CENTRO UNIVERSITÁRIO DE BRASÍLIA- UniCEUB

FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS LICADAS – FATECS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

DAISY MARÇAL SOARES PEREIRA

SISTEMA DE COMUNICAÇÃO ALTERNATIVO PARA PESSOAS COM DISTÚRBIOS NEUROPSICOMOTORES GRAVES

BRASÍLIA – DF

1° SEMESTRE DE 2014

DAISY MARÇAL SOARES PEREIRA

SISTEMA DE COMUNICAÇÃO ALTERNATIVO PARA PESSOAS COM DISTÚRBIOS NEUROPSICOMOTORES GRAVES

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.ª. MSc. Luciano Henrique Duque

BRASÍLIA

JULHO, 2014

DAISY MARÇAL SOARES PEREIRA

SISTEMA DE COMUNICAÇÃO ALTERNATIVO PARA PESSOAS COM DISTÚRBIOS NEUROPSICOMOTORES GRAVES

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.ª. MSc. Luciano Henrique Duque

Este Trabalho foi julgado adequado para a obtencão do Tíıtulo de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Msc. Prof. Abiézer Amarilia Fernandes

Coordenador do Curso

Banca examinadora:

_____

Msc. Prof. Luciano Henrique Duque

Orientador

_____

Msc. Prof. Luís Cláudio Lopes de Araujo

Convidado 1

_____

Msc. Prof. Francisco Javier de Obaldía Díaz

Convidado 2

Dedico esta monografia a todos que de alguma forma contribuíram para que eu chegasse até aqui. Agradeço principalmente a minha família pela paciência e confiança depositada em mim para que este momento tão importante fosse alcançado e ao meu namorado por todo esforço e motivação que me passou durante todo o projeto. Sou muito grata a todos!

Daisy Marçal Soares Pereira

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por permitir que este sonho se realizasse.

Agradeço aos meus pais, Dalton e Ira, por ter me dado todo apoio e incentivo necessário durante todo o curso. As minha irmãs, Danielle, Débora e Denise por todas as vezes que precisei não hesitaram em me ajudar.

Agradeço a Instituição por ter me proporcionado todas a ferramentas para a conclusão deste desafio. Agradeço a todos os professores pois sem eles a realização desta etapa não se realizaria. Agradeço ao meu orientador, o Professor Luciano Duque por todo tempo dedicado a este projeto e também a uma pessoa de grande importância para a realização deste trabalho, Robson Mamédio que não mediu esforços ao me auxiliar e orientar durante todo desenvolvimento do trabalho.

E por último mas não menos importante agradeço ao meu namorado, Denis, por todo esforço, apoio e dedicação para que esse projeto se concretiza-se.

A todos, muito Obrigada!!

Daisy Marçal Soares Pereira

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes”

Marthin Luther King

SUMÁRIO

Lista de Figuras

LISTA DE SIGLAS

EMG	Eletromiografico/ Eletromiografia
GER	Grupo de Engenharia e Reabilitação
MUAP	Motor Unit Action Potencial

RESUMO

Este trabalho tem como objetivo auxiliar na comunicação de pessoas com distúrbios neuropsicomotores graves. Para o cenário implementado foi considerada a condição conhecida como síndrome do encarceiramento. Utilizando-se de sinais eletromiográficos captados na musculatura orbicular da face é possível promover a comunicação entre pessoas com esta síndrome e o mundo externo.

Palavras chaves: eletromiografia, emg, síndrome do encarceiramento

ABSTRACT

This work aims to assist in the communication of people with severe neurological motor disorders. For the implemented scenario it was considered the condition known as Locked-in syndrome. Using electromyographic signals captured in orbicularis muscles of the face it's possible to promote communication between people with this syndrome and the external world.

Keywords: electromyographic, SEMG, Locked-in syndrome.

CAPÍTULO 1 INTRODUÇÃO

Eletromiografia é uma técnica que propõe estudar os fenômenos bioelétricos que correm no corpo humano no interior das membranas celulares na musculatura humana e em todo sistema nervoso. O espectro de frequências dos sinais mioelétricos estendem-se entre 10Hz e 500Hz para a maioria dos eletrodos de superfície.

No cérebro humano, há uma grande quantidade de atividade neural que define como somos e influi no que fazemos. O amplificador de eletromiografia é possibilitado mediante o uso de instrumentos eletrônicos que disponibilizam informações relacionadas à resposta, referente aos fenômenos bioelétricos captada quando ocorre um determinado esforço muscular (DUQUE e LOBO, 2013). Dependendo da área da captura do sinal e do eletrodo utilizado, sua amplitude e frequência podem ser alterados. O sinal elétrico obtido pela musculatura ocular possui entre 10µV e 100 µV.

Pessoas com distúrbios neuropsicomotor, como Paralisia Cerebral, Traumatismo Crânio-encefálico, Síndromes Genéticas, entre outras tem uma dificuldade enorme de se comunicar devido ao comprometimento motor grave, mas que possuem o desenvolvimento cognitivo normal. Essas pessoas, apesar de ter um potencial cognitivo normal e de compreenderem a linguagem oral, podem encontrar dificuldade na linguagem expressiva, considerando que esta envolve os músculos do aparelho fonatório, podendo ocasionar assim uma dificuldade na comunicação dos sentimentos, desejos ou opiniões (MAMÉDIO, DUQUE e WIECHERT, 2011).

A única forma de comunicação dos pacientes acometidos desta síndrome é a movimentação ocular. Trabalhando com foco nessa limitação o projeto apresenta uma forma de comunicação mais eficaz de forma a permitir os mais variados tipos de diálogos que, por meio de sinais miográficos emitidos pelo paciente e captados pelo sistema, permite a transformação em palavras, frases e textos.

A técnica para captação dos sinais eletromiográficos do musculo ocular utilizada neste trabalho é similar ao exame de Eletrooculograma (EOG) apenas na disposição dos eletrodos. A referida técnica permite medir a diferença de potencial ocular e decorre da utilização de dois eletrodos em cada extremidade dos olhos.

Utilizando-se da eletromiografia, este trabalho objetiva oferecer um sistema de comunicação alternativa para as pessoas que sofrem da síndrome do encarceramento (*Locked-in Syndrome*), que se caracteriza basicamente por tetraplegia, anatria e preservação do nível de consciência, além de certa movimentação ocular pela qual o paciente se comunica. O equipamento proposto será dividido em duas etapas: *hardware* que propiciará captação e processamento dos sinais e *software*, que receberá o sinal processado para formação de palavras e textos.

Para conversão do sinal analógico para digital será utilizada a entrada de áudio do computador utilizado para o projeto. O *software* será desenvolvido em linguagem Java o que será de suma importância neste projeto, considerando que após o *hardware* captar o envio do sinal pelo paciente, o *software* deverá identificar qual a opção de texto que estava sendo apresentada e iniciar a

formação de palavras conforme sequenciamento, por meio de um teclado que será apresentado ao paciente.

1.1 MOTIVAÇÃO

Durante o curso de engenharia de computação, é possível conhecer o quão vasta é a abrangência da área de atuação de engenharia. A cada aula, um novo conhecimento era adquirido e um novo desejo era despertado. A vontade de utilizar todos aqueles conhecimentos foi se transformando num crescente sentimento de que, por alguma forma, houvesse a possibilidade de ajudar ou até mesmo mudar a vida das pessoas. Em uma das oportunidades durante o curso, é possível acompanhar o trabalho realizado pelo Grupo de Engenharia e Reabilitação(GER-UniCEUB), e perceber que este projeto poderia ter por fundamento tudo o que os alunos do grupo já tinham desenvolvido até ali e, mediante a aplicação conhecimentos adquiridos durante o curso, propiciar o desenvolvimento de algo que pudesse ser um grande passo para essa conquista.

No dia 18 de janeiro de 2011, o humorista Shaolin sofreu um grave acidente na Rodovia Federal BR 230 em Campina Grande-PB. Foi socorrido e internado no Hospital de Trauma da Cidade e pouco tempo depois, foi transferido para o Hospital das Clínicas em São Paulo, onde ficou internado durante cinco meses. Logo após, recebeu alta e voltou para casa, em Campina Grande. No ano de 2012, Shaolin foi submetido a um teste de consciência, com a utilização de equipamentos importados, onde foi verificado que existiam reações, mesmo que involuntárias, em seu sistema nervoso. Dois anos após o acidente, um

programa da emissora Record, Programa da Tarde, fez uma matéria com o humorista, tendo sido utilizado um aparelho sueco que em sua versão mais completa chega a custar R$75 mil (setenta e cinco mil reais). Com auxílio deste aparelho Shaolin conseguiu se comunicar com a equipe de reportagem.

Em 2012, quando tiveram início os projetos do GER-UniCEUB, durante algumas apresentações ocorridas na semana da engenharia, promovida pelo UniCEUB, e tendo conhecimento do caso Shaolin, ao pesquisar sobre o assunto é possível identificar que existem muitas pessoas que estão presas em seu próprio corpo e sem qualquer tipo de comunicação com ambiente externo.

O aparelho sueco, que foi doado como um presente para o humorista, possui um alto preço. Mesmo que seja adquirido em sua versão mais básica, que custa em torno de R$ 30 mil (trinta mil reais), ainda assim pode-se considerar que possui um valor alto. A partir daí nasceu a ideia do projeto, qual seja, propor um dispositivo que tenha um baixo custo e que possa ser facilmente desenvolvido.

Apesar de não ter implementado tudo o que a tecnologia atual pode oferecer, o dispositivo tem em suas características mais básicas o suficiente para permitir uma comunicação entre o usuário e o mundo externo. O presente projeto satisfaz, assim, dois objetivos principais: utilizar-se dos conhecimentos adquiridos no curso e de alguma forma influenciar e/ou melhorar vidas.

1.2 OBJETIVOS

O **objetivo geral** deste projeto é desenvolver um protótipo de um dispositivo para a comunicação alternativa de pessoas com distúrbios neuropsicomotor grave.

Os **objetivos específicos** são aqueles que visam a atingir o objetivo geral. Nesse contexto, os objetivos específicos pautam-se em:

- Projetar e desenvolver um dispositivo capaz de capturar, filtrar, amplificar e transmitir o sinal analógico para um meio digital, de forma a se evitar a perda ou captação errônea desse sinal, permitindo-se uma alta confiabilidade no dispositivo desenvolvido.
- Confeccionar e testar os dispositivos simulados na etapa anterior.
- Projetar uma interface de EMG ocular para captação dos sinais musculares responsáveis pelo piscar dos olhos.
- Desenvolver de um *software* específico (*Firmware*) com as ferramentas de desenvolvimento Eclipse (Eclipse Foundation - 2004) e NetBeans (Sun Microsystems - 2000) para que tal dispositivo possa se comunicar com um computador via entrada de áudio.
- Desenvolver um editor de texto em JAVA 7, para que o paciente possa ver e ler tudo aquilo que escreva na tela do computador, utilizando o dispositivo de comunicação alternativa.

- Integrar *hardware* e *software* e testar tal integração para garantir a estabilidade e confiabilidade do projeto.

1.3 METODOLOGIA

Para o desenvolvimento do projeto é necessário que seja realizada uma revisão bibliográfica sobre sinais eletromiográficos, desenvolvimento de *software* em linguagem Java e um estudo sobre distúrbios neuropsicomotores graves, de modo a associá-los ao projeto.

Para facilitar o desenvolvimento do projeto, este é dividido em etapas, quais sejam:

- Etapa 1: Desenvolvimento um estudo sobre comunicação alternativa e sinais Eletromiográficos, pesquisa que será o norte de todo o projeto, pois vai embasar todo o desenvolvimento do *hardware* para captura do sinal. Paralelamente a isso, será também realizado um estudo sobre como vivem as pessoas com distúrbios neuropsicomotores graves e quais são suas principais necessidades e dificuldades para manter uma comunicação. Além disso será necessário, ainda que seja efetuado um estudo sobre programação em Linguagem Java para tratamento do sinal e auxilio no desenvolvimento do editor de texto que será apresentado ao paciente.
- Etapa 2: Projetar circuito de captação e amplificação de sinal. Utilizando-se do amplificador operacional TLC 274 e filtros RC.
- Etapa 3: Adquirir peças indicadas no projeto e confeccionar placas de circuito impresso de acordo com projeto elaborado na etapa anterior. Após a finalização da confecção, deverão ser testadas as placas para se garantir o seu funcionamento.

- Etapa 4: Desenvolver o *software* na Linguagem de programação escolhida para o projeto. Testar funcionalidades do *software* para garantir que irá atender aos objetivos propostos.
- Etapa 5: Integrar h*ardware* e *software* desenvolvidos e garantir a total funcionalidade desta integração. Após os teste desta etapa de integração será garantida a funcionalidade do projeto como também sua real aplicabilidade.

Complementando a metodologia, os equipamentos listados abaixo serão utilizados:

- Osciloscópio Digital, Multímetro Digital, Gerador de Sinal.
- *Software* para desenvolvimento em programação,

1.4 RESULTADOS ESPERADOS

Ao final do projeto é esperado um protótipo que atenda aos requisitos levantados e apresentados anteriormente no trabalho que apoiaram a aluna na escolha do tema proposto.

O protótipo deverá identificar as sinalizações enviadas pelo paciente que estará observando um *software* de comunicação. Ele deve ser de fácil implementação, com peças que possam ser facilmente adquiridas em lojas próprias para produtos eletrônicos. O *software* deve ser auto instalável e identificar de forma independente o *hardware* envolvido na primeira etapa do projeto. Com a finalização deste, poder-se-á mostrar o quanto a engenharia pode influenciar na vida e no cotidiano de todas as pessoas, e que mesmo um simples

protótipo como o proposto, pode ser viável e de grande ajuda, quando não é possível o acesso a produtos de última geração.

1.5 ORGANIZAÇÃO

Este trabalho será estruturado da seguinte forma: o Capítulo 1 apresenta uma breve explicação sobre o tema a ser abordado neste trabalho e descreve, ainda que superficialmente, os objetivos deste projeto. O capítulo 2 trata sobre a referência teórica do projeto e justifica o contexto no qual o projeto se insere. O capítulo 3 apresenta o dispositivo proposto para solução do problema, detalhando-se a estrutura e seus componentes. O capítulo 4 mostra como o dispositivo se comporta em testes realizados e proporciona que uma avalição do seu funcionamento seja feita. E por fim, o Capítulo 5 apresenta as considerações finais sobre o trabalho, as dificuldades encontradas e as sugestões para trabalhos futuros.

CAPÍTULO 2 REFERENCIAL TEÓRICO

2.1 ELETROMIOGRAFIA

Em 1780, um anatomista italiano, Luigi Galvani (1737-1798), percebeu que os músculos das pernas de uma rã dissecada retorciam quando estimulados por uma corrente elétrica, o que ele definiu como Eletricidade Animal (ASIMOV, 1993). Na mesma época em que Galvani desenvolvia esta pesquisa, Alejandro Volta (1745-1827), ensina Física na Universidade de Pavia em Via Taramelli, na Itália. Enquanto estudava o fenômeno descrito por Galvani, chegou à conclusão de que metais podiam produzir eletricidade e que músculos e nervos são apenas condutores de eletricidade. E acabou por invalidar a descoberta de Galvani (ASIMOV, 1993).

Em uma nova tentativa de provar a ocorrência da Eletricidade Animal, Galvani comprovou que a fonte de energia para a contração do músculo era de origem animal. Para isso, colocou em contato os músculos da pata de uma rã com o nervo ciático de outra. A confirmação de que os dois cientistas estavam certos veio apenas com o tempo e a evolução da ciência (DURAN, 2011).

O sinal registrado como eletromiográfico é captado quando um complexo processo para iniciar um potencial de ação no corpo é desencadeado. O MUAP (Motor Unit Action Potencial) ou potencial de ação é a unidade fundamental do sinal eletromiográficos. Como indicado na Figura 2.1, a ativação

das fibras musculares gera uma superposição de sinais que ao serem adicionados originam o MUAP, indicado na figura por h[t] (BUTTON, 2014).

