UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
ESCOLA DE ENGENHARIA - DELET
CURSO DE ENGENHARIA DE COMPUTAÇÃO

**BRUNO LANDAU ALBRECHT**

# CONTROLE DE UMA CADEIRA DE RODAS MOTORIZADA ATRAVÉS DE ELETROMIOGRAFIA EM UMA PLATAFORMA EMBARCADA

Porto Alegre
(2010)

**BRUNO LANDAU ALBRECHT**

# CONTROLE DE UMA CADEIRA DE RODAS MOTORIZADA ATRAVÉS DE ELETROMIOGRAFIA EM UMA PLATAFORMA EMBARCADA

Projeto de diplomação apresentado ao Instituto de Informática da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia da Computação.

Área de concentração: Instrumentação Eletro-Eletrônica, Instrumentação Biomédica.

ORIENTADOR: Prof. Dr. Alexandre Balbinot

Porto Alegre

(2010)

## DEDICATÓRIA

*Dedico este trabalho à minha família e minha namorada,*

*que nunca deixaram de acreditar.*

## AGRADECIMENTOS

*À minha família*

*pelo amor e apoio.*

*Aos meus pais em especial*

*pela educação de qualidade que puderam me prover.*

*À minha namorada Thabata*

*pelo amor incondicional, apesar da distância física que nos separa,*

*pela paciência para entender a distância ainda maior nos momentos difíceis*

*e por jamais deixar de acreditar no meu potencial.*

*Aos meus colegas de faculdade e da Falker*

*pelas discussões infindáveis sobre assuntos muitas vezes insignificantes,*

*pelo mate durante a aula e o truco com bolacha nos intervalos,*

*pelo conhecimento passado e o companheirismo para todas as horas.*

*Ao professor e orientador Dr. Alexandre Balbinot*

*pela orientação de qualidade, mostrando que ainda há professores que se importam,*

*pelos ensinamentos, incentivos e conselhos.*

*A todos os demais professores*

*pelo conhecimento transmitido.*

*À Universidade Federal do Rio Grande do Sul.*

*Ao Laboratório de Instrumentação Eletro-Eletrônica (IEE).*

*Aos grupos do Programa de Educação Tutorial (PET).*

*E a todos que direta ou indiretamente contribuíram nesta longa jornada.*

## RESUMO

Este trabalho apresenta a criação de um método para adquirir sinais mioelétricos provenientes do movimento das pálpebras e utilizá-los no controle de uma cadeira de rodas motorizada. O sistema é composto de um eletromiógrafo com ganho e filtragem configuráveis digitalmente através de um microcontrolador e um algoritmo de detecção de piscadas voluntárias e utilização destas para a ativação de uma cadeira de rodas. Através do eletromiógrafo, são tratados analogicamente biosinais de ambas as pálpebras posicionando-se eletrodos próximos dos músculos Orbicularis Oculi. Os sinais são filtrados através de um filtro de capacitores chaveados cuja frequência de corte pode ser configurada e amplificados com ganhos também configuráveis. Os sinais mioelétricos são então convertidos para dados digitais através de um conversor analógico digital de 13 bits e um algoritmo de detecção de piscadas é ativado. A partir de um protocolo pré-determinado, foi possível relacionar o movimento das pálpebras com comandos que são transmitidos à cadeira de rodas motorizada. Foi realizada uma validação de cada estágio da plataforma de hardware com sinais conhecidos e então realizados testes com sinais mioelétricos do músculo bíceps. A utilização dos biosinais relacionados ao movimento das pálpebras se mostrou satisfatória no que diz respeito ao controle da cadeira de rodas.



**Palavras-Chave:** biosinais, eletromiografia, processamento digital de sinais, cadeira de rodas motorizada, microcontroladores

## ABSTRACT

This paper presents the creation of a method to acquire myoelectric signals from the eyelids movement and use them to control a motorized wheelchair. The system consists of an electromyograph with digitally programmable gain and filtering via a microcontroller and a detection algorithm and use of voluntary blinks for enabling a wheelchair. Through the electromyograph, biosignals from both eyelids are treated analogically by positioning electrodes near muscle Orbicularis Oculi. The signals are filtered by a switched capacitor filter whose cutoff frequency can be set and amplified with gains also configurable. The myoelectric signals are then converted to digital data through a 13 bits analog to digital converter and a blink detection algorithm is activated. From a pre-determined protocol, it was possible to relate the eyelids movement with commands that are transmitted to the motorized wheelchair. The validation of each stage of the hardware platform was done with well known signals and then tests were performed with myoelectric signals from the biceps muscle. The use of biosignals related to movement of the eyelids was considered satisfactory with regard to the control of the wheelchair.



**Keywords:** biosignals, electromyography, digital signal processing, motorized wheelchair, microcontrollers.

**SUMÁRIO**

1 INTRODUÇÃO....13
1.1 Justificativa....13
1.2 Objetivos....14

2 REVISÃO BIBLIOGRÁFICA....15
2.1 Tecnologia Assistiva....15
2.1.1 Cadeira de rodas....17
2.2 Sinal Mioelétrico....18
2.2.1 Contração muscular....19
2.2.2 Musculatura facial....21
2.2.3 Eletromiografia (EMG)....24
2.2.4 Eletromiografia do sinal da piscada....28
2.2.5 Um breve relato sobre alguns trabalhos relacionados ao uso de EMG no controle de cadeiras de rodas....28

3 METODOLOGIA EXPERIMENTAL....30
3.1 Plataforma de Hardware....31
3.1.1 Eletrodos semi-ativos....31
3.1.2 Filtragem....32
3.1.3 Amplificação....39
3.1.4 Conversão analógico para digital....42
3.1.5 Flexibilidade....44
3.1.6 Cadeira de Rodas Motorizada....46
3.1.7 Alimentação da plataforma de hardware....47
3.2 Plataforma de Software....48
3.2.1 Interface com o usuário....49
3.2.2 Aquisição de dados....50
3.2.2.1 Taxa de Amostragem....51
3.2.3 Tratamento digital do sinal....52
3.2.4 Algoritmos de calibração....54
3.2.5 Detecção de piscadas....55
3.2.6 Controle da cadeira de rodas motorizada....56
3.2.7 Envio de dados....57

4 RESULTADOS E DISCUSSÕES....59
4.1 Validação da plataforma de hardware....59
4.2 Validação da plataforma de software....68
4.3 Validação com sinais reais....71
4.3.1 Ruídos devido à alimentação....73
4.4 Controle da cadeira de rodas....75

5 CONCLUSÕES....78

6 SUGESTÕES PARA TRABALHOS FUTUROS....79
6.1 Melhorias da plataforma de hardware....79
6.2 Melhorias da plataforma de software....80

REFERÊNCIAS BIBLIOGRÁFICAS....82

ANEXO A.1.....87
ANEXO A.2.....88
ANEXO A.3.....89
ANEXO A.4.....90
ANEXO A.5.....91
ANEXO A.6.....92
ANEXO A.7.....93
ANEXO A.8.....94
ANEXO A.9.....95
ANEXO A.10.....96
ANEXO A.11.....97
ANEXO A.12.....98
ANEXO A.13.....99
ANEXO A.14.....127
ANEXO A.15.....129
ANEXO A.16.....134
ANEXO A.17.....135

## LISTA DE FIGURAS

Figura 2.1 - Diversas categorias de tecnologias assistivas.....17
Figura 2.2 - (a) Cadeira de rodas manual comum e (b) cadeira de rodas elétrica guiada por joystick e (c) cadeira de rodas adaptada para esportes e (d) cadeira de rodas antiga, feita de madeira.....18
Figura 2.3 - Célula Nervosa.....20
Figura 2.4 - Potenciais de ação para diferentes raios de fibra nervosa.....20
Figura 2.5 - Junção Neuromuscular.....21
Figura 2.6 - Músculos Faciais.....22
Figura 2.7 - (a) Unidade motora , e (b) fibras intercaladas.....23
Figura 2.8 - Efeito da tetanização.....24
Figura 2.9 - (a) Eletrodos não Invasivos Passivos e (b) Eletrodos não Invasivos Semi-Ativos e (c) Eletrodos não Invasivos Semi-Ativos e (d) Eletrodos Invasivos.....26
Figura 2.10 - EMG do bíceps exercitando-se com 10 kgf.....27
Figura 3.1 - Diagrama de blocos do sistema proposto.....30
Figura 3.2 - Diagrama de blocos da plataforma de aquisição de sinais analógicos.. 31
Figura 3.3 - Esquemático do eletrodo semi-ativo.....32
Figura 3.4 - Filtro passa-altas utilizado.....33
Figura 3.5 - Diagrama de Bode do primeiro estágio de filtragem.....33
Figura 3.6 - Esquemático do filtro de capacitores chaveados passa baixas Chebychev de quarta ordem.....34
Figura 3.7 - Efeito de serrilhamento da saída do CI MF10.....36
Figura 3.8 - Esquemático do filtro passa baixas com frequência de corte fixa.....38
Figura 3.9 - Diagrama de Bode do filtro passa baixas com frequência de corte fixa. 38
Figura 3.10 - Filtro passa altas com frequência de corte em 1,59 Hz e tensão de referência em 2,5 V.....39
Figura 3.11 - Diagrama de Bode do filtro passa altas utilizado.....39
Figura 3.12 - Esquemático do estágio de amplificação com ganho programável.....41
Figura 3.13 - Placa USB-6008 da National Instruments. .....42
Figura 3.14 - Diagrama de blocos funcionais do CI MCP3304.....44
Figura 3.15 - Esquemático do circuito digital.....46
Figura 3.16 - (a) Cadeira de rodas motorizada e (b) foto aproximada do motor.....47
Figura 3.17 - Fluxograma da plataforma de software.....49
Figura 3.18 - Simulação da tela de comandos da plataforma de hardware.....50
Figura 3.19 - Fluxograma do tratamento digital dos sinais.....52
Figura 3.20 - Fluxograma do algoritmo de detecção de piscadas.....55
Figura 3.21 - Fluxograma do controle da cadeira de rodas motorizada.....57
Figura 4.1 - Simulação do eletrodo semi ativo.....60
Figura 4.2 - Imagem do osciloscópio realizando uma medição do circuito do eletrodo semi-ativo, onde CH2 é a entrada e CH4 é a saída.....60
Figura 4.3 - Foto do eletrodo semi ativo utilizado em um dos canais.....60
Figura 4.4 - Simulação do estágio de entrada.....61
Figura 4.5 - Imagem do osciloscópio realizando uma medição no circuito de entrada, onde CH2 é a entrada e CH1 é a saída.....61
Figura 4.6 - Foto do estágio de entrada.....62
Figura 4.7 - Imagem do osciloscópio mostrando o efeito de serrilhamento no sinal de saída do CI MF10.....63
Figura 4.8 - Medições com o osciloscópio do estágio de filtragem, onde CH2 é a saída e CH3 é a entrada com frequência (a) 303 Hz e (b) 1689 Hz.....63

Figura 4.9 - Foto da placa de circuito impresso criada para o estágio de filtragem…63
Figura 4.10 - (a) Simulação do circuito de amplificação com ganho configurado em 5 vezes e (b) em 10 vezes.....64
Figura 4.11 - (a) Imagem do osciloscópio realizando uma medição no circuito de amplificação com ganho configurado em 5 vezes e (b) em 10 vezes.....65
Figura 4.12 - Foto da placa do estágio de amplificação.....65
Figura 4.13 - Foto da parte digital da plataforma de hardware.....66
Figura 4.14 - (a) Foto da face superior da placa de distribuição dos sinais e (b) foto da face inferior da mesma.....67
Figura 4.15 - Foto do circuito de alimentação da plataforma de hardware.....68
Figura 4.16 - Simulação da comunicação Serial do microcontrolador.....69
Figura 4.17 - Imagem do software Realterm realizando a comunicação Serial com o microcontrolador.....69
Figura 4.18 - (a) Imagem do osciloscópio simulado realizando medições do sinal de controle da frequência de corte do estágio de filtragem em 500 Hz e (b) em 300 Hz.....70
Figura 4.19 - (a) Imagem do osciloscópio realizando medições do sinal de controle da frequência de corte do estágio de filtragem em 500 Hz e (b) em 200 Hz.....71
Figura 4.20 - Validação do canal 1 de aquisição com sinal mioelétrico do músculo bíceps exercitando-se com peso de 5kgf.....72
Figura 4.21 - Validação do canal 2 de aquisição com sinal mioelétrico do músculo bíceps exercitando-se com peso de 5kgf.....73
Figura 4.22 - Sinal do músculo bíceps em repouso com a aquisição dos dados em um notebook conectado à rede elétrica.....74
Figura 4.23 - Sinal do músculo bíceps em repouso com a aquisição dos dados em um notebook operando através de sua bateria interna.....74
Figura 4.24 - Posicionamento dos eletrodos para as medições.....75
Figura 4.25 - Comandos “andar para frente” e “parar” ativados por sinais da piscada dos olhos.....76
Figura 4.26 - Comandos “andar para trás” e “parar” ativados por sinais da piscada dos olhos.....76
Figura 4.27 - Comandos “andar para a direita”, tentativa de ativação do comando “andar para frente” e comando “parar” ativados por sinais da piscada dos olhos.....77

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| CAT | Comitê de Ajudas Técnicas |
| EEG | Eletroencefalografia |
| EMG | Eletromiografia |
| EOG | Eletrooculografia |
| ACh | Acetilcolina |
| MUAP | Motor Unit Action Potential |
| MUAPT | Motor Unit Action Potential Train |
| LABVIEW | Laboratory Virtual Instrument Engineering |
| DC | Corrente Contínua |
| CI | Circuito Integrado |
| SNR | Signal to Noise Ratio |
| TTL | Transistor-Transistor Logic |
| ASCII | Código Padrão Americano para o Intercâmbio de Informação |
| SPI | Serial Peripheral Interface Bus |
| DIP | Dual In-Line Package |
| SOIC | Small-outline integrated circuit |
| SSOP | Shrink Small-Outline Package |
| SMD | Surface Mount Devices |

## LISTA DE TABELAS

Tabela 2.1 - Sinais Biomédicos.....27
Tabela 3.1 - Valores de ganho escolhidos e as resistências correspondentes.....41
Tabela 3.2 - Resolução do sinal de entrada do sistema para os diversos ganhos do estágio de amplificação.....43
Tabela 3.3 - Codificação de cada byte das médias enviadas pela comunicação Serial RS-232C, onde o bit 7 corresponde ao mais significativo.....58
Tabela 3.4 - Codificação do byte correspondente ao estado atual da cadeira de rodas.....58

# 1 INTRODUÇÃO

Pessoas portadoras de deficiências físicas e/ou neurológicas apresentam grandes problemas não somente de integração à sociedade, mas também de comunicação, em função de suas dificuldades na realização de tarefas simples. Através do grande desenvolvimento científico observado nos últimos anos, esta situação tem mudado, basicamente devido à popularização de tecnologias assistivas. As inúmeras pesquisas realizadas nesse campo estão proporcionando uma maior independência dos portadores de necessidades especiais, facilitando a reintegração à sociedade.

Uma das áreas mais desenvolvidas dentro do campo de tecnologias assistivas é a da utilização dos biosinais no desenvolvimento de equipamentos que facilitam o dia-a-dia de pessoas portadoras de necessidades especiais. Dentre esses sinais, pode-se destacar os mioelétricos (sinais musculares), obtidos através de um equipamento denominado eletromiógrafo, que podem ser utilizados como entradas em sistemas de interação com cadeiras de rodas, próteses experimentais, etc.

