulador, e a diferença entre eles, compondo o sinal de erro.

Figura 46: Sinal do computador, sinal do emulador e sinal de erro.

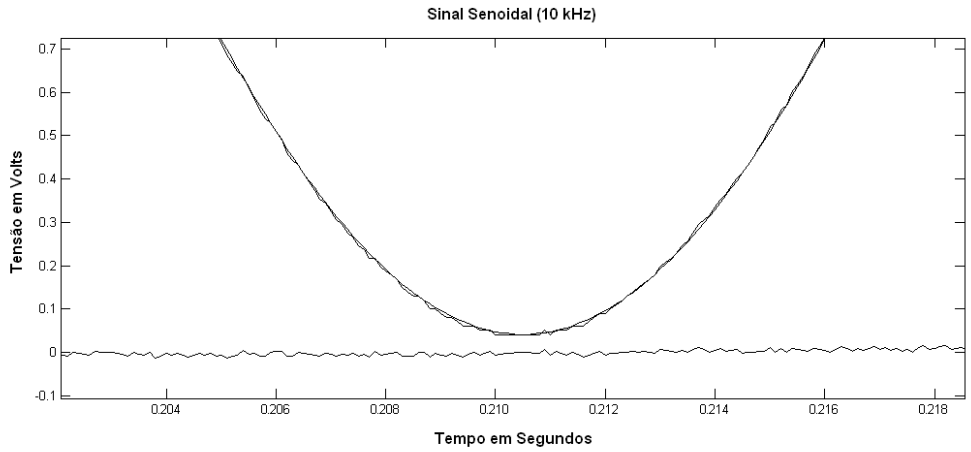


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Uma aproximação, feita a partir da mudança de escala da Figura 46, é mostrada na Figura 47. Nesta aproximação percebe-se a diferença entre os sinais digitais fornecidos pelo computador e os sinais analógicos coletados na saída do emulador.

Nota-se que o erro (diferença entre os sinais), apesar de ficar muito próximo de zero, existe, e é variável de acordo com a frequência de amostragem, como se observa nas análises estatísticas de erro efetuadas no final do capítulo.

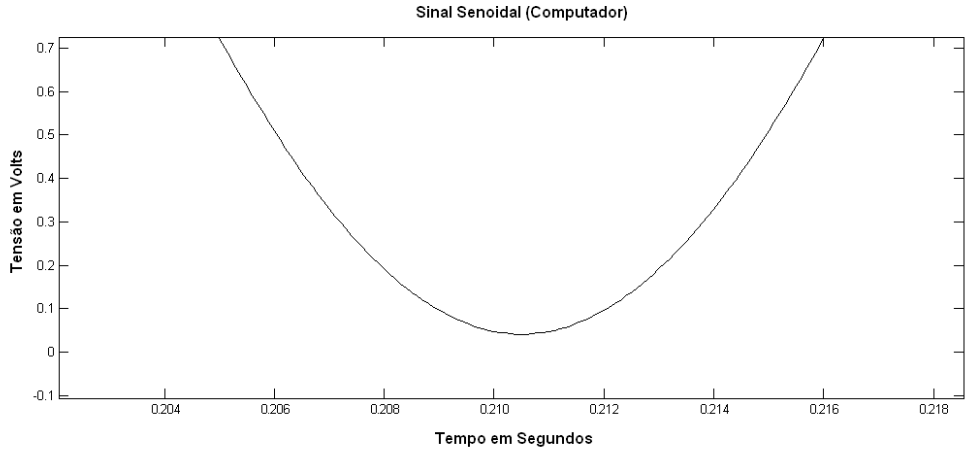
Figura 47: Sinal do computador, sinal do emulador e sinal de erro (alta resolução).



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

As Figuras 48, 49 e 50, representam as formas de onda ilustradas na imagem acima, plotados separadamente. A Figura abaixo ilustra a forma de onda construída com as amostras digitais fornecidas pelo computador.

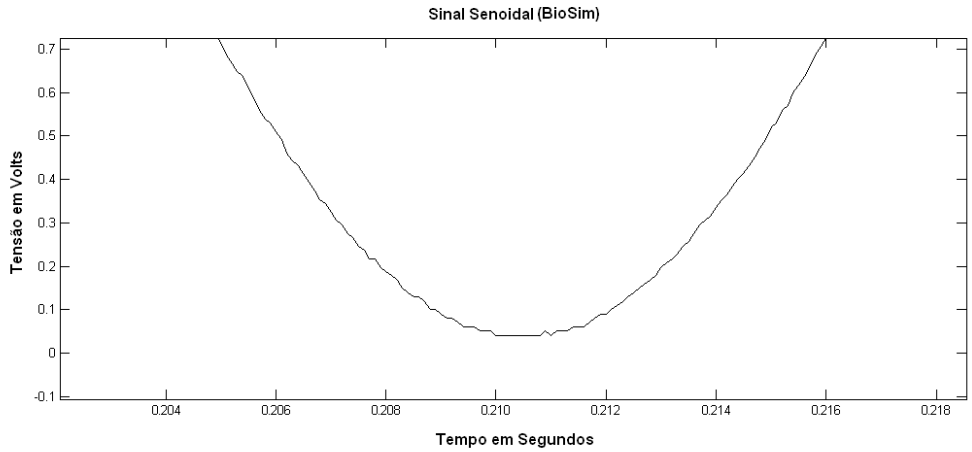
Figura 48: Forma de onda gerada pelo computador.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Já a Figura 49, ilustra a forma de onda construída com as amostras analógicas, provenientes do emulador de sinais biológicos, BioSim.

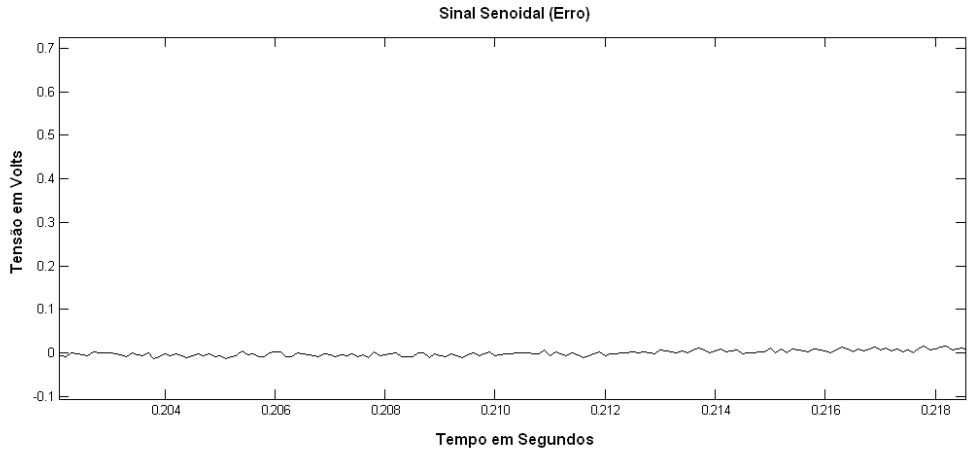
Figura 49: Forma de onda gerada pelo emulador.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O sinal de erro entre as amostras é ilustrado na Figura 50. Vale lembrar que o conversor D/A MCP4922 contido no equipamento, recebe um sinal digital referente a um determinado valor de tensão e reproduz esta tensão na saída do equipamento. Portanto a comparação é feita entre um sinal virtual, gerado pelo computador, e um sinal real gerado pelo equipamento e coletado em um de seus canais.

Figura 50: Sinal de erro entre as amostras.

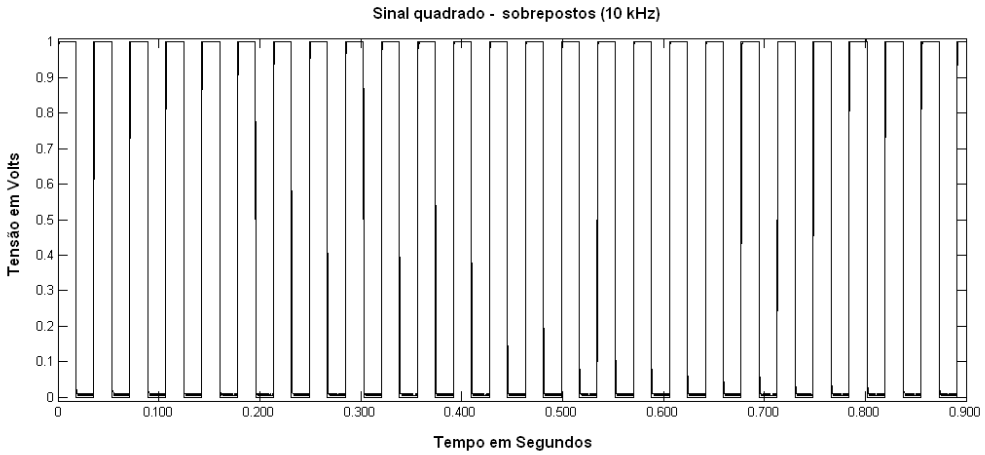


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

5.3 – Sinais Retangulares

Os sinais retangulares, representados na Figura 51, foram utilizados para observar a resposta do emulador quando se trata de sinais como interrupções bruscas.

Figura 51: Sinais Retangulares.



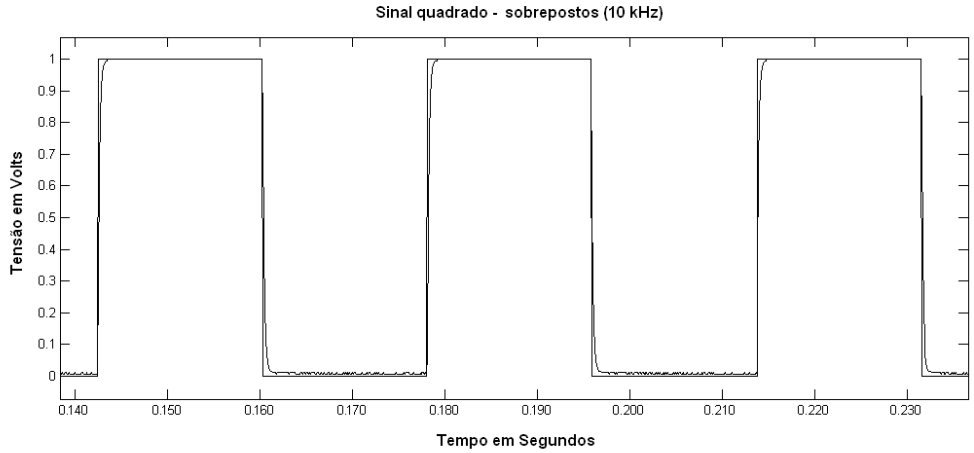
Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Os sinais retangulares gerados pelo emulador mostraram uma resposta diferente tanto dos sinais biológicos quanto dos sinais senoidais.

A Figura 52 ilustra a sobreposição das formas de onda digitais e analógicas, nela pode-se observar um atraso na construção do sinal, no final das bordas de subida e bordas de descida.

Esta pequena distorção, não influencia na frequência do sinal retangular, e de acordo com os testes, estas distorções aumentam ou diminuem de acordo com a frequência amostrada.

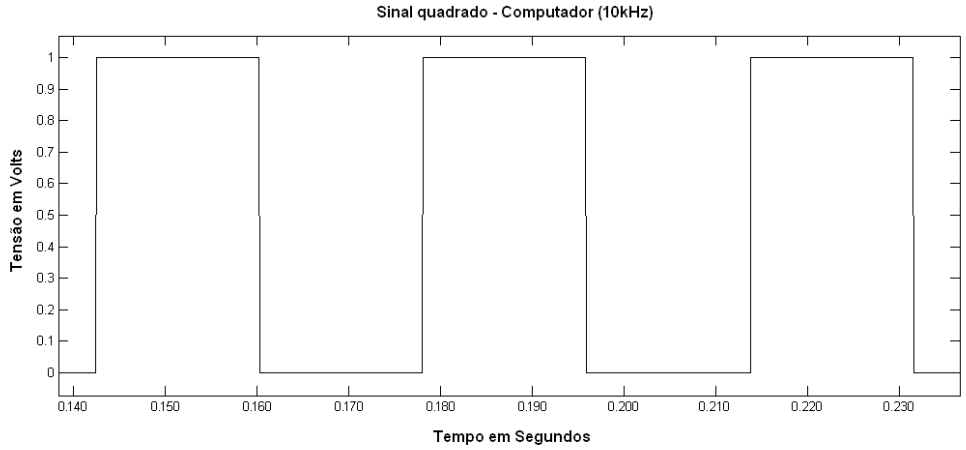
Figura 52: Sobreposição das formas de onda digitais e analógicas a 10 kHz.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A Figura 53 representa o sinal digital fornecido pelo computador.

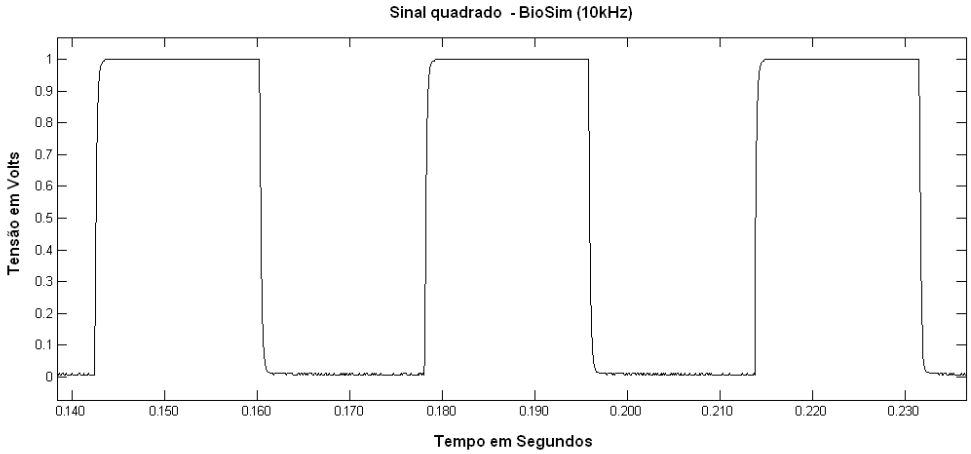
Figura 53: Sinal retangular digital fornecido pelo computador



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A Figura 54 a seguir representa a forma de onda analógica gerada pelo emulador de sinais. Nesta frequência nota-se na borda inferior (próximo do zero) um sinal de erro muito pequeno, mas com uma resposta diferente ao da borda superior (próximo de 1 volt).