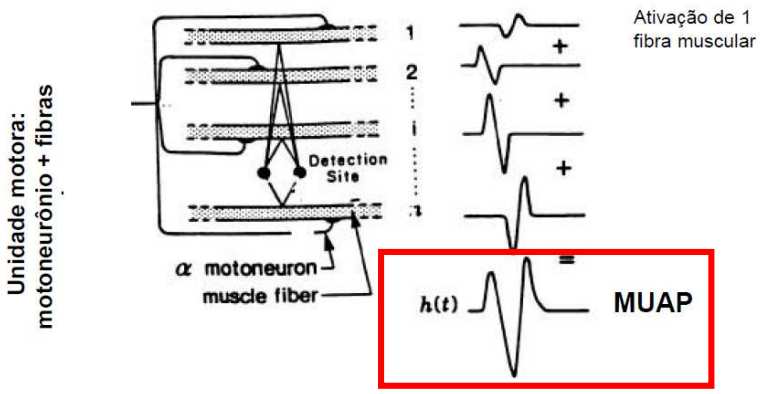


Figura 2.1 Demonstração MUAP (BUTTON, 2014)

Para mostrar o referido processo, a Figura 2.2 ilustra os potenciais de ação ou MUAP´s detectados, em seguida o somatório desses sinais transformado em sinais EMG´s.

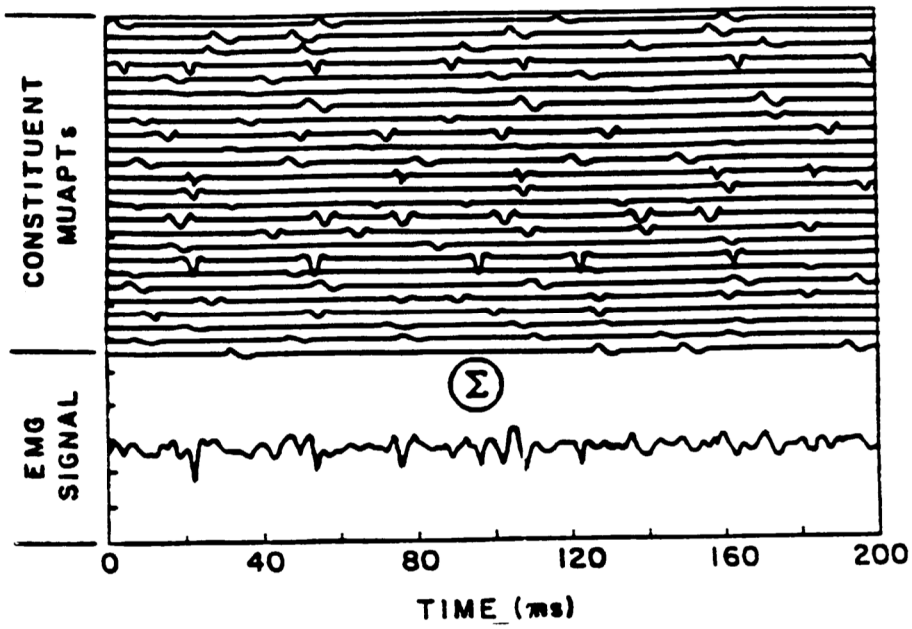


Figura 2.2 Somatório de MUAP´s (BUTTON, 2014)

O sinal de eletromiografia é definido pela equação (MAMÉDIO, DUQUE e WIECHERT, 2011):

$$EMG(t) = \sum_{j=1}^{N} SPAUM\ j(t) + n(t)$$

Onde:

EMG(t): representa o sinal de eletromiografia

SPAUM: representa a série de potenciais de ação da unidade motora

n(t): representa o ruído

t: representa o tempo da amostra

Sua aplicação se encontra em diversas áreas considerando que sua implementação permite uma eficaz avaliação do comportamento dos músculos, em especial quanto à existência ou não de uma lesão na área indicada como também no que concerne nos esportes de alto rendimento (MARCHETTI e DUARTE, 2006). Dependendo do eletrodo utilizado e da técnica escolhida os sinais EMG podem ter uma frequência entre 20Hz e 10kHz e apresentar amplitudes entre 100µV e 90mV (GODOI, 2013).

Os sinais EMG podem ser o resultado de vários outros sinais advindos de uma combinação de eletrodos, podendo ser classificados como monopolar, bipolar e multipolar (DUCHENE e GOUBEL, 1993). Neste projeto será utilizada a configuração bipolar, por ser a mais indicada quando se trata de exercícios de contração voluntária (MARCHETTI e DUARTE, 2006). Esta configuração se caracteriza por dois eletrodos posicionados de forma a captar o sinal do músculo e outro, colocado de preferência em uma área óssea para que possa atuar com um neutro do sistema (Figura 2.3).

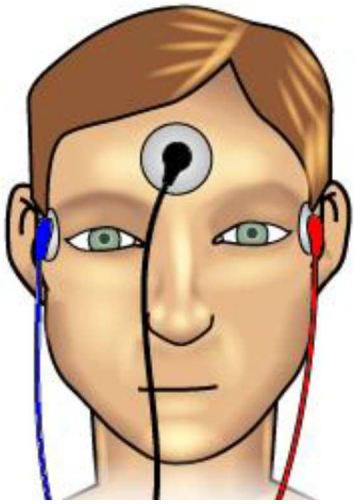


Figura 2.3 Ilustração de aplicação dos eletrodos (MARCHETTI e DUARTE, 2006)

Outra aplicação da captação do sinal EMG é na detecção de neuropatias, desnervação e síndromes de aprisionamento (CIPRIANO, 1999).

2.2 DISTURBIOS NEUROPSICOMOTORES GRAVES

Distúrbios neuropsicomotores graves podem ser considerados aqueles que acometem qualquer tipo de locomoção de uma pessoa. Na literatura médica podem ser encontrados diversos distúrbios que se inserem nessa definição.

Para este trabalho, serão considerados os casos em que o distúrbio ocasione no paciente um estado de tetraplégico, anártico mas que mantenha suas atividades cerebrais normais com isso possibilitando-lhe apenas a movimentação ocular.

Este tipo de acometimento deve ser diferenciado do estado de coma vigil, definido tecnicamente como o estado em que o paciente, apesar de ter os

olhos abertos, não apresenta atividades cerebrais compatíveis com a indicação de consciência.

Para auxiliar os médicos na identificação desse nível de consciência, existe uma escala conhecida como *Glascow Coma Scale (GCS)* ou Escala de Glascow, a qual pode ser utilizada como um fator facilitador para identificação do nível de consciência do paciente, auxiliando ainda na definição do grau de comprometimento crânio-encefálico (AENFERMAGEM, 2008).

Seu funcionamento está baseado na pontuação obtida pelo paciente, de acordo com sua resposta em testes ortopédicos e neurológicos (CIPRIANO, 1999). A Figura 2.4 mostra a pontuação de acordo com respostas obtidas ao se realizarem os testes.

ESCALA DE COMA DE GLASGOW

VARIÁVEIS		ESCORE
Abertura ocular	Espontânea À voz À dor Nenhuma	4 3 2 1
Resposta verbal	Orientada Confusa Palavras inapropriadas Palavras incompreensivas Nenhuma	5 4 3 2 1
Resposta motora	Obedece comandos Localiza dor Movimento de retirada Flexão anormal Extensão anormal Nenhuma	6 5 4 3 2 1

TOTAL MÁXIMO	TOTAL MÍNIMO	INTUBAÇÃO
15	3	8

Figura 2.4 Escala de Glascow (AENFERMAGEM, 2008)

A pontuação mínima por grupo de avaliação é 1, e se refere aos grupos abertura ocular, resposta verbal e resposta motora. A pontuação mínima a ser atingida por um paciente é 3 e o máximo é 15, sendo que quanto mais baixa, menor o nível de consciência encontrado.

A síndrome do encarceramento, distúrbio encontrado na literatura médica que mais se aproxima do objeto estudado por este projeto, possui tal dominação por se referir a pacientes com normais níveis de consciência, mas que se encontram aprisionadas em seu próprio corpo. Essa síndrome é um raro distúrbio neurológico que compromete todos os músculos voluntários do corpo, exceto os responsáveis pelos movimentos oculares (NAPA VALLEY REGISTER, 2012).

Presas a uma cama, essas pessoas têm uma total dependência das outras ao seu redor. A comunicação, na maioria das vezes, está limitada a perguntas elaboradas diretamente ao paciente, com respostas combinadas do tipo: pisque uma vez para sim e duas para não.

De acordo com a literatura especializada, há relatos de casos sobre a completa recuperação de pacientes com a síndrome do encarceramento. Um levantamento relacionado a 35 (trinta e cinco) casos revistos na literatura médica obteve como retorno: dois casos de pacientes que sobreviveram por mais de três anos presos em suas cadeiras de roda, um outro similar que durou mais de um ano, havendo ainda um que passou pouco mais de dois anos se comunicando apenas pela utilização do código Morse com os olhos (FILHO e GOMES, 2012).

2.3 JAVA

Java é uma linguagem de programação lançada pela Sun Microsystems em 1995 (Java, 2014) . Possui como principal atrativo a

portabilidade entre plataformas, com isso evitando a necessidade de manutenção no código de linguagem para adaptação.

O atrativo destacado no parágrafo anterior, teve fundamental relevância para sua escolha na implementação do projeto. Sua independência em relação a um *hardware* específico para execução dispensa a necessidade de desenvolver um *software* para cada plataforma disponível no mercado. Isso influencia diretamente no preço final do projeto, tendo em vista a facilidade de acesso a computadores nos dias atuais.

As Figura 2.5 e Figura 2.6 mostram a diferença de funcionamento entre as outras linguagens de programação disponíveis no mercado e linguagem Java.

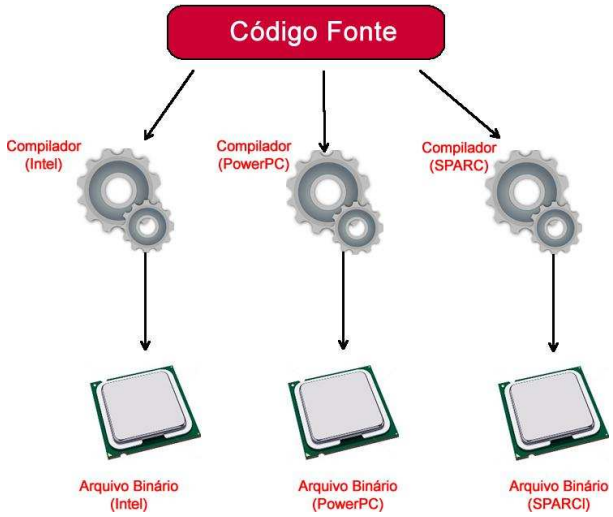


Figura 2.5 Compilação de um programa desenvolvido por outras linguagens (PALMEIRA, 2014)

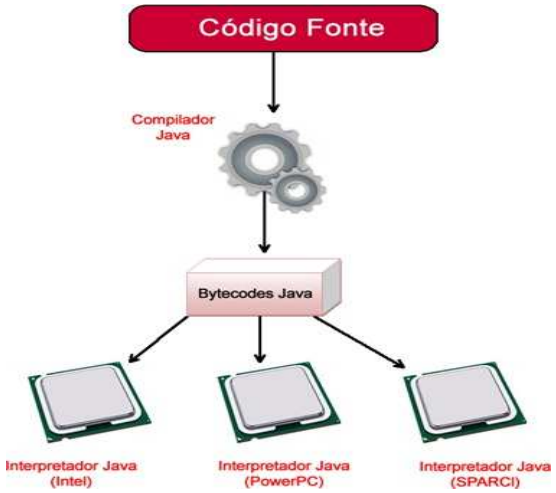


Figura 2.6 Compilação de um programa desenvolvido por Java (PALMEIRA, 2014)

O que possibilita essa independência de execução é um emulador conhecido como Máquina Virtual Java ou JVM (Java Virtual Machine), o termo Máquina virtual se refere a uma máquina simulada rodando dentro de outra máquina real, o que possibilita a criação de uma estrutura com componentes genéricos e permite a execução de sistemas incompatíveis com a máquina real.

Na Figura 2.7 encontra-se algumas justificativas pela qual o modo pelo qual o Java está presente no dia a dia de pessoas de todo o mundo.

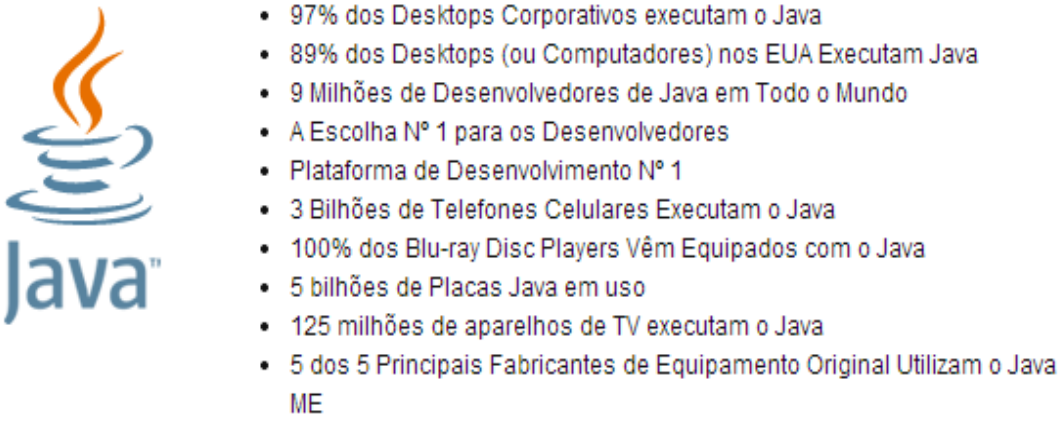


Figura 2.7 Principais Informações sobre a Linguagem Java (Java, 2014)

CAPÍTULO 3 DESENVOLVIMENTO

Neste capítulo será abordado sobre todo o processo de desenvolvimento do sistema e detalhado cada passo e cada decisão a ser tomada.

3.1 DIAGRAMA EM BLOCOS DO PROJETO

O sistema é composto por três etapas de *hardware*, como apresentado no diagrama em blocos da Figura 3.1, assim sendo: captação de sinal pelos eletrodos de superfície, um circuito de entrada e amplificação do sinal e amplificador de áudio e um computador atuando como conversor AD e hospedeiro do *software*.

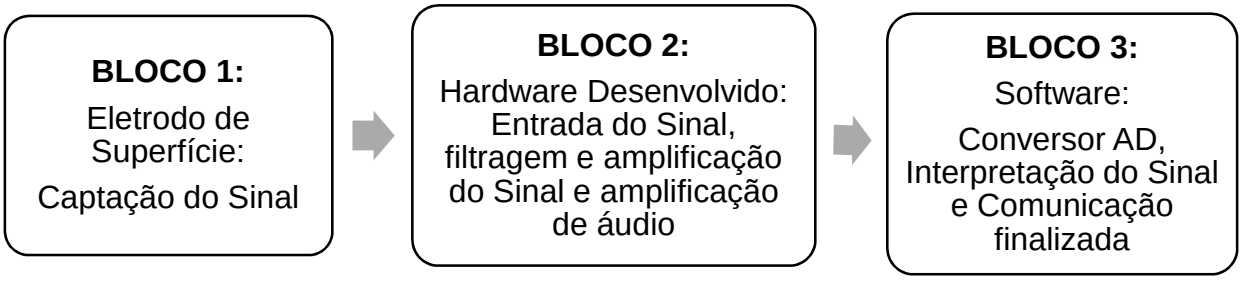


Figura 3.1 Diagrama em blocos

O projeto funciona conforme o fluxograma indicado na Figura 3.2, onde cada etapa é planejada de tal forma que o sistema dispensa qualquer tipo de intervenção para o seu funcionamento. Durante o desenvolvimento deste documento, cada etapa será descrita para que não existam dúvidas quanto ao funcionamento do sistema.