## 1.1 Justificativa

Portadores de deficiências físicas sofrem muita discriminação devido às suas dificuldades em realizar tarefas simples do dia-a-dia, como, por exemplo, caminhar. Esse tipo de problema acaba acarretando na dependência desta pessoa a outras pessoas, por exemplo, para se locomover. Pode-se citar ainda casos em que o indivíduo consegue locomover-se sozinho, mas isso acaba por inutilizar os braços, impossibilitando-o de carregar algum objeto. Os dependentes de cadeiras de rodas têm disponíveis modelos motorizados, o que facilita sua locomoção, porém, por serem, em sua maioria, controladas através de um joystick, não resolvem o problema da ocupação dos braços. Além disso, portadores de deficiências motoras podem não conseguir utilizar cadeiras de rodas comuns, novamente necessitando de alguém para auxiliá-los a se mover.

O sinal mioelétrico é o sinal de controle muscular do corpo humano que contém a informação da intenção do usuário contrair um músculo e, consequentemente, realizar um movimento, por exemplo, piscar os olhos. A

utilização desse artifício permite a criação de um protocolo para controlar objetos, tais como a cadeira de rodas motorizada, sem a necessidade de ocupar mãos e braços para isso. Entretanto, sinais desta natureza são difíceis de utilizar, devido às baixas amplitudes captadas, sendo necessário projetar sistemas altamente imunes a ruídos.

Finalmente, a aquisição dos sinais mioelétricos pode ser realizada facilmente com eletrodos não invasivos, o que diminui o desconforto causado por eletrodos invasivos. Além disso, a utilização de eletrodos de superfície é mais prática para o usuário, uma vez que podem ser colocados, retirados e higienizados rapidamente e sem representar perigos de saúde.

## 1.2 Objetivos

O presente trabalho tem como objetivo o estudo, simulação e desenvolvimento de um sistema portátil que, através de eletrodos não invasivos (também chamados de eletrodos de superfície) e um eletromiógrafo, capture os biosinais relacionados ao movimento de piscada do olho, interprete-os e utilize-os para o controle do movimento de uma cadeira de rodas motorizadas. Considerando a possibilidade de distinguir tais sinais, será criado ainda um protocolo para facilitar sua utilização, bem como um método de calibração dinâmica de tal sistema.

# 2 REVISÃO BIBLIOGRÁFICA

## 2.1 Tecnologia Assistiva

Tecnologia assistiva é o termo utilizado para designar uma grande variedade de equipamentos que visam aumentar a habilidade funcional de pessoas com deficiências e idosos e, consequentemente, ampliar a independência e inclusão social do indivíduo. No Brasil, no final de 2006, a Secretaria Especial dos Direitos Humanos (SEDH/PR), através da portaria número 142, instituiu o Comitê de Ajudas Técnicas (CAT), que reuniu um grupo de especialistas e representantes de órgãos governamentais com o objetivo de apresentar propostas de políticas governamentais e parcerias entre a sociedade civil e órgãos públicos referentes à área de tecnologia assistiva; estruturar as diretrizes da área de conhecimento; realizar levantamento dos recursos humanos que atualmente trabalham com o tema; detectar os centros regionais de referência, objetivando a formação de rede nacional integrada; estimular nas esferas federal, estadual, municipal, a criação de centros de referência; propor a criação de cursos na área de tecnologia assistiva, bem como o desenvolvimento de outras ações com o objetivo de formar recursos humanos qualificados e propor a elaboração de estudos e pesquisas, relacionados com o tema da tecnologia assistiva. De tal reunião, o CAT aprovou, no final do ano seguinte, o conceito de Tecnologia Assistiva como "*uma área do conhecimento, de característica interdisciplinar que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social.*" (CORDE, 2007). Dando um grande passo no sentido de incentivar o desenvolvimento da tecnologia assistiva no Brasil, em junho de 2009, através da lei 11958, e em outubro de 2009, através do decreto 6980, é criada a Subsecretaria Nacional de Promoção dos Direitos das Pessoas com Deficiências, cujo objetivo é articular e coordenar políticas públicas voltadas para pessoas com deficiência.

Apesar da preocupação com tecnologia assistiva ser um assunto novo no Brasil, este já é um assunto recorrente em outros países. Nos Estados Unidos, por exemplo, onde há cerca de 54 milhões de pessoas, ou 20,6% da população, com algum tipo de deficiência, há leis específicas desde os anos 70 (NIST, 2010),

englobando inclusive um capítulo inteiro na legislação do país (ESTADOS UNIDOS DA AMÉRICA, 2006).

Por ser um termo de grande abrangência, costuma-se classificar a tecnologia assistiva em diversas categorias, com o intuito de organizar a utilização, prescrição, estudo e pesquisa de recursos e serviços. Dentre as categorias, pode-se citar os auxílios para a vida diária e prática, abrangendo materiais e produtos que favoreçam a autonomia e independência em tarefas rotineiras ou facilitam o cuidado de pessoas em situação de dependência de auxílio, em atividades como cozinhar, alimentar-se, vestir-se, etc; a comunicação aumentativa e alternativa, destinada a atender pessoas com problemas de comunicação utilizando recursos como pranchetas com simbologia gráfica, letras ou palavras escritas, ou através do computador, com softwares específicos; os recursos de acessibilidade ao computador, como forma de facilitar o acesso aos equipamentos de pessoas com disfunções motoras ou sensoriais através, por exemplo, de teclados modificados, softwares de reconhecimento de voz, impressoras em braile, etc; os sistemas de controle de ambiente, provendo às pessoas com deficiências de locomoção, uma forma fácil e rápida de interagir com os objetos de suas casas; os projetos arquitetônicos para acessibilidade, que garantem acesso, funcionalidade e mobilidade a todas as pessoas, independentemente de suas condições físicas e sensoriais, através de adaptações estruturais, como rampas, elevadores e banheiros especiais; as órteses e próteses, como forma de substituir partes ausentes do corpo com peças artificiais ou de garantir-lhes um melhor funcionamento, posicionamento e/ou estabilização; a adequação postural, cujo objetivo é garantir posturas alinhadas, estáveis e com boa distribuição do peso corporal; os auxílios de mobilidade, abrangendo desde bengalas a cadeiras de rodas elétricas que auxiliem a mobilidade pessoal; os auxílios para cegos ou pessoas com visão debilitada, incluindo equipamentos falados, como relógios e termômetros, além de auxílios ópticos, softwares leitores de tela e impressoras braile; os auxílios para pessoas com disfunções auditivas, como aparelhos para surdez e sistemas de alerta táctil-visual; e adaptações em veículos, que possibilitem portadores de deficiência física dirigir um automóvel e acessá-lo, como rampas e elevadores para cadeiras de rodas (BERSH, 2008). Como exemplo, a Figura 2.1 apresenta alguns exemplos de tecnologia assistiva, como pernas mecânicas, cadeiras de rodas motorizadas e etc.

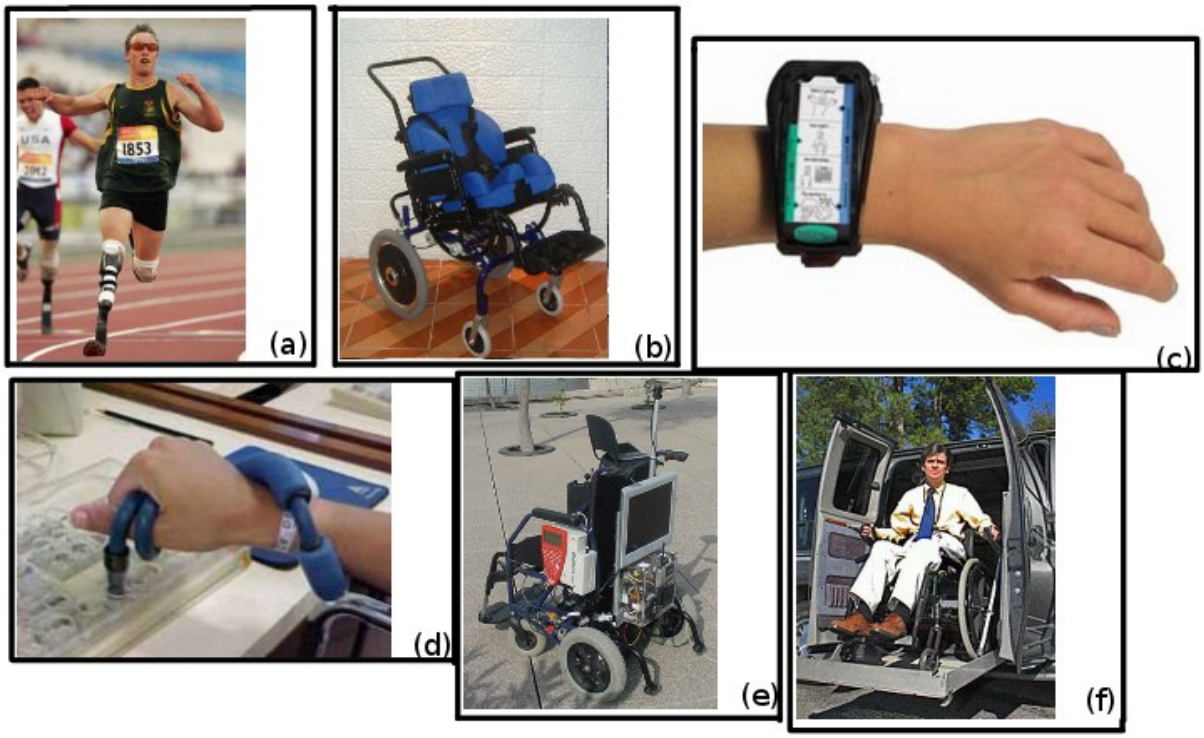



**Figura 2.1 - Diversas categorias de tecnologias assistivas.**
**Fonte – (a), (b), (c) e (f) Adaptados de BERSH, 2008, e (d) adaptado de GALVÃO FILHO *et al*, 2008, e (e) adaptado de CORTÉS *et al*, 2007.**

### 2.1.1 Cadeira de rodas

Identificada em gravuras chinesas datadas do século VI, a cadeira de rodas teve poucas melhorias desde o início de sua documentação formal, no século XVI. Primeiramente utilizada como meio de transporte, ao invés de prover mobilidade a pessoas com deficiências, a cadeira, feita então com madeira, dotada de rodas grandes na parte da frente e pequenas atrás, ganhou motores elétricos no início do século XX. Já durante a Primeira Guerra Mundial, as cadeiras de rodas começaram a ser utilizadas por deficientes, com acionamento realizado por um simples botão liga-desliga, sem controle de velocidade. Finalmente na década de 1960, um joystick foi adaptado para prover um melhor controle da direção e velocidade, mas somente no final da década de 1990 iniciaram-se os estudos da utilização de biosinais para interagir com as cadeiras de rodas (CLARK, 1997). Na Figura 2.2, pode-se visualizar alguns modelos de cadeiras de rodas, desde as feitas de madeira, até as especialmente feitas para esportes.


**Figura 2.2 - (a) Cadeira de rodas manual comum e (b) cadeira de rodas elétrica guiada por *joystick* e (c) cadeira de rodas adaptada para esportes e (d) cadeira de rodas antiga, feita de madeira.**
**Fontes – (a) SUNRISE MEDICAL, 2010, e (b) PRIDE MOBILITY, 2010, e (c) WHEELCHAIR NETWORK, 2010, e (d) MUSEUM OF DISABILITY, 2010.**

## 2.2 Sinal Mioelétrico

Observados pela primeira vez pelo cientista Luigi Galvani, em 1791, em sapos, os sinais mioelétricos representam o reflexo involuntário da passagem de corrente elétrica pelas fibras musculares durante as contrações musculares, conforme descoberto por Frenchman Dubois-Reymond, em 1849. A intensificação dos estudos a respeito da morfologia dos sinais mioelétricos durante o século XX levou ao surgimento da eletromiografia (EMG), na qual eram utilizados tubos catódicos e eletrodos específicos que, ligados a um osciloscópio, permitiam visualizar tais sinais. Pela utilização, pela primeira vez, desta maneira, o método

rendeu o prêmio Nobel em Medicina e Fisiologia em 1944 aos cientistas Herbert Gasser e Joseph Erlanger (FAVIEIRO, 2009). Atualmente, os métodos de aquisição dos sinais mioelétricos estão bastante evoluídos, compreendendo eletrônica analógica e digital, o que provê maior precisão nas medições com um custo reduzido.

### 2.2.1 Contração muscular

Para ser realizada a contração muscular, diversos processos ocorrem antes, tais como a condução do impulso nervoso pelo neurônio motor, a transmissão sináptica, e a transmissão dos potencias de ação do músculo, que, somados, representam o sinal mioelétrico. Finalmente, ocorre a contração das fibras musculares e, por consequência, o movimento muscular (MARIEB *et al*, 2006).

Os neurônios, ou células nervosas, podem gerar dois tipos de sinais: potenciais graduados (*graded potentials*), que são atenuados de acordo com a distância que percorrem, são comumente encontrados em neurônios sensores e têm por objetivo ativar ou inibir os potenciais de ação; e os potenciais de ação (*action potentials*), que não são atenuados de acordo com a distância percorrida, são comumente encontrados em neurônios motores e têm por objetivo propagar o impulso nervoso nos neurônios. O neurônio motor, que normalmente pode ser encontrado no Sistema Nervoso Central, é composto pelo corpo celular, dendritos e um axônio, conforme pode ser visualizado na Figura 2.3. O corpo celular engloba um núcleo e um nucléolo bem definido, enquanto os dendritos são geralmente pequenas extensões citoplasmáticas espessas e altamente ramificadas, cuja função é receber os impulsos elétricos e conduzi-los em direção ao corpo celular. O axônio, também chamado de fibra nervosa quando possui um maior comprimento, tem por objetivo transmitir os sinais neurais para a célula nervosa seguinte ou para as células efetoras (*effector cells*), como músculos e glândulas. (MARIEB *et al*, 2006).

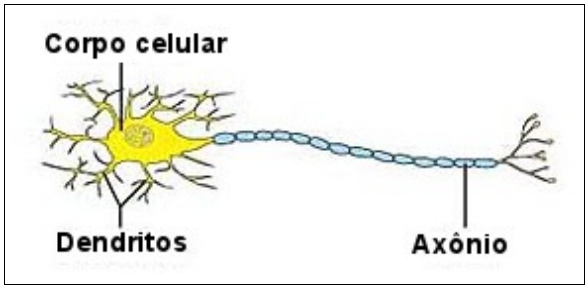


**Figura 2.3 - Célula Nervosa.**
**Fonte - Adaptado de MARIEB *et al*, 2006.**

O potencial de ação neural não é utilizado somente por células nervosas, mas por diversas outras, todavia, a utilização pelo sistema nervoso é extensa, sendo utilizada como meio de comunicação entre neurônios e entre neurônios e tecidos, como músculos e glândulas, podendo percorrer distâncias grandes pelos axônios (do fim da medula espinhal ao músculo do pé, por exemplo). Tal efeito é iniciado quando ocorre uma mudança abrupta no potencial de repouso negativo da membrana do neurônio, tornando-se positivo, e, em seguida, seu término é caracterizado pela volta do potencial a seu estado negativo (Figura 2.4). Cada ciclo dura poucos milissegundos, mas pode ser seguido por outros diversos, o que indica a formação de um trem de impulsos (*spike train*), permitindo, com o aumento ou diminuição da frequência, que os neurônios modulem sua comunicação (BRONZINO, 1995).