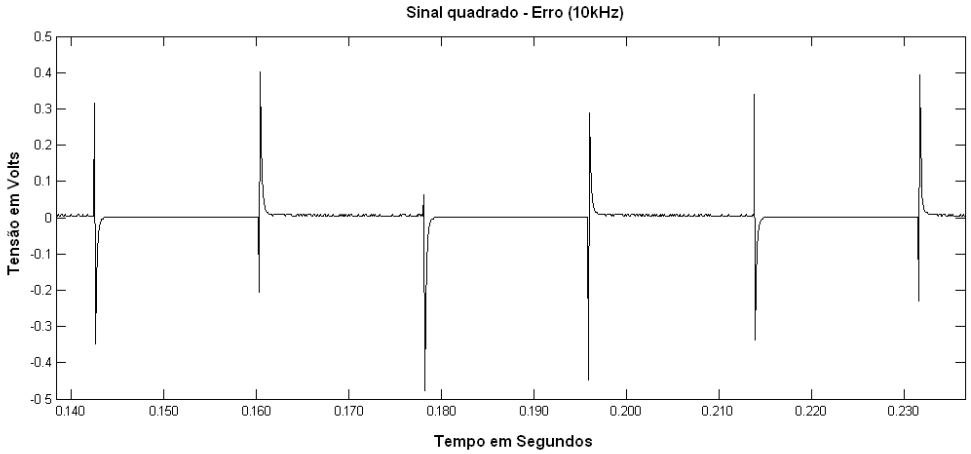
Figura 54: Forma de onda analógica gerada pelo emulador de sinais.



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

Analisando o sinal de erro na Figura 55, pode-se observar com mais facilidade a diferença da resposta da onda no pico (parte superior) e no vale (parte inferior). Esta resposta acontece porque o emulador trabalha com um filtro passivo de valor fixo na saída do sinal, e de acordo com a frequência utilizada para a construção da forma de onda, o sinal apresenta respostas distintas.

Figura 55: Sinal de erro onda retangular em 10 kHz

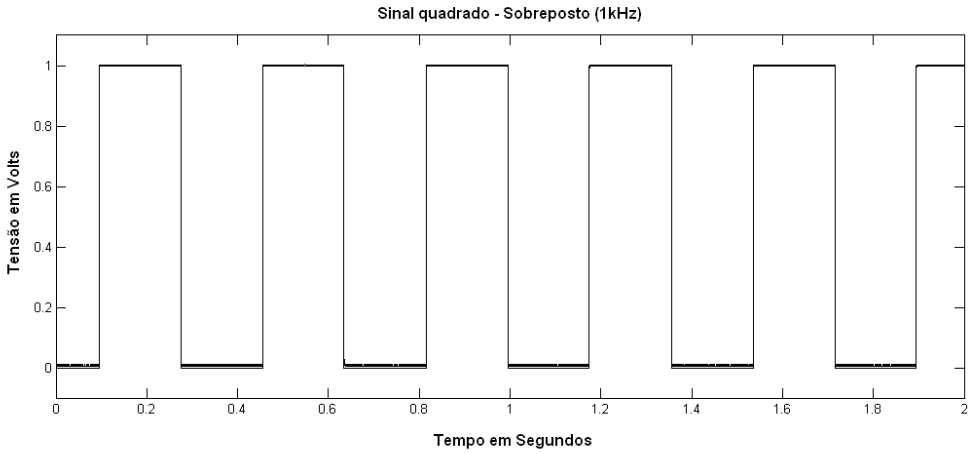


Fonte: Sistema BioSim desenvolvido pelo autor (2013).

O sinal analisado acima, Figura 55, possui uma frequência de 10 kHz, enquanto que o sinal apresentado a seguir na Figura 56, é amostrado com uma velocidade dez vezes inferior,

1 kHz, e como pode ser observado, a resposta do equipamento é diferenciado entre as duas frequências.

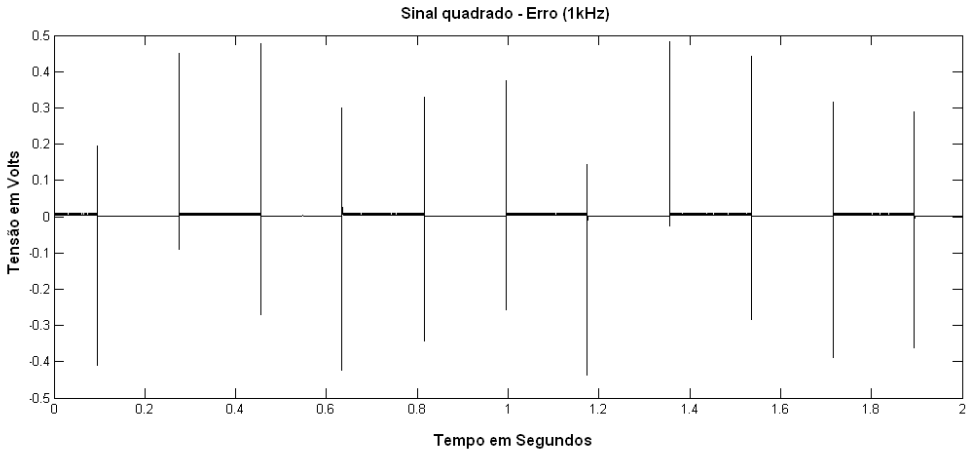
Figura 56: Sobreposição das formas de onda digitais e analógicas a 1 kHz



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

A comparação entre os sinais adquiridos com taxas diferentes fica evidente quando analisamos a Figura abaixo. Este é o gráfico de erro a 1 kHz, e como pode ser observado, apresenta uma menor distorção em relação ao gráfico ilustrado na Figura 45 (10 kHz).

Figura 57: Sinal de erro onda retangular em 1 kHz



Fonte: Sistema BioSim desenvolvido pelo autor (2013).

5.4 – Análise estatística do erro

Para que os resultados gerados pelo emulador possam ser estatisticamente comprovados, foi calculado o desvio padrão das amostras em frequências de 1 kHz, 5 kHz, 10 kHz, 15 kHz e 20 kHz.

A Tabela 1 contempla o valor do desvio padrão do erro em um sinal eletromiográfico, de amplitude máxima de 2,378 volts, em diferentes frequências.

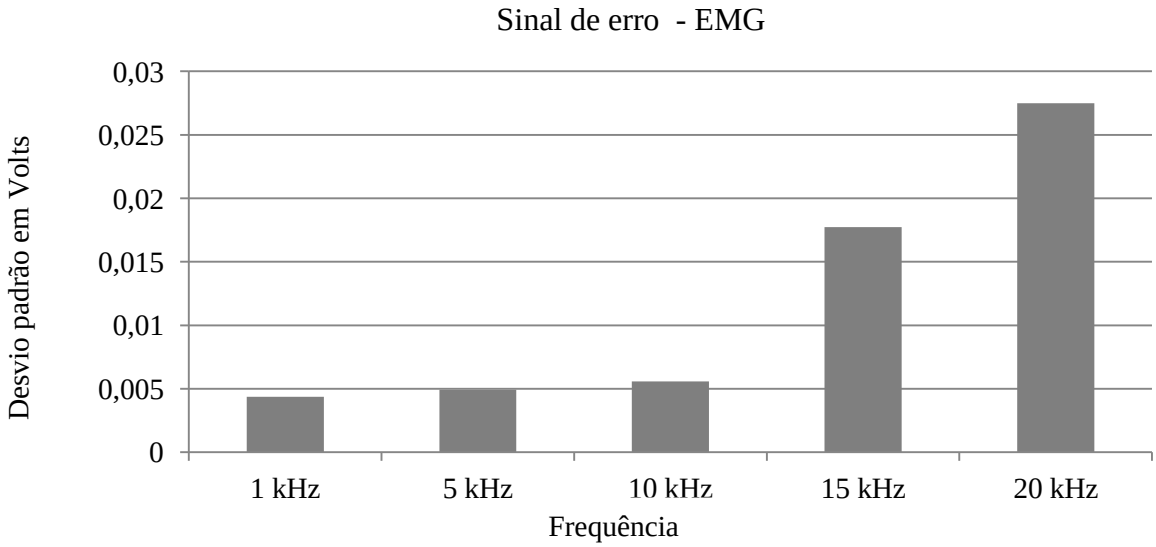
Tabela 1: Valor do desvio padrão do erro em um sinal eletromiográfico.

Sinal de Erro – EMG					
Frequência das amostras	1 kHz	5 kHz	10 kHz	15 kHz	20 kHz
Desvio Padrão – Volts	0,004376	0,004915	0,005584	0,017738	0,027495

Fonte: Próprio Autor (2013).

A representação gráfica destes valores pode ser vista a seguir, na Figura 58.

Figura 58: Representação gráfica do desvio padrão sinal eletromiográfico.



Fonte: Próprio Autor (2013).

Assim como no sinal eletromiográfico, o desvio padrão foi calculado para a onda retangular, com amplitude de 1 Volt e apresentou os resultados ilustrados na tabela 2.

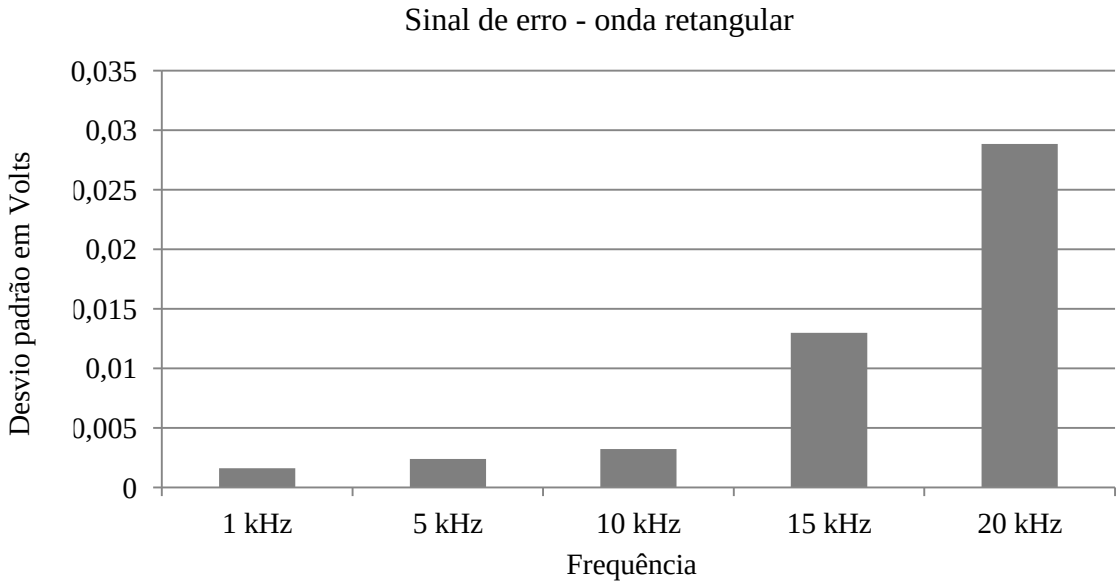
Tabela 2: Valor do desvio padrão do erro em um sinal retangular.

Sinal de erro – onda retangular					
Frequência das amostras	1 kHz	5 kHz	10 kHz	15 kHz	20 kHz
Desvio Padrão – Volts	0,001612	0,002388	0,003228	0,012996	0,028849

Fonte: Próprio Autor (2013).

A representação gráfica dos valores do desvio padrão do sinal de erro em uma onda retangular é mostrada na Figura a seguir.

Figura 59: Representação gráfica do desvio padrão do erro em um sinal retangular



Fonte: Próprio Autor (2013).

O desvio padrão do erro observado entre as amostras e em diferentes frequências, tanto do sinal eletromiográfico como na onda retangular, obtiveram resultados muito satisfatórios quando comparados com a amplitude dos sinais apresentados.

A resposta do sinal observado nas frequências mais altas, mesmo mostrando um desvio padrão bem maior em relação às frequências mais baixas, não compromete a qualidade do sinal gerado pelo emulador, e permite afirmar que este equipamento pode, perfeitamente, ser utilizado em substituição aos seres humanos quando houver necessidade de um novo processo de coleta de sinais biológicos e o propósito da pesquisa for o seu processamento.

Assim a qualidade do sinal emulado garante tanto a otimização da pesquisa, como a implementação e avaliação de diversas ferramentas de análise.

Capítulo 6

Considerações finais

O trabalho conseguiu, com êxito, alcançar o seu objetivo geral de desenvolver um hardware emulador de sinais biológicos para o ensino e aprendizagem na área de engenharia biomédica.

O emulador visa a substituição da etapa da coleta de sinais nas pesquisas e também irá auxiliar as novas tecnologias provenientes da utilização de sinais biológicos.

Durante o desenvolvimento da pesquisa, foi possível criar um sistema para conversão de arquivos contendo sinais digitais em sinais analógicos; projetar um sistema computacional, capaz de converter arquivos gerados por equipamentos de coleta de sinais biológicos em arquivos apropriados para leitura e emulação; além de desenvolver um equipamento, intitulado BioSim, capaz de gerar sinais analógicos provenientes de arquivos contendo sinais digitas; e realizar simulações para validar o sistema desenvolvido, que correspondem aos objetivos específicos propostos no início desta pesquisa.

Assim uma das principais contribuições do presente trabalho foi a construção de um emulador de sinais que possui um baixo custo, mais acessível do que os emuladores existentes no mercado.

Logo o emulador/simulador BioSim pode ser utilizado por instituições de ensino e pesquisa, não excluindo clínicas e hospitais que necessitam deste tipo de instrumento para o desenvolvimento de suas atividades.

6.1 – Trabalhos Futuros

Apesar da pesquisa ter apresentado ótimos resultados no desenvolvimento de um equipamento de emulação e simulação de sinais, deve-se levar em consideração o avanço tecnológico dos componentes utilizados para a sua construção.