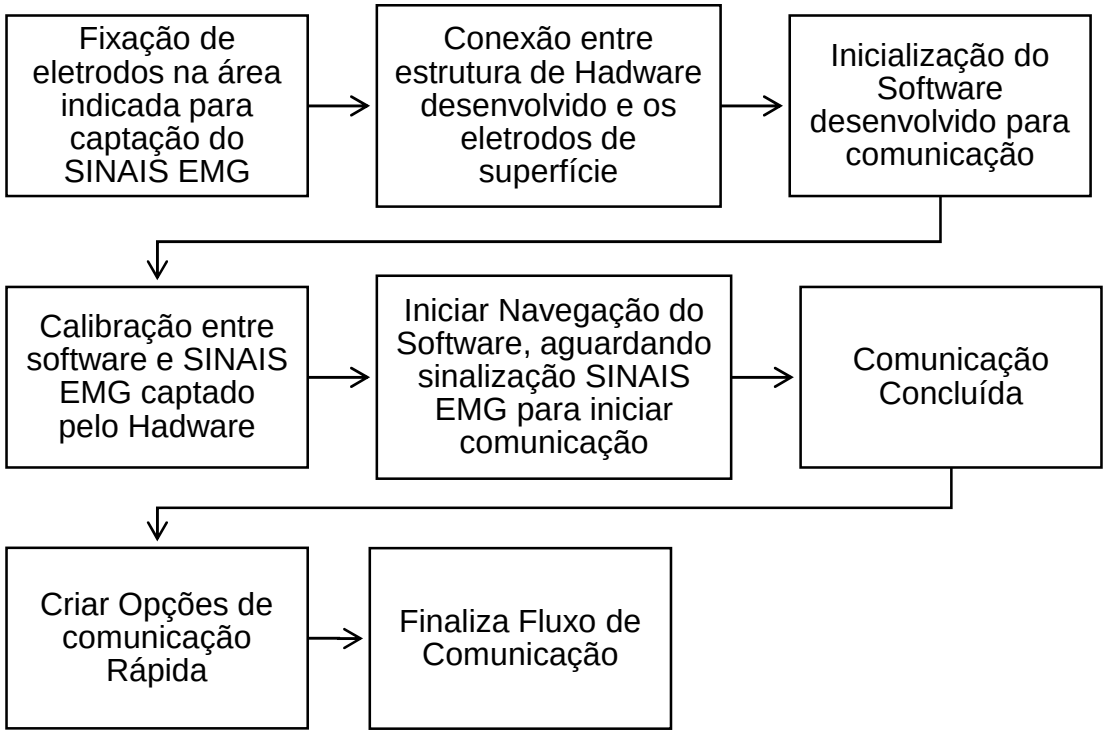


Figura 3.2 Sequência de funcionamento do sistema (Fonte: Daisy Marçal)

3.1.1 BLOCO 1: CAPTAÇÃO DO SINAL

Os eletrodos de superfície mostrados na Figura 3.3, utilizados para captar os SINAIS EMG´s, são da série Kendall Medi-Trace Foam 200. A interface pele-eletrodo permite a detecção do sinais EMG de modo estável e com baixo ruído. Para sua utilização, é recomendável que seja associado a um gel condutor presente no eletrodo. Esses eletrodos são do tipo passivo, ou seja, apenas detectam os sinais EMG e os enviam ao circuito responsável pela amplificação e filtragem dos sinais.

Figura 3.3 Eletrodo de Superfície

Os eletrodos fixados na pele comportam-se como um filtro passa-baixa com o principal objetivo de evitar a passagem de sinais de alta frequência e, conforme implementação no projeto, dificultar a passagem de outros sinais biomédicos presentes nas musculaturas próximas ao músculo medido (MARCHETTI e DUARTE, 2006). No projeto, os eletrodos foram fixados conforme apresentado na Figura 3.4.

Figura 3.4 Eletrodos Aplicados (Fonte: Daisy Marçal)

A colocação dos eletrodos de superfície foi feita de acordo com a Figura 3.5. Os mesmo foram posicionados na região Orbicular do olho, musculatura responsável pela ação de piscar.

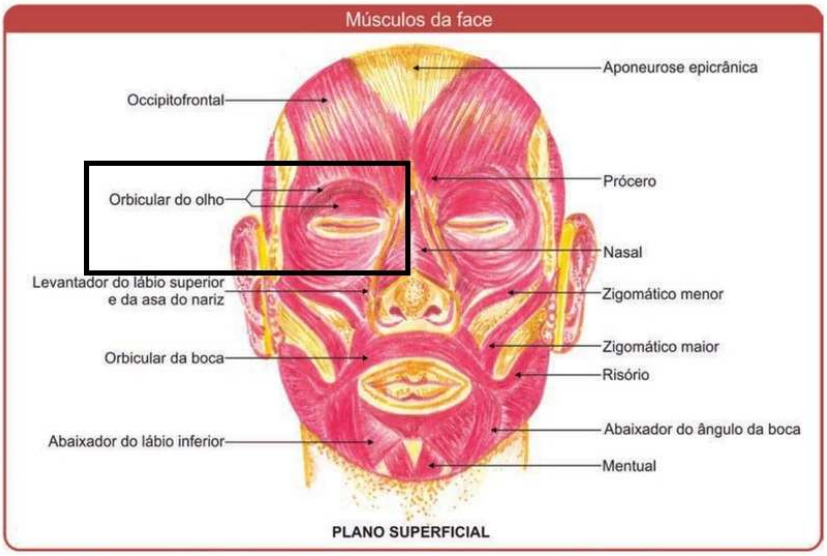


Figura 3.5 Ilustração da musculatura da face (FALAVIGNA e TORNATTO F., 2013)

É recomendável que, ao se fazer a aplicação dos eletrodos, deve-se observar a direção em relação à fibra muscular, conforme Figura 3.6, pois o MUAP é transmitido no sentido das fibras, sendo por isso aconselhável que se estabeleça o mesmo sentido para a colocação dos eletrodos visando uma melhor captação dos sinais.

Figura 3.6 Fibras Musculares Região Orbicular (Fonte: google)

Após a etapa de captação, o sinal de EMG deve ser filtrado e amplificado. Para isso são utilizados como referência alguns estudos já desenvolvidos no âmbito do GER-UniCEUB, relacionados à amplificação de sinais EMG.

3.1.2 BLOCO 2: FILTRAGEM E AMPLIFICAÇÃO DE SINAL

O amplificador diferencial, responsável pela captação do sinal EMG e pela eliminação de ruído, é composto por um amplificador Operacional TLC 274 (TEXAS INSTRUMENTS, 2014) e possui a configuração de amplificador diferencial, conforme esquema elétrico apresentado na Figura 3.7. O TLC274 é um amplificador operacional de alta sensibilidade e alta impedância, que possibilita a captação dos sinais EMG (MAMÉDIO, DUQUE e WIECHERT, 2011).

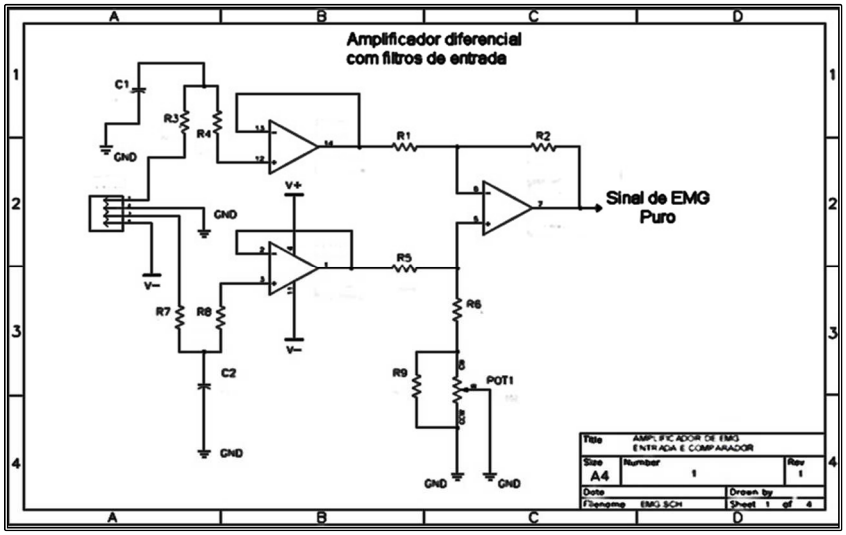


Figura 3.7 Esquema elétrico Amplificador Diferencial (MAMÉDIO, DUQUE e WIECHERT, 2011)

O referido amplificador deverá ser associado a um amplificador de áudio, conforme esquema elétrico mostrado na Figura 3.8, sendo esta uma etapa que possibilitará direcionar esse sinal para o receptor (computador), possibilitando assim, realizar-se a parte da etapa final de tratamento e interpretação do sinal.

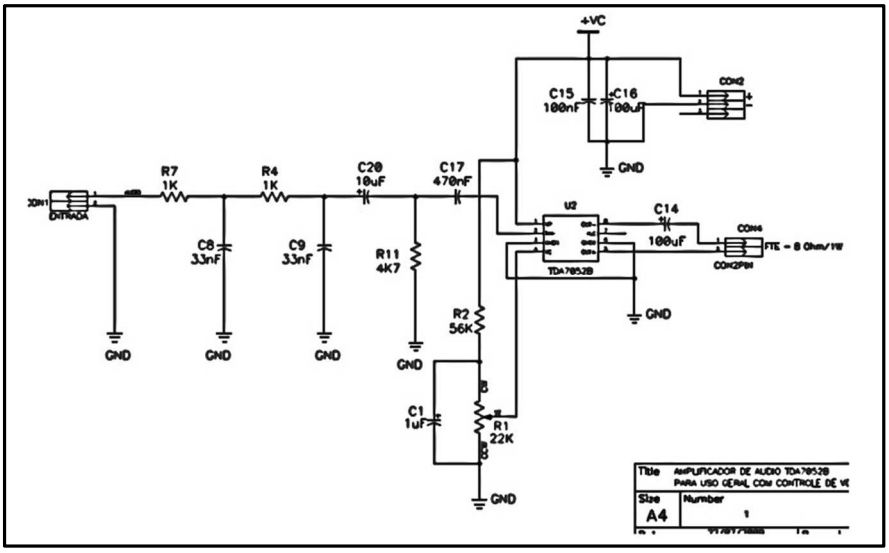


Figura 3.8 Esquema elétrico Amplificador de áudio (MAMÉDIO, DUQUE e WIECHERT, 2011)

O amplificador diferencial possui na sua entrada dois filtros RC para eliminação de ruídos ou outras atividades em musculaturas próximas (DUQUE e LOBO, 2013). Esse *hardware* também influencia-se pelo posicionamento e pela configuração dos eletrodos, permitindo apenas a transmissão da frequência oriunda dos sinais EMG.

Pode-se obter a frequência de corte dos filtros por meio da fórmula:

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

Onde:

f = frequência;

R = Resistor em Ohms

C = Capacitor em uF

O amplificador diferencial irá atuar como amplificador dos sinais EMG capturados pelos eletrodos. Ele foi montado na placa de circuitos conforme mostrado na Figura 3.9.

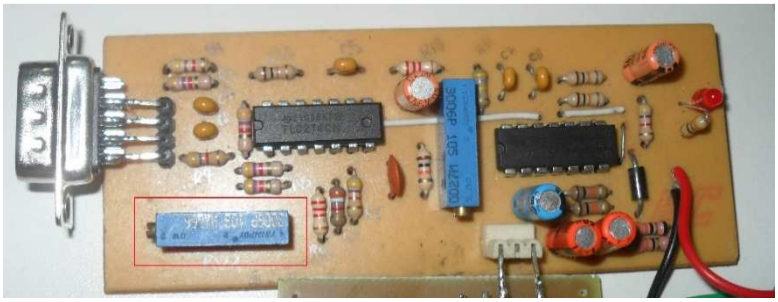


Figura 3.9 Amplificador de sinais EMG (Fonte: Daisy Marçal)

Depois de terem sido executadas as etapas expostas nos parágrafos anteriores, tornou-se necessária a produção de um circuito para amplificação de áudio, de modo a permitir a utilização da placa de entrada de áudio do computador. Dessa forma, elimina-se a necessidade da utilização de um conversor analógico-digital (AD) no projeto, reduzindo-se assim tanto o seu custo final, quanto o nível de complexidade relativo à execução do projeto.

Primeiramente montou-se o circuito de amplificação em uma Protoboard, conforme Figura 3.10, visando a realização dos testes iniciais de filtragens e amplificação.

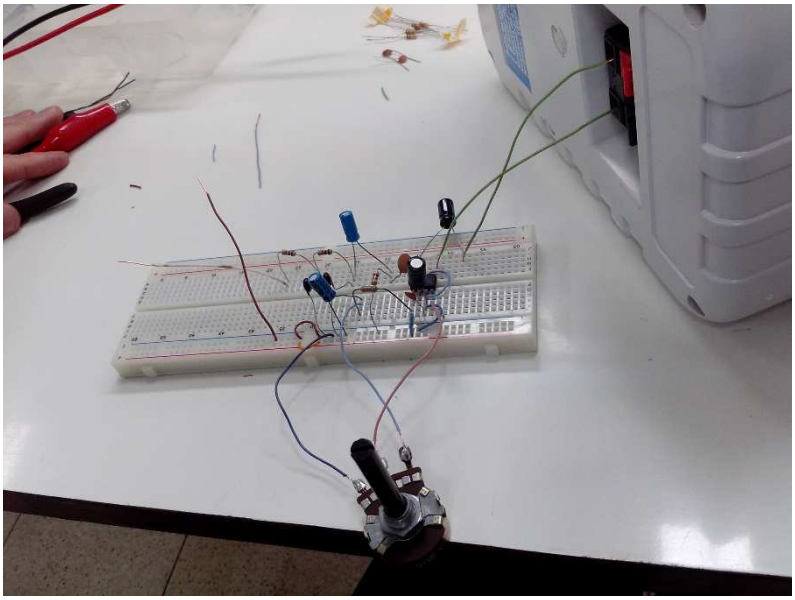


Figura 3.10 Amplificador de Áudio na Protoboard (Fonte: Daisy Marçal)

Para os testes iniciais, utilizou-se um gerador de sinal calibrado na frequência de 1Khz e também uma caixa de som. Ao se injetar o sinal do gerador, ocorreu a filtragem e a amplificação do sinal, tendo o mesmo sendo emitido pelo autofalante na saída do protótipo. O sinal emitido pelo autofalante tem como o frequência os mesmos 1Khz do sinal de entrada. Após garantir o seu perfeito funcionamento o circuito foi montando em uma placa de circuito perfurada como mostrado na Figura 3.11.

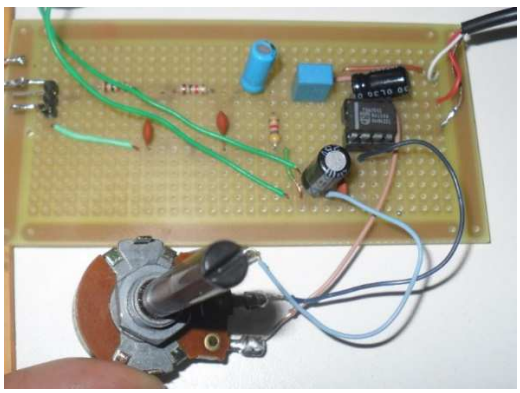


Figura 3.11 Amplificador de áudio na placa de circuito perfurada (Fonte: Daisy Marçal)

Na etapa seguinte, procedeu-se à integração do amplificador de EMG e do amplificador de áudio, conforme demonstrado na Figura 3.12. Utilizou-se ainda do osciloscópio e do gerador de sinal, este último tendo sido calibrado para gerar um sinal de 1Khz e com 1V de amplitude de pico a pico, e o osciloscópio para mensurar o sinais de calibração. Verifica-se que tensões acima disso podem danificar o amplificador operacional TLC274 (MAMÉDIO, DUQUE e WIECHERT, 2011).

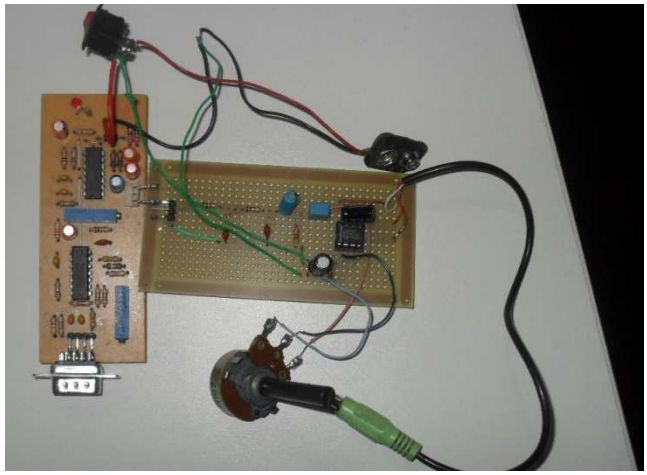


Figura 3.12 Imagem integração dos circuitos de amplificação e áudio (Fonte: Daisy Marçal)

Este amplificador combinado com o CMRR (Commom-Mode Rejection Ratio), apontado na Figura 3.13, mostrou-se a melhor escolha para o desenvolvimento deste projeto (TEXAS INSTRUMENTS, 2014, p. 5). O CMRR é também um amplificador operacional que nesta implementação tem como função equalizar os sinais de entrada captados pelos eletrodos.