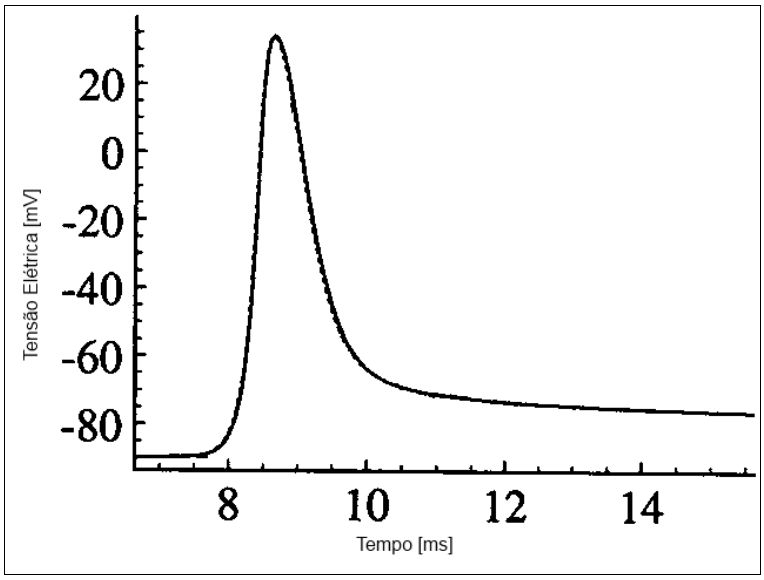



**Figura 2.4 - Potenciais de ação para diferentes raios de fibra nervosa.**
**Fonte - Adaptado de BRONZINO, 1995.**

Segundo MARIEB *et al* (2006), a comunicação entre os neurônios motores e o músculo se dá através de uma junção neuromuscular com a placa motora, que nada mais é do que a região da membrana plasmática de uma fibra muscular (sarcolema) onde se dá o encontro entre o axônio e o músculo, permitindo desencadear a contração muscular. Nesta área, o neurônio forma o chamado botão sináptico (*synaptic bouton*), através do qual será liberado o neurotransmissor

acetilcolina (ACh). Basicamente, quando um impulso nervoso chega à placa motora, ocorre a liberação do neurotransmissor, que atua na membrana da placa motora de forma a aumentar a permeabilidade de íon de sódio (Na+), cujo fluxo (do sódio para a placa motora) causa a despolarização da membrana, implicando, finalmente, em um potencial de ação, que, propagado através da fibra muscular, causa a contração do músculo. Logo após a liberação da ACh, uma enzima chamada acetilcolinesterase também é liberada, causando a quebra do neurotransmissor, permitindo a repolarização da membrana e deixando-a pronta para o próximo estímulo. Na Figura 2.5, pode-se visualizar a junção neuromuscular.

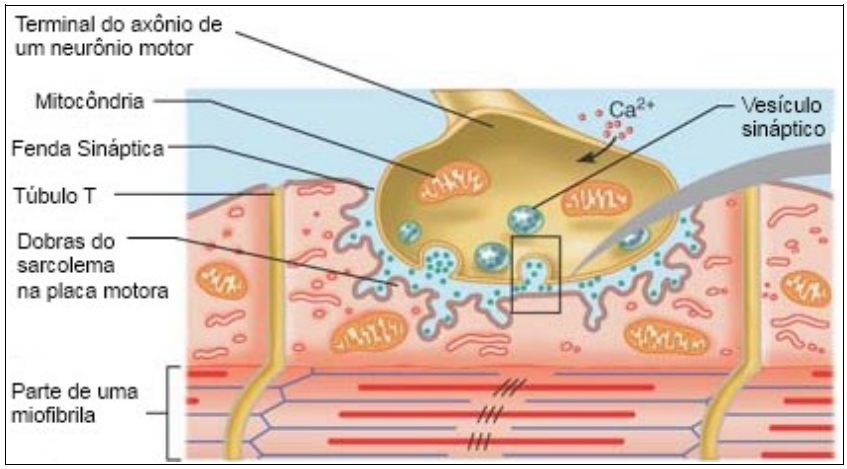



**Figura 2.5 - Junção Neuromuscular.**
**Fonte – Adaptado de MARIEB *et al, 2006*.**

### 2.2.2 Musculatura facial

A face humana é composta por uma complexa cadeia de músculos que, conforme são contraídos ou relaxados, formam a expressão facial. Por serem derivados do *facial somitomere*, tais músculos são inervados pelo nervo facial. Na Figura 2.6, pode-se visualizar alguns destes músculos com seus nomes. Como percebe-se, os músculos ao redor dos olhos, chamados Orbicularis Oculi, assim como aqueles ao redor dos lábios, chamados Orbicularis Oris, são músculo cujos fascículos possuem padrão circular, ou seja, são organizados como anéis concêntricos. Tal padrão tem por objetivo controlar a abertura e fechamento dos orifícios do corpos, neste caso, olhos e boca, respectivamente (STERN, 2003).

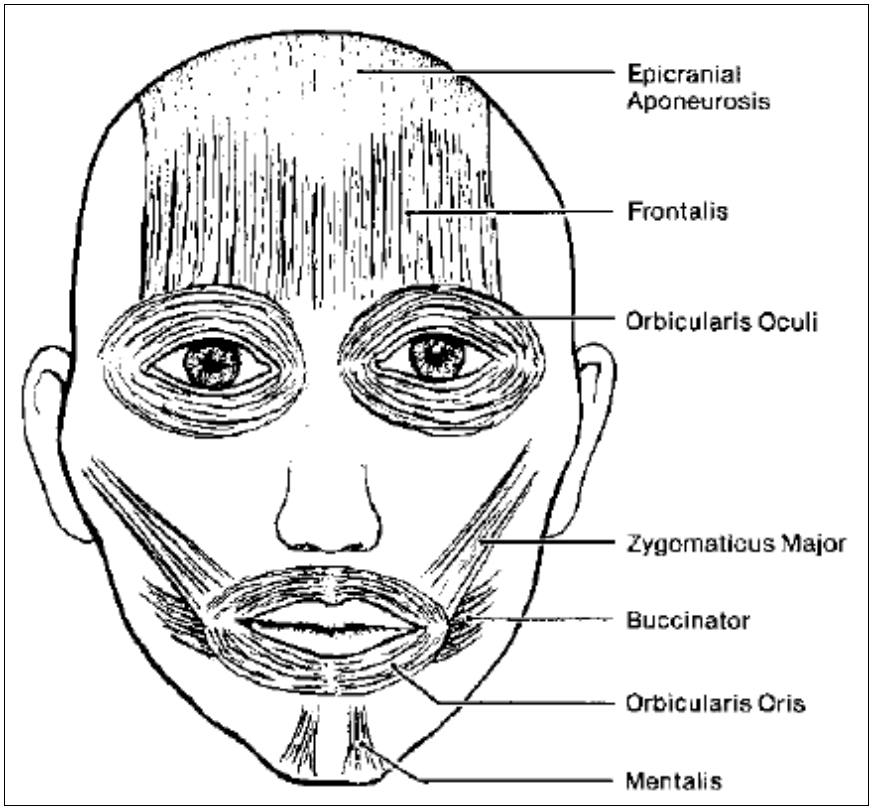



**Figura 2.6 - Músculos Faciais.**
**Fonte – Adaptado de STERN, 2003.**

O Orbicularis Oculi é um músculo fino que circunda a borda da órbita dos olhos, caracterizado por ser um esfíncter tripartido da pálpebra, cuja função é proteger os olhos do excesso de luz e de lesões. O músculo consiste de três porções com funções diferentes: a porção palpebral, a porção lacrimal e a porção orbital. A porção palpebral age involuntariamente, fechando as pálpebras gentilmente, como quando dormimos ou piscamos. Ela se origina do ligamento palpebral, passa por cima de cada pálpebra e insere-se na rafe palpebral. Quando as pálpebras superiores estão abertas, as fibras desta porção arqueiam-se para cima, até que contraiam-se, fazendo com que as pálpebras fechem. Já as pálpebras inferiores movem-se muito pouco durante o movimento de abrir e fechar dos olhos, inclusive causando em pessoas de idade mais avançada a queda deste tecido devido a seu próprio peso. A porção lacrimal desenha as pálpebras e as extremidades dos canais lacrimais e comprime-os contra a superfície do globo ocular, além de comprimir o saco lacrimal, colocando-a (a porção) no local mais favorável para receber as lágrimas. Tal porção é composta, também, pelo músculo de Riolan, cuja função é manter as pálpebras juntas e a passagem lacrimal à prova de água. A porção orbital consiste de fibras musculares que circundam a periferia

orbital às raízes das pálpebras. Essa porção é responsável pelos movimentos voluntários de tentar fechar os olhos sem fechar as pálpebras, interpondo tanta pele quanto possível entre o mundo externo e o globo ocular (STERN, 2003).

Cada músculo do corpo humano é inervado por diversos neurônios motores, que partem da medula espinhal, de forma que cada grupo de fibras musculares inervadas por somente uma fibra nervosa motora formam a chamada unidade motora, como pode-se visualizar na Figura 2.7(a). O número de fibras musculares por unidade motora pode variar significativamente, de acordo com a precisão necessária ao músculo, de três, em músculos externos dos olhos, a cento e vinte fibras, para músculos com movimentos mais grosseiros como o gastrocnêmio (músculo da perna). Em geral, as fibras de diferentes unidades motoras são inervadas intercaladamente, como pode-se visualizar na Figura 2.7(b) (BRONZINO, 1995).

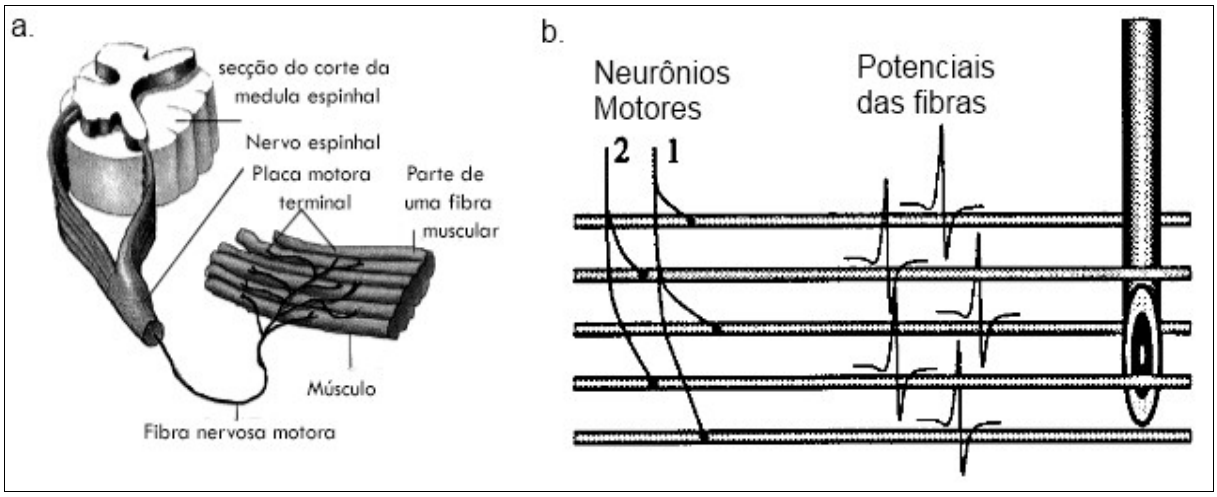



**Figura 2.7 - (a) Unidade motora , e (b) fibras intercaladas.**
**Fonte – (a) BELTRAMINI, 1999 e (b) adaptado de BRONZINO, 1995.**

Além do número de unidades motoras utilizadas, a força de um músculo depende da frequência com que cada uma é estimulada, ou seja, quanto maior a frequência, é atingido um momento em que cada nova contração ocorre antes do término da primeira, causando uma sobreposição de contrações, aumentando sua força. Quando tal frequência atinge um ponto crítico, ocorre o efeito da tetanização, ou seja, as contração praticamente fundem-se, aparecendo, então como uniforme e contínua (GUYTON, 2006). Na Figura 2.8, pode-se visualizar esse efeito em um gráfico da força da contração do músculo pela frequência de estimulação.

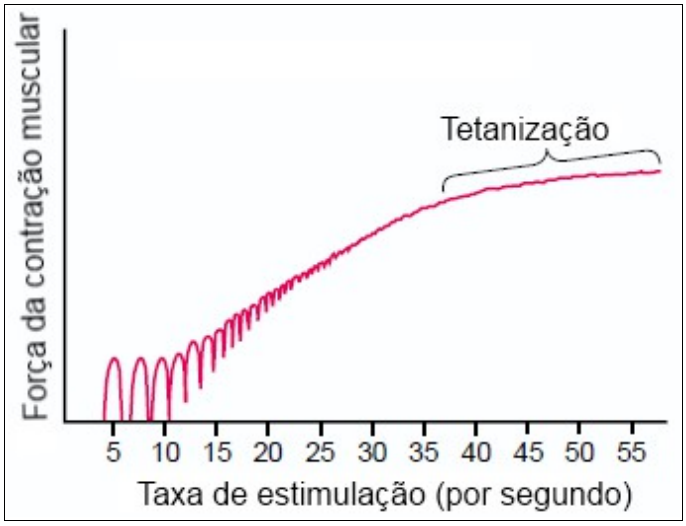



**Figura 2.8 - Efeito da tetanização.**
**Fonte – Adaptado de GUYTON, 2006.**

Apesar do músculo em geral contrair-se como um todo, todas as fibras musculares de uma unidade motora não são estimuladas simultaneamente, devido às diferentes distâncias que os potenciais de ação devem percorrer através do axônio do neurônio motor e à natureza aleatória das descargas de ACh nas junções neuromusculares. Além disso, como a duração desses potenciais de ação possuem um período pequeno (de 2 a 10 ms), para que um músculo consiga sustentar uma contração por mais tempo, é necessário que as unidades motoras sejam estimuladas continuamente. O resultado da soma algébrica em um instante de tempo desses potenciais de ação em uma unidade é chamada potencial de ação da unidade motora (em inglês, MUAP ou *Motor Unit Action Potential*) e sua sequência é denominada trem de potenciais de ação da unidade motora (em inglês, MUAPT ou *Motor Unit Action Potential*) (BRONZINO, 1995).

### 2.2.3 Eletromiografia (EMG)

O movimento e a posição dos membros do corpo humano são controlados pelo fluxo de sinais elétricos entre os músculos e o sistema nervoso, os já mencionados potenciais de ação. O registro e estudo desses sinais elétricos denomina-se Eletromiografia (EMG) e tem sido alvo de utilização desde em clínicas

médicas, como forma de diagnóstico, até em sistema de interação entre o homem e equipamentos eletrônicos.