O mercado atual dispõe de dispositivos mais baratos e mais eficientes do que os utilizados na pesquisa. A tecnologia dos microcontroladores progrediu muito com a popularização das placas Arduino, que contam hoje, por exemplo com o Arduino DUE, que utiliza um núcleo ARM de 32 bits, que possui um processamento em 84Mhz, 96 KBytes de SRAM, 512 KBytes de memória Flash para o código e um controlador de DMA, que pode aliviar o CPU de fazer tarefas de memória intensiva (ARDUINO, 2014).

A plataforma Arduino DUE, é mais poderosa e mais fácil de programar do que as utilizadas pela MAPLE, além de contar com um conversor D/A embutido de 12 bits, o que economizaria os conversores MCP4922 da Microchip utilizados no projeto.

O avanço tecnológico afetou não só os microcontroladores, como também os sistemas computacionais, permitindo a utilização de Tablets para o processamento do *software* BioSim como conversor de sinais.

Utilizando a base de dados e a experiência adquirida na pesquisa realizada a sujestão para trabalhos futuros é a criação de um sistema compacto, utilizando outra modalidade de computadores, neste caso, um tablet, interagindo com placas Arduino DUE, além da utilização de filtros ativos com regulação automática para as diversas frequências de amostragem desejadas.

O novo sistema deve contar também com ajustes de *offset* e amplificadores na saída dos canais para que o usuário consiga alterar a posição e as dimensões do sinal, diretamente no equipamento, sem a necessidade de gerar outros arquivos de dados via *software*.

Referências

AKAY, M. Biomedical Signal Processing, Academic Press, 1994. Challis & Kitney, Biomedical Signal Processing (Time-domain methods), Tutorial paper, Med. & Biol. & Comput. 1990, 28, 509-524

APLICAÇÃO PRÁTICA DA FISIOLOGIA DO EXERCÍCIO. Eletromiografia. Disponível em <http://aplicacoesfisioex.wordpress.com/>. Acessado em 15/08/2013.

ARDUINO. Disponível em <http://www.arduino.cc/> Acessado dia 15/02/2012.

BEMMEL, J.; MUSEN, Mark A. Sinal e Imagem. *Handbook of Medical Informatics* - Capítulo 9. Disponível em < http://im.med.up.pt/sinal_imagem/sinal_imagem.html> Acessado em 12/12/2013.

BERGER, Pedro de A.; NASCIMENTO, Francisco A. de O.; ROCHA, Adson F. da; CARVALHO, Joao L. A. A New Wavelet-Based Algorithm for Compression of Emg Signals. Proceedings of the 29th Annual International, Conference of the IEEE EMBS, Cité Internationale, Lyon, France, August 23-26, 2007

BERNARDES, Wellington Maycon Santos; ANDRADE, Adriano O.; MIOTTO, Guilherme Alessandri Alvim; SÁ, Ângela Abreu de Rosa de. Decomposição e análise de sinais eletromiográficos. Disponível em www.biolab.eletrica.ufu.br/admin/downloads/Ceel2007_062.pdf Acessado em 10/11/2013.

BIOLABOR MEDICINA DIAGNÓSTICA. Importância do Sono. Disponível em: < http://www.biolabor.com.br/informativo.php?ref=60>. Acessado em 12/12/2013.

BOTTER, A.; MERLETTI, R.; MINETTO, M. A. Pulse charge and not waveform affects M-wave properties during progressive motor unit activation. Disponível em <http://www.sciencedirect.com/science/article/pii/S1050641108000527>. Acessado em 14/09/2013.

BOUDREAUX-BARTELS, GF. Mixed mite-frequency signal transformations,The Transforms handbook (A.D. poularikas ed), CRC Press, Boca Raton(FL), 829-885, 1996.

CISOTTO, M. V.; Ferreira, M. G. V. Proposta De Simulador/Emulador Para Avaliação E Testes Do Processador Do Sistema Brasileiro De Coleta De Dados. 2º Workshop em Engenharia e Tecnologias Espaciais. 2011.

CIUCIU P, ABRY P, RABRAIT C, WENDT H, ROCHE A. Leader-Based Multi fractal Analysis for Evi fMRI time series: Ongoing vs Task-Related Brain Activity.2007 ISBI : 404-407.

DATAHOMINIS. Disponível em < http://www.datahominis.com.br/ >. Acessado em 28/12/2013.

DING H, HUANG Z, SONG Z, YAN Y. Hilbert—Huang transform based signal analysis for the characterization of gas—liquid two phase flow. 2007. Flow measurement and instrumentation:18, 37-46.

EYKERN, L. A van.; GEISLER, H.C.; GRAMSBERGEN, A. A new technique for simultaneously recording EMG and movements in experimental animals.Disponível em <http://www.sciencedirect.com/science/article/pii/S1385299X0000043X>. Acessado em 14/09/2013.

FARIAS, T. M. T. Sistema Embarcado para um Monitor Holter que Utiliza o Modelo PPM na Compressão de Sinais ECG. Universidade Federal da Paraíba, Centro de Ciências Exatas e da Natureza, Departamento de Informática, Programa de Pós-Graduação em Informática. João Pessoa, 2010.

FLUKE BIOMEDICAL. Disponível em <http://pt.flukebiomedical.com/Biomedical/usen/Products/Patient-Simulators-Fast.htm>. Acessado em 13/09/2013.

GEDDES, L. A.; BAKER, L. E. Principles of Applied Biomedical Instrumentation, 3rd ed., New York: Wiley, 1989.).

GIORGI, F. S.; PERINI, D.; MAESTRI, M.; GUIDA, M.; PIZZANELLI, C.; CASERTA, A.; IUDICE, A.; BONANNI, E. Usefulness of a simple sleep-deprived EEG protocol for epilepsy diagnosis in *de novo* subjects. Disponível em <http://www.sciencedirect.com/science/article/pii/S1388245713006767>. Acessado em 14/09/2013.

HLAWATSCH F, BOUDREAUX-BARTELS GF. Linear and quadratic time-frequency signal representations,1992. IEEE signal proc. Mag. 9, 21-27.

HONGBO X, WANG Z. Mean frequency derived Hilbert-Huang transform with application to fatigue EMG signal analysis, 2007; Computer methods and programs in biomedicine (82);114:120.

HUANG N, SHEN Z, LONG S, WU M, SHIH H, ZHENG Q, YEN N, TUNG C, LIU H. The Empirical mode decomposition and the Hilbert sepctrum for nonlinear and non-sationary time series analysis, 1998, Proc R. Soc. Lond. A, 454, 903-995.

KARTHIK, R. ECG Simulation Using MATLAB – Principle of Fourier Series. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/10858> Acesso em: 13/09/2013.

LEAFLABS. Disponível em <http://leaflabs.com/docs/index.html> Acessado dia 23/05/2012.

LI, X.; JEFFERYS, J.; FOX J.; YAO X. Neuronal Population Oscillation Analysis of Rat Hippocampus during Epileptic Seizure, 2008; Neural Networks, 21; 1105-1111.

MATHWORKS. Disponível em <http://www.mathworks.com/products/matlab/>. Acessado em 15/11/2013.

MCCDAQ. Disponível em <www.mccdaq.com>. Acessado em 15/11/2013.

MERLETTI, R.; PARKER, P.“Electromyography: Engineering and Noninvasive Applications,” Hoboken, NJ: John Wiley & Sons, 2004

MICROCHIP. Disponível em <www.microchip.com/wwwproducts>. Acessado em 13/08/2013.

PINTO, Rui. Disponível em <http://ibeb.fc.ul.pt/downloads/TESE_RuiPinto.pdf.> Acessado em 15/11/2013.

RAO R, HSU E-C. Hilbert-Huang Transform Analysis Of Hydrological And Environmental Time Series. 2008. Water Science and Technology Library-Vol. 60.

RIEGER, Josef. EEG signal analysis . 2006. Disponível em: <http://gerstner.felk.cvut.cz/biolab/X33BMI/Rieger_minimum.pdf>Acessado em 12/12/2013.

ROCHA, A. F da; CARVALHO, J. L. A.; BERGER, P. A.; NASCIMENTO, F. A. O. Processamento de Sinais Biológicos. In: Lourdes Mattos Brasil. (Org.). Informática em Saúde. 1ed. Brasília: Universa/Eduel, 2008, v. 1, p. 397-426

SALISBURY J, SUN Y. Rapid screening test for sleep apnea using a nonlinear and nonstationary signal processing technique, 2007. Medical Engineering and Physics, 29(3):336-43.

SECRETARIA DA SAÚDE DO ESTADO DA BAHIA. Cihdott do HGPV recebe eletroencefalograma. 2010. Disponível em: < http://www2.saude.ba.gov.br/hgpv/noticia_009.2010-CIHDOTT_HGPV_RECEBE_ELETROENCEFALOGRAMA.htm>. Acessado em: 27/12/2013.

SPES MEDICA BRASIL. Disponível em < http://spesmedicabrasil.com.br/>. Acessado em 11/12/2013.

TANG J, ZOU Q, TANG Y, LIU B, ZHANG Xiao-kai. Hilbert-Huang Transform for ECG De-Noising, 2007.

_____. Bioinformatics and Biomedical Engineering; 664-667. Bedrosian E; A Product theorem for Hilbert Transform, 1963; Proceedings of the IEEE, 51, 868-869.

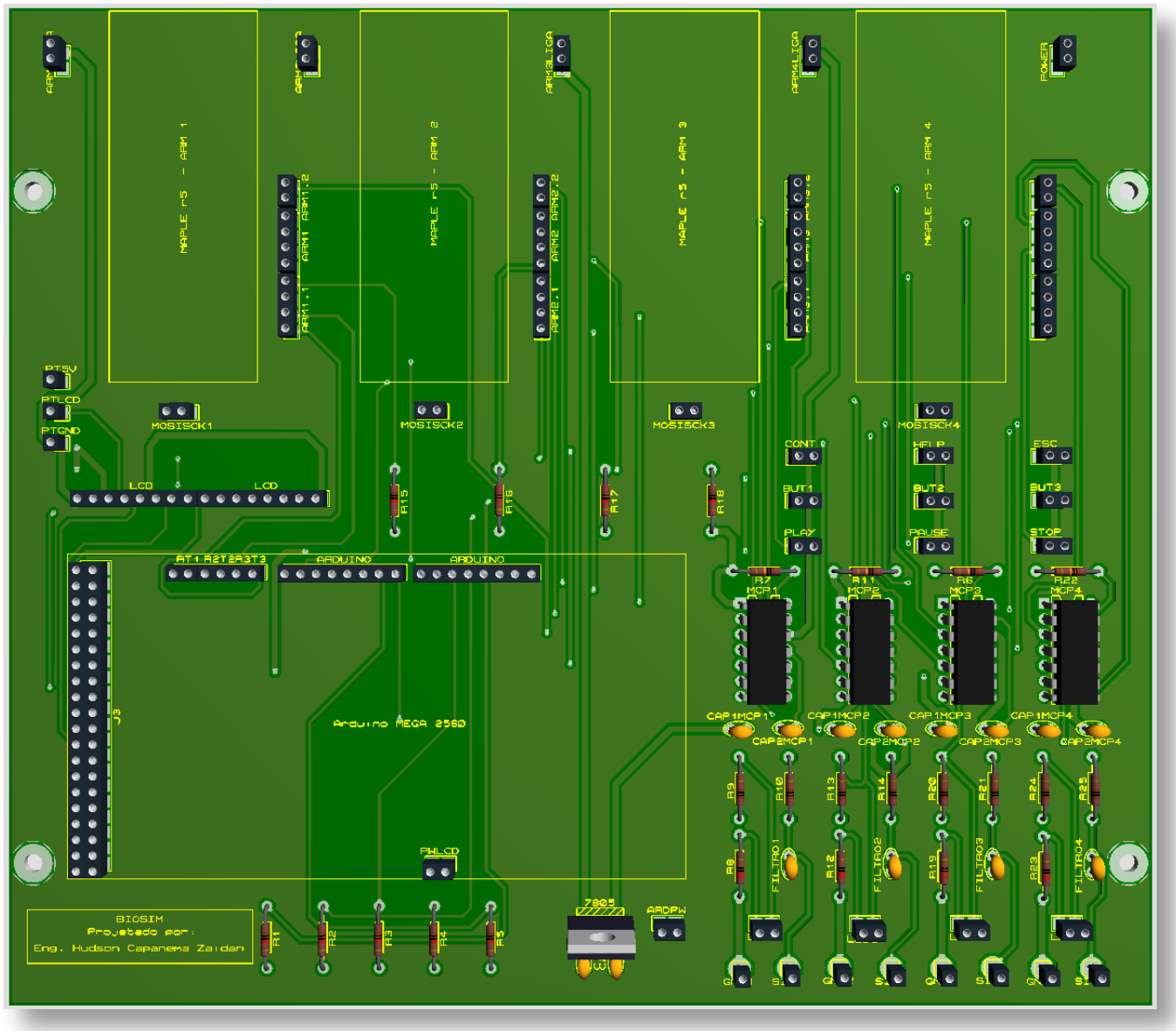
THAKOR, N. V. Biopotentials and Electrophysiology Measurement. In The Measurements, Instrumentation and Sensors Handbook. Ed. Webster, JG. CRC Press. 1999.