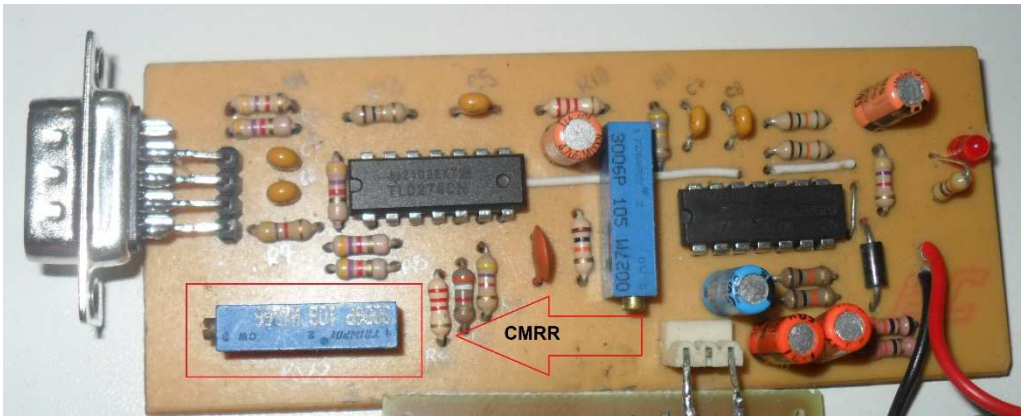


Figura 3.13 CMRR (Fonte: Daisy Marçal)

A citada técnica de calibração tem o intuito de simular um ruído na entrada do amplificador diferencial como também de eliminá-lo. Essa eliminação é possível utilizando-se o CMRR, onde utilizando-se de um gerador de sinais configurado para gerar um sinal de 1kHz e 1V vpp conectado aos dois eletrodos ativos, sendo a rejeição ajustada no CMRR, conforme mostrado na Figura 3.13.

Depois de verificado o sinal de entrada e o sinal de saída, ajustou-se o CMRR conforme descrito no parágrafo anterior, até que o sinal de saída do circuito tenha sido igual a 0, garantindo assim a calibração do protótipo.

Após as etapas anteriores, o circuito apresentava-se pronto para ser testado, eliminando-se o gerador de sinais e utilizando-se dos eletrodos de superfície. Ainda com o auxílio do osciloscópio como visualizador de saída dos sinais gerados, os eletrodos foram aplicados na região orbicular do paciente e iniciaram-se as medições reais, como mostrado na Figura 3.14.

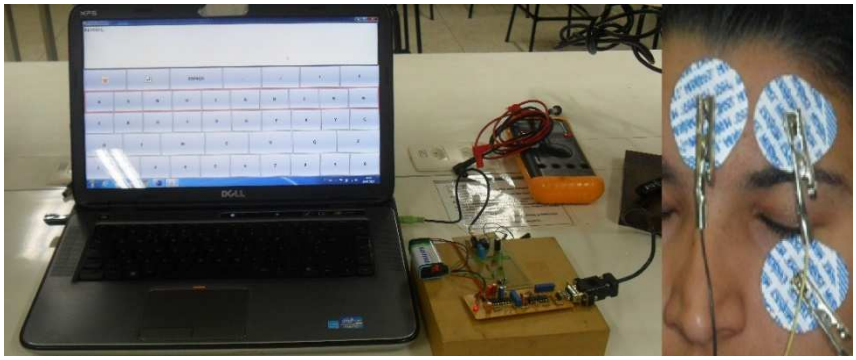


Figura 3.14 Circuito completamente integrado e pronto para execução da próxima etapa (Fonte: Daisy Marçal)

Depois de aplicados os eletrodos, foram gerados sinais EMG voluntários para o teste do circuito. Como demonstrado na Figura 3.15, o sinal de saída mostrou-se de acordo com os resultados esperados, o que possibilitou a inicialização da etapa seguinte do projeto, relativa à integração entre o *hardware* e o *software*.

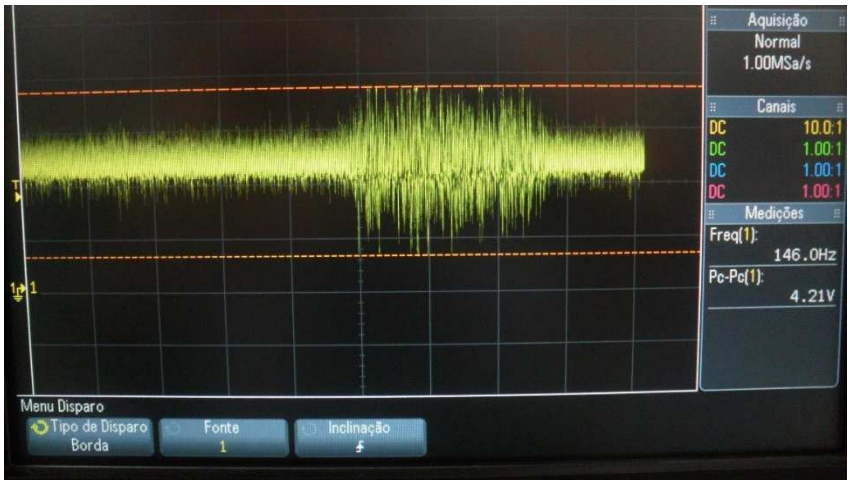


Figura 3.15 Sinal EMG representado no osciloscópio (Fonte: Daisy Marçal)

A integração entre *hardware* e *software* se dá primeiramente por meio da conversão do sinal analógico para digital, pois, conforme já abordado neste documento, o sinal EMG se trata de um sinal analógico e para que seja entendido pela máquina deve ser convertido em um sinal digital.

Visando à solução deste problema escolheu-se a placa de som mostrada na Figura 3.16, que é comum a quase todos os computadores de utilização pessoal disponíveis no mercado. Sua solução é suficiente para captação do sinal EMG.

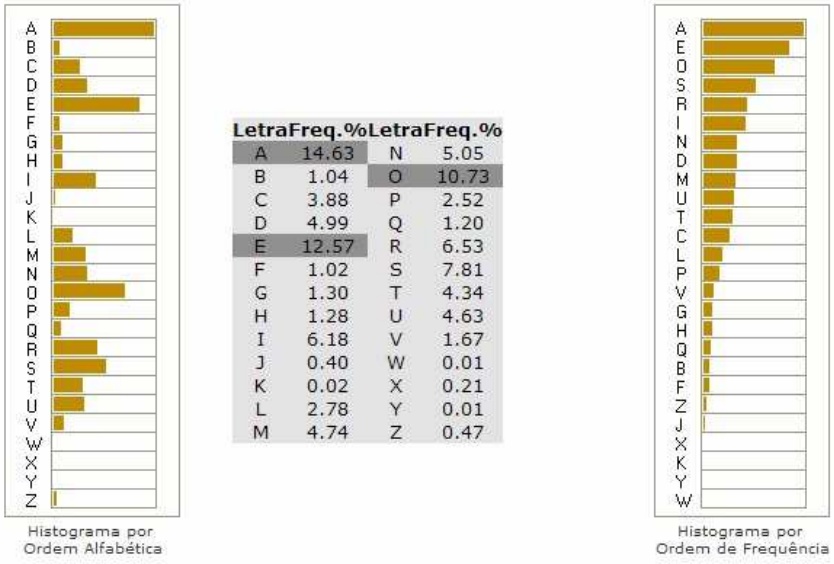
Figura 3.16 Placa de áudio disponível em computadores pessoais (Fonte: Daisy Marçal)

3.1.3 BLOCO 3: *SOFTWARE*

O projeto requer também o desenvolvimento de um *software* capaz de identificar a captura do sinal, com isso possibilitar a comunicação. Também é necessário desenhar o seu fluxo de funcionamento para que seja possível uma maior otimização, pois se tratando de um sistema lento qualquer ganho durante sua execução pode fazer uma grande diferença.

A primeira escolha a ser feita relaciona-se quanto ao *layout* do teclado virtual a ser apresentado para o usuário. O *layout* escolhido para o projeto refere-se ao teclado a que os usuários já mostra estar acostumados no dia a dia. Em seguida deve-se decidir quanto ao sequenciamento de letras a ser apresentado.

Segundo o site Aldeia NumaBoa, ao analisar seis textos escritos por escritores brasileiros conhecidos, mas de épocas diferentes, com 157.764 palavras e 725.511 letras, identificou-se a frequência de utilização de letras do alfabeto na língua portuguesa (ALDEIA NUMABOA, 2014), conforme mostrado na Figura 3.17.



Letra	Freq.%	Letra	Freq.%
A	14.63	N	5.05
B	1.04	O	10.73
C	3.88	P	2.52
D	4.99	Q	1.20
E	12.57	R	6.53
F	1.02	S	7.81
G	1.30	T	4.34
H	1.28	U	4.63
I	6.18	V	1.67
J	0.40	W	0.01
K	0.02	X	0.21
L	2.78	Y	0.01
M	4.74	Z	0.47

Figura 3.17 Frequência de utilização das letras do alfabeto na língua portuguesa (ALDEIA NUMABOA, 2014)

Quando organizadas por frequência de utilização, as letras assumem a organização indicada pela Figura 3.18.

1. A, E, O
2. S, R, I
3. N, D, M, U, T, C
4. L, P, V, G, H, Q, B, F
5. Z, J, X, K, W, Y

Figura 3.18 Organização quanto a frequência de utilização (ALDEIA NUMABOA, 2014)

A partir das informações conhecidas acima, tornou-se possível a projeção de um teclado que viesse a melhorar a performance do protótipo. A solução que se mostrou mais eficaz refere-se à divisão do teclado alfabético em três linhas de distribuição, colocando-se como primeira letra de cada linha aquela que seja mais utilizada, formando assim o teclado indicado na Figura 3.19.

Figura 3.19 Teclado formado de acordo com frequência de utilização (Fonte: Daisy Marçal)

Apresenta-se como necessária também a inserção de teclas de atalho para otimização da utilização do sistema, ou seja, ao invés de se criar uma nova linha com vários caracteres, deve ser criada uma nova linha com teclas de atalhos para novas telas de navegação com os caracteres restantes. Desta forma, elimina-se a necessidade de o sistema navegar em teclas menos utilizadas durante sua execução. A referida linha criada deu-se o nome de acesso rápido (Figura 3.20), a qual apresenta, entre suas opções, o acesso a uma página de favoritos, onde estariam salvas frases mais utilizadas ou até mesmo após um tempo de utilização do *software* frases salvas pelo próprios pacientes que por ventura venham a utilizar o protótipo.

Figura 3.20 Linha de acesso rápido (Fonte: Daisy Marçal)

No caso em estudo, para o usuário que irá auxiliar o paciente na utilização do protótipo, será disponibilizado um acesso para configuração e calibração manual do sistema, no intuito de se deixar o sistema mais estável, dispensando-se ainda a necessidade de alteração no código fonte toda vez em que fosse necessária uma alteração.

No que concerne à imagem a ser apresentada para o utilizador do sistema, chegou-se ao *layout* final apresentado na Figura 3.21.

Figura 3.21 *Layout* final *software* (Fonte: Daisy Marçal)

Posteriormente, torna-se necessária a criação de um fluxo de funcionamento do *software*, mostrado na Figura 3.22, que sirva de direcionamento para o desenvolvimento do *software*.

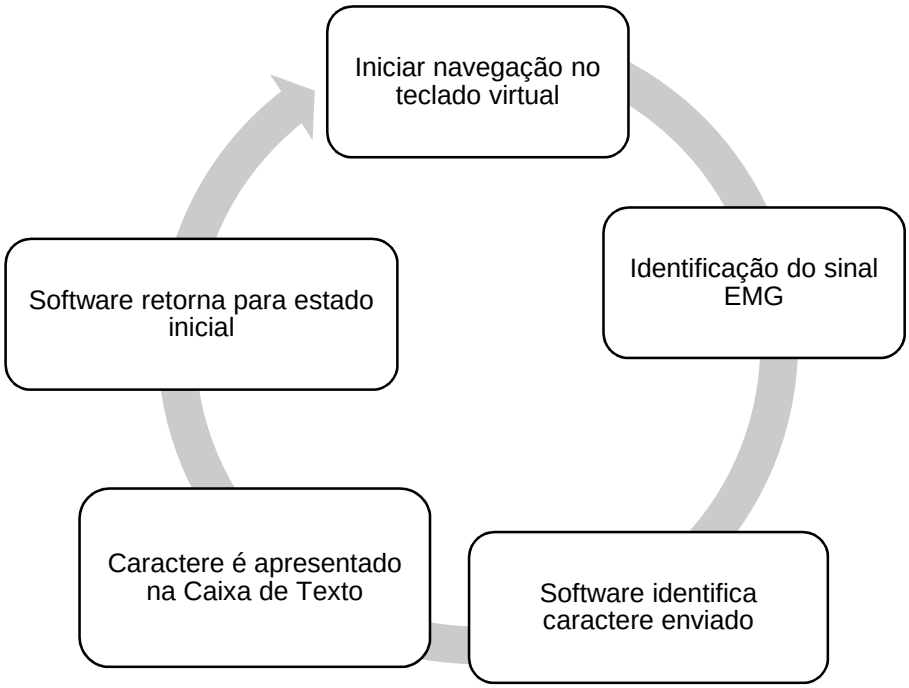


Figura 3.22 Fluxo de funcionamento do *Software* (Fonte: Daisy Marçal)

Partindo-se para codificação de execução do sistema, a primeira implementação a ser feita é a navegação. Por se tratar de um sistema autônomo, decidiu-se que o sentido de deslocamento do ponteiro para seleção do paciente seria primeiro na vertical e depois na horizontal, conforme codificação indicada na Listagem 3.1.

```
public void run(){
        setWalk(true);
        while(isWalk()) {
            try {
                for(Block block: getSidewalk()){
                    highlightComponent(block.getKeyLine());

                    for(int i =0 ; i<5; i++){
                        if(hasSignal()){
                            resetSignal();
                            sleep(1000l);
                            for(JButton button: block.getKeys()){

                                highlightComponent(button);

                                for(int j=0; j<5 ; j++){
                                    if(hasSignal()){
                                        resetSignal();
                                        sleep(1000l);
                                        button.doClick();
                                        this.reset = true;
                                        disacknowledgeButton(button);
                                        break;
                                    }

                                    sleep(getSleepInterval());
                                }
                                disacknowledgeButton(button);
                                if(reset){
                                    this.reset = false;
                                    break;
                                }
                            }
                        }
                        sleep(getSleepInterval());
                    }
                    disacknowledgeButton(block.getKeyLine());
                }
            }
            catch( Exception e ) {}
        }
```

Listagem 3.1 Codificação para navegação entre os Blocos do teclado – Classe KeyboardWalker

Em seguida, as linhas de navegação foram divididas em blocos e ao serem acessadas cada botão se torna um bloco independente para acionamento. Assim que selecionada a tecla alvo do paciente o sistema retorna para um nível hierárquico superior.

Conforme dito anteriormente, enquanto navega, o sistema fica aguardando a sinalização do paciente. O sinal é capturado na entrada de áudio do computador conforme mostrado na Listagem 3.2.

```
public void run() {
        int qtdeBytesLidos;
        byte[] data = new byte[this.bufferSize];

        line.start();

        System.out.println("Iniciou a linha...");

        try {
            while (!parar) {
                qtdeBytesLidos = line.read(data, 0,
data.length);
                out.write(data, 0, qtdeBytesLidos);
            }
            line.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Listagem 3.2 Implementação de Captura do Áudio

Depois de capturado o áudio, precisa-se verificar para garantir a existência ou não do sinal EMG na amostra capturada. Essa verificação realiza-se pelo processo da classe *LeitorDeAmostraDetectorDeSinal,* conforme mostrada na Listagem 3.3.