Para registrar tais sinais, podem ser utilizados eletrodos invasivos (intramusculares) ou não invasivos (de superfície), conforme podem ser visualizados na Figura 2.9, e são geralmente confeccionados com Prata e cobertos com Cloreto de Prata (Ag-AgCl), por ser uma liga metálica não polarizável, evitando o aparecimento de *offsets* na medição. Uma vez que os eletrodos intramusculares são geralmente compostos de finos fios metálicos ou agulhas, permitem a captação de sinais bastante precisos (altas amplitudes, espectro de frequência mais largo e maior SNR), em regiões bem definidas, evitando interferências de outros músculos, porém, causam grande desconforto ao usuário, além de não permitir uma grande repetibilidade dos experimentos. Os eletrodos de superfície, por sua vez, são constituídos por pequenas placas que ficam grudadas à pele do usuário, portanto, são facilmente aplicáveis ao usuário. Apesar disso, possuem limitações quanto à precisão de seus sinais, que ficam tipicamente em uma faixa de 0 a 500 Hz em amplitudes máximas de 10 mV, sendo muito recomendada a utilização de pasta ou gel condutor, a fim de diminuir a impedância de contato eletrodo-pele. Deve-se ainda ressaltar que a utilização de eletrodos de superfície permite apenas a captação de sinais mioelétricos próximos à superfície da pele e são mais suscetíveis às interferências de outros músculos, fazendo do seu correto posicionamento característica importante do EMG (ORTOLAN, 2002). Na tabela 52.1 de BRONZINO (1995), podem ser encontrados diversos tipos de eletrodos sugeridos para os mais variados tipos de medições de biosinais.

Os eletrodos de superfície podem ainda ser classificados como ativos, quando possuem um primeiro banco de amplificação e/ou filtragem analógica muito próxima do contato eletrodo-pele, tendo como principal vantagem o aumento da razão sinal-ruído, ou passivos, quando toda a parte eletrônica do sistema está localizada longe do contato eletrodo-pele. Há ainda uma classificação intermediária denominada semi-ativa, na qual o primeiro banco de amplificação e/ou filtragem encontra-se no meio do cabo que leva o sinal do eletrodo em si à parte eletrônica do sistema, mais próximo do contato eletrodo-pele. Quanto à configuração utilizada, esta pode tanto ser monopolar, na qual são utilizados apenas dois eletrodos - um para o sinal e um para a referência, geralmente localizado longe do eletrodo de sinal, em lugares com pouca atividade bioelétrica -, quanto bipolar, onde são utilizados três eletrodos - dois para o sinal, geralmente em configuração diferencial,

e um para a referência. Na Figura 2.10, é possível visualizar, como exemplo, o sinal mioelétrico proveniente de uma eletromiografia do músculo bíceps enquanto era exercitado com 10 kgf de peso, utilizando eletrodos de superfície em modo bipolar passivo.

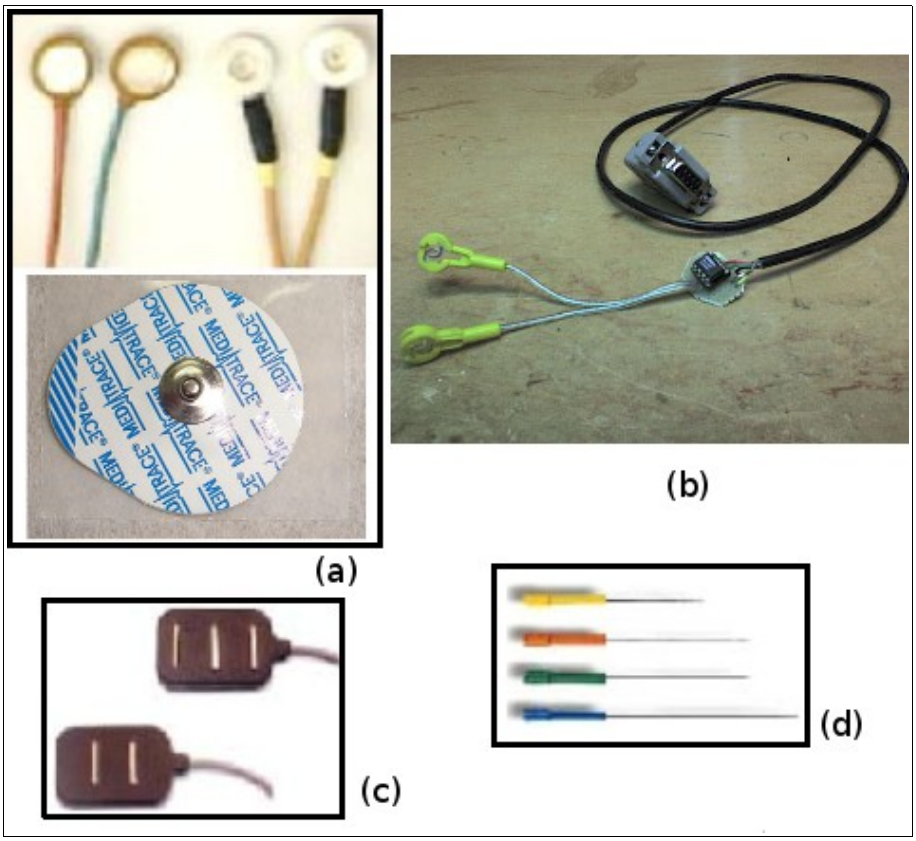



**Figura 2.9 - (a) Eletrodos não Invasivos Passivos e (b) Eletrodos não Invasivos Semi-Ativos e (c) Eletrodos não Invasivos Semi-Ativos e (d) Eletrodos Invasivos.**
**Fonte – Adaptado de FAVIEIRO, 2009.**

**Tabela 2.1 - Sinais Biomédicos**

| Tipo do Sinal | Amplitude | Tipo de Eletrodo | Comentários |
|---|---|---|---|
| Potencial de Ação | 10μV a 100mV | Microeletrodos | Medida invasiva do potencial da membrana da célula |
| Eletroneurograma (ENG) | 5μV a 10mV | Agulha | Potencial do feixe de nervos |
| Eletrorretinograma (ERG) | 0,5μV a 1mV | Microeletrodos | Potencial evocado por flashes |
| Eletro-oculograma (EOG) | 10μV a 5 mV | Superfície | Potencial córneo-retinal |
| Eletroencefalograma (EEG) | | | |
| Superfície | 2 a 100μV | Superfície | Potencial multicanal do escalpo |
| Ondas Delta | 2 a 100μV | Superfície | Crianças jovens, sono profundo e |

| | | | |
|---|---|---|---|
| | | | patologias |
| Ondas Teta | 2 a 100μV | Superfície | Áreas central e temporal durante estado de alerta |
| Ondas Alfa | 2 a 100μV | Superfície | Acordado, relaxado, olhos fechados |
| Ondas Beta | 2 a 100μV | Superfície | |
| Fusos de sono | 50 a 100μV | Superfície | Sequências de 0,2 a 0,6s |
| Complexos-K | 100 a 200μV | Superfície | Sequências durante sono moderado e profundo |
| Potenciais Evocados | 0,1 a 20μV | Superfície | Resposta do potencial do cérebro a estímulos |
| Eletrocorticograma | | Agulha | Gravações de superfícies exposta do cérebro |
| Eletromiografia (EMG) | | | |
| Fibra Única | 1 a 10μV | Agulha | Potenciais de ação de uma única fibra muscular |
| Potencial de Ação da Unidade Motora | 100μV a 2mV | Agulha | |
| EMG de Superfície | | | |
| Músculo Esquelético | 50μV a 5mV | Superfície | |
| Músculo Liso | | Superfície | |
| Eletrocardiograma (ECG) | 1 a 10mV | Superfície | |
| ECG de Alta Frequência | 100μV a 2mV | Superfície | |

**Fonte — Adaptado de BRONZINO, 1995.**

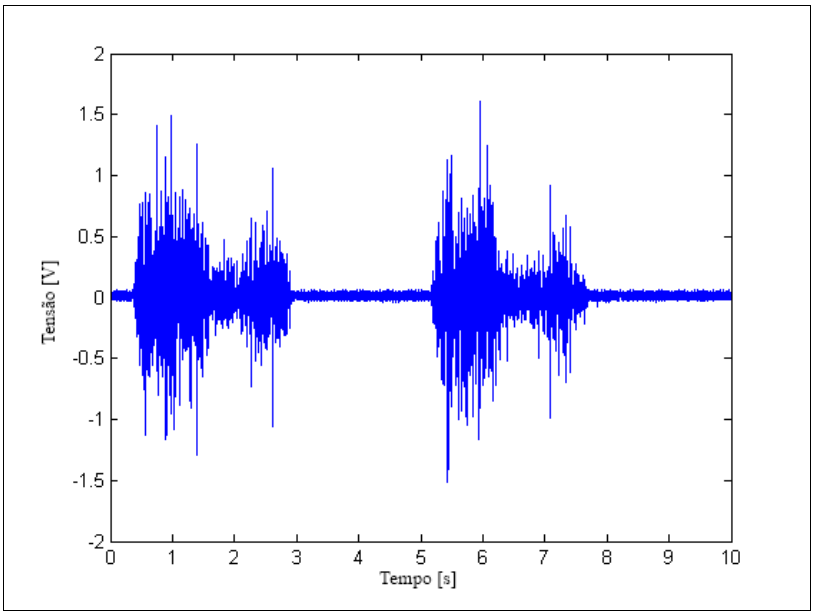



**Figura 2.10 - EMG do bíceps exercitando-se com 10 kgf.**

### 2.2.4 Eletromiografia do sinal da piscada

O movimento da piscada do olho é consequência da atividade de três músculos (dois esqueléticos e um liso) e da disposição mecânica da pálpebra. Os músculos esqueléticos (o sempre ativo elevador da pálpebra e o normalmente em repouso orbicularis oculi), sobre os quais temos controle, provêm a força dominante que atuam sobre a pálpebra, enquanto o músculo de Mueller, sobre o qual, por ser um músculo liso, não temos controle, provê uma força adicional para manter as pálpebras abertas. Finalmente, o arranjo dos ligamentos e dos pontos de inserção dos músculos esqueléticos produzem uma força constante com o intuito de fechar as pálpebras. Para o movimento de piscar, portanto, o sistema nervoso desliga o elevador da pálpebra, enquanto o orbicularis oculi faz com que a pálpebra abaixe, até que, de forma que pálpebra levante, o segundo tem seu movimento cessado e o primeiro volta a funcionar. (EVINGER *et al*, 1991)

A partir de diversas medições da eletromiografia do movimento da piscada bem controlados, pôde-se perceber, a partir da análise de Fourier, que 90% da energia do sinal reside abaixo de 90Hz. Testes de piscadas por reflexo, voluntárias e espontâneas mostraram padrões bastante semelhantes, mas, principalmente, demonstraram que o músculo orbicularis oculi possui o papel principal no movimento da piscada, uma vez que faz com que a pálpebra abaixe (EVINGER *et al*, 1991).

### 2.2.5 Um breve relato sobre alguns trabalhos relacionados ao uso de EMG no controle de cadeiras de rodas

Trabalhos recentes na área indicam o crescimento no interesse de criar novas interfaces homem-máquina, no sentido de facilitar a vida dos indivíduos. Especificamente relacionado à utilização de biosinais no controle de cadeiras de rodas, grande parte dos estudos concentra-se em utilizar as técnicas de Eletrooculografia (EOG), no sentido dos movimentos dos olhos, como forma de controle (KUO, 2009 e BAREA, 2003), enquanto diversos outros abordam a Eletroencefalografia (EEG) para o controle (REBSAMEN, 2007; PIRES, 2008; e KHARE *et al*, 2010). Apesar de complexos e em nível tecnológico avançado, os experimentos são normalmente realizados em laboratórios e outros ambientes internos controlados, sem demonstrar o desempenho em locais externos, mais

suscetíveis a ruídos. VANDERWERF (2002) demonstrou que devido a diferentes tipos de estímulos, tais como aqueles encontrados em ambientes reais, o movimento da piscada pode diferir tanto em amplitude como em duração.

Ainda que diversos estudos no sentido de caracterizar o movimento da piscada têm sido realizados, a utilização destes, no controle de máquinas, ainda não foi devidamente explorado, o que é visível ante a pequena quantidade de artigos publicados nesta área. De toda forma, alguns esforços têm sido empregados no sentido de criar um robô movido pelos sinais mioelétricos orbiculares (ALONSO, 2009).

Outros estudos no sentido de criar cadeiras de rodas controladas através de biosinais especificamente provenientes do movimento da piscada dos olhos têm sido realizados também no Brasil (BASTOS FILHO, 2006). É possível também encontrar trabalhos que utilizam os músculos extensor e flexor do pescoço (CHOI *et al*, 2006), o movimento das sombrancelhas e olhos (TSUI *et al*, 2007), o músculo esternocleidomastóideo (HAN *et al*, 2003), os músculos tríceps braquial e bíceps braquial (OONISHI *et al*, 2008), variações na pressão do ouvido devido a movimentos da língua (MACE *et al*, 2010), a aspiração e expiração do ar através do nariz (PLOTKIN *et al*, 2010) e movimentos da testa, olhos e mandíbula (WEI *et al*, 2009) para o controle de cadeiras de rodas motorizadas.

Considerando a pouca quantidade de artigos relacionando biosinais ao controle de máquinas em ambientes reais (fora de laboratórios), além da baixa amplitude medida com eletrodos de superfície na aquisições de EOG e EEG, o que encarece enormemente o sistema, decidiu-se utilizar esta nova abordagem do movimento da piscada no controle dos equipamentos. Uma vez que será avaliado um sinal muscular, sua amplitude terá maior significância ante o ruído comumente presente em ambientes não controlados, o que implicará em um hardware mais simples e dados mais confiáveis.

# 3 METODOLOGIA EXPERIMENTAL

Devido à evidente necessidade de adquirir dados com uma boa confiabilidade e analisá-los de forma precisa, decidiu-se desenvolver uma nova plataforma de aquisição de dados, bem como um algoritmo customizado para sua análise. A plataforma de hardware, composta de um banco de filtragem, amplificadores e conversores A/D foi especialmente desenvolvida para dar grande flexibilidade às medidas, a partir da possibilidade de configurar com precisão todos seus parâmetros. Os dados convertidos para a plataforma digital são então analisados por um computador portátil, que tomará decisões a respeito da movimentação de uma cadeira de rodas, de acordo com regras previamente configuradas. Nas seções a seguir será explicado em detalhes o aparato experimental desenvolvido, cujo diagrama de blocos que pode ser analisado na Figura 3.1.