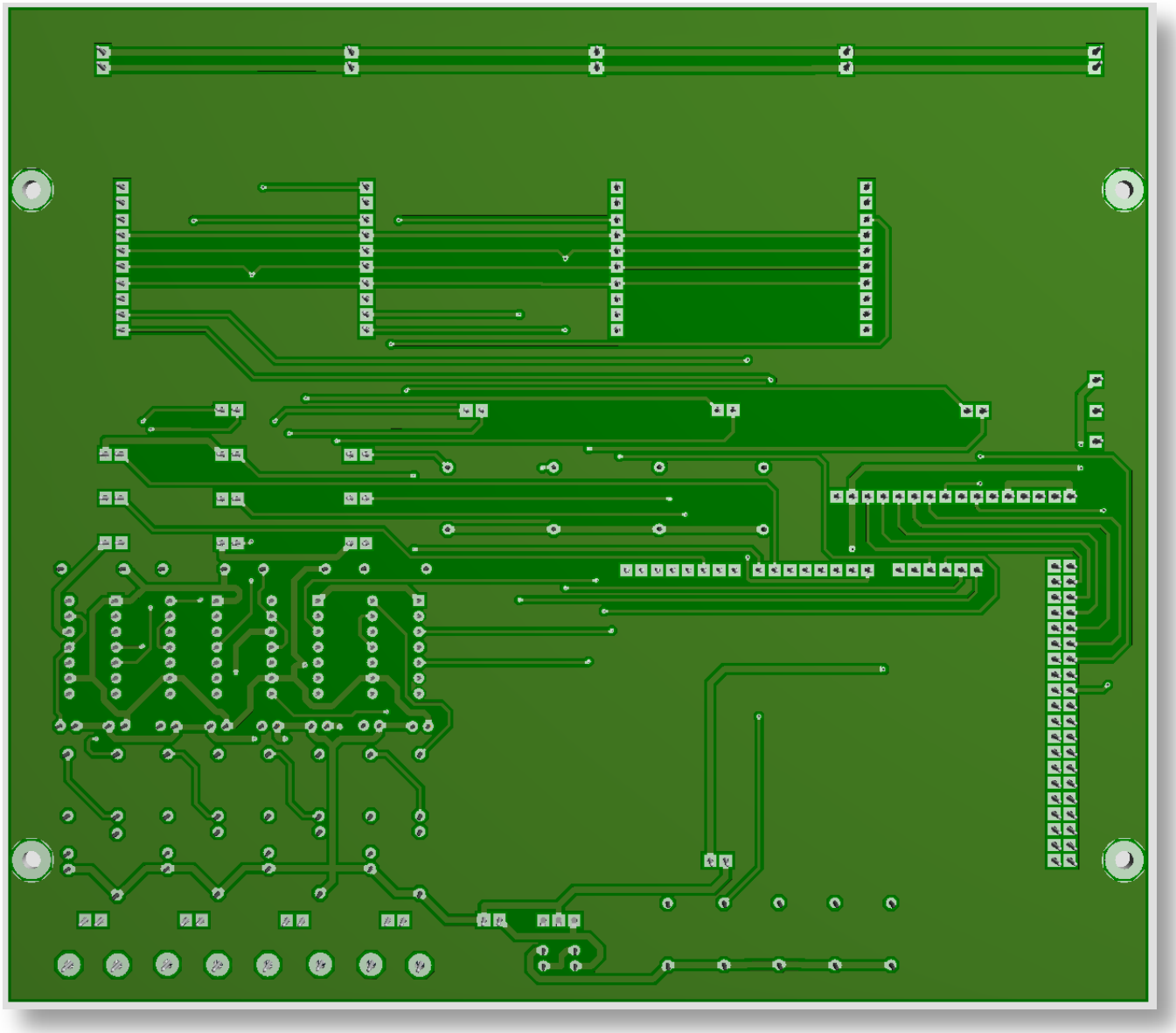
THE UNIVERSITY OF TENNESSEE, COLLEGE OF VETERINARY MEDICINE. Outpatient Electrodiagnostics. Disponível em <http://www.vet.utk.edu/utmri/outpatient-electrodiagnostics.php>. Acessado em 28/12/2013.

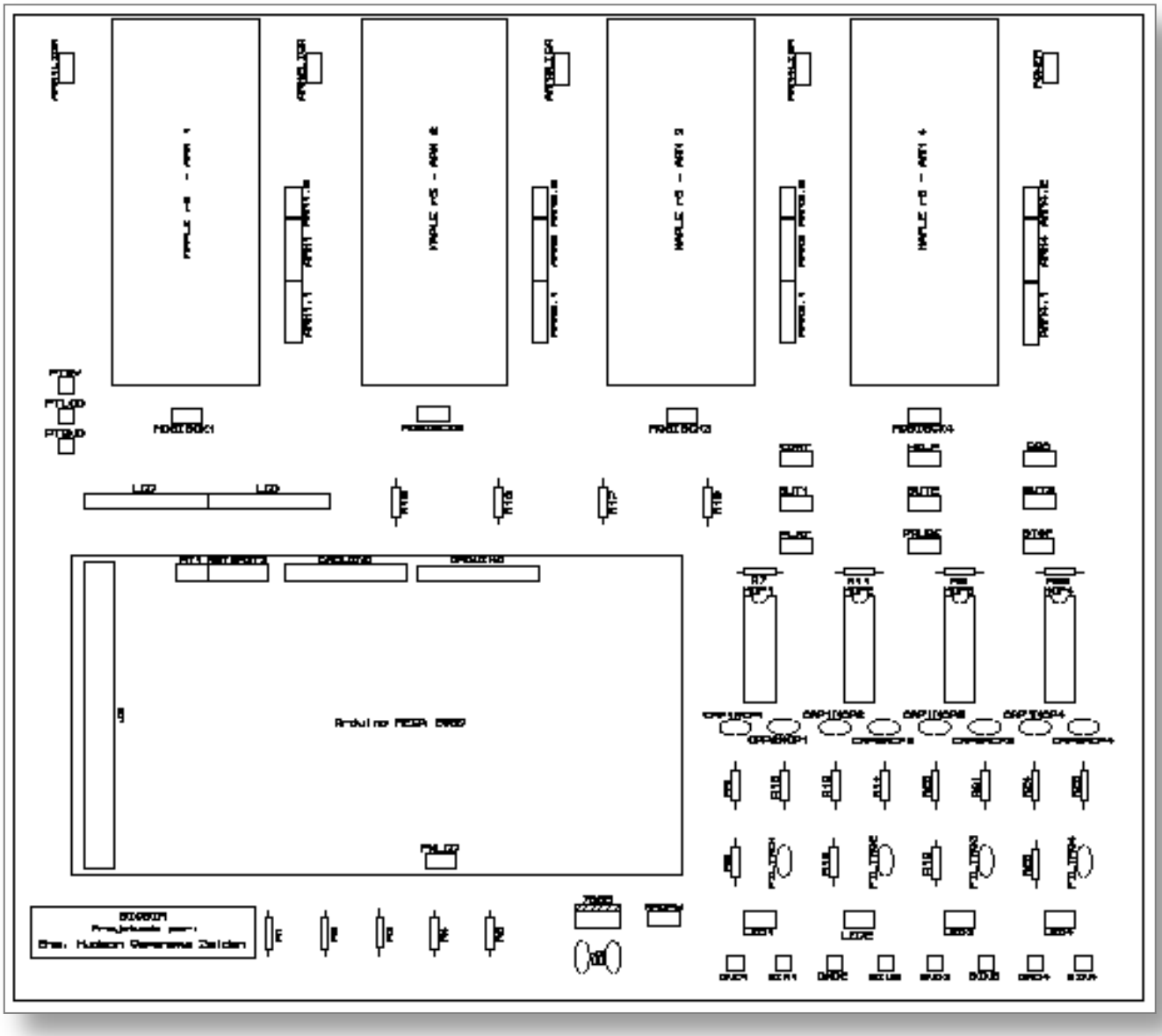
ZHANG, X; LAI, K. K., WANG, S.Y. A new Approach for Crude Oil Price Analysis Based on. Empirical Mode Decomposition, 2008; Energy Economics 30(3), 905 – 918).

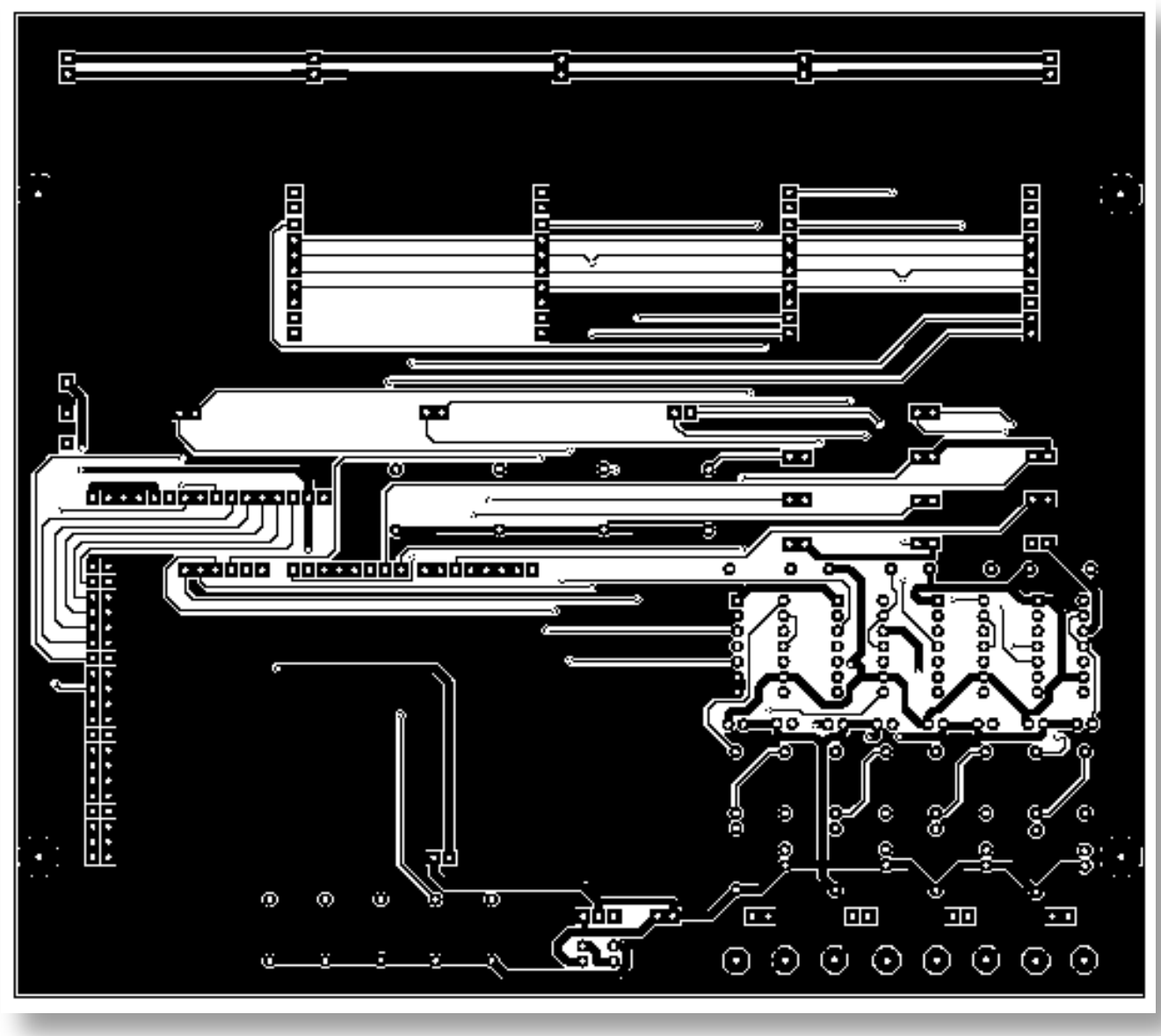
ANEXO I

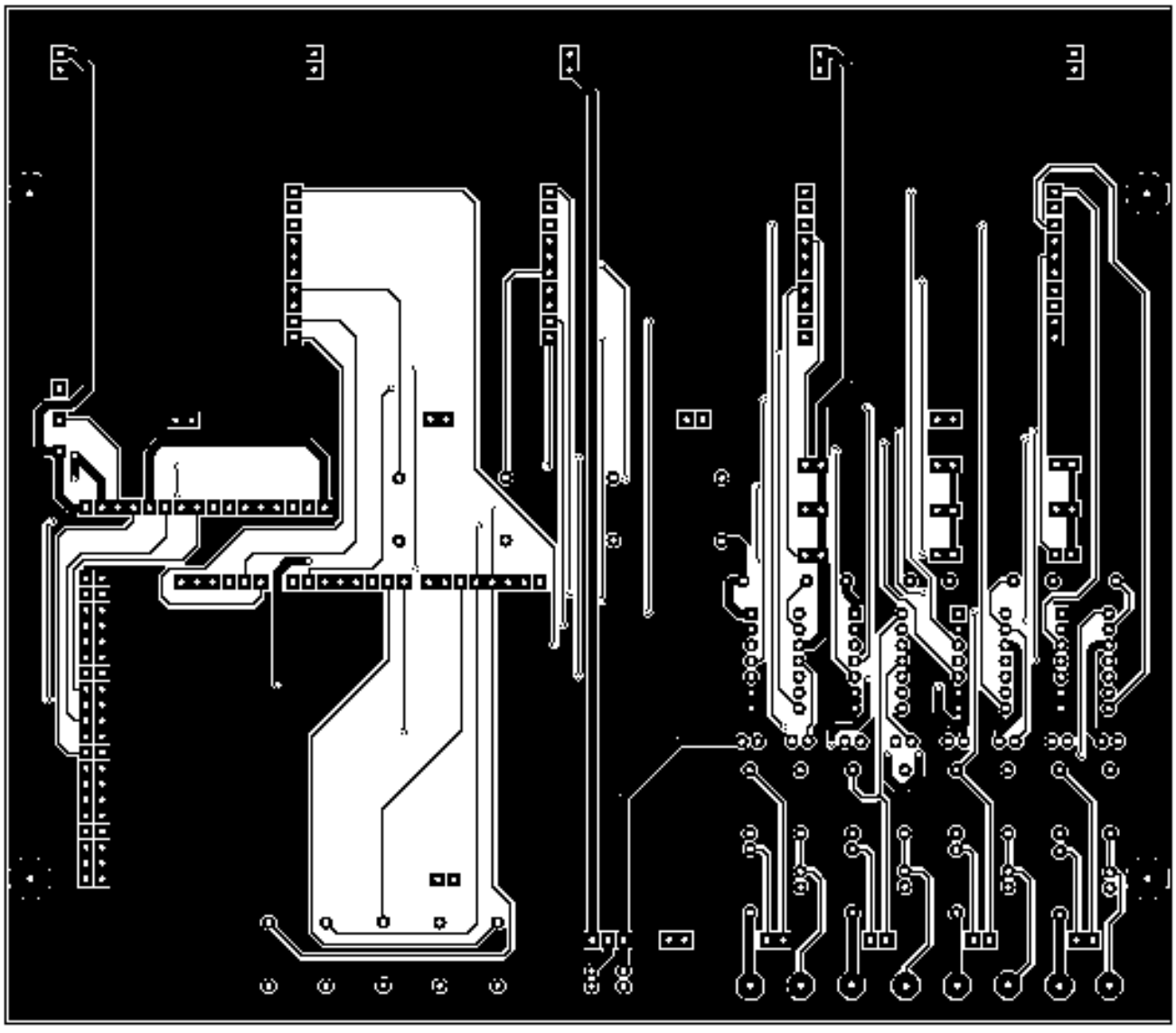
PLACA DE CIRCUITO IMPRESSO






Arduino MEGA 2560





ANEXO II

LINHAS DE CÓDIGO DO SOFTWARE BIOSIM

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.IO;
using System.Media;
using System.Globalization;
using System.Threading;
using System.Diagnostics;



namespace Projeto_emulador
{
   public partial class Form1 : Form
   {
      string leitura;
      string leitura2;
      string[] vet_leitura = new string[50];
      int linhas = 0;
      string[,] matriz;
      private Thread trd;