```
public void run() {
        try {
            int periodoMedia = 5;
            int[] mediaBuffer = new int[periodoMedia];
            int amplitude;
            double limiar = 180.0;
            boolean temSinal = false;
            double media = 0.0;
            int i = 0;
            while (!parar) {
                amplitude = ais.lerAmostra();

                mediaBuffer[i % periodoMedia] = amplitude;

                if (i % periodoMedia == 0) {
                    media /= periodoMedia;

                    System.out.println("Media: " + media);
                    if (!temSinal && media > limiar) {
                        temSinal = true;
                        for(ISignalListener listener:
signalListeners){
                            listener.notifySignal();
                        }
                        temSinal = false;
                    }
                    media = 0.0;

                }
                media += Math.abs(amplitude);
                i++;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Listagem 3.3 Verificação de amostras de Sinais de entrada

O áudio capturado deve ter uma taxa de amostragem de 8000 amostras por segundo de 16 bits cada, capturada em um canal mono. Esse formato de áudio é definido no construtor da classe *LeitorDeSinalDeAudio*, conforme demonstrado na Listagem 3.4.

```
public LeitorDeSinalDeAudio(AudioFormat af, int bufferSize)
                throws LineUnavailableException {
        this.format = af;
        this.bufferSize = bufferSize;

        DataLine.Info info = new DataLine.Info(TargetDataLine.class,
format);
        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Não tem suporte");
        } else {
            System.out.println("Tem suporte");
        }

        line = (TargetDataLine) AudioSystem.getLine(info);
    }

    public LeitorDeSinalDeAudio() throws LineUnavailableException {
        this(new AudioFormat(8000.0f, 16, 1, true, true), 2048);
    }
```

Listagem 3.4 Definição formato de áudio

Em seguida à captação e verificação do áudio, o sistema informa a classe KeyboardListen que, ao receber o sinal, escreve na caixa de texto destinada a auxiliar na comunicação. Esta etapa desenvolve-se conforme demonstrado na Listagem 3.5

```
public void actionPerformed(ActionEvent e) {
    JTextArea textBox = getMainForm().getTextBox();
    if(e.getActionCommand().equals("RETURN")){
        textBox.setText(textBox.getText() + "\n");
    } else if(e.getActionCommand().equals("SPACE")){
        textBox.setText(textBox.getText() + " ");
    } else {
        textBox.setText(textBox.getText() + e.getActionCommand());
    }
}
```

Listagem 3.5 Implementação de código de escrita

CAPÍTULO 4 TESTES E RESULTADOS

Os testes necessários para garantir o funcionamento do sistema final são realizados a cada etapa concluída, pois assim tem-se um risco mínimo de que a ocorrência de um erro venha prejudicar o resultado.

A placa de amplificador diferencial, é responsável pela captação do sinal EMG, tendo sido a primeira a iniciar a fase de testes. Na entrada de sinal foi ligado um gerador de sinal configurado para gerar um sinal de 1KHz e 1 V vpp e tendo sido a saída validada por um osciloscópio conectado na saída do circuito.

A segunda fase de teste, feita após a integração dos circuitos de filtragem e amplificação de sinal, serviu para que o sistema fosse calibrado para eliminação de ruídos, a Figura 4.1 mostra a sequência de calibração do sistema.

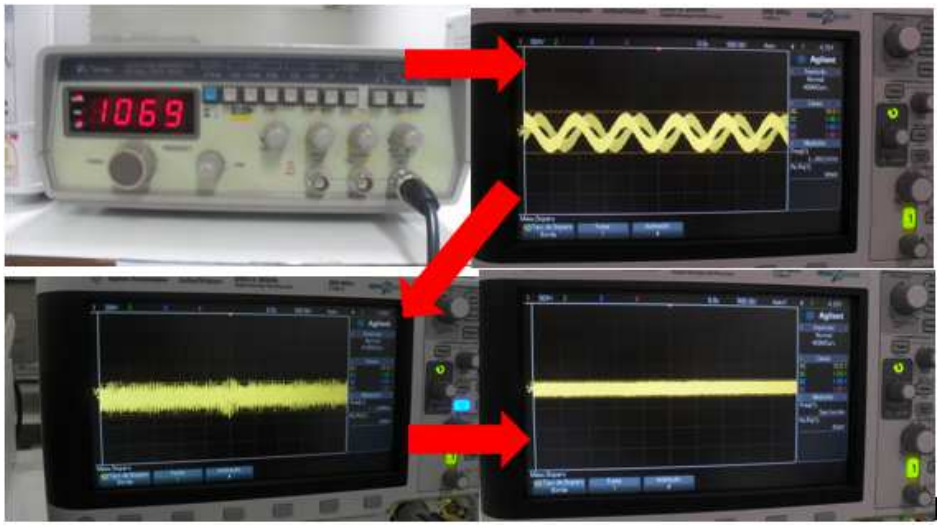


Figura 4.1 Sequência de calibração do Sistema

(Fonte: Daisy Marçal)

Depois de calibrado, o sistema deve ser ligado ao computador utilizado para auxiliar na comunicação. Para validação do sinal recebido pelo

computador antes que o *software* tenha sido inicializado, torna-se necessário a utilização do programa *Goldwave*. Sua versão grátis disponível na internet é suficiente para a verificação do sinal EMG capturado pelo *Hardware*. A figura 4.2 mostra o sinal adquirido no teste da etapa descrita.

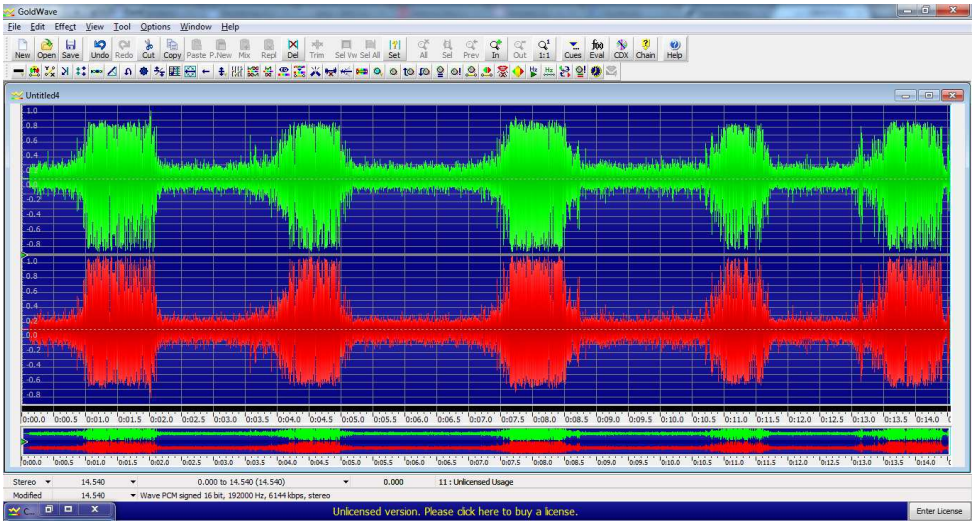


Figura 4.2 Sinal EMG captado pelo áudio do computador representado no GOLDWAVE (Fonte: Daisy Marçal)

Ainda em um outra etapa de testes será necessário a medida das tensões de saída do sistema. Para isso utilizando-se do osciloscópio digital para validar os valores de saída foi obtido um sinal com frequência de 146Hz e 4.21V de pico a pico, monstrado na Figura 4.3. Esse valor de 4.21 pode parecer alto mas se justifica pela presença de amplificadores de sinais existentes no projeto para melhor captação, tratamento e interpretação.

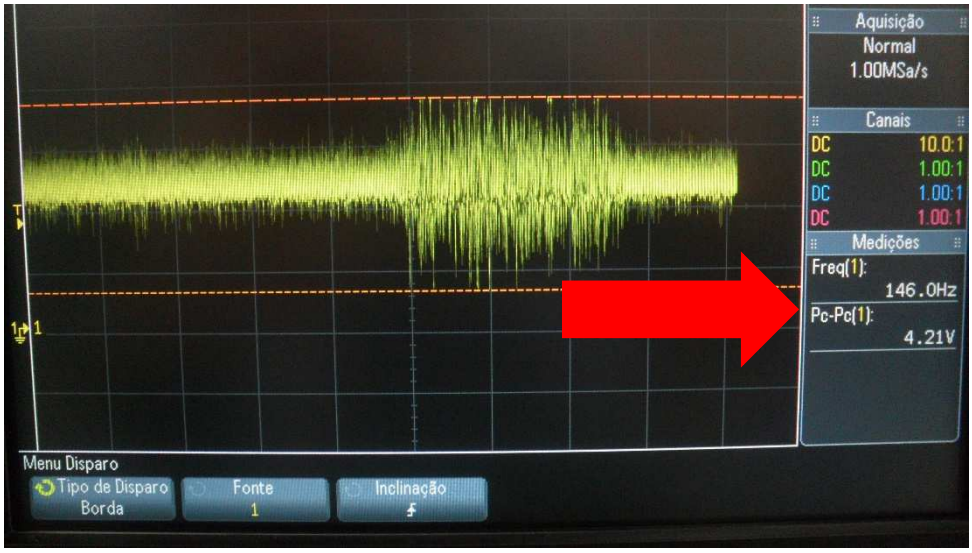


Figura 4.3 Sinal demonstrado em osciloscópio digital (Fonte: Daisy Marçal)

Uma vez validado o sinal, pode-se garantir que o *hardware* está funcionando corretamente e que o sistema está pronto para sua integração com a parte gráfica do projeto. Durante a inicialização do sistema gráfico desenvolvido para o projeto é feito um teste para validar se o computador escolhido oferece suporte para a captação do sinal, a resposta do sistema pode ser vista conforme Figura 4.4.

Figura 4.4 Validação de suporte do computador para inicialização do sistema (Fonte: Daisy Marçal)

Com validação do suporte, o *software* é iniciado. Para um primeiro teste foi feito a simulação da comunicação com a palavra “OI”, conforme mostrado nas Figura 4.5 até Figura 4.8.

Figura 4.5 *Software* Inicia navegação

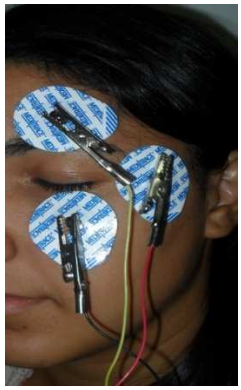


Figura 4.6 *Software* detecta sinalização enviada para iniciar escrita

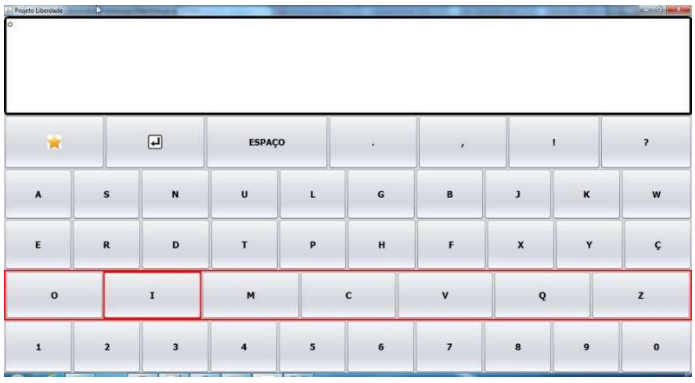


Figura 4.7 Após detecção *software* escreve conforme indicação do paciente

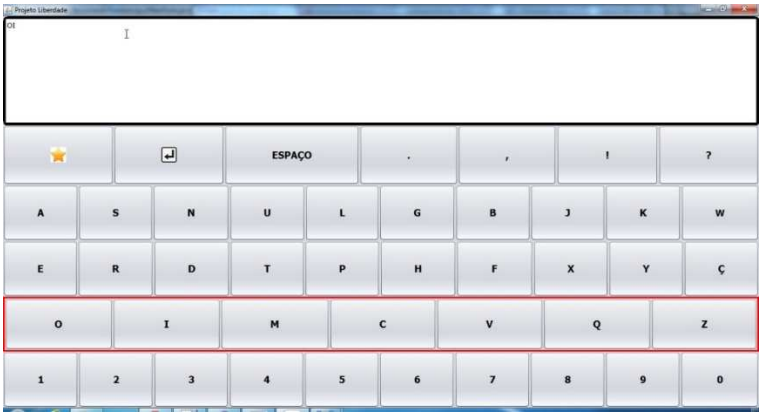


Figura 4.8 Finalização da escrita “OI” conforme teste proposto

Outra funcionalidade a ser testada é a pagina de acesso aos favoritos do sistema, mostrado pela Figura 4.9, a função se mostrou satisfatória e com um ganho real de tempo de execução.

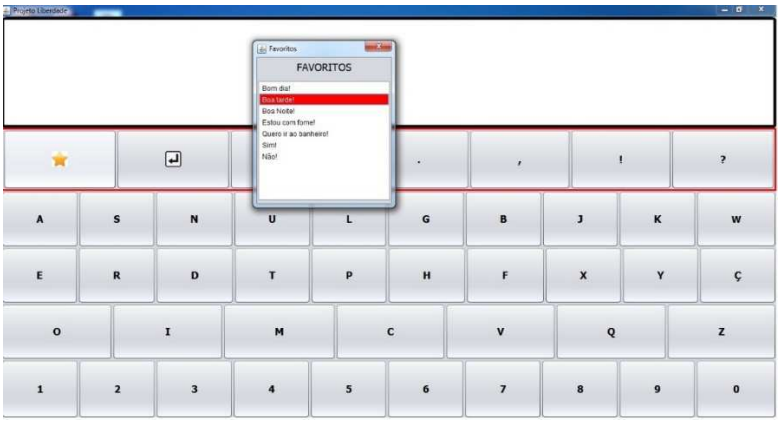


Figura 4.9 Tela Frases favoritas (Fonte: Daisy Marçal)

Tendo como parâmetros os resultados dos testes realizados acima é possível afirmar que o protótipo atende aos objetivos propostos e está pronto para ser apresentado.

CAPÍTULO 5 CONCLUSÃO E SUGESTÃO DE TRABALHOS FUTUROS

Por meio deste projeto é possível mostrar a relevância da engenharia como um diferencial na vida das pessoas. A cada dia que passa e a cada tecnologia nova inventada, tem-se deixado perder várias oportunidades de utilização da ciência como forma de ajudar ao próximo, muito vezes descartando-se ideias por se priorizar apenas o lucro.

O projeto atende ao objetivo inicial de possibilitar a comunicação para pessoas com distúrbios neuropsicomotores graves com o mundo externo. Essa possibilidade de se conquistar tal independência é vista como um grande diferencial e uma mudança drástica, tanto no que se refere ao comportamento do paciente tratado, como na interpretação do ponto de vista médico em relação à situação por este enfrentada.

Esse projeto está aberto a novas possibilidades de desenvolvimento que aqui não foram implementadas, por falta de tempo hábil para execução. O cenário tecnológico atual permite que a comunicação ultrapasse os limites do ambiente hospitalar e da comunicação apenas entre o paciente, seus familiares e amigos próximos, que por ventura venham a visita-lo.

A integração deste projeto com o mundo virtual pode significar uma nova abordagem na vida desses pacientes, possibilitando-lhes a troca de mensagens de textos, elaboração de diários virtuais onde possam falar de suas vidas e trocar informações com desconhecidos assim como é feito habitualmente em sites de relacionamento e outros afins.

Por isso é tido como um grande passo para este projeto a possibilidade de integração ao mundo externo. Entende-se então que, este projeto é apenas um primeiro passo, sem limites de crescimento e com infinitas possibilidades de melhoria e aplicação.

BIBLIOGRAFIA

AENFERMAGEM. Aenfermagem. Aenfermagem, 28 nov. 2008. Disponivel em: <http://aenfermagem.com.br/materia/escala-de-coma-de-glasgow/>.

ALDEIA NUMABOA. Aldeia Numaboa. Aldeia Numaboa, 22 fev. 2014. Disponivel em: <http://www.numaboa.com.br/criptografia/criptoanalise/310-Frequencia-no-Portugues>.

ASIMOV, I. Cronologia das ciências e das descobertas. In: ASIMOV, I. Cronologia das ciências e das descobertas. Tradução de Ana Zelma Campos. [S.l.]: Civilização Brasileira S/A, 1993. p. 391.

BUTTON, V. L. S. IA 748 - Instrumentação Biomédica. scribd, 16 abril 2014. Disponivel em: <http://pt.scribd.com/doc/85235099/Eletromiografo-Instrumentacao-Biomedica-2002>. Acesso em: 02 mar. 2014.