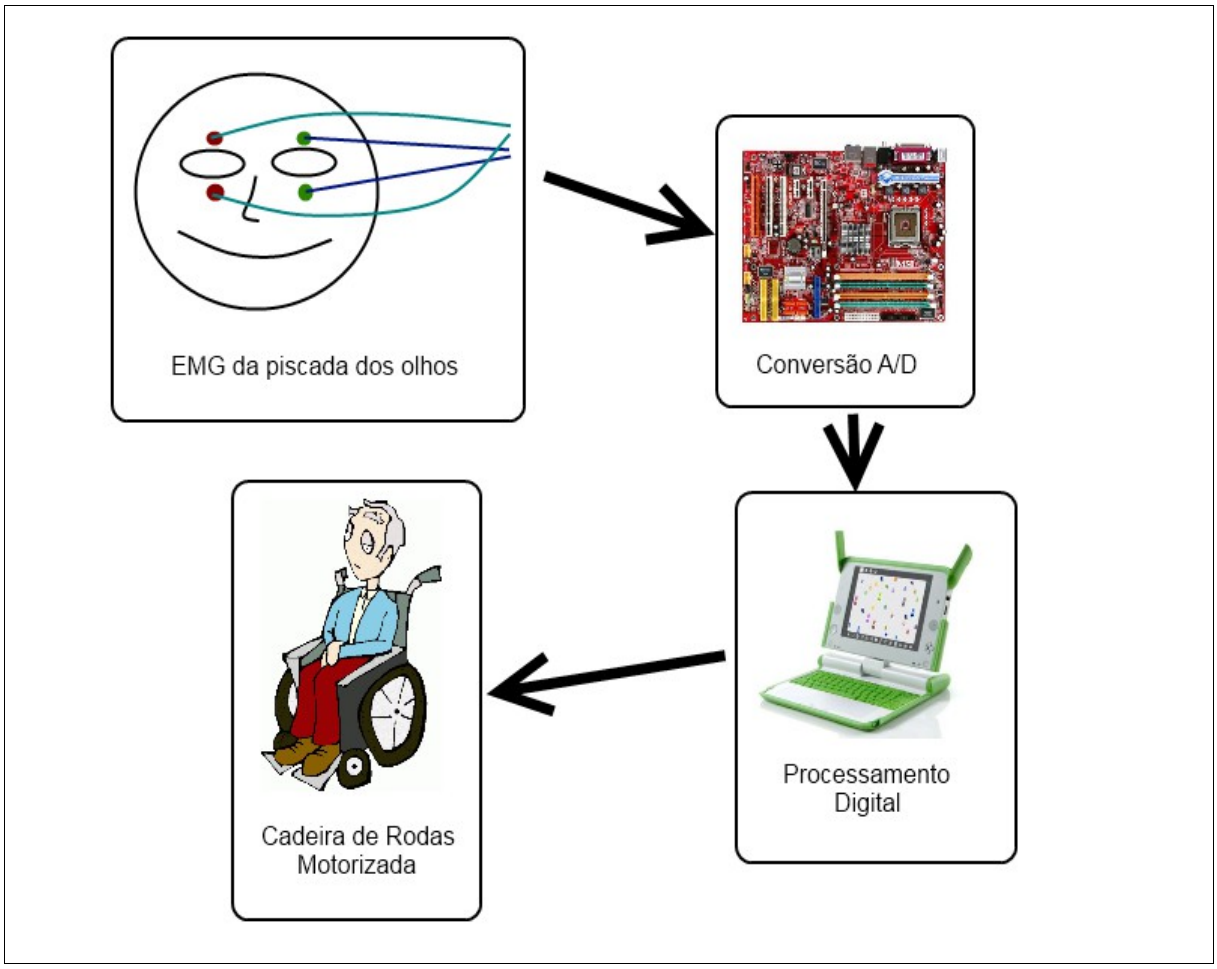



**Figura 3.1 - Diagrama de blocos do sistema proposto.**

## 3.1 Plataforma de Hardware

Desenvolvida com o objetivo de permitir uma máxima flexibilidade quanto as aquisições dos dados provenientes da piscada dos olhos, o hardware do eletromiógrafo pode ser dividido em quatro setores diferentes: captação semi-ativa, filtragem, amplificação e conversão analógico/digital. O objetivo de tal plataforma é manipular analogicamente o sinal dos eletrodos de forma a minimizar o ruído e permitir a captura dos dados com maior precisão possível para que então seja analisada em um sistema embarcado, que será responsável por atuar na movimentação da cadeira de rodas. Na Figura 3.2 é mostrado o diagrama de blocos detalhados da plataforma de aquisição de sinais mioelétricos. Nela, somente é mostrado o diagrama para aquisição de um canal, mas na plataforma desenvolvida, há dois canais em aquisição paralela, um para cada olho. Esse capítulo possui como objetivo o detalhamento do projeto de hardware de cada um dos canais, cujos circuitos serão iguais.

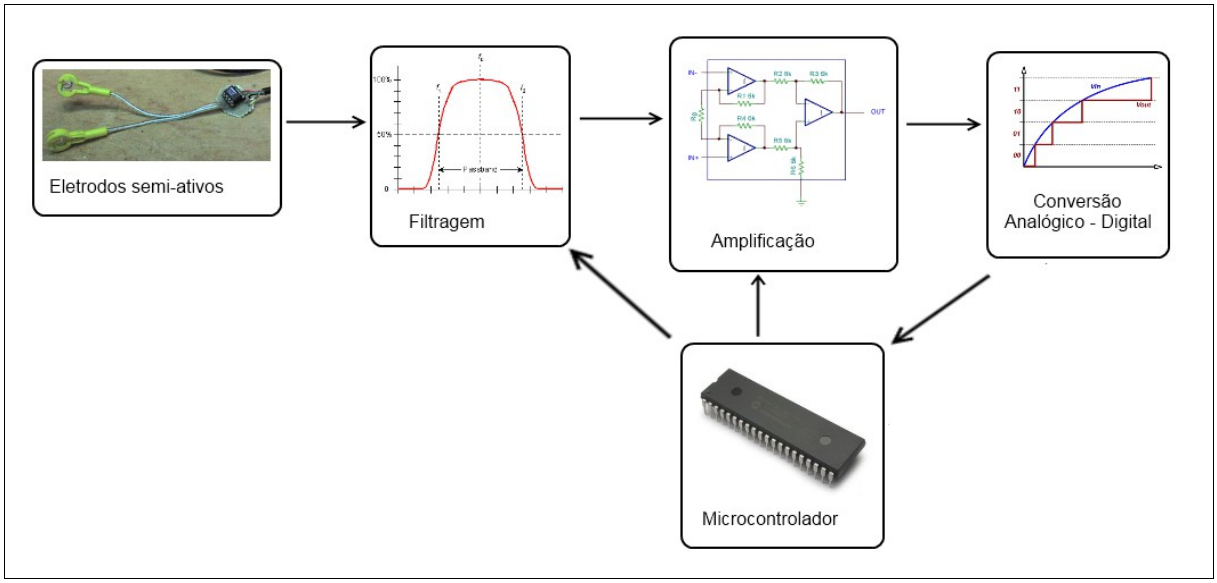



**Figura 3.2 - Diagrama de blocos da plataforma de aquisição de sinais analógicos.**

### 3.1.1 Eletrodos semi-ativos

Considerando-se a baixa amplitude do sinal mioelétrico, compreendido na faixa de poucos micro volts a poucos milivolts, decidiu-se utilizar a técnica de eletrodos semi-ativos, ou seja, a utilização de uma pequena amplificação do sinal o

mais próximo possível dos eletrodos. A amplificação de 10,33 vezes, conforme pode ser verificado na equação 3.1, foi realizada utilizando-se o amplificador de instrumentação INA126 da fabricante Texas Instruments, em topologia diferencial e com referência conectada ao sinal de 2,5 V do circuito e a um eletrodo de referência, de forma que o sinal fique centrado em 2,5 V, conforme pode ser verificado na Figura 3.3.

$$Ganho = 5 + \frac{80\,k\Omega}{R_g} = 5 + \frac{80\,k\Omega}{15\,k\Omega} = 5 + 5{,}33 = 10{,}33 \qquad \text{(Equação 3.1)}$$

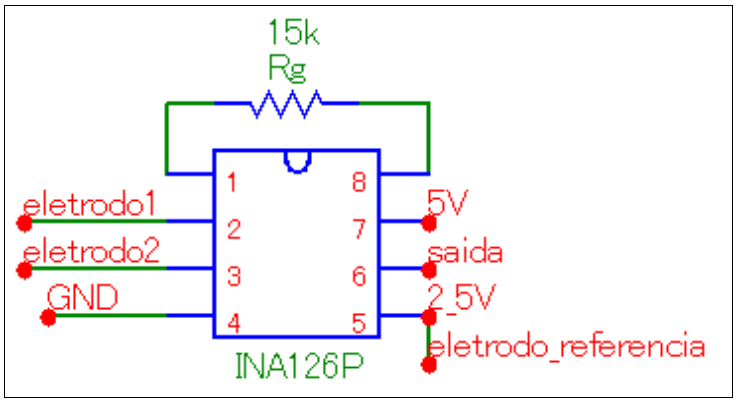



**Figura 3.3 - Esquemático do eletrodo semi-ativo.**

### 3.1.2 Filtragem

O sinal proveniente dos eletrodos ainda contém muito ruído quando é amplificado em seu primeiro estágio, fazendo-se necessário uma série de filtros. Além de aumentar a relação sinal ruído, os filtros desenvolvidos têm a característica de permitir a modificação de suas frequências de corte, dando grande flexibilidade ao circuito, uma vez que é possível utilizar apenas a faixa de frequência de interesse ao experimento.

O primeiro estágio da filtragem é composto por um filtro passa altas de primeira ordem passivo, com frequência de corte fixa em 1,59 Hz e tensão de referência em 5V, cujo objetivo é retirar a componente DC original do sinal e centrá-lo em 5V. A necessidade de evitar valores constantes implica em um maior aproveitamento do *span* do conversor A/D, característica muito desejada em

sistemas nos quais a resolução dos sinais possui grande importância. Após o filtro, há um estágio de *buffer*, de forma a aumentar a impedância de saída do circuito. Na Figura 3.4 é possível visualizar o esquemático do filtro utilizado, enquanto na Figura 3.5 encontra-se o diagrama de Bode do mesmo, criados através do software Micro-Cap.

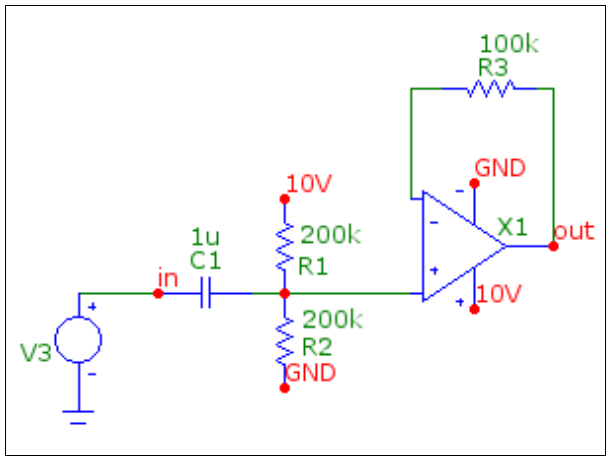



**Figura 3.4 - Filtro passa-altas utilizado.**

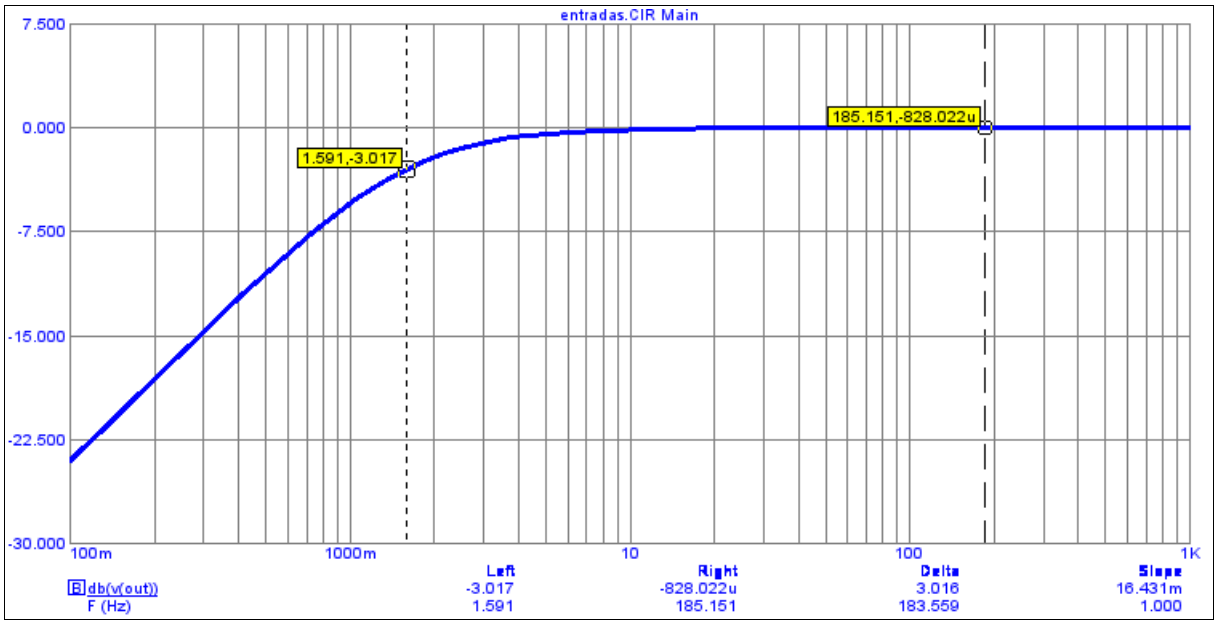



**Figura 3.5 - Diagrama de Bode do primeiro estágio de filtragem.**

Após esse primeiro estágio do circuito, foi implementado um filtro adaptativo. Esse segundo estágio de filtragem, responsável por evitar os ruídos em alta frequência tem por mais significativa característica a possibilidade de ter sua

frequência de corte configurada, uma vez que será utilizado um circuito integrado com capacitores chaveados.

O CI MF10, ou sua versão melhorada LMF100, fabricados pela empresa National Semiconductor, possui encapsulados dois filtros ativos de capacitores chaveados, cuja frequência é controlada externamente através de um sinal digital cuja frequência deve ser cem vezes maior do que a frequência de corte desejada. Isso permite a criação de qualquer tipo de filtro, seja passa-altas, passa-baixas, passa-faixa, rejeita-faixa ou ainda um passa-tudo, possibilitando ainda modificar o fator de qualidade e ganho do filtro através de poucos resistores. Na figura 3.6, é apontada a topologia utilizada, criada com o software KiCad, onde “FREQ” é o sinal de controle da frequência de corte gerado pelo microcontrolador, que será descrito na seção 3.1.5.

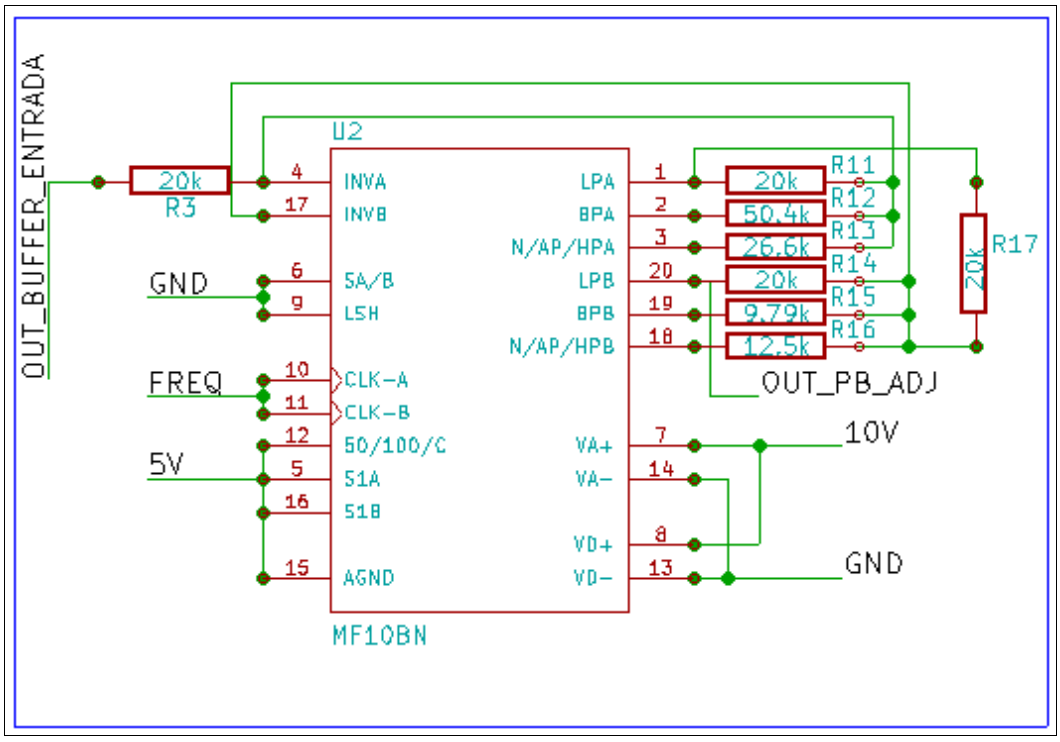



**Figura 3.6 - Esquemático do filtro de capacitores chaveados passa baixas Chebychev de quarta ordem.**

Para este projeto, decidiu-se por utilizar os dois filtros em topologia passa-baixas, resultando em uma topologia Chebychev de quarta ordem com ondulação máxima em 0,1 dB e ganho unitário em DC. Para tal especificação, o fator de qualidade do primeiro estágio, $Q_A$, deve ser 2,183 e o modificador da frequência

deve ser de 1,153 ( $f_{0A}=f_{corte}\times 1{,}153$ ); já para o segundo estágio de filtragem, o fator de qualidade, $Q_B$, deve ser de 0,619 e o modificador da frequência deve ser de 0,789 ( $f_{0B}=f_{corte}\times 0{,}789$ ). Isso significará que a frequência de corte do primeiro filtro será 1,153 vezes a desejada, enquanto a do segundo filtro será 0,789 vezes a desejada, o que não seria muito interessante, uma vez que seria necessário gerar duas frequências diferentes (NATIONAL SEMICONDUCTOR, 2005).