      RollingPointPairList listaPlotagem1 = new RollingPointPairList(500000);
      RollingPointPairList listaPlotagem2 = new RollingPointPairList(500000);
      RollingPointPairList listaPlotagem3 = new RollingPointPairList(500000);
      RollingPointPairList listaPlotagem4 = new RollingPointPairList(500000);
      char[] delimiterChars = { '\t' };


      public Form1()
      {
         InitializeComponent();
      }

      private void btn_Abrir_Click(object sender, EventArgs e)
      {
         try
         {

            openFileDialog1.Filter = "Text files (*.txt)|*.txt";
```

```
openFileDialog1.Title = "Open an Text File";
openFileDialog1.FileName = "";
DialogResult resultado = openFileDialog1.ShowDialog();
if (!(resultado == System.Windows.Forms.DialogResult.Cancel))
{
   int a = 0;
   linhas = 0;

   System.IO.StreamReader Vleitor = new System.IO.StreamReader(openFileDialog1.FileName);
   labelFileSelect.Text = " Arquivo Selecionado: " + openFileDialog1.FileName;
   if (Form1.ActiveForm.Text == "BioSim - File Converter")
   {
      labelFileSelect.Text = "    Selected File:   " + openFileDialog1.FileName;
   }
   labelFileSelect.Visible = true;


   while (!Vleitor.EndOfStream)
   {
      linhas++;
      leitura = Vleitor.ReadLine();
      if (a == 0)
      {
         leitura = Vleitor.ReadLine();
         vet_leitura = leitura.Split(delimiterChars);
         a = 1;
      }
   }
   Vleitor.Close();
   Vleitor.Dispose();
}

System.IO.StreamReader Vleitor2 = new System.IO.StreamReader(openFileDialog1.FileName);

while (!Vleitor2.EndOfStream)
{
   matriz = new string[linhas + 1, vet_leitura.Length];

   for (int j = 0; j <= linhas; j++)
   {
      leitura2 = Vleitor2.ReadLine();

      vet_leitura = leitura2.Split(delimiterChars);

      for (int k = 0; k < vet_leitura.Length; k++)
      {
         matriz[j, k] = vet_leitura[k];
      }
   }

}
Vleitor2.Close();
```

```
        for (int j = 0; j < vet_leitura.Length - 1; j++)
        {
            CB_Canal1.Items.Add(j + 1);
            CB_Canal2.Items.Add(j + 1);
            CB_Canal3.Items.Add(j + 1);
            CB_Canal4.Items.Add(j + 1);
        }

        picturedown2.Visible = true;
        picturedown1.Visible = true;
        picturedown3.Visible = true;
        picturedown4.Visible = true;
        CB_Canal1.BackColor = Color.White;
        CB_Canal2.BackColor = Color.White;
        CB_Canal3.BackColor = Color.White;
        CB_Canal4.BackColor = Color.White;
        SystemSounds.Exclamation.Play();
        btn_Abrir.Enabled = false;
        btn_Converter.Enabled = true;
        btn_Converter.Focus();
        progressBar_Ch1.Value = 0;
        progressBar_Ch2.Value = 0;
        progressBar_Ch3.Value = 0;
        progressBar_Ch4.Value = 0;
        progressBar_save1.Value = 0;
        progressBar_save2.Value = 0;
        progressBar_save3.Value = 0;
        progressBar_save4.Value = 0;

        listaPlotagem1.Clear();
        zed1.GraphPane.CurveList.Clear();
        listaPlotagem2.Clear();
        zed2.GraphPane.CurveList.Clear();
        listaPlotagem3.Clear();
        zed3.GraphPane.CurveList.Clear();
        listaPlotagem4.Clear();
        zed4.GraphPane.CurveList.Clear();
        zed1.Refresh();
        zed2.Refresh();
        zed3.Refresh();
        zed4.Refresh();

    }
    catch
    {
        MessageBox.Show("Este não é um arquivo válido", "          Atenção");
    }
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}
```

```
    private void Form1_Load(object sender, EventArgs e)
    {
        btn_Abrir.Select();

        zed1.GraphPane.Title.FontSpec.Size = 22;
        zed1.GraphPane.Title.Text = "   Data1";
        zed1.GraphPane.XAxis.Title.FontSpec.Size = 22;
        zed1.GraphPane.XAxis.Title.Text = "Segundos";
        zed1.GraphPane.YAxis.Title.FontSpec.Size = 22;
        zed1.GraphPane.YAxis.Title.Text = "Sinal Biológico";

        zed1.GraphPane.XAxis.Scale.FontSpec.Size = 20;
        zed1.GraphPane.XAxis.Scale.Min = 0;
        zed1.GraphPane.XAxis.Scale.Max = 30;
        zed1.GraphPane.XAxis.Scale.MinorStep = 0.05;
        zed1.GraphPane.XAxis.Scale.MajorStep = 1;

        zed1.GraphPane.YAxis.Scale.FontSpec.Size = 20;
        zed1.GraphPane.YAxis.Scale.Min = Convert.ToSingle(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
        zed1.GraphPane.YAxis.Scale.Max = Convert.ToSingle(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
        zed1.GraphPane.YAxis.MajorGrid.IsZeroLine = false;

        zed1.GraphPane.Fill = new Fill(Color.Gainsboro);
        zed1.GraphPane.Chart.Fill = new Fill(Color.Black);

        zed1.AxisChange();
        zed1.Invalidate();
        zed1.Refresh();

        /*****************************************************************/

        zed2.GraphPane.Title.FontSpec.Size = 22;
        zed2.GraphPane.Title.Text = "   Data2";
        zed2.GraphPane.XAxis.Title.FontSpec.Size = 22;
        zed2.GraphPane.XAxis.Title.Text = "Segundos";
        zed2.GraphPane.YAxis.Title.FontSpec.Size = 22;
        zed2.GraphPane.YAxis.Title.Text = "Sinal Biológico";

        zed2.GraphPane.XAxis.Scale.FontSpec.Size = 20;
        zed2.GraphPane.XAxis.Scale.Min = 0;
        zed2.GraphPane.XAxis.Scale.Max = 30;
        zed2.GraphPane.XAxis.Scale.MinorStep = 0.05;
        zed2.GraphPane.XAxis.Scale.MajorStep = 1;

        zed2.GraphPane.YAxis.Scale.FontSpec.Size = 20;
        zed2.GraphPane.YAxis.Scale.Min = Convert.ToSingle(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
        zed2.GraphPane.YAxis.Scale.Max = Convert.ToSingle(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
        zed2.GraphPane.YAxis.MajorGrid.IsZeroLine = false;
```

```
        zed2.GraphPane.Fill = new Fill(Color.Gainsboro);
        zed2.GraphPane.Chart.Fill = new Fill(Color.Black);

        zed2.AxisChange();
        zed2.Invalidate();
        zed2.Refresh();

        /*************************************************************************/

        zed3.GraphPane.Title.FontSpec.Size = 22;
        zed3.GraphPane.Title.Text = "   Data3";
        zed3.GraphPane.XAxis.Title.FontSpec.Size = 22;
        zed3.GraphPane.XAxis.Title.Text = "Segundos";
        zed3.GraphPane.YAxis.Title.FontSpec.Size = 22;
        zed3.GraphPane.YAxis.Title.Text = "Sinal Biológico";

        zed3.GraphPane.XAxis.Scale.FontSpec.Size = 20;
        zed3.GraphPane.XAxis.Scale.Min = 0;
        zed3.GraphPane.XAxis.Scale.Max = 30;
        zed3.GraphPane.XAxis.Scale.MinorStep = 0.05;
        zed3.GraphPane.XAxis.Scale.MajorStep = 1;

        zed3.GraphPane.YAxis.Scale.FontSpec.Size = 20;
        zed3.GraphPane.YAxis.Scale.Min = Convert.ToSingle(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
        zed3.GraphPane.YAxis.Scale.Max = Convert.ToSingle(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
        zed3.GraphPane.YAxis.MajorGrid.IsZeroLine = false;

        zed3.GraphPane.Fill = new Fill(Color.Gainsboro);
        zed3.GraphPane.Chart.Fill = new Fill(Color.Black);

        zed3.AxisChange();
        zed3.Invalidate();
        zed3.Refresh();

        /*************************************************************************/

        zed4.GraphPane.Title.FontSpec.Size = 22;
        zed4.GraphPane.Title.Text = "   Data4";
        zed4.GraphPane.XAxis.Title.FontSpec.Size = 22;
        zed4.GraphPane.XAxis.Title.Text = "Segundos";
        zed4.GraphPane.YAxis.Title.FontSpec.Size = 22;
        zed4.GraphPane.YAxis.Title.Text = "Sinal Biológico";

        zed4.GraphPane.XAxis.Scale.FontSpec.Size = 20;
        zed4.GraphPane.XAxis.Scale.Min = 0;
        zed4.GraphPane.XAxis.Scale.Max = 30;
        zed4.GraphPane.XAxis.Scale.MinorStep = 0.05;
        zed4.GraphPane.XAxis.Scale.MajorStep = 1;

        zed4.GraphPane.YAxis.Scale.FontSpec.Size = 20;
```

```
        zed4.GraphPane.YAxis.Scale.Min = Convert.ToSingle(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
        zed4.GraphPane.YAxis.Scale.Max = Convert.ToSingle(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
        zed4.GraphPane.YAxis.MajorGrid.IsZeroLine = false;

        zed4.GraphPane.Fill = new Fill(Color.Gainsboro);
        zed4.GraphPane.Chart.Fill = new Fill(Color.Black);

        zed4.AxisChange();
        zed4.Invalidate();
        zed4.Refresh();

    }

    private void btn_Converter_Click(object sender, EventArgs e)
    {

        //Cleaning Zedgraph control
        listaPlotagem1.Clear();
        zed1.GraphPane.CurveList.Clear();
        listaPlotagem2.Clear();
        zed2.GraphPane.CurveList.Clear();
        listaPlotagem3.Clear();
        zed3.GraphPane.CurveList.Clear();
        listaPlotagem4.Clear();
        zed4.GraphPane.CurveList.Clear();
        trackBar2.Enabled = false;
        btn_bra.Enabled = false;
        btn_uk.Enabled = false;


        //Starting thread for data conversion
        this.trd = new Thread(new ThreadStart(this.ThreadConvertData));
        trd.IsBackground = true;
        trd.Start();
    }

    public delegate string GetStringFromObject(int chan);
    public string strObjMethod(int chan)
    {
        string returnStr = "";

        switch (chan)
        {
            case 1:
                returnStr = CB_Canal1.Text;
                break;
            case 2:
                returnStr = CB_Canal2.Text;
                break;
            case 3:
                returnStr = CB_Canal3.Text;
```

```
            break;
        case 4:
            returnStr = CB_Canal4.Text;
            break;
    }

    return returnStr;
}


public delegate void ZedGraphCode();
public void ZedGraphCodeMethod()
{
    zed1.GraphPane.AddCurve("", listaPlotagem1, Color.LightGreen, SymbolType.None);
    zed1.Invalidate();
    zed1.Refresh();

    zed2.GraphPane.AddCurve("", listaPlotagem2, Color.LightPink, SymbolType.None);
    zed2.Invalidate();
    zed2.Refresh();

    zed3.GraphPane.AddCurve("", listaPlotagem3, Color.SkyBlue, SymbolType.None);
    zed3.Invalidate();
    zed3.Refresh();

    zed4.GraphPane.AddCurve("", listaPlotagem4, Color.Yellow, SymbolType.None);
    zed4.Invalidate();
    zed4.Refresh();

    CB_Canal1.Items.Clear();
    CB_Canal2.Items.Clear();
    CB_Canal3.Items.Clear();
    CB_Canal4.Items.Clear();

    CB_Canal1.Text = "Selecione o Canal";
    CB_Canal2.Text = "Selecione o Canal";
    CB_Canal3.Text = "Selecione o Canal";
    CB_Canal4.Text = "Selecione o Canal";

    picturedown1.Visible = false;
    picturedown2.Visible = false;
    picturedown3.Visible = false;
    picturedown4.Visible = false;

    btn_Abrir.Enabled = true;
    btn_Converter.Enabled = false;
    btn_salvar.Enabled = true;
    btn_salvar.Focus();
    trackBar2.Enabled = true;
    btn_bra.Enabled = true;
    btn_uk.Enabled = true;

}
```

```
        private void ThreadConvertData()
        {

            try
            {
                double xmin = Convert.ToDouble(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
                double xmax = Convert.ToDouble(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
                double result1, result2, result3, result4;
                double eixoy1, eixoy2, eixoy3, eixoy4;
                int resultfinal1, resultfinal2, resultfinal3, resultfinal4;
                string plotter1, plotter2, plotter3, plotter4;

                double time = Convert.ToDouble(matriz[1, 0],
System.Globalization.CultureInfo.InvariantCulture);


                textBoxInterv.Invoke(new Action(() =>
                {
                    textBoxInterv.Text = Convert.ToString(time,
System.Globalization.CultureInfo.InvariantCulture);
                }));


                double totaltime = Convert.ToSingle(linhas) * (time);
                decimal totaltimedec = Convert.ToDecimal(totaltime);
                zed1.GraphPane.XAxis.Scale.Max = Convert.ToInt32(totaltimedec);
                zed2.GraphPane.XAxis.Scale.Max = Convert.ToInt32(totaltimedec);
                zed3.GraphPane.XAxis.Scale.Max = Convert.ToInt32(totaltimedec);
                zed4.GraphPane.XAxis.Scale.Max = Convert.ToInt32(totaltimedec);
                decimal Freq = Convert.ToDecimal(linhas / totaltimedec);

                textBoxFreq.Invoke(new Action(() =>
                {
                    textBoxFreq.Text = Convert.ToString(decimal.Round(Freq, 0));
                }));


                for (int i = 0; i <= linhas; i++)
                {

                    string CB_Canal1Text = (string)CB_Canal1.Invoke(new GetStringFromObject(strObjMethod),
1);

                    if ((CB_Canal1Text != "Selecione o Canal") && (CB_Canal1Text != "  Select  Channel"))
                    {
                        result1 = Convert.ToDouble(matriz[i, (Convert.ToInt16(CB_Canal1Text))],
System.Globalization.CultureInfo.InvariantCulture);