CIPRIANO, J. J. Manual Fotográfico De Testes Ortopédicos e Neurológicos. Atlanta: Manole LTDA, 1999.

CUBIEBOARD , 13 maio 2014. Disponivel em: <http://www.cubieboard.org>.

DUCHENE, J.; GOUBEL, F. Surface electromyogram during voluntary contraction: processing tools and relation to physiological events. Critical Reviews in Biomedical Engineering, New York, 21, 1993. 313 - 397.

DUQUE, L.; LOBO, R. Dispositivo de Comunicação Alternativa para Pessoas com Paralisia Cerebral, 2013.

DURAN, J. E. R. Biofísica: conceitos e aplicações. São Paulo: Pearson Prentice Hall, 2011.

FALAVIGNA, A.; TORNATTO F., A. J. Anatomia Humana. Caxias do Sul: EDUCS, 2013.

FILHO, F.; GOMES, M. D. P. SciELO - Scientific Electronic Library Online. SciELO Brasil, Brasilia, 06 fev. 2012. Disponivel em: <http://www.scielo.br/pdf/anp/v40n3/13.pdf>. Acesso em: 05 mar. 2014.

GODOI, T. PRÓTESE MIOELÉTRICA CONTROLADA POR REDES NEURAIS. Brasília: [s.n.], 2013.

HENEINE, I. F. Biofísica Básica. São Paulo: Atheneu, 2006. p. 227.

INDIEGOGO , 17 abr. 2014. Disponivel em: <indiagoo.com>.

JAVA. Oracle, 01 abr. 2014. Disponivel em: <http://java.com/pt_BR/download/faq/helpful_concepts.xml>.

MAMÉDIO, R.; DUQUE, L.; WIECHERT, D. Dispositivo de Comunicação Alternativa para pessoas com Paralisia Cerebral. Anuário da produção de iniciação cientifica discente, Brasília, 2011.

MARCHETTI, P. H.; DUARTE, M. Instrumentação em Eletromiografia. FEFISO, São Paulo, 2006. Disponivel em: <www.fefiso.edu.br/grupoestudo/pdfs/06.pdf>. Acesso em: 2014 abr. 2014.

NAPA VALLEY REGISTER. Napa Valley Register. Napa Valley Register, 15 ago. 2012. Disponivel em: <Survivor of 'locked-in syndrome' now able to lead nearly normal life>. Acesso em: 06 maio 2014.

PALMEIRA, T. V. V. DEVMEDIA. DEVMEDIA, 01 maio 2014. Disponivel em: <http://www.devmedia.com.br/java-historia-e-principais-conceitos/25178>. Acesso em: 05 maio 2014.

TEXAS INSTRUMENTS. Texas Instruments. Texas Instruments, 02 maio 2014. Disponivel em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/28864/TI/TLC274.html>. Acesso em: 02 abr. 2014.