De forma que fosse possível utilizar esta topologia gerando apenas uma frequência, utilizou-se os filtros em modo 3 (NATIONAL SEMICONDUCTOR, 1999), o qual permite mudar a frequência de corte com apenas dois resistores. Desta forma, partindo-se de um resistor de entrada do primeiro estágio ($R_{1A}$) de 20 kΩ, foram necessários os resistores $R_{4A}$ igual ao $R_{1A}$, pois o ganho em DC $H_{OLPA}$ é unitário, $R_{2A}$ de 26.6 kΩ (Equação 3.2) e $R_{3A}$ de 5.04 kΩ (Equação 3.3). Já para o segundo estágio, partindo-se também de um resistor de entrada ($R_{1B}$) igual ao do primeiro estágio, foram necessários os resistores $R_{4B}$ igual ao $R_{1B,}$ $R_{2B}$ de 12.5 kΩ (Equação 3.4) e $R_{3B}$ de 9.79 kΩ (Equação 3.5).

$$R_{2A}=R_{4A}\times\frac{\left(f_{0A}\right)^2}{\left(\frac{f_{CLK}}{100}\right)}$$

$$\Rightarrow 2\times 10^4\times\left(\frac{f_{corte}\times 1{,}153}{f_{corte}}\right)^2$$

$$\Rightarrow 2\times 10^4\times\left(1{,}153\right)^2$$

$$\Rightarrow 2{,}66\times 10^4\ \sim\ 26{,}1\,k\,\Omega \qquad \text{(Equação 3.2)}$$

$$R_{3A}=Q_A\times\sqrt{R_{2A}\times R_{4A}}$$

$$\Rightarrow 2{,}183\times\sqrt{2{,}66\times 10^4\times 2\times 10^4}$$

$$\Rightarrow 5{,}04\times 10^4\ \sim\ 51{,}1\,k\,\Omega \qquad \text{(Equação 3.3)}$$

$$R_{2B} = R_{4B} \times \frac{(f_{0B})^2}{\left(\frac{f_{CLK}}{100}\right)}$$

$$\Rightarrow 2 \times 10^4 \times \left(\frac{f_{corte} \times 0{,}789}{f_{corte}}\right)^2$$

$$\Rightarrow 2 \times 10^4 \times (0{,}789)^2$$

$$\Rightarrow 1{,}25 \times 10^4 \sim 1{,}27\, k\,\Omega \qquad \text{(Equação 3.4)}$$

$$R_{3B} = Q_B \times \sqrt{R_{2B} \times R_{4B}}$$

$$\Rightarrow 0{,}619 \times \sqrt{1{,}25 \times 10^4 \times 2 \times 10^4}$$

$$\Rightarrow 9{,}79 \times 10^3 \sim 2 \times 4{,}87\, k\,\Omega \qquad \text{(Equação 3.5)}$$

Apesar de ter sido projetado para funcionar como um filtro passa baixas, o CI MF10, por seu princípio de funcionamento insere algumas componentes em alta frequência, mais especificamente, em frequências maiores do que a metade daquela utilizada para controle (sinal “FREQ” da Figura 3.6). Este efeito é causado pelo fato do CI ser um filtro de dados amostrados, de forma que a amplitude do sinal de saída muda a cada período de amostragem (período do sinal FREQ da Figura 3.6), e pode ser visualizado na Figura 3.7.

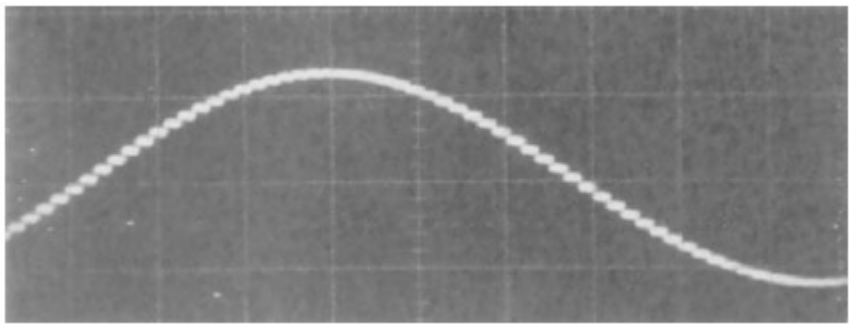


**Figura 3.7 - Efeito de serrilhamento da saída do CI MF10.**
**Fonte – Adaptado de NATIONAL SEMICONDUCTOR, 1999.**

Outro problema encontrado no CI MF10 é sua tensão de *offset*, que pode ser significativa, conforme o circuito final escolhido. As Equações 3.6a e 3.6b mostram as fórmulas utilizadas para o cálculo deste efeito em ambos os filtros encapsulados

no CI, onde $V_{OS1}$ é o *offset* dos amplificadores operacionais internos, de cerca de ±5 mV; $R_{PA}$ é dado pelo paralelo entre os resistores $R_1$, $R_2$ e $R_3$; $R_{PA}$ é dado pelo paralelo entre os resistores $R_5$, $R_6$ e $R_7$; $V_{OS2}$ tem o valor de -300 mV, quando utilizada a razão 100:1 do sinal "FREQ" da Figura 3.6 e a frequência de corte do filtro; e $V_{OS3}$ tem o valor -140 mV, quando a razão 100:1 é utilizada (NATIONAL SEMICONDUCTOR, 1999). Os resistores $R_1$, $R_2$, $R_3$, $R_4$, $R_5$, $R_6$, $R_7$ e $R_8$ estão representados na Figura 3.6 pelos resistores $R_3$, $R_{13}$, $R_{12}$, $R_{11}$, $R_{17}$, $R_{16}$, $R_{15}$ e $R_{14}$, respectivamente. As Equações 3.7a e 3.7b mostram os cálculos das resistências $R_{PA}$ e $R_{PB}$, e a equação 3.8, o cálculo da tensão de *offset* resultante para o filtro completo.

$$V_{OS}(FILTRO1)=V_{OS1}\times\left[1+\frac{R_4}{R_{PA}}\right]-V_{OS2}\times\left(\frac{R_4}{R_2}\right)-V_{OS3}\times\left(\frac{R_4}{R_3}\right) \quad \text{(Equação 3.6a)}$$

$$V_{OS}(FILTRO2)=V_{OS1}\times\left[1+\frac{R_8}{R_{PB}}\right]-V_{OS2}\times\left(\frac{R_8}{R_6}\right)-V_{OS3}\times\left(\frac{R_8}{R_7}\right) \quad \text{(Equação 3.6b)}$$

$$\begin{aligned} R_{PA} &= R_1 \| R_2 \| R_3 = 20k \| 26{,}6k \| 50{,}4k \\ &\Rightarrow \left(\frac{1}{20k}+\frac{1}{26{,}6k}+\frac{1}{50{,}4k}\right)^{-1} \\ &\Rightarrow \frac{20k\times 26{,}6k\times 50{,}4k}{(26{,}6k)\times(50{,}4k)+(20k)\times(50{,}4k)+(20k)\times(26{,}6k)} \\ &\Rightarrow \frac{26812{,}8G}{1340{,}64M+1008M+532M}=5{,}128k\,\Omega \end{aligned} \quad \text{(Equação 3.7a)}$$

$$\begin{aligned} R_{PB} &= R_5 \| R_6 \| R_7 = 20k \| 12{,}5k \| 9{,}79k \\ &\Rightarrow \left(\frac{1}{20k}+\frac{1}{12{,}5k}+\frac{1}{9{,}79k}\right)^{-1} \\ &\Rightarrow \frac{20k\times 12{,}5k\times 9{,}79k}{(12{,}5k)\times(9{,}79k)+(20k)\times(9{,}79k)+(20k)\times(12{,}5k)} \\ &\Rightarrow \frac{2447{,}5G}{122{,}375M+195{,}8M+250M}=3{,}204k\,\Omega \end{aligned} \quad \text{(Equação 3.7b)}$$

$$V_{OS}=V_{OS}(FILTRO1)+V_{OS}(FILTRO1) \quad \text{(Equação 3.8)}$$

$$\Rightarrow(\pm 10)\times\left[2+\frac{R_4}{R_{PA}}+\frac{R_8}{R_{PB}}\right]+600\times\left[\frac{R_4}{R_2}+\frac{R_8}{R_6}\right]+280\times\left[\frac{R_4}{R_3}+\frac{R_8}{R_7}\right]$$

$$\Rightarrow(\pm 10)\times[2+3{,}9+6{,}24]+600\times[0{,}75+1{,}6]+280\times[0{,}4+2{,}04]$$

$$\Rightarrow(\pm 121{,}4)+1410+683{,}2 \;=\; 2093{,}2\pm 121{,}4\,mV$$

Devido ao efeito do serrilhamento do sinal de saída do filtro, optou-se por inserir um último estágio de filtragem, composto de um filtro passa baixas ativo de segunda ordem do tipo Butterworth, em topologia MFB Single-Ended e frequência de corte fixa em 500 Hz e ganho de 6 dB. O esquemático do filtro passa baixas pode ser visualizado na Figura 3.8 e seu diagrama de Bode na Figura 3.9, criados através do software Micro-Cap.

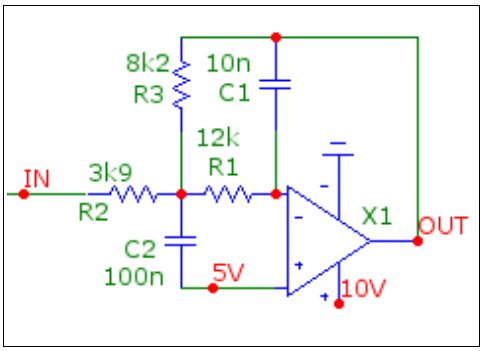



**Figura 3.8 - Esquemático do filtro passa baixas com frequência de corte fixa.**

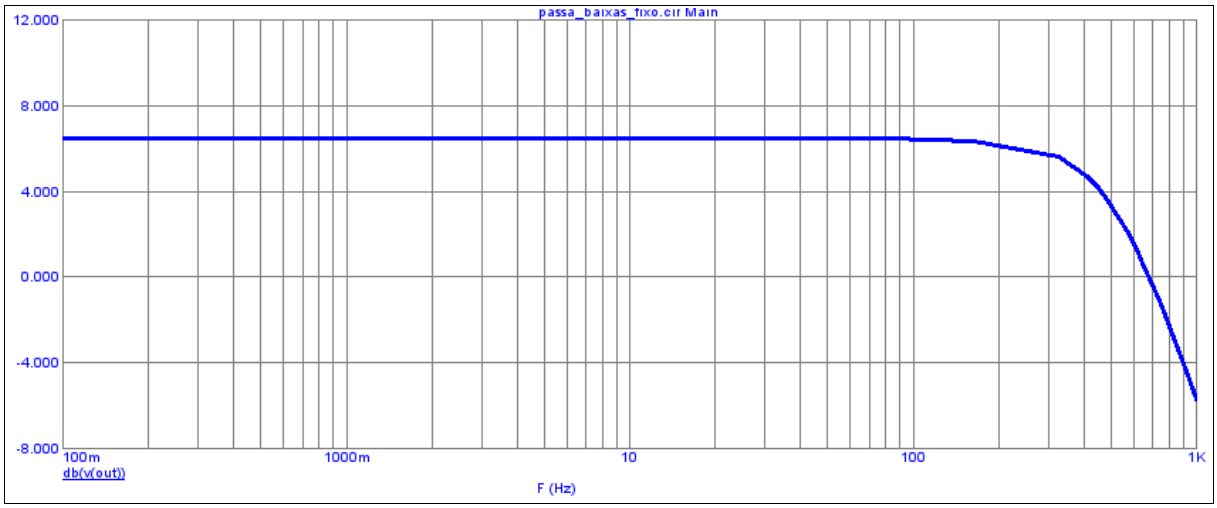



**Figura 3.9 - Diagrama de Bode do filtro passa baixas com frequência de corte fixa.**

De forma a evitar o efeito da tensão de *offset*, foi inserido um filtro passa altas passivo com frequência de corte em 1,59 Hz e referência em 2,5 V, seguido de um

amplificador não inversor com ganho unitário, cujo objetivo é centrar o sinal em 2,5 V, de forma semelhante à que foi utilizado no primeiro estágio de filtragem do circuito. Neste estágio, porém a tensão de referência e menor devido à limitação do multiplexador analógico utilizado na amplificação do sinal, que será apresentada no capítulo 3.1.3. O esquemático e o diagrama de Bode do filtro podem ser visualizados nas Figuras 3.10 e 3.11, respectivamente.