                        listaPlotagem1.Add((i * time), result1);// convert1 é o valor real da variavel texto lida e
convertida para ponto flutuante e plotada no Gráfico
```

```
                eixoy1 = ((result1 - xmin) * (4095 / (xmax - xmin)) * 0.874);//87.4% de 5 volts, é a taxa máxima de conversão do mcp4922 sem distorções ou seja 4.37V;

                resultfinal1 = Convert.ToInt16(eixoy1);// resultfinal1 é o valor convertido em 12 Bits que será gravado no cartão

                plotter1 = resultfinal1.ToString();

                if (plotter1.Length == 1)
                {
                   plotter1 = 0 + plotter1;
                }
                if (plotter1.Length == 2)
                {
                   plotter1 = 0 + plotter1;
                }
                if (plotter1.Length == 3)
                {
                   plotter1 = 0 + plotter1;
                }

                listBoxConfere1.Items.Add(plotter1);

                progressBar_Ch1.Invoke(new Action(() =>
                {
                   progressBar_Ch1.Value = i;
                }));

                progressBar_Ch1.Invoke(new Action(() =>
                {
                   progressBar_Ch1.Maximum = linhas;
                }));

            }

            string CB_Canal2Text = (string)CB_Canal1.Invoke(new GetStringFromObject(strObjMethod), 2);

            if ((CB_Canal2Text != "Selecione o Canal") && (CB_Canal2Text != "  Select  Channel"))
            {
                result2 = Convert.ToSingle(matriz[i, (Convert.ToInt16(CB_Canal2Text))], System.Globalization.CultureInfo.InvariantCulture);

                listaPlotagem2.Add((i * time), result2);// convert2 é o valor real da variável texto lida e convertida para ponto flutuante e plotada no Gráfico

                eixoy2 = ((result2 - xmin) * (4095 / (xmax - xmin)) * 0.874);//87.4% de 5 volts, é a taxa máxima de conversão do mcp4922 sem distorções ou seja 4.37V

                resultfinal2 = Convert.ToInt16(eixoy2);// resultfinal3 é o valor convertido em 12 Bits que será gravado no cartão
```

```
                plotter2 = resultfinal2.ToString();

                if (plotter2.Length == 1)
                {
                    plotter2 = 0 + plotter2;
                }
                if (plotter2.Length == 2)
                {
                    plotter2 = 0 + plotter2;
                }
                if (plotter2.Length == 3)
                {
                    plotter2 = 0 + plotter2;
                }

                listBoxConfere2.Items.Add(plotter2);

                progressBar_Ch2.Invoke(new Action(() =>
                {
                    progressBar_Ch2.Value = i;
                }));

                progressBar_Ch2.Invoke(new Action(() =>
                {
                    progressBar_Ch2.Maximum = linhas;
                }));
            }

            string CB_Canal3Text = (string)CB_Canal1.Invoke(new GetStringFromObject(strObjMethod),
3);

            if ((CB_Canal3Text != "Selecione o Canal") && (CB_Canal3Text != "  Select  Channel"))
            {
                result3 = Convert.ToSingle(matriz[i, (Convert.ToInt16(CB_Canal3Text))],
System.Globalization.CultureInfo.InvariantCulture);

                listaPlotagem3.Add((i * time), result3);// convert3 é o valor real da variavel texto lida e
convertida para ponto flutuante e plotada no Gráfico

                eixoy3 = ((result3 - xmin) * (4095 / (xmax - xmin)) * 0.874);//87.4% de 5 volts, é a taxa
máxima de conversão do mcp4922 sem distorções ou seja 4.37V;

                resultfinal3 = Convert.ToInt16(eixoy3);// resultfinal3 é o valor convertido em 12 Bits que
será gravado no cartão

                plotter3 = resultfinal3.ToString();

                if (plotter3.Length == 1)
                {
                    plotter3 = 0 + plotter3;
                }
                if (plotter3.Length == 2)
                {
```

```
                plotter3 = 0 + plotter3;
            }
            if (plotter3.Length == 3)
            {
                plotter3 = 0 + plotter3;
            }

            listBoxConfere3.Items.Add(plotter3);

            progressBar_Ch3.Invoke(new Action(() =>
            {
                progressBar_Ch3.Value = i;
            }));

            progressBar_Ch3.Invoke(new Action(() =>
            {
                progressBar_Ch3.Maximum = linhas;
            }));
        }

        string CB_Canal4Text = (string)CB_Canal1.Invoke(new GetStringFromObject(strObjMethod),
4);

        if ((CB_Canal4Text != "Selecione o Canal") && (CB_Canal4Text != "  Select  Channel"))
        {
            result4 = Convert.ToSingle(matriz[i, (Convert.ToInt16(CB_Canal4Text))],
System.Globalization.CultureInfo.InvariantCulture);

            listaPlotagem4.Add((i * time), result4);// convert4 é o valor real da variavel texto lida e
convertida para ponto flutuante e plotada no Gráfico

            eixoy4 = ((result4 - xmin) * (4095 / (xmax - xmin)) * 0.874);//87.4% de 5 volts, é a taxa
máxima de conversão do mcp4922 sem distorções ou seja 4.37V;

            resultfinal4 = Convert.ToInt16(eixoy4);// resultfinal4 é o valor convertido em 12 Bits que
será gravado no cartão

            plotter4 = resultfinal4.ToString();

            if (plotter4.Length == 1)
            {
                plotter4 = 0 + plotter4;
            }
            if (plotter4.Length == 2)
            {
                plotter4 = 0 + plotter4;
            }
            if (plotter4.Length == 3)
            {
                plotter4 = 0 + plotter4;
            }

            listBoxConfere4.Items.Add(plotter4);
```

```
                progressBar_Ch4.Invoke(new Action(() =>
            {
               progressBar_Ch4.Value = i;
            }));

                progressBar_Ch4.Invoke(new Action(() =>
                {
                   progressBar_Ch4.Maximum = linhas;
                }));
               }
            }

            //updating zedpgraph control
            zed1.Invoke(new ZedGraphCode(ZedGraphCodeMethod));

         }
         catch
         {
            MessageBox.Show("Erro na conversão do arquivo");
         }
      }


      private void btn_salvar_Click(object sender, EventArgs e)
      {
         try
         {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "Binary files (*.bin)|*.bin";
            saveFileDialog1.Title = "Save an Bin File";
            saveFileDialog1.FileName = "data1";
            DialogResult resultado = saveFileDialog1.ShowDialog();

            if (resultado != System.Windows.Forms.DialogResult.Cancel)
            {
               if (progressBar_Ch1.Value == linhas)
               {
                  FileStream stream1 = new FileStream(saveFileDialog1.FileName, FileMode.Create);
                  BinaryWriter arquivobin1 = new BinaryWriter(stream1);


                  for (int i = 0; i < listBoxConfere1.Items.Count; i++)
                  {

arquivobin1.Write(Convert.ToUInt16(listBoxConfere1.Items[i],System.Globalization.CultureInfo.Invariant
Culture));
                     progressBar_save1.Maximum = listBoxConfere1.Items.Count;
                     progressBar_save1.Value = i;
                  }
                  arquivobin1.Close();
               }
```

```
            if (progressBar_Ch2.Value == linhas)
            {
               saveFileDialog1.FileName = "data2.bin";
               FileStream stream2 = new FileStream(saveFileDialog1.FileName, FileMode.Create);
               BinaryWriter arquivobin2 = new BinaryWriter(stream2);
               for (int i = 0; i < listBoxConfere2.Items.Count; i++)
               {

arquivobin2.Write(Convert.ToUInt16(listBoxConfere2.Items[i],System.Globalization.CultureInfo.Invariant
Culture));
                  progressBar_save2.Maximum = listBoxConfere2.Items.Count;
                  progressBar_save2.Value = i;
               }
               arquivobin2.Close();
            }

            if (progressBar_Ch3.Value == linhas)
            {
               saveFileDialog1.FileName = "data3.bin";
               FileStream stream3 = new FileStream(saveFileDialog1.FileName, FileMode.Create);
               BinaryWriter arquivobin3 = new BinaryWriter(stream3);
               for (int i = 0; i < listBoxConfere3.Items.Count; i++)
               {

arquivobin3.Write(Convert.ToUInt16(listBoxConfere3.Items[i],System.Globalization.CultureInfo.Invariant
Culture));
                  progressBar_save3.Maximum = listBoxConfere3.Items.Count;
                  progressBar_save3.Value = i;
               }
               arquivobin3.Close();
            }

            if (progressBar_Ch4.Value == linhas)
            {
               saveFileDialog1.FileName = "data4.bin";
               FileStream stream4 = new FileStream(saveFileDialog1.FileName, FileMode.Create);
               BinaryWriter arquivobin4 = new BinaryWriter(stream4);
               for (int i = 0; i < listBoxConfere4.Items.Count; i++)
               {

arquivobin4.Write(Convert.ToUInt16(listBoxConfere4.Items[i],System.Globalization.CultureInfo.Invariant
Culture));
                  progressBar_save4.Maximum = listBoxConfere4.Items.Count;
                  progressBar_save4.Value = i;
               }
               arquivobin4.Close();
            }
            btn_Converter.Enabled = false;
         }
      }
      catch
      {
         MessageBox.Show("Erro ao tentar salvar os arquivos");
```

```
    }
}



private void btn_bra_Click(object sender, EventArgs e)
{
    try
    {
        btn_Abrir.Text = "Abrir";
        btn_Converter.Text = "Converter";
        btn_salvar.Text = "Salvar";
        btn_sair.Text = "Sair";
        labeltimeinterval.Text = "Intervalo entre amostras";
        labelsampfreq.Text = "Frequência de amostragem";
        labelRange.Text = "Faixa de Entrada do Conversor A/D";
        if (labelFileSelect.Visible == true)
        {
            labelFileSelect.Text = " Arquivo Selecionado: " + openFileDialog1.FileName;
        }
        Form1.ActiveForm.Text = "BioSim - Conversor de Arquivos";
        zed1.GraphPane.XAxis.Title.Text = "Segundos";
        zed2.GraphPane.XAxis.Title.Text = "Segundos";
        zed3.GraphPane.XAxis.Title.Text = "Segundos";
        zed4.GraphPane.XAxis.Title.Text = "Segundos";
        zed1.GraphPane.YAxis.Title.Text = "Sinal Biológico";
        zed2.GraphPane.YAxis.Title.Text = "Sinal Biológico";
        zed3.GraphPane.YAxis.Title.Text = "Sinal Biológico";
        zed4.GraphPane.YAxis.Title.Text = "Sinal Biológico";
        zed1.Refresh();
        zed2.Refresh();
        zed3.Refresh();
        zed4.Refresh();
        CB_Canal1.Text = "Selecione o Canal";
        CB_Canal2.Text = "Selecione o Canal";
        CB_Canal3.Text = "Selecione o Canal";
        CB_Canal4.Text = "Selecione o Canal";
    }
    catch
    {
        MessageBox.Show("Erro inesperado");
    }
}

private void btn_usa_Click(object sender, EventArgs e)
{
    try
    {
        btn_Abrir.Text = "Open";
        btn_Converter.Text = "Convert";
        btn_salvar.Text = "Save";
        btn_sair.Text = "Exit";
```

```
        labeltimeinterval.Text = "Time interval";
        labelsampfreq.Text = "Sampling Frequency";
        labelRange.Text = "A/D Converter Range";
        if (labelFileSelect.Visible == true)
        {
           labelFileSelect.Text = "    Selected File:   " + openFileDialog1.FileName;
        }
        Form1.ActiveForm.Text = "BioSim - File Converter";
        zed1.GraphPane.XAxis.Title.Text = "Seconds";
        zed2.GraphPane.XAxis.Title.Text = "Seconds";
        zed3.GraphPane.XAxis.Title.Text = "Seconds";
        zed4.GraphPane.XAxis.Title.Text = "Seconds";
        zed1.GraphPane.YAxis.Title.Text = "Biological Signal";
        zed2.GraphPane.YAxis.Title.Text = "Biological Signal";
        zed3.GraphPane.YAxis.Title.Text = "Biological Signal";
        zed4.GraphPane.YAxis.Title.Text = "Biological Signal";
        zed1.Refresh();
        zed2.Refresh();
        zed3.Refresh();
        zed4.Refresh();
        CB_Canal1.Text = "  Select  Channel";
        CB_Canal2.Text = "  Select  Channel";
        CB_Canal3.Text = "  Select  Channel";
        CB_Canal4.Text = "  Select  Channel";
    }
    catch
    {
        MessageBox.Show("Erro inesperado");
    }
}

private void CB_Canal1_SelectedIndexChanged(object sender, EventArgs e)
{
    btn_Converter.Enabled = true;
}

private void CB_Canal2_SelectedIndexChanged(object sender, EventArgs e)
{
    btn_Converter.Enabled = true;
}

private void CB_Canal3_SelectedIndexChanged(object sender, EventArgs e)
{
    btn_Converter.Enabled = true;
}

private void CB_Canal4_SelectedIndexChanged(object sender, EventArgs e)
{
    btn_Converter.Enabled = true;
}