APÊNDICE A - Código

```
package br.com.tomas.emgprocessing.sinal;

import java.io.IOException;

import java.io.InputStream;

public class AmostraAudioInputStream extends AmostraInputStream {

    public AmostraAudioInputStream(InputStream in) {

            super(in);

    }

    public int lerAmostra() throws IOException {

            byte high = (byte) in.read();

            byte low = (byte) in.read();

            int amostra = getSixteenBitSample(high, low);

            return amostra;

    }


    private int getSixteenBitSample(int high, int low) {

            return (high << 8) + (low & 0x00ff);

    }

}


package br.com.tomas.emgprocessing.sinal;
```

```
import java.io.FilterInputStream;

import java.io.IOException;

import java.io.InputStream;


/**

 * Classe abstrata que representa um filtro sobre um InputStream para recuperar

 * amostras de acordo com o formato especfico do InputStream

 *

 * @author

 *

 */

public abstract class AmostraInputStream extends FilterInputStream {


    protected AmostraInputStream(InputStream in) {

            super(in);

    }


    public abstract int lerAmostra() throws IOException;


}


package br.com.tomas.emgprocessing.sinal;
```

```
public interface AmostraListener {

    void amostrasChegaram(Integer[] amostras);

}

package br.com.tomas.emgprocessing.sinal;

import java.util.ArrayList;
import java.util.List;

public abstract class LeitorDeAmostras {
    protected List<AmostraListener> listeners = new ArrayList<AmostraListener>();
    protected AmostraInputStream ais;

    public LeitorDeAmostras(AmostraInputStream ais) {
            this.ais = ais;
    }

    public void addAmostraListener(AmostraListener al) {
            this.listeners.add(al);
    }
```

```
    public void removeAmostraListener(AmostraListener al) {
        this.listeners.remove(al);
    }

    public abstract void iniciarLeitura();

    public abstract void parar();

}

package br.com.tomas.emgprocessing.sinal;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import br.uniceub.freedom.signal.listener.ISignalListener;

public class LeitorDeAmostrasDetectorDeSinal extends LeitorDeAmostras
implements Runnable {

    private boolean parar = false;
```

```
private List<ISignalListener> signalListeners = new ArrayList<ISignalListener>();


public LeitorDeAmostrasDetectorDeSinal(AmostraInputStream ais) {
        super(ais);
        this.ais = ais;
}


@Override
public void run() {
        try {
                int periodoMedia = 5;
                int[] mediaBuffer = new int[periodoMedia];
                int amplitude;
                double limiar = 180.0;
                boolean temSinal = false;
                double media = 0.0;
                int i = 0;
                while (!parar) {
                        amplitude = ais.lerAmostra();


                        mediaBuffer[i % periodoMedia] = amplitude;


                        if (i % periodoMedia == 0) {
```

```
                                        media /= periodoMedia;


                                        System.out.println("Media: " + media);
                                        System.out.println("Limiar: " + limiar);
                                        if (!temSinal && media > limiar) {
                                                temSinal = true;
                                                for(ISignalListener listener:
signalListeners){
                                                        listener.notifySignal();
                                                }
                                                temSinal = false;
                                        }
                                        media = 0.0;


                                }
                                media += Math.abs(amplitude);
                                i++;
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
```

```
    public void iniciarLeitura() {
            new Thread(this).start();
    }

    public void addSignalListener(ISignalListener signalListener){
            this.signalListeners.add(signalListener);
    }

    @Override
    public void parar() {
            parar = true;
    }
}
```

```
package br.com.tomas.emgprocessing.sinal;

import java.io.InputStream;

public interface LeitorDeSinal {

    InputStream comecarLeitura() throws LeituraDeSinalException;

    void pararLeitura();
```

```
}


package br.com.tomas.emgprocessing.sinal;


import java.io.IOException;

import java.io.InputStream;

import java.io.PipedInputStream;

import java.io.PipedOutputStream;


import javax.sound.sampled.AudioFormat;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.DataLine;

import javax.sound.sampled.LineUnavailableException;

import javax.sound.sampled.TargetDataLine;



public class LeitorDeSinalDeAudio implements LeitorDeSinal, Runnable {

    private TargetDataLine line;

    private AudioFormat format;

    private PipedOutputStream out;

    private int bufferSize;

    private boolean parar;
```

```
public LeitorDeSinalDeAudio(AudioFormat af, int bufferSize)
                throws LineUnavailableException {
        this.format = af;
        this.bufferSize = bufferSize;

        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);
        if (!AudioSystem.isLineSupported(info)) {
                throw new LineUnavailableException("Sem suporte de audio!");
        }
        System.out.println("Audio Suportado!");

        line = (TargetDataLine) AudioSystem.getLine(info);
}

public LeitorDeSinalDeAudio() throws LineUnavailableException {
        this(new AudioFormat(8000.0f, 16, 1, true, true), 2048);
}

public InputStream comecarLeitura() throws LeituraDeSinalException {
        try {
                line.open(format);

                System.out.println("Tamanho do buffer: " + line.getBufferSize());
```

```
                    System.out.println("Tamanho do frame: "
                            + line.getFormat().getFrameSize());
                    System.out
                            .println("Qtde canais: " +
line.getFormat().getChannels());

                    PipedInputStream in = new PipedInputStream(this.bufferSize);

                    out = new PipedOutputStream(in);

                    Thread threadLeitora = new Thread(this, "LeitoraDeSinal");
                    threadLeitora.start();

                    return in;
                } catch (LineUnavailableException e) {
                    throw new LeituraDeSinalException(e);
                } catch (IOException e) {
                    throw new LeituraDeSinalException(e);
                }
            }

            @Override
            public void run() {
```

```
            int qtdeBytesLidos;
            byte[] data = new byte[this.bufferSize];

            line.start();

            System.out.println("Iniciou a linha...");

            try {
                while (!parar) {
                    qtdeBytesLidos = line.read(data, 0, data.length);
                    out.write(data, 0, qtdeBytesLidos);
                }
                line.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void pararLeitura() {
            this.parar = true;
        }
    }
```

```
package br.com.tomas.emgprocessing.sinal;

public class LeituraDeSinalException extends Exception {
    private static final long serialVersionUID = 1L;

    public LeituraDeSinalException(Exception e) {
            super(e);
    }

    public LeituraDeSinalException() {}
}

package br.uniceub.freedom.gui;

import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JPanel;

public class Block {

    private JPanel keyLine;
```

```
private ArrayList<JButton> keys;

public Block(JPanel keyLine, ArrayList<JButton> keys){
        this.keyLine = keyLine;
        this.keys = keys;
}

public JPanel getKeyLine() {
        return keyLine;
}

public void setKeyLine(JPanel keyLine) {
        this.keyLine = keyLine;
}

public ArrayList<JButton> getKeys() {
        return keys;
}

public void setKeys(ArrayList<JButton> keys) {
        this.keys = keys;
}
```

```
}

package br.uniceub.freedom.gui;

import java.awt.Component;

import javax.swing.JDialog;
import javax.swing.JList;

import br.uniceub.freedom.signal.listener.ISignalListener;

public class FavoriteWalker extends Thread implements ISignalListener{

    private JDialog favoriteDialog;

    public FavoriteWalker(JDialog favoriteDialog) {
            this.favoriteDialog = favoriteDialog;
    }

    @Override
    public void notifySignal() {

    }
```

```
    @SuppressWarnings("unused")
    private JList<String> getFavoriteListComponent(){
            for(Component j: this.favoriteDialog.getComponents())
                if(j instanceof JList){
                    return (JList)j;
                }
            return null;
    }

}

package br.uniceub.freedom.gui;

import java.awt.Color;
import java.util.ArrayList;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.border.Border;

import br.uniceub.freedom.signal.listener.ISignalListener;
```

```
public class KeyboardWalker extends Thread implements ISignalListener {

    private boolean walk;
    private long sleepInterval = 300;
    private MainForm mainForm;
    private boolean signal = false;
    private boolean reset = false;
    public KeyboardWalker(MainForm mainForm) {
            this.mainForm = mainForm;
    }

    public boolean isWalk() {
            return walk;
    }

    public void setWalk(boolean walk) {
            this.walk = walk;
    }

    public long getSleepInterval() {
            return sleepInterval;
    }
```

```
public void setSleepInterval(long sleepInterval) {
        this.sleepInterval = sleepInterval;
}

public ArrayList<Block> getSidewalk(){
        return this.mainForm.getSidewalk();
}

public void repaint(){
        this.mainForm.repaint();
}

public Border getHighlightBorder(){
        return BorderFactory.createLineBorder(Color.RED, 3);
}

public Border getDefaultBorder(){
        return null;
}

public void run(){
        setWalk(true);
```

```
                    while(isWalk()) {
                        try {
                            for(Block block: getSidewalk()){
                                highlightComponent(block.getKeyLine());

                                for(int i =0 ; i<5; i++){
                                    if(hasSignal()){
                                        sleep(1000l);
                                        resetSignal();
                                        for(JButton button:
block.getKeys()){
        highlightComponent(button);
                                            for(int j=0; j<5 ; j++){
                                                if(hasSignal()){

        sleep(1000l);
        resetSignal();
        button.doClick();
        this.reset = true;
        disacknowledgeButton(button);
```

```
                                                                    break;
                                                                }


            sleep(getSleepInterval());
                                                            }


            disacknowledgeButton(button);
                                                            if(reset){
                                                                this.reset      =
false;
                                                                break;
                                                            }
                                                        }
                                                    }


                                                    sleep(getSleepInterval());
                                                }


                                                disacknowledgeButton(block.getKeyLine());
                                            }


                                        }
                                        catch( Exception e ) {}
```

```
            }
}

private void disacknowledgeButton(JComponent component) {
            component.setBorder(getDefaultBorder());
            repaint();
}

private void highlightComponent(JComponent component) {
            component.setBorder(getHighlightBorder());
            repaint();
}

@Override
public synchronized void  notifySignal() {
            this.signal = true;
}

public boolean hasSignal() {
            return signal;
}

public void resetSignal(){
```

```
                this.signal = false;
        }




}

package br.uniceub.freedom.gui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;

import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JProgressBar;
import javax.swing.JWindow;

public class SplashScreen extends JWindow {

    /**
     *
     */
    private static final long serialVersionUID = 6860546259742255067L;
```

```
 private JLabel jLabelSplashImage;
private JLabel jLabelTextoCarregamento;
private static JLabel jLabelTextoDinamicoPlugins;
private static JProgressBar jProgressBarSistema;

public SplashScreen() {
 initComponents();
 this.setVisible(true);
 }

 private void initComponents() {
            /**
    * Inicializando as variavaeis utilizadas
    */
   jProgressBarSistema = new JProgressBar();
   jLabelSplashImage = new JLabel();
   jLabelTextoCarregamento = new JLabel();
   jLabelTextoDinamicoPlugins = new JLabel();

   /**
    * Carregando a imagem do Splash e adicionando a imagem ao componente
    * jLabelSplashImage
```

```
                */
                ImageIcon imageIcon = new
ImageIcon(getClass().getResource("/uniceub.png"));
                jLabelSplashImage.setIcon(imageIcon);

                /**
                 * Definindo dinamicamente o tamando do container segundo o tamanho da
imagem.
                 */
                this.setMinimumSize(new
java.awt.Dimension(imageIcon.getIconWidth(),imageIcon.getIconHeight()));
                jLabelSplashImage.setBounds(0, 0, imageIcon.getIconWidth(),
imageIcon.getIconHeight());

                /**
                 * A definicao do layout=null e importante para possibilitar que os componentes
                 * fiquem sobrescritros em tempo de execucao
                 */
                getContentPane().setLayout(null);

                /**
                 * Definindo a localizacao do splash no centro da tela
                 */
                Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
```

```
this.setLocation((screen.width - this.getSize().width) / 2, (screen.height - this.getSize().height) / 2);

/**
 * Setando parametros da variavel jProgressBarSistema
 */
jProgressBarSistema.setForeground(new Color(0,0,204));
jProgressBarSistema.setPreferredSize(new java.awt.Dimension(148, 5));
jProgressBarSistema.setBounds(0, 266, imageIcon.getIconWidth(), 5);
jProgressBarSistema.setBorderPainted(false);
jProgressBarSistema.setIndeterminate(true);

/**
 * Adicionando o jProgressBarSistema a classe SplashJProgressBar
 */
getContentPane().add(jProgressBarSistema);

/**
 * Setando parametros da variavel jProgressBarSistema
 */
jLabelTextoCarregamento.setForeground(new java.awt.Color(0,0,204));
jLabelTextoCarregamento.setFont(new java.awt.Font("DialogInput", 0, 11));
jLabelTextoCarregamento.setText("Carregando...");
```

```
jLabelTextoCarregamento.setBounds(280, 285, 80, 20);


/**
 * Adicionando o jProgressBarSistema a classe SplashJProgressBar
 */
this.getContentPane().add(jLabelTextoCarregamento);


/**
 * Setando parametros da variavel jProgressBarSistema
 */
jLabelTextoDinamicoPlugins.setForeground(new java.awt.Color(0,0,204));
jLabelTextoDinamicoPlugins.setFont(new java.awt.Font("DialogInput", 0, 11));
jLabelTextoDinamicoPlugins.setBounds(360, 285, 230, 20);
/**
 * Adicionando o jProgressBarSistema a classe SplashJProgressBar
 */
this.getContentPane().add(jLabelTextoDinamicoPlugins);


/**
 * O Ultimo item adicionado no conteiner deve ser o componente que comtem
 * a imagem do Splah
 */
this.getContentPane().add(jLabelSplashImage);
```

```
        this.pack();
    }

}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package br.uniceub.freedom.gui.listener;

import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import br.uniceub.freedom.gui.MainForm;


public class FavoriteListener implements ListSelectionListener {

    private MainForm mainForm;
```

```
        public FavoriteListener(MainForm mainForm){
            this.mainForm = mainForm;
            addListener();
        }

        @Override
        public void valueChanged(ListSelectionEvent e) {
            getMainForm().getTextBox().setText(getMainForm().getTextBox().getText()+
"\n" + getMainForm().getFavoriteList().getSelectedValue());
            getMainForm().getFavoriteDialog().dispose();
        }

        private void addListener() {
            getMainForm().getFavoriteList().addListSelectionListener(this);
        }

        public MainForm getMainForm() {
            return mainForm;
        }

        public void setMainForm(MainForm mainForm) {
            this.mainForm = mainForm;
        }
```

```
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package br.uniceub.freedom.gui.listener;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JTextArea;

import br.uniceub.freedom.gui.Block;
import br.uniceub.freedom.gui.MainForm;

public class KeyboardListener implements ActionListener {

    private MainForm mainForm;
```

```
public KeyboardListener(MainForm mainForm){
    this.mainForm = mainForm;
    addListener();
}

public MainForm getMainForm() {
    return mainForm;
}

public void setMainForm(MainForm mainForm) {
    this.mainForm = mainForm;
}

@Override
public void actionPerformed(ActionEvent e) {
    JTextArea textBox = getMainForm().getTextBox();
    if(e.getActionCommand().equals("RETURN")){
        textBox.setText(textBox.getText() + "\n");
    } else if(e.getActionCommand().equals("SPACE")){
        textBox.setText(textBox.getText() + " ");
    } else {
        textBox.setText(textBox.getText() + e.getActionCommand());
```

```
        }


    }


    private void addListener() {
        for(Block block: getMainForm().getSidewalk()){
                for(JButton button: block.getKeys()){
                        button.addActionListener(this);
                }
        }
    }


}


package br.uniceub.freedom.gui.util;


import java.awt.Component;


import javax.swing.JList;
import javax.swing.ListCellRenderer;


public class FavoriteCellRenderer implements ListCellRenderer<String>{
```

```
        @Override
        public Component getListCellRendererComponent(JList<? extends String> list,
                String value, int index, boolean isSelected, boolean
cellHasFocus) {
            return null;
        }



    }

    package br.uniceub.freedom.gui.util;

    import javax.swing.AbstractListModel;

    public class FavoriteModel extends AbstractListModel<String>{

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        private String[] favoritos = { "Bom dia!", "Boa tarde!", "Boa Noite!", "Estou com
fome!", "Quero ir ao banheiro!", "Sim!", "Não!" };
```

```
    @Override
    public int getSize() {
            return favoritos.length;
    }

    @Override
    public String getElementAt(int index) {
            return favoritos[index];
    }



}

package br.uniceub.freedom.signal.listener;

public interface ISignalListener {

    void notifySignal();

}

/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */


package br.uniceub.freedom.gui;


import java.awt.Font;
import java.awt.Toolkit;
import java.io.InputStream;
import java.util.ArrayList;


import javax.sound.sampled.LineUnavailableException;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;


import br.com.tomas.emgprocessing.sinal.AmostraAudioInputStream;
```

```
import br.com.tomas.emgprocessing.sinal.LeitorDeAmostras;

import br.com.tomas.emgprocessing.sinal.LeitorDeAmostrasDetectorDeSinal;

import br.com.tomas.emgprocessing.sinal.LeitorDeSinal;

import br.com.tomas.emgprocessing.sinal.LeitorDeSinalDeAudio;

import br.com.tomas.emgprocessing.sinal.LeituraDeSinalException;

import br.uniceub.freedom.gui.listener.FavoriteListener;

import br.uniceub.freedom.gui.listener.KeyboardListener;

import br.uniceub.freedom.gui.util.FavoriteModel;



public class MainForm extends javax.swing.JFrame {

    /**

     *

     */

    private static final long serialVersionUID = 8895063620081115659L;

    private ArrayList<JButton> keyboardLine1 = new ArrayList<JButton>();

    private ArrayList<JButton> keyboardLine2 = new ArrayList<JButton>();

    private ArrayList<JButton> keyboardLine3 = new ArrayList<JButton>();

    private ArrayList<JButton> keyboardLine4 = new ArrayList<JButton>();

    private ArrayList<JButton> keyboardLine5 = new ArrayList<JButton>();

    private ArrayList<Block> sidewalk = new ArrayList<Block>();
```

```
@SuppressWarnings("unused")
 private KeyboardListener keyboardListener;
@SuppressWarnings("unused")
 private FavoriteListener favoriteListener;
private LeitorDeAmostras leitorAmostras;
private SplashScreen splashScreen;


/**
 * Creates new form MainForm
 */
public MainForm() {
 initSplash();
    initComponents();
    initkeyboardsButtons();
    initSidewalk();
    initSampleReader();
    initWalker();
    keyboardListener = new KeyboardListener(this);
    favoriteListener = new FavoriteListener(this);
    setExtendedState(JFrame.MAXIMIZED_BOTH);
    setLocationRelativeTo(null);
    setVisible(true);
}
```

```
    private void initSplash() {
        this.splashScreen = new SplashScreen();
    }

    private void initWalker() {
        KeyboardWalker walker = new KeyboardWalker(this);

((LeitorDeAmostrasDetectorDeSinal)getLeitorDeAmostras()).addSignalListener(walker);
        walker.start();
    }

    /**
    * This method is called from within the constructor to initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is always
    * regenerated by the Form Editor.
    */
    @SuppressWarnings({ "unchecked", "rawtypes" })
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {
```

```
favoriteDialog = new javax.swing.JDialog();

favoritePane = new javax.swing.JPanel();

favoriteLabel = new javax.swing.JLabel();

favoriteListPane = new javax.swing.JScrollPane();

favoriteList = new javax.swing.JList();

textRow = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

textBox = new javax.swing.JTextArea();

firstRow = new javax.swing.JPanel();

favoriteButton = new javax.swing.JButton();

returnButton = new javax.swing.JButton();

spaceButton = new javax.swing.JButton();

dotButton = new javax.swing.JButton();

commaButton = new javax.swing.JButton();

exclamationButton = new javax.swing.JButton();

questionButton = new javax.swing.JButton();

secondRow = new javax.swing.JPanel();

oneButton = new javax.swing.JButton();

twoButton = new javax.swing.JButton();

threeButton = new javax.swing.JButton();

fourButton = new javax.swing.JButton();

fiveButton = new javax.swing.JButton();
```

```
sixButton = new javax.swing.JButton();

sevenButton = new javax.swing.JButton();

eightButton = new javax.swing.JButton();

nineButton = new javax.swing.JButton();

zeroButton = new javax.swing.JButton();

thirdRow = new javax.swing.JPanel();

aButton = new javax.swing.JButton();

sButton = new javax.swing.JButton();

nButton = new javax.swing.JButton();

uButton = new javax.swing.JButton();

lButton = new javax.swing.JButton();

gButton = new javax.swing.JButton();

bButton = new javax.swing.JButton();

jButton = new javax.swing.JButton();

kButton = new javax.swing.JButton();

wButton = new javax.swing.JButton();

fourthRow = new javax.swing.JPanel();

eButton = new javax.swing.JButton();

rButton = new javax.swing.JButton();

dButton = new javax.swing.JButton();

tButton = new javax.swing.JButton();

pButton = new javax.swing.JButton();

hButton = new javax.swing.JButton();
```

```
                fButton = new javax.swing.JButton();
                xButton = new javax.swing.JButton();
                yButton = new javax.swing.JButton();
                cedillaButton = new javax.swing.JButton();
                fifthRow = new javax.swing.JPanel();
                oButton = new javax.swing.JButton();
                iButton = new javax.swing.JButton();
                mButton = new javax.swing.JButton();
                cButton = new javax.swing.JButton();
                vButton = new javax.swing.JButton();
                qButton = new javax.swing.JButton();
                zButton = new javax.swing.JButton();

favoriteDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE
);
                favoriteDialog.setTitle("Favoritos");
                favoriteDialog.setAlwaysOnTop(true);
                favoriteDialog.setMinimumSize(new java.awt.Dimension(236, 213));
                favoriteDialog.setModal(true);

                favoriteLabel.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
                favoriteLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
                favoriteLabel.setLabelFor(favoriteList);
```

```
                favoriteLabel.setText("FAVORITOS");

                favoriteListPane.setMaximumSize(new java.awt.Dimension(236, 213));
                favoriteListPane.setMinimumSize(new java.awt.Dimension(236, 213));
                favoriteListPane.setPreferredSize(new java.awt.Dimension(236, 213));

                favoriteList.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1,
1));
                favoriteList.setModel(new FavoriteModel());

favoriteList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
                favoriteList.setMaximumSize(new java.awt.Dimension(107, 150));
                favoriteList.setMinimumSize(new java.awt.Dimension(107, 150));
                favoriteList.setPreferredSize(new java.awt.Dimension(107, 150));
                favoriteListPane.setViewportView(favoriteList);

                javax.swing.GroupLayout favoritePaneLayout = new
javax.swing.GroupLayout(favoritePane);
                favoritePane.setLayout(favoritePaneLayout);
                favoritePaneLayout.setHorizontalGroup(

favoritePaneLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(favoriteLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addGroup(favoritePaneLayout.createSequentialGroup()
                    .addComponent(favoriteListPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 236,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE))
            );
            favoritePaneLayout.setVerticalGroup(
favoritePaneLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(favoritePaneLayout.createSequentialGroup()
                    .addComponent(favoriteLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(favoriteListPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            );

            javax.swing.GroupLayout favoriteDialogLayout = new
javax.swing.GroupLayout(favoriteDialog.getContentPane());
            favoriteDialog.getContentPane().setLayout(favoriteDialogLayout);
            favoriteDialogLayout.setHorizontalGroup(
```

```
favoriteDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(favoritePane,    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                );
                favoriteDialogLayout.setVerticalGroup(

favoriteDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(favoriteDialogLayout.createSequentialGroup()
                        .addComponent(favoritePane,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 0, Short.MAX_VALUE))
                );

                setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
                setTitle("Projeto Liberdade");
                setPreferredSize((Toolkit.getDefaultToolkit()).getScreenSize());

                textRow.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
                textRow.setPreferredSize(null);

                jScrollPane1.setBorder(new                javax.swing.border.LineBorder(new
java.awt.Color(0, 0, 0), 5, true));
```

```
                textBox.setColumns(20);
                textBox.setFont(new java.awt.Font("Verdana", Font.BOLD, 25)); // NOI18N
                textBox.setLineWrap(true);
                textBox.setRows(5);
                textBox.setToolTipText("Caixa de Texto");
                textBox.setBorder(javax.swing.BorderFactory.createEmptyBorder(2, 2, 2, 0));
                jScrollPane1.setViewportView(textBox);

                javax.swing.GroupLayout textRowLayout = new
javax.swing.GroupLayout(textRow);
                textRow.setLayout(textRowLayout);
                textRowLayout.setHorizontalGroup(

textRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1)
                );
                textRowLayout.setVerticalGroup(

textRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1,    javax.swing.GroupLayout.DEFAULT_SIZE,
130, Short.MAX_VALUE)
                );

                firstRow.setPreferredSize(null);
```

```
                favoriteButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                favoriteButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Fav-icon.png")));
                favoriteButton.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        favoriteButtonActionPerformed(evt);
                    }
                });

                returnButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                returnButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/237271-Keyboard_enter-128.png"))); //