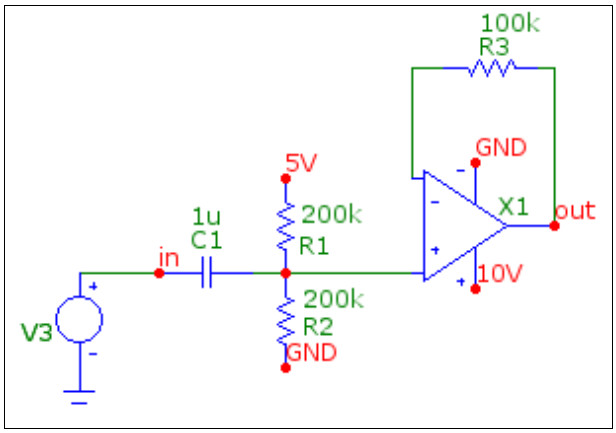



**Figura 3.10 - Filtro passa altas com frequência de corte em 1,59 Hz e tensão de referência em 2,5 V.**

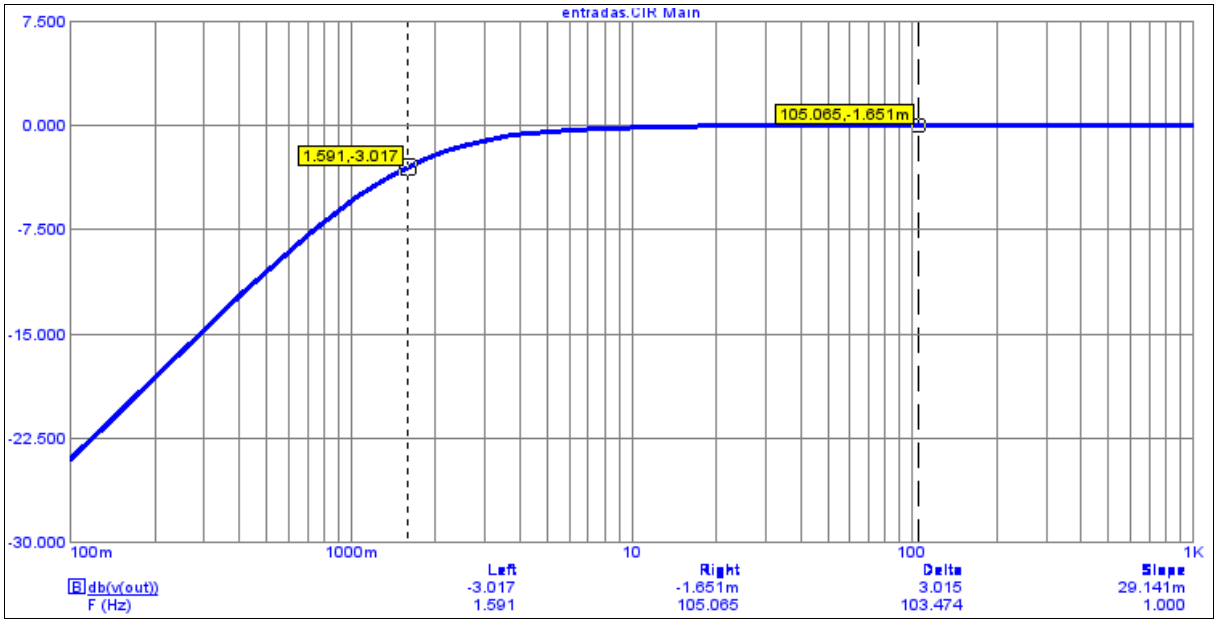



**Figura 3.11 - Diagrama de Bode do filtro passa altas utilizado.**

### 3.1.3 Amplificação

O bloco referente à amplificação de saída do sistema tem por objetivo permitir o máximo aproveitamento do *span* do conversor A/D, regulando a amplitude do sinal filtrado. Inicialmente, imaginou-se utilizar o CI PGA204, pois possui ganho programável, porém, como os ganhos possíveis são bastante limitados (apenas ganhos de 1, 10, 100 ou 1000), optou-se por utilizar o INA129, da fabricante Texas Instruments, um amplificador de instrumentação com ganho regulável de 1 a 10000 através de apenas um resistor externo.

Para permitir o ajuste dinâmico do ganho, decidiu-se por utilizar um multiplexador analógico, porém, por ser ser responsável pela escolha do resistor de ganho do circuito, é necessário que possua uma resistência de estado ligado inferior ao menor resistor necessário para o maior ganho especificado. Uma vez que o maior ganho desejado é de 1000 V/V, é necessária a utilização de um resistor de 49.9 Ω, implicando na utilização de um multiplexador com $R_{on}$ menor ou igual a este. Correspondendo a esta especificação, foi escolhido o CI MAX4617, da fabricante Maxim Integrated Products, cujo $R_{on}$ típico com alimentação única de 5 volts é de 8 Ω. A utilização deste multiplexador em específico permite a configuração de até 9 ganhos, onde 8 são escolhidos de acordo com os resistores e o ganho unitário é alcançado desligando-se o CI, através de seu pino de controle.

A escolha dos ganhos programáveis foi realizada levando-se em conta a baixa amplitude do sinal mioelétrico, de forma que os valores ideais foram calculados e, levando-se em conta os 8 Ω do multiplexador já presentes, os valores comerciais mais próximos foram escolhidos. Na Tabela 3.1, pode-se visualizar os ganhos utilizados, bem como os valores ideais calculados, os resistores escolhidos dentro da série E48 e o ganho real utilizando os valores comerciais. Já na figura 3.12, pode-se visualizar o esquemático do estágio de amplificação utilizado na plataforma de hardware desenvolvida, criado através do software KiCad, onde o sinal “OUTPUT_BUFFER” é o sinal de entrada do circuito de amplificação. Este nome diz respeito ao amplificador não inversor de ganho unitário da entrada do módulo de amplificação. Já os sinais “MUX_SEL0”, “MUX_SEL1”, “MUX_SEL2” e “MUX_SEL3” são os sinais de controle de ganho, cuja origem é o microcontrolador descrito no capítulo 3.1.5.

**Tabela 3.1 - Valores de ganho escolhidos e as resistências correspondentes.**

| Ganho Ideal | Resistência Ideal (Ω) | Resistência Real (Ω) | Ganho Real | Ganho Real em dB |
|---|---|---|---|---|
| 5 | 12350 | 12100 | 5,08 | 14,1 |

| | | | | |
|---|---|---|---|---|
| 10 | 5489 | 5360 | 10,2 | 20,2 |
| 50 | 1008 | 1000 | 50,01 | 34,0 |
| 100 | 499 | 487 | 100,8 | 40,1 |
| 250 | 198 | 187 | 254,33 | 48,1 |
| 500 | 99 | 90,9 | 500,49 | 54,0 |
| 750 | 66 | 59 | 738,31 | 57,4 |
| 1000 | 49,4 | 42,2 | 985,06 | 59,9 |

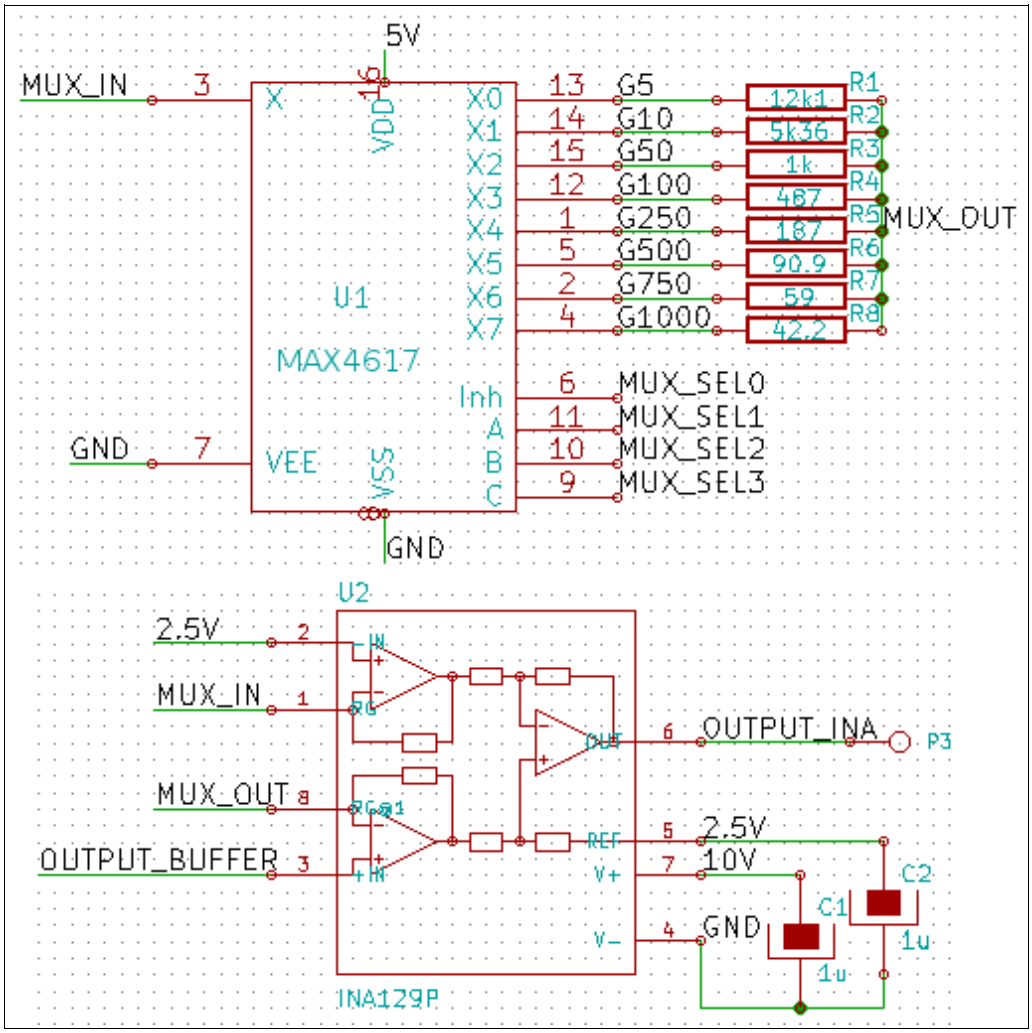



**Figura 3.12 - Esquemático do estágio de amplificação com ganho programável.**

Após o circuito de amplificação, foi inserido um último estágio de manipulação do sinal, composto de um filtro passa altas passivo com frequência de corte em 1,59 Hz e tensão de referência de 2,5 V, seguido de um *buffer* de tensão. Este estágio é semelhante ao utilizado na saída do estágio de filtragem, como descrito no capítulo 3.1.2, e tem por objetivo evitar que quaisquer valores DC que tenham sido amplificados pelo CI INA129 acabem por interferir na resolução do conversor AD.

### 3.1.4 Conversão analógico para digital

Visando em um primeiro momento minimizar a complexidade, mantendo a confiabilidade do sistema, optou-se por não inserir a conversão dos sinais analógicos para a plataforma digital já no hardware projetado. Ao invés, utilizou-se a placa comercial USB-6008, da fabricante National Instruments, que possui uma interface bastante simplificada com o software LABVIEW, da mesma fabricante, através da conexão USB. Essa placa de conversão A/D possui resolução de 12 bits, com até 10 mil amostras por segundo e capacidade de aquisição de até 8 canais e pode ser vista na Figura 3.13. Considerando a frequência máxima do sinal proveniente da plataforma de hardware, configurada para não ultrapassar os 500 Hz, tal taxa de amostragem é mais do que suficiente para a aplicação onde será utilizada.


**Figura 3.13 - Placa USB-6008 da National Instruments.**

Ao utilizar uma plataforma comercial, faz-se necessária a utilização de um computador para coletar os dados, tratá-los e então efetuar o controle da cadeira de rodas. Uma vez que um dos objetivos desta plataforma de hardware é ser portátil, permitindo ao usuário movimentar-se com maior liberdade, a necessidade de ter um computador conectado a ela contradiz tal premissa. Desta forma, preferiu-se utilizar

um conversor analógico para digital na própria plataforma, composto pelo CI MCP3304, da fabricante Microchip. Este CI possui resolução de 13 bits, com até 100 mil amostras por segundo distribuídas em até 8 canais de aquisição. Sua interface de comunicação é serial do tipo SPI, permitindo ao microcontrolador utilizado adquirir dados com certa facilidade.

O diagrama de blocos funcionais do circuito integrado pode ser visualizado na Figura 3.14 e, na Tabela 3.2, é possível verificar a resolução aproximada do sinal dos eletrodos, para os diversos ganhos do estágio de amplificação, considerando que o número de bits do conversor analógico digital efetivamente utilizados é 12, conforme será explicado no Capítulo 3.2, e que os ganhos fixos do eletrodo semi ativo e do estágio de filtragem são 20 e 6 dB, respectivamente.

**Tabela 3.2 - Resolução do sinal de entrada do sistema para os diversos ganhos do estágio de amplificação.**

| Ganho do estágio de amplificação (dB) | Ganho Total (dB) | Resolução (nV) |
|---|---|---|
| 0,0 | 26,0 | 30517,6 |
| 14,1 | 40,1 | 6007,4 |
| 20,2 | 46,2 | 2991,9 |
| 34,0 | 60,0 | 610,2 |
| 40,1 | 66,1 | 302,8 |
| 48,1 | 74,1 | 120,0 |
| 54,0 | 80,0 | 61,0 |
| 57,4 | 83,4 | 41,3 |
| 59,9 | 85,9 | 31,0 |

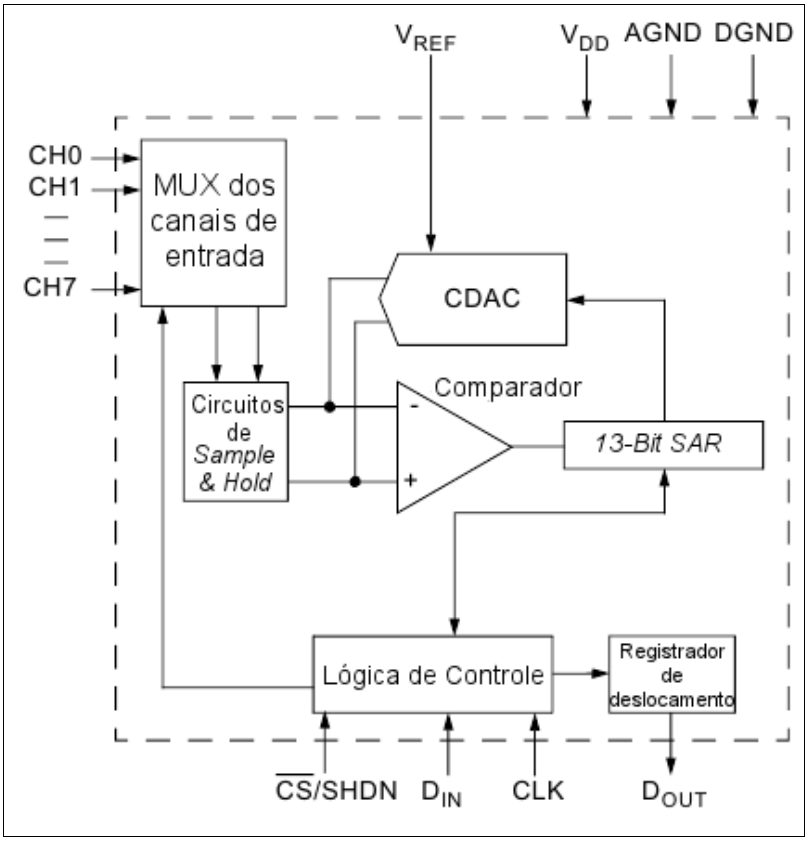



**Figura 3.14 - Diagrama de blocos funcionais do CI MCP3304.**
**Fonte – MICROCHIP, 2008.**

### 3.1.5 Flexibilidade

A plataforma de hardware foi projetada com o objetivo de ser tão flexível quanto possível, permitindo a escolha da frequência de corte do filtro passa baixas e o ganho de saída do estágio de amplificação de ambos os canais. Para realizar tal tarefa, decidiu-se utilizar um microcontrolador PIC16F877A, da Microchip Technology, escolhido pela facilidade de encontrá-lo no varejo. Considerando que ambos os canais de aquisição devam possuir sinais com amplitudes e espectros de frequência semelhantes, o usuário somente poderá configurar a amplificação e filtragem de ambos os canais, significando que os circuitos devem funcionar de forma semelhante. Tais configurações estarão disponíveis através da comunicação Serial RS-232C com o computador.

O microcontrolador utilizado possui um núcleo de 8 bits, com a possibilidade de realizar instruções em 200 ns armazenadas em um máximo de 8 mil instruções na memória FLASH. Possui ainda 368 bytes de memória RAM e 256 bytes de memória não volátil EEPROM. Dentre seus periféricos, pode-se destacar os dois

temporizadores de 8 bits e um de 16 bits, comunicações $I^2C$, SPI e Serial RS232, dois módulos de PWM com resolução de 10 bits e até 8 canais de conversores A/D com 10 bits de resolução, além de diversos pinos para propósitos gerais.