private void btn_sair_Click(object sender, EventArgs e)
{
```

```
            Application.Exit();
        }

        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            trackBar2.Minimum = -100;
            trackBar2.Maximum = -1;
            decimal valortrak = Convert.ToDecimal(trackBar2.Value);
            decimal valortraker1 = valortrak / 10;
            decimal valortraker2 = valortrak / -10;
            zed1.GraphPane.YAxis.Scale.Min = Convert.ToSingle(valortraker1,
System.Globalization.CultureInfo.InvariantCulture);
            zed2.GraphPane.YAxis.Scale.Min = Convert.ToSingle(valortraker1,
System.Globalization.CultureInfo.InvariantCulture);
            zed3.GraphPane.YAxis.Scale.Min = Convert.ToSingle(valortraker1,
System.Globalization.CultureInfo.InvariantCulture);
            zed4.GraphPane.YAxis.Scale.Min = Convert.ToSingle(valortraker1,
System.Globalization.CultureInfo.InvariantCulture);
            zed1.GraphPane.YAxis.Scale.Max = Convert.ToSingle(valortraker2,
System.Globalization.CultureInfo.InvariantCulture);
            zed2.GraphPane.YAxis.Scale.Max = Convert.ToSingle(valortraker2,
System.Globalization.CultureInfo.InvariantCulture);
            zed3.GraphPane.YAxis.Scale.Max = Convert.ToSingle(valortraker2,
System.Globalization.CultureInfo.InvariantCulture);
            zed4.GraphPane.YAxis.Scale.Max = Convert.ToSingle(valortraker2,
System.Globalization.CultureInfo.InvariantCulture);
            zed1.Refresh();
            zed2.Refresh();
            zed3.Refresh();
            zed4.Refresh();
        }


        private void txt_max_Leave(object sender, EventArgs e)
        {
            try
            {
                double txtmax = Convert.ToDouble(txt_max.Text,
System.Globalization.CultureInfo.InvariantCulture);
                if (txtmax >= 0 && txtmax <= 10)
                {
                    zed1.GraphPane.YAxis.Scale.Max = txtmax;
                    zed2.GraphPane.YAxis.Scale.Max = txtmax;
                    zed3.GraphPane.YAxis.Scale.Max = txtmax;
                    zed4.GraphPane.YAxis.Scale.Max = txtmax;
                    zed1.Refresh();
                    zed2.Refresh();
                    zed3.Refresh();
                    zed4.Refresh();
                }
                else
                {
                    MessageBox.Show("os valores devem variar de 00.00 a +10.00");
```

```
                txt_max.Text = "10.00";
            }
        }
        catch
        {
            MessageBox.Show("os valores devem variar de 00.00 a +10.00");
            txt_max.Text = "10.00";
        }
    }

    private void txt_min_Leave(object sender, EventArgs e)
    {
        try
        {
            double txtmin = Convert.ToDouble(txt_min.Text,
System.Globalization.CultureInfo.InvariantCulture) * -1;
            if (txtmin >= -10 && txtmin <= 0)
            {
                zed1.GraphPane.YAxis.Scale.Min = txtmin;
                zed2.GraphPane.YAxis.Scale.Min = txtmin;
                zed3.GraphPane.YAxis.Scale.Min = txtmin;
                zed4.GraphPane.YAxis.Scale.Min = txtmin;
                zed1.Refresh();
                zed2.Refresh();
                zed3.Refresh();
                zed4.Refresh();
            }
            else
            {
                MessageBox.Show("os valores devem variar de -10.00 a 00.00");
                txt_min.Text = "10.00";
            }

        }
        catch
        {
            MessageBox.Show("os valores devem variar de -10.00 a 00.00");
            txt_min.Text = "10.00";
        }
    }

  }

}
```

ANEXO III

LINHAS DE CÓDIGO DO FIRMWARE DA PLACA MAPLE

```
/*********************************************************************/
/*              Programa que efetua a leitura do cartão de memória e  */
/*                    envia os dados para o conversor DAC             */
/*********************************************************************/
/*********************************************************************/
/*                              Bibliotecas                          */
/*********************************************************************/
#include <CircularBuffer.h>
#include <SdFat.h>
#include <MCP4922.h>
#include <stdint.h>
/*********************************************************************/
/*                              Defines                              */
/*********************************************************************/
//Timer rate in microseconds
//#define TIMER_RATE 100
//Tamanho dos pacote de dados que será lido do cartão
// em bytes
#define DATA_SIZE 1024
//Número de amostras máximo pra gerar um pedido de busca de pacotes no cartão
#define NSAMPLES_TO_GENERATE_REQUEST 512
//Pinos utilizados para o DAC
#define DAC_CS 7
```

```
#define DAC_LATCH 9
/*****************************************************************/
/*                          OBJETOS                              */
/*****************************************************************/
CircularBuffer Cbuff;
 //SPI para o sdcard
HardwareSPI spisd(1);
Sd2Card card(spisd, 8, true);
//Objetos para manipular o sistema de arquivos no cartão sd
SdVolume volume;
SdFile root;
SdFile file;
//Obj Timer para garantir a tx de amostragem do sinal a ser reproduzido
HardwareTimer timer(2);
//DAC
MCP4922 dac(DAC_CS, DAC_LATCH);
/*****************************************************************/
/*                       Variáveis Globais                       */
/*****************************************************************/
//buffer para onde são transferidos os dados do cartão
uint16_t data[DATA_SIZE];
volatile boolean bDACRequest = false; //flag que indica o pedido de mais amostras para o DAC
volatile uint32 samplescounter = 0; //Registra o número de amostras já retirada do buffer para gerar os pedidos
uint32 numberofsamples = 0; //Numero total de amostras lidas do arquivo
volatile uint32 timeinus = 0; //Tempo total gasto para ler todas as amostras em uS
float time = 0, rate = 0; //Variaveis para calculo do tempo total em segundos e taxa de amostragem
int16 c = 0; //Variavel usada para receber um dado do buffer circular
char armazena;
```

```
float tempo;
int a = 0;
int b = 0;
float dados = -1;
int botao1 = 3;
int botao2 = 6;
int val1;
/*************************************************************8*****/
/*                          Função consumidora                     */
/*****************************************************************/
void TimerHandler(void)
{
  //Verifica se o buffer não está vazio, caso ele esteja é retornado -1
  if((c = Cbuff.Get()) != -1)
  {
    //Envia o dado presente no buffer circular para o DAC
    dac.sendToA((uint16)c);
    dac.updateOutputs();
  }
   //Atualiza o número de amostra lido e verifica se é necessário
  //gerar um pedido de mais amostras
  samplescounter++;
  timeinus++; //variavel para contar quanto tempo foi gasto
  if(samplescounter > NSAMPLES_TO_GENERATE_REQUEST)
  {
    samplescounter = 0;
    bDACRequest = true;
  }
}
```

```
/*********************************************************************/
/*                              Funções                              */
/*********************************************************************/
/*
  Função para configuração e inicialização do timer
*/
void TimerConfig(void)
{
   // Pause the timer while we're configuring it
  timer.pause();
  // Set up period
  timer.setPeriod(tempo);//(TIMER_RATE); // in microseconds
  // Set up an interrupt on channel 1
  timer.setChannel1Mode(TIMER_OUTPUT_COMPARE);
  timer.setCompare(TIMER_CH1, 1);  // Interrupt 1 count after each update
  timer.attachCompare1Interrupt(TimerHandler);
  // Refresh the timer's count, prescale, and overflow
  timer.refresh();
  // Start the timer counting
  timer.resume();
}
/*********************************************************************/
/*                              SETUP                                */
/*********************************************************************/
void setup() {
  pinMode(botao1,INPUT);
  pinMode(botao2,INPUT);
  Serial2.begin(9600);
```

```
  //Inicializa a USB
  SerialUSB.begin();
  // initialize the SD card
  if (!card.init())
  SerialUSB.println("card.init failed.");
  delay(100);
  // initialize a FAT volume
  if (!volume.init(&card,1))
    SerialUSB.println("volume.init failed.");
  // open the root directory
  if (!root.openRoot(&volume))
    SerialUSB.println("openRoot failed.");
}
/*****************************************************************************/
/*                                    LOOP                                   */
/*****************************************************************************/
void loop()
{
  val1=digitalRead(botao1);
  val2=digitalRead(botao2);
  if(Serial2.available()){
    armazena=Serial2.read();
    if(armazena>='0' && armazena<='9'){
      if(dados<0){
        dados = (int)armazena-48;
      }
      else{
        dados = dados*10+((int)armazena-48);
      }
```

```
  }
  else{
    if(dados>=0){
   //   SerialUSB.println(data);
      tempo = 1000000/dados;
    }
    dados=-1;
  }
}
if ((val1 == LOW)||(a==1)){
//envia um sinal para o arduino de que a emulação começou
Serial2.println('x');
//Inicializa o buffer circular
Cbuff.Init();
SerialUSB.println("aperte qualquer tecla para iniciar");
//  while (!SerialUSB.available()); //Aguarda a chegada de um caractere
//  SerialUSB.read();
// open a file
 uint32_t size = 1, totalsize = 0;
 if (file.open(&root, "data.bin", O_READ)) //Tenta abrir o arquivo
 {
  SerialUSB.println("data.bin iniciado.");
 //Leitura do cabeçalho onde está o numero de bytes do arquivo
  file.read(&totalsize, sizeof(uint32));
 //imprime o tamanho do aquivo em bytes
  SerialUSB.print("tamanho do arquivo: ");
  SerialUSB.print(totalsize);
  SerialUSB.println(" Bytes.");
  //Imprime o número de amostras
```

```
  SerialUSB.print("Total de amostras lidas: ");
  SerialUSB.print(totalsize/sizeof(uint16));
  SerialUSB.println(".");
  SerialUSB.print("Tempo: ");
  SerialUSB.print(tempo);
//Leitura das amostras
  totalsize = 0;
  //Aqui é inserido uma quantidade inicial de amostras no buffer circular
  for(uint32 i = 0; i < 4; i++)
  {
    size = file.read(data,DATA_SIZE); //Faz um pedido de DATA_SIZE bytes para o cartão
  //Armazena os dados lidos no cartão no buffer circular
    Cbuff.PutN(data,(size/sizeof(uint16)));
    totalsize += size;
  }
  SerialUSB.println();
  SerialUSB.println("Iniciando!");
  //Função consumidora é habilitada a consumir
  TimerConfig();
  while(size)
  {
  //Espera pedido de dados gerado pela função consumidora.
    if(bDACRequest == true) {
      //reseta o pedido de dados
      bDACRequest = false;
      //Solicita os daddos para o cartão
      size = file.read(data,DATA_SIZE);
      totalsize += size;
      //Armazena no buffer circular o numero de dados
```

```
    //que foi lido do cartão
    CbufANEXOf.PutN(data, (size/sizeof(uint16)));
   }
 }
 //Aguarda o buffer acabar de ser completamente consumido pela função timerhandler
 while(Cbuff.ConsPos() != Cbuff.ProductorPos());
//Desabilita o timer para parar de consumir o buffer, pois não haverá mais amostras
 timer.pause();
 SerialUSB.println("Fim");
 SerialUSB.println();
 SerialUSB.println("INFORMACOES.");
 //Calcula e imprime do numero de amostras
 numberofsamples = (totalsize/sizeof(uint16));
 SerialUSB.print("Numero de amostras: ");
 SerialUSB.print(numberofsamples);
 SerialUSB.println(".");
  //Calcula e imprime o tempo gasto
 time = ((float)(timeinus*tempo))/1000000;
 SerialUSB.print("Tempo gasto: ");
 SerialUSB.print(time);
 SerialUSB.println(" s.");
 //Calcula e imprime a taxa de amostragem
 rate = ((float)numberofsamples)/time;
 SerialUSB.print("Taxa de amostragem: ");
 SerialUSB.print(rate);
 SerialUSB.println(" APS.");
  //Zera o DAC
 dac.sendToA(0);
 dac.updateOutputs();
```

```
  //reseta todas as variáveis utilizadas para que um novo
  //processamento seja feito
  numberofsamples = 0;
  time = 0;
  rate = 0;
  totalsize = 0;
  size = 1;
  timeinus = 0;
  samplescounter = 0;
  bDACRequest = false;
  //Fim
  SerialUSB.println();
  SerialUSB.println("Ok!");
  SerialUSB.println();
  if(val2==LOW){
  a=1;
  }
  if(val2==HIGH){
  Serial2.println('y');
  a=0;
  }
 }
 else
 {
  SerialUSB.println("file.open falhou");
 }
//Fecha o arquivo para abri-lo novamente na proxima iteração da void loop()
 file.close();
 }
```

```
}
```

ANEXO IV

LINHAS DE CÓDIGO DO FIRMWARE DA PLACA ARDUINO MEGA

```
/*****************************************************************/
/*                                                               */
/*     Programa para controle da interface Homem-Máquina e gerenciamento      */
/*          dos canais do EMULADOR DE SINAIS BIOLÓGICOS.         */
/*                                                               */
/*****************************************************************/
#include <LiquidCrystal.h>
#include <Keypad.h>
// inicializa a biblioteca com os numeros dos pinos do arduino
LiquidCrystal lcd(43, 39, 29, 27, 25, 23);
int ledPin = 13;
char num[5];
char key;
int x = 0,y = 0;
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] ={{    '1','2','3'     },
                        {    '4','5','6'     },
                        {    '7','8','9'     },
                        {    '*','0','#'     }};
byte rowPins[ROWS] = { 42, 38, 44, 48}; //connect to the row pinouts of the keypad
byte colPins[COLS] = { 52, 50, 40}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
int but1, but2, but3, btnesc, btnenter;//variáveis referentes aos botões
```

```
int a=0, b=0, c=0, i, w=0, habilitateclado = 1, sdcardinit = 0,valordefault = 10040,
habilitaesc1 = 0, habilitaesc2 = 0, habilitaesc3 = 0, habilitaesc4 = 0; //variáveis de controle

int ln = 0,cl = 9;//variaveis para controle do cursor.

unsigned short int valor1,valor2,valor3,valor4;

/*****************************************************************/

/*                    Rotina que gerencia ajuda                  */

/*****************************************************************/

void help()

{

  lcd.clear();

  lcd.print("PARA EMULAR O SINAL");

  lcd.setCursor(0,1);

  lcd.print("DESEJADO, AJUSTE A ");

  lcd.setCursor(0,2);

  lcd.print("FREQUENCIA DO CANAL");

  lcd.setCursor(0,3);

  lcd.print("E PRESSIONE 'ENTER'");

  delay(7000);

  lcd.clear();

  lcd.setCursor(0,1);

  lcd.print("  *** EXEMPLO ***  ");

  pisca();