NOI18N
                returnButton.setToolTipText("Return");
                returnButton.setActionCommand("RETURN");

                spaceButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                spaceButton.setText("ESPAÇO");
                spaceButton.setActionCommand("SPACE");
```

```
                dotButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
                dotButton.setText(".");

                commaButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                commaButton.setText(",");

                exclamationButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                exclamationButton.setText("!");

                questionButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                questionButton.setText("?");

                javax.swing.GroupLayout firstRowLayout = new
javax.swing.GroupLayout(firstRow);
                firstRow.setLayout(firstRowLayout);
                firstRowLayout.setHorizontalGroup(

firstRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addGroup(firstRowLayout.createSequentialGroup()
```

```
                .addComponent(favoriteButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(returnButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(spaceButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(dotButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(commaButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(exclamationButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addGap(0, 0, 0)

                .addComponent(questionButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addGap(0, 0, 0))
        );
        firstRowLayout.setVerticalGroup(

firstRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                .addComponent(favoriteButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(returnButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(spaceButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(dotButton,   javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(commaButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(exclamationButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(questionButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
        );
```

```
secondRow.setPreferredSize(null);

oneButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
oneButton.setText("1");

twoButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
twoButton.setText("2");

threeButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
threeButton.setText("3");

fourButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
fourButton.setText("4");

fiveButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
fiveButton.setText("5");

sixButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
sixButton.setText("6");

sevenButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
```

```
        sevenButton.setText("7");

        eightButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
        eightButton.setText("8");

        nineButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
        nineButton.setText("9");

        zeroButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
        zeroButton.setText("0");

        javax.swing.GroupLayout secondRowLayout = new javax.swing.GroupLayout(secondRow);
        secondRow.setLayout(secondRowLayout);
        secondRowLayout.setHorizontalGroup(

secondRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
            .addGroup(secondRowLayout.createSequentialGroup()
                .addComponent(oneButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(twoButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
```

```
                .addGap(0, 0, 0)
                .addComponent(threeButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(fourButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(fiveButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(sixButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(sevenButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(eightButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
```

```
                    .addComponent(nineButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                    .addGap(0, 0, 0)
                    .addComponent(zeroButton,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                    .addGap(0, 0, 0))
                );
                secondRowLayout.setVerticalGroup(

secondRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addComponent(oneButton,  javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                    .addComponent(twoButton,  javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                    .addComponent(threeButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                    .addComponent(fourButton,  javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                    .addComponent(fiveButton,  javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(sixButton,    javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addComponent(sevenButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)

                .addComponent(eightButton,
javax.swing.GroupLayout.Alignment.CENTER,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)

                .addComponent(nineButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

                .addComponent(zeroButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)

            );


            thirdRow.setPreferredSize(null);


            aButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N

            aButton.setText("A");


            sButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N

            sButton.setText("S");


            nButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
```

```
nButton.setText("N");

uButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
uButton.setText("U");

lButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
lButton.setText("L");

gButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
gButton.setText("G");

bButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
bButton.setText("B");

jButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
jButton.setText("J");

kButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
kButton.setText("K");

wButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 17)); // NOI18N
wButton.setText("W");
```

```
javax.swing.GroupLayout thirdRowLayout = new javax.swing.GroupLayout(thirdRow);
thirdRow.setLayout(thirdRowLayout);
thirdRowLayout.setHorizontalGroup(

thirdRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
.addGroup(thirdRowLayout.createSequentialGroup()
.addComponent(aButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
.addComponent(sButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
.addComponent(nButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
.addComponent(uButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
.addComponent(lButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
.addComponent(gButton, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
.addGap(0, 0, 0)
```

```
                .addComponent(bButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(jButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(kButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(wButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0))
            );
            thirdRowLayout.setVerticalGroup(

thirdRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                .addComponent(aButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(sButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(nButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(uButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(lButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(gButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(bButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(jButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(kButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(wButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
            );

            fourthRow.setPreferredSize(null);

            eButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
            eButton.setText("E");
```

```
rButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
rButton.setText("R");

dButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
dButton.setText("D");

tButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
tButton.setText("T");

pButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
pButton.setText("P");

hButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
hButton.setText("H");

fButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
fButton.setText("F");

xButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
xButton.setText("X");

yButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
```

```
                yButton.setText("Y");

                cedillaButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); //
NOI18N
                cedillaButton.setText("Ç");

                javax.swing.GroupLayout fourthRowLayout = new
javax.swing.GroupLayout(fourthRow);
                fourthRow.setLayout(fourthRowLayout);
                fourthRowLayout.setHorizontalGroup(

fourthRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addGroup(fourthRowLayout.createSequentialGroup()
                        .addComponent(eButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(rButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(dButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(tButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
```

```
                .addComponent(pButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(hButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(fButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(xButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(yButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(cedillaButton,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0))
            );
            fourthRowLayout.setVerticalGroup(

fourthRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                .addComponent(eButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(rButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(dButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(tButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(pButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(hButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(fButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(xButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(yButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(cedillaButton,
javax.swing.GroupLayout.Alignment.CENTER, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
            );
```

```
fifthRow.setPreferredSize(null);

oButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
oButton.setText("O");

iButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
iButton.setText("I");

mButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
mButton.setText("M");

cButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
cButton.setText("C");

vButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
vButton.setText("V");

qButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
qButton.setText("Q");

zButton.setFont(new java.awt.Font("Tahoma", Font.BOLD, 18)); // NOI18N
zButton.setText("Z");
```

```
                javax.swing.GroupLayout fifthRowLayout = new
javax.swing.GroupLayout(fifthRow);
                fifthRow.setLayout(fifthRowLayout);
                fifthRowLayout.setHorizontalGroup(

fifthRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addGroup(fifthRowLayout.createSequentialGroup()
                        .addComponent(oButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(iButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(mButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(cButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(vButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(qButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
```

```
                .addGap(0, 0, 0)
                .addComponent(zButton, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addGap(0, 0, 0))
            );
            fifthRowLayout.setVerticalGroup(

fifthRowLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                .addComponent(oButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(iButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(mButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(cButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(vButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addComponent(qButton, javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(zButton,      javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)

                );


                javax.swing.Group Layout          layout          =          new
javax.swing.Group Layout(getContentPane());

                getContentPane().set Layout(layout);

                layout.setHorizontalGroup(

                   layout.createParallelGroup(javax.swing.Group Layout.Alignment.CENTER)

                   .addComponent(textRow,      javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)

                   .addComponent(firstRow,javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)


.addComponent(secondRow,javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)

                   .addComponent(thirdRow,javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)

                   .addComponent(fourthRow,javax.swing.Group Layout.Alignment.CENTER,
javax.swing.Group Layout.PREFERRED_SIZE,    javax.swing.Group Layout.PREFERRED_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(fifthRow,javax.swing.GroupLayout.Alignment.CENTER,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(textRow, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(firstRow, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(thirdRow,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(fourthRow,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                        .addGap(0, 0, 0)
                        .addComponent(fifthRow, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                        .addGap(0, 0, 0)
```

```
                .addComponent(secondRow,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.PREFERRED_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0))
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private    void    favoriteButtonActionPerformed(java.awt.event.ActionEvent    evt)
{//GEN-FIRST:event_favoriteButtonActionPerformed
            favoriteDialog.setVisible(true);
            initFavNavigation();
        }

        private void initFavNavigation() {

         }



         /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
```

```
                /* Set the Nimbus look and feel */
                //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
                /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
                 * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
                 */
                try {
                    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                        if ("Nimbus".equals(info.getName())) {
                            javax.swing.UIManager.setLookAndFeel(info.getClassName());
                            break;
                        }
                    }
                } catch (ClassNotFoundException ex) {
                    java.util.logging.Logger.getLogger(MainForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
                } catch (InstantiationException ex) {
                    java.util.logging.Logger.getLogger(MainForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
            } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainForm.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainForm.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
            }
            //</editor-fold>

            /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new MainForm();
                }
            });
        }

        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JButton aButton;
        private javax.swing.JButton bButton;
        private javax.swing.JButton cButton;
        private javax.swing.JButton cedillaButton;
```

```
private javax.swing.JButton commaButton;

private javax.swing.JButton dButton;

private javax.swing.JButton dotButton;

private javax.swing.JButton eButton;

private javax.swing.JButton eightButton;

private javax.swing.JButton exclamationButton;

private javax.swing.JButton fButton;

private javax.swing.JButton favoriteButton;

private javax.swing.JDialog favoriteDialog;

private javax.swing.JLabel favoriteLabel;

private javax.swing.JList favoriteList;

private javax.swing.JScrollPane favoriteListPane;

private javax.swing.JPanel favoritePane;

private javax.swing.JPanel fifthRow;

private javax.swing.JPanel firstRow;

private javax.swing.JButton fiveButton;

private javax.swing.JButton fourButton;

private javax.swing.JPanel fourthRow;

private javax.swing.JButton gButton;

private javax.swing.JButton hButton;

private javax.swing.JButton iButton;

private javax.swing.JButton jButton;

private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JButton kButton;
private javax.swing.JButton lButton;
private javax.swing.JButton mButton;
private javax.swing.JButton nButton;
private javax.swing.JButton nineButton;
private javax.swing.JButton oButton;
private javax.swing.JButton oneButton;
private javax.swing.JButton pButton;
private javax.swing.JButton qButton;
private javax.swing.JButton questionButton;
private javax.swing.JButton rButton;
private javax.swing.JButton returnButton;
private javax.swing.JButton sButton;
private javax.swing.JPanel secondRow;
private javax.swing.JButton sevenButton;
private javax.swing.JButton sixButton;
private javax.swing.JButton spaceButton;
private javax.swing.JButton tButton;
private javax.swing.JTextArea textBox;
private javax.swing.JPanel textRow;
private javax.swing.JPanel thirdRow;
private javax.swing.JButton threeButton;
private javax.swing.JButton twoButton;
```

```
private javax.swing.JButton uButton;
private javax.swing.JButton vButton;
private javax.swing.JButton wButton;
private javax.swing.JButton xButton;
private javax.swing.JButton yButton;
private javax.swing.JButton zButton;
private javax.swing.JButton zeroButton;
// End of variables declaration//GEN-END:variables

public JButton getAButton() {
    return aButton;
}

public void setAButton(JButton aButton) {
    this.aButton = aButton;
}

public JButton getBButton() {
    return bButton;
}

public void setBButton(JButton bButton) {
    this.bButton = bButton;
```

```
}

public JButton getCButton() {
    return cButton;
}

public void setCButton(JButton cButton) {
    this.cButton = cButton;
}

public JButton getCedillaButton() {
    return cedillaButton;
}

public void setCedillaButton(JButton cedillaButton) {
    this.cedillaButton = cedillaButton;
}

public JButton getCommaButton() {
    return commaButton;
}

public void setCommaButton(JButton commaButton) {
```

```
    this.commaButton = commaButton;
}

public JButton getDButton() {
    return dButton;
}

public void setDButton(JButton dButton) {
    this.dButton = dButton;
}

public JButton getDotButton() {
    return dotButton;
}

public void setDotButton(JButton dotButton) {
    this.dotButton = dotButton;
}

public JButton getEButton() {
    return eButton;
}
```

```
public void setEButton(JButton eButton) {
    this.eButton = eButton;
}

public JButton getEightButton() {
    return eightButton;
}

public void setEightButton(JButton eightButton) {
    this.eightButton = eightButton;
}

public JButton getExclamationButton() {
    return exclamationButton;
}

public void setExclamationButton(JButton exclamationButton) {
    this.exclamationButton = exclamationButton;
}

public JButton getFButton() {
    return fButton;
}
```

```
public void setFButton(JButton fButton) {
    this.fButton = fButton;
}

public JButton getFavoriteButton() {
    return favoriteButton;
}

public void setFavoriteButton(JButton favoriteButton) {
    this.favoriteButton = favoriteButton;
}

public JDialog getFavoriteDialog() {
    return favoriteDialog;
}

public void setFavoriteDialog(JDialog favoriteDialog) {
    this.favoriteDialog = favoriteDialog;
}

public JLabel getFavoriteLabel() {
    return favoriteLabel;
```

```
}

public void setFavoriteLabel(JLabel favoriteLabel) {
    this.favoriteLabel = favoriteLabel;
}

public JButton getFiveButton() {
    return fiveButton;
}

public void setFiveButton(JButton fiveButton) {
    this.fiveButton = fiveButton;
}

public JButton getFourButton() {
    return fourButton;
}

public void setFourButton(JButton fourButton) {
    this.fourButton = fourButton;
}
```

```
public JButton getGButton() {
    return gButton;
}

public void setGButton(JButton gButton) {
    this.gButton = gButton;
}

public JButton getHButton() {
    return hButton;
}

public void setHButton(JButton hButton) {
    this.hButton = hButton;
}

public JButton getIButton() {
    return iButton;
}

public void setIButton(JButton iButton) {
    this.iButton = iButton;
}
```

```
public JButton getJButton() {
    return jButton;
}

public void setJButton(JButton jButton) {
    this.jButton = jButton;
}

public JButton getKButton() {
    return kButton;
}

public void setKButton(JButton kButton) {
    this.kButton = kButton;
}

public JButton getLButton() {
    return lButton;
}

public void setLButton(JButton lButton) {
    this.lButton = lButton;
```

```
}

public JButton getMButton() {
    return mButton;
}

public void setMButton(JButton mButton) {
    this.mButton = mButton;
}

public JButton getNButton() {
    return nButton;
}

public void setNButton(JButton nButton) {
    this.nButton = nButton;
}

public JButton getNineButton() {
    return nineButton;
}

public void setNineButton(JButton nineButton) {
```

```
    this.nineButton = nineButton;
}

public JButton getOButton() {
    return oButton;
}

public void setOButton(JButton oButton) {
    this.oButton = oButton;
}

public JButton getOneButton() {
    return oneButton;
}

public void setOneButton(JButton oneButton) {
    this.oneButton = oneButton;
}

public JButton getPButton() {
    return pButton;
}
```

```
public void setPButton(JButton pButton) {
    this.pButton = pButton;
}

public JButton getQButton() {
    return qButton;
}

public void setQButton(JButton qButton) {
    this.qButton = qButton;
}

public JButton getQuestionButton() {
    return questionButton;
}

public void setQuestionButton(JButton questionButton) {
    this.questionButton = questionButton;
}

public JButton getRButton() {
    return rButton;
}
```

```
public void setRButton(JButton rButton) {
    this.rButton = rButton;
}

public JButton getReturnButton() {
    return returnButton;
}

public void setReturnButton(JButton returnButton) {
    this.returnButton = returnButton;
}

public JButton getSButton() {
    return sButton;
}

public void setSButton(JButton sButton) {
    this.sButton = sButton;
}

public JPanel getSecondRow() {
    return secondRow;
```

```
}

public void setSecondRow(JPanel secondRow) {
    this.secondRow = secondRow;
}

public JButton getSevenButton() {
    return sevenButton;
}

public void setSevenButton(JButton sevenButton) {
    this.sevenButton = sevenButton;
}

public JButton getSixButton() {
    return sixButton;
}

public void setSixButton(JButton sixButton) {
    this.sixButton = sixButton;
}

public JButton getSpaceButton() {
```

```
    return spaceButton;
}

public void setSpaceButton(JButton spaceButton) {
    this.spaceButton = spaceButton;
}

public JButton getTButton() {
    return tButton;
}

public void setTButton(JButton tButton) {
    this.tButton = tButton;
}

public JButton getThreeButton() {
    return threeButton;
}

public void setThreeButton(JButton threeButton) {
    this.threeButton = threeButton;
}
```

```
public JButton getTwoButton() {
    return twoButton;
}

public void setTwoButton(JButton twoButton) {
    this.twoButton = twoButton;
}

public JButton getUButton() {
    return uButton;
}

public void setUButton(JButton uButton) {
    this.uButton = uButton;
}

public JButton getVButton() {
    return vButton;
}

public void setVButton(JButton vButton) {
    this.vButton = vButton;
}
```

```
public JButton getWButton() {
    return wButton;
}

public void setWButton(JButton wButton) {
    this.wButton = wButton;
}

public JButton getXButton() {
    return xButton;
}

public void setXButton(JButton xButton) {
    this.xButton = xButton;
}

public JButton getYButton() {
    return yButton;
}

public void setYButton(JButton yButton) {
    this.yButton = yButton;
```

```
}

public JButton getZButton() {
    return zButton;
}

public void setZButton(JButton zButton) {
    this.zButton = zButton;
}

public JButton getZeroButton() {
    return zeroButton;
}

public void setZeroButton(JButton zeroButton) {
    this.zeroButton = zeroButton;
}

public ArrayList<JButton> getKeyboardLine1() {
        return keyboardLine1;
 }
```

```
public ArrayList<JButton> getKeyboardLine2() {
        return keyboardLine2;
}

public ArrayList<JButton> getKeyboardLine3() {
        return keyboardLine3;
}

public ArrayList<JButton> getKeyboardLine4() {
        return keyboardLine4;
}

public ArrayList<JButton> getKeyboardLine5() {
        return keyboardLine5;
}

public JTextArea getTextBox() {
   return textBox;
}
```

```
public void setTextBox(JTextArea textBox) {
    this.textBox = textBox;
}

public JList getFavoriteList() {
    return favoriteList;
}

public void setFavoriteList(JList favoriteList) {
    this.favoriteList = favoriteList;
}

public LeitorDeAmostras getLeitorDeAmostras(){
    return this.leitorAmostras;
}

public void setLeitorDeAmostras(LeitorDeAmostras leitorAmostras){
    this.leitorAmostras = leitorAmostras;
}

public ArrayList<Block> getSidewalk() {
            return sidewalk;
```

```
}



private void initkeyboardsButtons() {
//Linha 1
keyboardLine1.add(getFavoriteButton());
  keyboardLine1.add(getReturnButton());
  keyboardLine1.add(getSpaceButton());
  keyboardLine1.add(getDotButton());
  keyboardLine1.add(getCommaButton());
  keyboardLine1.add(getExclamationButton());
  keyboardLine1.add(getQuestionButton());

  //Linha2
  keyboardLine2.add(getOneButton());
  keyboardLine2.add(getTwoButton());
  keyboardLine2.add(getThreeButton());
  keyboardLine2.add(getFourButton());
  keyboardLine2.add(getFiveButton());
  keyboardLine2.add(getSixButton());
  keyboardLine2.add(getSevenButton());
  keyboardLine2.add(getEightButton());
  keyboardLine2.add(getNineButton());
```

```
keyboardLine2.add(getZeroButton());


//Linha3
keyboardLine3.add(getAButton());
keyboardLine3.add(getSButton());
keyboardLine3.add(getNButton());
keyboardLine3.add(getUButton());
keyboardLine3.add(getLButton());
keyboardLine3.add(getGButton());
keyboardLine3.add(getBButton());
keyboardLine3.add(getJButton());
keyboardLine3.add(getKButton());
keyboardLine3.add(getWButton());


keyboardLine4.add(getEButton());
keyboardLine4.add(getRButton());
keyboardLine4.add(getDButton());
keyboardLine4.add(getTButton());
keyboardLine4.add(getPButton());
keyboardLine4.add(getHButton());
keyboardLine4.add(getFButton());
keyboardLine4.add(getXButton());
keyboardLine4.add(getYButton());
```

```
    keyboardLine4.add(getCedillaButton());


    //Linha5
    keyboardLine5.add(getOButton());
    keyboardLine5.add(getIButton());
    keyboardLine5.add(getMButton());
    keyboardLine5.add(getCButton());
    keyboardLine5.add(getVButton());
    keyboardLine5.add(getQButton());
    keyboardLine5.add(getZButton());


}


private void initSidewalk() {
 sidewalk.add(new Block(firstRow, keyboardLine1));
 sidewalk.add(new Block(thirdRow, keyboardLine3));
 sidewalk.add(new Block(fourthRow, keyboardLine4));
 sidewalk.add(new Block(fifthRow, keyboardLine5));
 sidewalk.add(new Block(secondRow, keyboardLine2));
 }


private void initSampleReader() {
    try{
```

```
                    LeitorDeSinal leitor = new LeitorDeSinalDeAudio();
                    InputStream is = leitor.comecarLeitura();
                    AmostraAudioInputStream aais = new AmostraAudioInputStream(is);

                    setLeitorDeAmostras(new LeitorDeAmostrasDetectorDeSinal(aais));
                    getLeitorDeAmostras().iniciarLeitura();
                } catch (LineUnavailableException lue){
                    JOptionPane.showMessageDialog(this, "Não há suporte de Audio!", "Error",
JOptionPane.ERROR_MESSAGE);
                    System.exit(ERROR);
                } catch (LeituraDeSinalException lse){
                        JOptionPane.showMessageDialog(this, "Ocorreu um problema na leitura
do sinal", "Error", JOptionPane.ERROR_MESSAGE);
                        System.exit(ERROR);
                }
            }
        }
```