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print(" CANAL 1: 10000 Hz ");

  lcd.setCursor(0,1);

  lcd.print(" CANAL 2:  5000 Hz ");
```

```
  lcd.setCursor(0,2);
  lcd.print(" CANAL 3:   500 Hz ");
  lcd.setCursor(0,3);
  lcd.print(" CANAL 4:   100 Hz ");
  delay(7000);
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print(" PRESSIONE 'ENTER' ");
  pisca();
  seleciona();
}
/*********************************************************************/
/*                          Rotina que gerencia intermitência                          */
/*********************************************************************/
void pisca()
{
  for(int p=0;p<4;p++)
  {
    lcd.noDisplay();
    delay(400);
    lcd.display();
    delay(400);
  }
}
/*********************************************************************/
/*      Rotina que substitui o "delay" podendo ser interrompida a qualquer tempo      */
/*********************************************************************/
```

```
void aguarda()
{
  but3 = HIGH;
  for(int s=0;s<201;s++)
  {
    if (but3 == LOW)    {
  help();    }
    delay(30);
  }
}
/*********************************************************************/
/*                          Rotina que gerencia a seleção                          */
/*********************************************************************/
void seleciona()
{
  lcd.clear();
  lcd.noBlink();
  lcd.setCursor(0,0);//Posiciona o cursor na coluna 0 e linha 1
  lcd.print("     SELECIONE     ");
  lcd.setCursor(0,1);//Posiciona o cursor na coluna 0 e linha 2
  lcd.print("1 -  EMULAR CANAIS  ");
  lcd.setCursor(0,2);//Posiciona o cursor na coluna 0 e linha 3
  lcd.print("2 -  AJUSTES        ");
  lcd.setCursor(0,3);//Posiciona o cursor na coluna 0 e linha 4
  lcd.print("3 -  AJUDA          ");
  a = 0;//desabilita a tecla '#' do teclado alfa numérico
}
```

```
/*****************************************************************/
/*                Rotina que gerencia as configurações de frequência                */
/*****************************************************************/
void ajustes() {
  lcd.clear();
  lcd.setCursor(0,0);//Posiciona o cursor na coluna 0 e linha 1
  lcd.print("PARA EMULAR O SINAL");
  lcd.setCursor(0,1);//Posiciona o cursor na coluna 0 e linha 2
  lcd.print("DESEJADO, AJUSTE A");
  lcd.setCursor(0,2);//Posiciona o cursor na coluna 0 e linha 3
  lcd.print("FREQUENCIA DO CANAL");
  lcd.setCursor(0,3);//Posiciona o cursor na coluna 0 e linha 4
  lcd.print("E PRESSIONE 'ENTER'");
  delay(7000);
  lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("CANAL 1:      Hz   ");
 lcd.setCursor(0,1);
 lcd.print("CANAL 2:      Hz   ");
 lcd.setCursor(0,2);
 lcd.print("CANAL 3:      Hz   ");
 lcd.setCursor(0,3);
 lcd.print("CANAL 4:      Hz   ");
 lcd.setCursor(9,0);
 lcd.blink();
 a = 1;//habilita a navegação no lcd pela tecla '#'
 }
```

```
/*********************************************************************/
/*                        Rotina que habilita a tela inicial         */
/*********************************************************************/
void telainicial()
{
  lcd.print(" EMULADOR DE SINAIS ");//Exibe a mensagem
  lcd.setCursor(0,2);
  lcd.print("     BIOLOGICOS    ");
  aguarda();// chama a função help e funciona como delay(6000)
  sdcard();
}
/*********************************************************************/
/*                        Rotina que verifica a condição do SD CARD  */
/*********************************************************************/
void sdcard()
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("    SD CARD1: OK    ");
  lcd.setCursor(0,1);
  lcd.print("    SD CARD2: OK    ");
  lcd.setCursor(0,2);
  lcd.print("    SD CARD3: OK    ");
  lcd.setCursor(0,3);
  lcd.print("    SD CARD4: OK    ");
  if (Serial3.available())
  {
```

```
  x = Serial3.read();
  if(x == 119)
  {
    lcd.setCursor(0,0);
    lcd.print("ERRO AO LER SD CARD1");
  }
}
if (Serial2.available())
{
  x = Serial2.read();
  if(x == 119)
  {
    lcd.setCursor(0,1);
    lcd.print("ERRO AO LER SD CARD2");
  }
}
if (Serial1.available())
{
  x = Serial1.read();
  if(x == 119)
  {
    lcd.setCursor(0,2);
    lcd.print("ERRO AO LER SD CARD3");
  }
}
if (Serial.available())
{
```

```
   x = Serial.read();
   if(x == 119)
   {
     lcd.setCursor(0,3);
     lcd.print("ERRO AO LER SD CARD4");
   }
  }
  sdcardinit = 1;
  aguarda();
  seleciona();
}
/*********************************************************************/
/*                                SETUP                              */
/*********************************************************************/
void setup()
{
  lcd.begin(20,4);//inicia o LCD com o número de colunas e linhas
  Serial1.begin(9600);
  Serial1.flush();
  Serial2.begin(9600);
  Serial2.flush();
  Serial3.begin(9600);
  Serial3.flush();
  Serial.begin(9600);
  Serial.flush();
  pinMode(53,OUTPUT);
  pinMode(8,OUTPUT);
```

```
  pinMode(12,INPUT);
  pinMode(11,INPUT);
  pinMode(10,INPUT);
  pinMode(7,INPUT);
  pinMode(6,INPUT);
  telainicial();
}
/*********************************************************************/
/*                                LOOP                               */
/*********************************************************************/
void loop()
{
  but1=digitalRead(12);
  but2=digitalRead(11);
  but3=digitalRead(10);
  btnesc=digitalRead(7);
  btnenter=digitalRead(6);
  key = keypad.getKey();
  if (sdcardinit == 0)
  {
    delay(50);
    sdcard();
  }
  if (Serial3.available())
  {
    x = Serial3.read();
    if(x == 120)
```

```
  {
    lcd.setCursor(0,0);
    lcd.print(" CANAL 1: EMULANDO  ");
    habilitaesc1 = 1;
  }
  if(x == 121)
  {
    lcd.setCursor(0,0);
    lcd.print(" CANAL 1:  PRONTO   ");
    habilitaesc1 = 0;
  }
  if(x == 122)
  {
    lcd.setCursor(0,0);
    lcd.print("DATA1 NAO ENCONTRADO");
    habilitaesc1 = 0;
  }
  if(x == 118)
  {
    lcd.setCursor(0,0);
    lcd.print(" CANAL 1:          ");
    habilitaesc1 = 0;
  }
}
if (Serial2.available())
{
  x = Serial2.read();
```

```
 if(x == 120)
 {
  lcd.setCursor(0,1);
  lcd.print(" CANAL 2: EMULANDO  ");
  habilitaesc2 = 1;
 }
 if(x == 121)
 {
  lcd.setCursor(0,1);
  lcd.print(" CANAL 2:  PRONTO   ");
  habilitaesc2 = 0;
 }
 if(x == 122)
 {
  lcd.setCursor(0,1);
  lcd.print("DATA2 NAO ENCONTRADO");
  habilitaesc2 = 0;
 }
  if(x == 118)
 {
  lcd.setCursor(0,1);
  lcd.print(" CANAL 2:          ");
  habilitaesc2 = 0;
 }
}
if (Serial1.available())
{
```

```
 x = Serial1.read();
 if(x == 120)
 {
  lcd.setCursor(0,2);
  lcd.print(" CANAL 3: EMULANDO  ");
  habilitaesc3 = 1;
 }
 if(x == 121)
 {
  lcd.setCursor(0,2);
  lcd.print(" CANAL 3:  PRONTO   ");
  habilitaesc3 = 0;
 }
 if(x == 122)
 {
  lcd.setCursor(0,2);
  lcd.print("DATA3 NAO ENCONTRADO");
  habilitaesc3 = 0;
 }
 if(x == 118)
 {
  lcd.setCursor(0,2);
  lcd.print(" CANAL 3:          ");
  habilitaesc3 = 0;
 }
}
if (Serial.available())
```

```
{
  x = Serial.read();
  if(x == 120)
  {
    lcd.setCursor(0,3);
    lcd.print(" CANAL 4: EMULANDO  ");
    habilitaesc4 = 1;
  }
  if(x == 121)
  {
    lcd.setCursor(0,3);
    lcd.print(" CANAL 4:  PRONTO   ");
    habilitaesc4 = 0;
  }
  if(x == 122)
  {
    lcd.setCursor(0,3);
    lcd.print("DATA4 NAO ENCONTRADO");
    habilitaesc4 = 0;
  }
  if(x == 118)
  {
    lcd.setCursor(0,3);
    lcd.print(" CANAL 4:          ");
    habilitaesc4 = 0;
  }
}
```

```
 if ((but1 == LOW)&&(w == 0))//ativado pelo botão esc
 {
 delay(300);
 Serial.println(valordefault);
 Serial1.println(valordefault);
 Serial2.println(valordefault);
 Serial3.println(valordefault);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("   CANAIS PRONTOS   ");
 lcd.setCursor(0,2);
 lcd.print("    APERTE PLAY     ");
 }
if ((but1 == LOW)&&(w == 1))
{
  delay(300);
 Serial.println(valor4);
 Serial1.println(valor3);
 Serial2.println(valor2);
 Serial3.println(valor1);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   CANAIS PRONTOS   ");
  lcd.setCursor(0,2);
  lcd.print("    APERTE PLAY     ");
}
```

```
  if ((but2 == LOW)&&(habilitaesc1 == 0)&&(habilitaesc2 == 0)&&(habilitaesc3 ==
0)&&(habilitaesc4 == 0))

  {

    delay(300);

    w = 1;

    ajustes();

  }

  if ((but3 == LOW)&&(habilitaesc1 == 0)&&(habilitaesc2 == 0)&&(habilitaesc3 ==
0)&&(habilitaesc4 == 0))

  {

    help();

    lcd.clear();

    lcd.noBlink();

    seleciona();

  }

  if ((btnesc == LOW)&&(habilitaesc1 == 0)&&(habilitaesc2 == 0)&&(habilitaesc3 ==
0)&&(habilitaesc4 == 0))

  {

    delay(300);

    seleciona();

    w = 0;

  }

/*********************************************************************/

 /*                        HABILITA TECLADO NUMÉRICO                  */

/*********************************************************************/

  if(a==1)

  {

    if ((key != NO_KEY)&&(habilitateclado == 1))
```

```
{
 num[i] = key;
 lcd.print(num[i]);
 i++;//incrementa o numero que ira ser enviado ao ARM
 if(i > 4)
 {
 habilitateclado = 0;
 }
}
  if ((btnenter == LOW)&&(b==0))// seta os valores de emulação do canal 1
 {
   delay(300);
   valor1 = atoi(&num[0]);
   if (valor1 > 20000)
   {
     lcd.setCursor(9,0);
     lcd.print("OVERLOAD  ");
     valor1 = 0;
     delay(2000);
   }
   lcd.setCursor(9,0);
   lcd.print("          ");
   lcd.setCursor(9,0);
   lcd.print(valor1);//confirma o valor digitado
   delay(100);
   for(i=0;i<6;i++)//zera todas as posições do vetor
   {
```

```
  num[i] = '\0';
  }
  i = 0;//reinicia o vetor de dados numéricos
  btnenter = HIGH;// força o pino ao nível alto
  lcd.setCursor(18,0);
  lcd.print("Ok");//confirma o envio de dados para o ARM
  lcd.setCursor(9,1);//posiciona o cursor na linha seguinte
  b = 1;
  habilitateclado = 1;//habilita teclado
}
if ((btnenter == LOW)&&(b == 1))// seta os valores de emulação do canal 2
{
  delay(300);
  valor2 = atoi(&num[0]);

  if (valor2 > 20000)
  {
    lcd.setCursor(9,1);
    lcd.print("OVERLOAD  ");
    valor2 = 0;
    delay(2000);
  }
  lcd.setCursor(9,1);
  lcd.print("          ");
  lcd.setCursor(9,1);
  lcd.print(valor2);//confirma o valor digitado
  delay(100);
```

```
  for(i=0;i<6;i++)//zera todas as posições do vetor
  {
  num[i] = '\0';
  }
  i = 0;
  btnenter = HIGH;
  lcd.setCursor(18,1);
  lcd.print("Ok");
  lcd.setCursor(9,2);
  b = 2;
  habilitateclado = 1;
}
if ((btnenter == LOW)&&(b == 2))// seta os valores de emulação do canal 3
{
  delay(300);
  valor3 = atoi(&num[0]);
  if (valor3 > 20000)
  {
    lcd.setCursor(9,2);
    lcd.print("OVERLOAD  ");
    valor3 = 0;
    delay(2000);
  }
  lcd.setCursor(9,2);
  lcd.print("          ");
  lcd.setCursor(9,2);
  lcd.print(valor3);
```

```
  delay(100);
  for(i=0;i<6;i++)//zera todas as posições do vetor
  {
  num[i] = '\0';
  }
  i = 0;
  btnenter = HIGH;
  lcd.setCursor(18,2);
  lcd.print("Ok");
  lcd.setCursor(9,3);
  b = 3;
  habilitateclado = 1;
}
if ((btnenter == LOW)&&(b == 3))// seta os valores de emulação do canal 4
{
  delay(300);
  valor4 = atoi(&num[0]);
  if (valor4 > 20000)
  {
    lcd.setCursor(9,3);
    lcd.print("OVERLOAD  ");
    valor4 = 0;
    delay(2000);
  }
  lcd.setCursor(9,3);
  lcd.print("          ");
  lcd.setCursor(9,3);
```

```
      lcd.print(valor4);//confirma valor digitado
      delay(100);
      for(i=0;i<6;i++)//zera todas as posições do vetor
      {
      num[i] = '\0';
      }
      i = 0;
      btnenter = HIGH;
      lcd.setCursor(18,3);
      lcd.print("Ok");
      lcd.setCursor(9,0);
      b = 4;
      habilitateclado = 1;
    }
    if (b==4)
    {
      delay(300);
      seleciona();
      b = 0;
      w = 1;
    }
  }
}
```