A vantagem de utilizar um sistema microcontrolado é a facilidade de modificar a frequência de corte do filtro passa baixas, uma vez que é dada por um centésimo da frequência de clock enviada ao CI de filtros de capacitores chaveados, conforme explicado no Capítulo 3.1.2. Além disso, a utilização do multiplexador analógico para o controle do ganho do estágio de amplificação do sinal exige pelo menos quatro entradas para cada canal. Considerando que são utilizados dois canais de medição e estes devem possuir ganhos e filtragens iguais, são necessários somente quatro pinos de propósito geral para o controle do multiplexador e um módulo de PWM para a geração do sinal de clock.

Uma vez que o microcontrolador será utilizado para o controle da cadeira de rodas, dispensando a necessidade de um computador para a análise dos dados, foi implementado um algoritmo de decisão do controle da cadeira, tendo como entrada os sinais mioelétricos adquiridos através do conversor analógico-digital MCP3304, conforme explicado no Capítulo 3.1.4. A aquisição dos dados é realizada através da comunicação SPI, uma comunicação do tipo mestre-escravo, na qual o microcontrolador faz o papel de mestre, requisitando dados de cada canal, e o conversor analógico-digital faz o papel de escravo, enviando dados requisitados.

A interface com o usuário foi implementada através da porta Serial RS-232C, por onde será possível enviar comandos específicos para a configuração da plataforma de hardware. Na Figura 3.15, é possível visualizar o esquemático do circuito digital utilizado na plataforma, realizado no software KiCad, incluindo o microcontrolador, o CI MAX232, da fabricante Texas Instruments, responsável por elevar o sinal digital da porta serial do PIC para a comunicação com um computador no padrão RS232, e o conversor analógico-digital MCP3304.

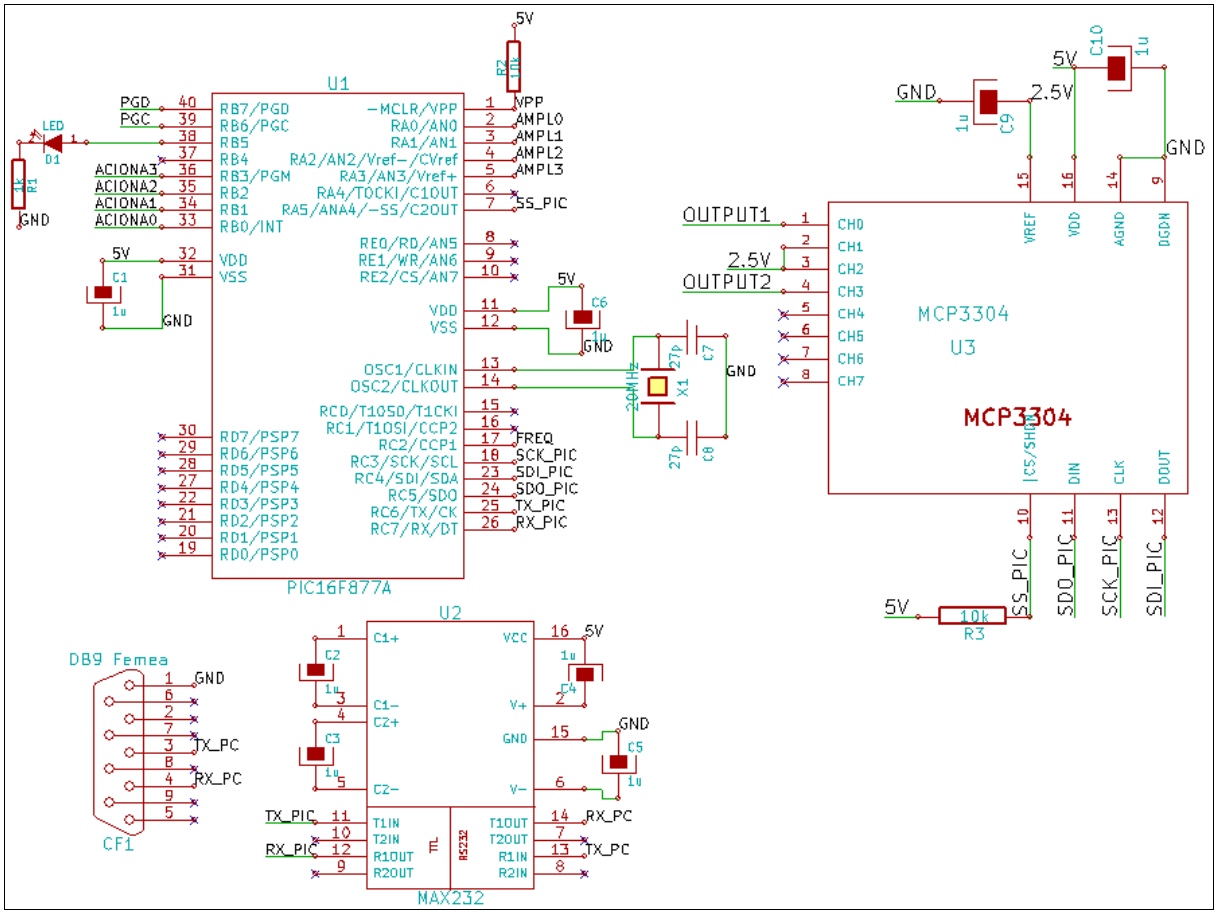



**Figura 3.15 - Esquemático do circuito digital.**

### 3.1.6 Cadeira de Rodas Motorizada

A cadeira de rodas utilizada era, originalmente, uma cadeira comum à qual foram adaptados motores de corrente contínua e um sistema simples de controle alimentado por uma bateria de carro de 12 V. A adaptação da cadeira foi realizada pelo engenheiro eletricista Daniel Fusco em seu trabalho de conclusão de curso, conforme FUSCO, 2010.

Em seu trabalho, dois motores DC sem redução foram fixados aos eixos das rodas e acionados através de um sinal de PWM proveniente de um microcontrolador e um estágio de potência. Este microcontrolador recebe como entrada quatro sinais de nível TTL, que indicam o movimento da cadeira de rodas (para frente, para trás, para a esquerda e para a direita), permitindo uma maior simplicidade no controle da mesma. O controle de velocidade não foi implementado, de forma que não há garantia de que ambas as rodas girarão de forma igual (FUSCO, 2010).

Devido à falta da redução nos motores, a cadeira, da forma como foi projetada, não obteve bons resultados, uma vez que os motores utilizados não

possuem torque suficiente. Dessa forma, foi necessário trocar os motores originais por dois motores da fabricante Imobras, modelo 04.012.12. Esses motores possuem redução integrada, que provê um torque nominal de 6 N.m, suficiente para movimentar a cadeira de rodas. Apesar da troca de motores, não foi necessário realizar mudanças no sistema de controle original, que se mostrou robusto o suficiente para os novos motores. Na Figura 3.16(a), pode-se verificar uma foto da cadeira de rodas motorizadas com os novos motores, e, na Figura 3.16(b), uma foto aproximado de um dos motores.

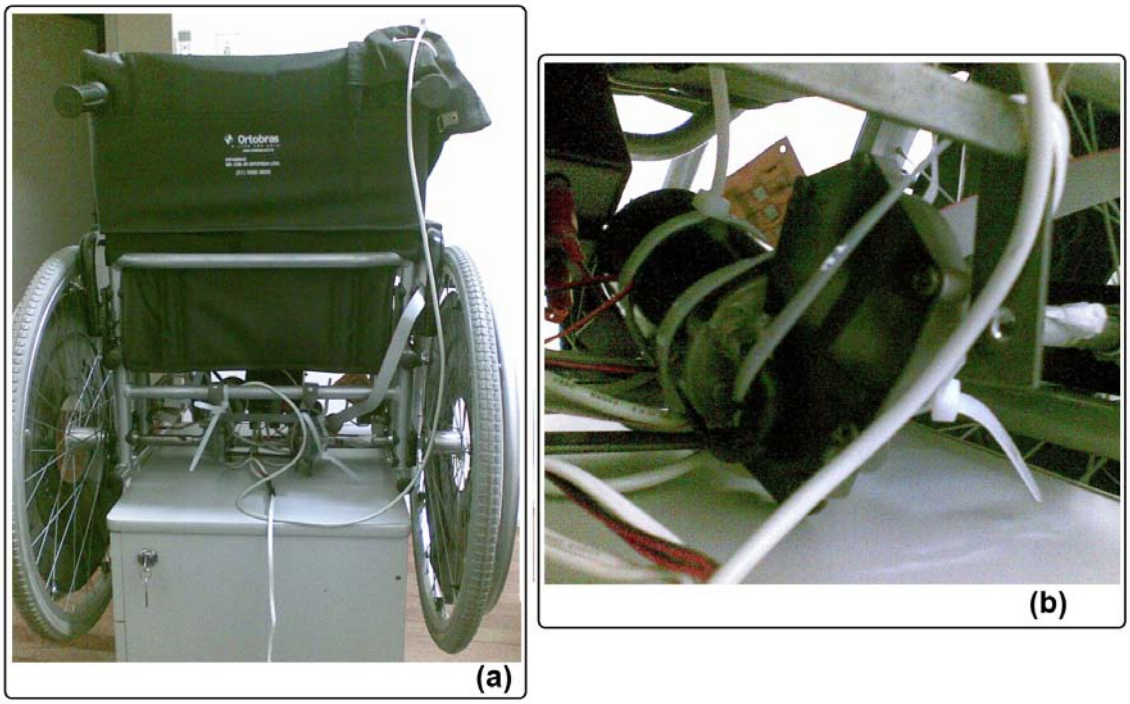



**Figura 3.16 - (a) Cadeira de rodas motorizada e (b) foto aproximada do motor.**

### 3.1.7 Alimentação da plataforma de hardware

Utilizando-se da mesma fonte de energia já presente no sistema de controle da cadeira de rodas motorizada, criou-se um circuito de regulagem das tensões, de forma que se pudesse alimentar a plataforma de hardware. Uma vez que há somente uma bateria de 12 V, não é possível utilizar tensões negativas no circuito, sendo necessário criar um circuito de terra virtual. Considerando que o filtro de capacitores chaveados necessita que sua alimentação seja entre 9 e 14 V e que o multiplexador analógico do estágio de amplificação necessita que sua alimentação seja entre 2 e 5,5 V, decidiu-se derivar da tensão da bateria, as tensões 10 V, 5 V e

2,5V. Para as tensões 10 e 5 V, foram utilizados os componentes LM7810 e LM7805, respectivamente, ambos da fabricante Fairchild Semiconductor, que possuem correntes de saída de até 1 A e regulagem da tensão de saída de ± 2 %. Para a tensão de 2,5V foi utilizado o componente AD780, que pode ser configurado também para uma tensão de saída de 3 V, possui uma corrente máxima de 10 mA e uma regulagem muito eficiente, chegando a 10 μV/V.

## 3.2 Plataforma de Software

Uma vez que o sinal será processado em um microcontrolador da família PIC16, decidiu-se utilizar a linguagem de programação C, uma vez que ferramentas gratuitas estão disponíveis no site da fabricante. A plataforma para o desenvolvimento do software embarcado foi a MPLAB IDE, disponibilizada pela Microchip, a qual incorpora diversas ferramentas, desde a gravação do microcontrolador, até a simulação do código instrução por instrução. Além disse, ainda permite a integração com diversos compiladores, facilitando o desenvolvimento dos softwares (MICROCHIP, 2010).

O compilador utilizado foi o HI-TECH C para PIC10/12/16, da fabricante Hi Tech, disponível gratuitamente, em versão *Lite*, no site da Microchip. Por ser gratuita, essa versão não possui diversas otimizações presentes nas versões Standard ou PRO, como otimizações relativas ao tamanho dos ponteiros das variáveis de acordo com suas utilizações e redução do *overhead* requerido pela troca de contextos causadas por interrupções (MICROCHIP, 2010), porém, não possui limitações que impeçam a criação da plataforma de software.

Para a simulação da integração da plataforma de software com o a plataforma de hardware, foi utilizado o software Proteus VSM, da fabricante Labcenter, e disponível gratuitamente em versão de demonstração no site da empresa. Por ser versão de demonstração, não é possível salvar, imprimir ou projetar os circuitos, mas é possível utilizar as amostras instaladas junto com o software (LABCENTER, 2010), o que é suficiente para a simulação da plataforma de software desenvolvida.

A gravação do microcontrolador foi realizada através de uma versão simplificada da ferramenta PicKit2. De propriedade do autor deste texto, a ferramenta é baseada no Pickit 2, da fabricante Microchip, e permite a gravação do software embarcado sem a necessidade de remover o microcontrolador do circuito.

O software embarcado é responsável pela configuração do ganho do estágio de amplificação, conforme explicado no Capítulo 3.1.3, pela configuração da frequência de corte do estágio de filtragem, conforme explicado no Capítulo 3.1.2, pela aquisição do sinal mioelétrico, conforme explicado no Capítulo 3.1.4, pelo processamento e análise do sinal digital e, finalmente, pelo controle da cadeira de rodas motorizada, de acordo com os sinais adquiridos. Além disso, o microcontrolador realizará a interface com o usuário através da comunicação serial RS-232C, recebendo comandos e exibindo informações em caracteres ASCII. Na Figura 3.17, é possível verificar o fluxograma simplificado da plataforma de software desenvolvida, criado através do software yEd da fabricante yWorks, o qual será explicada em maiores detalhes neste capítulo.

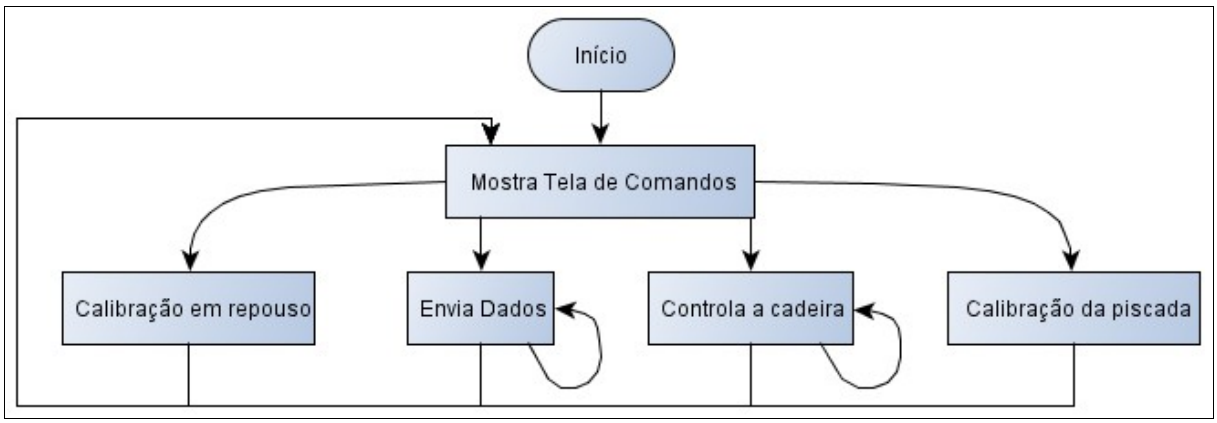



**Figura 3.17 - Fluxograma da plataforma de software.**

### 3.2.1 Interface com o usuário

Uma vez que a interface entre a plataforma de software e o usuário será realizada através de uma conexão serial RS-232C, optou-se por realizar o envio de caracteres ASCII para mostrar informações e receber comandos também através da mesma codificação, de forma que basta um computador com interface serial e um programa que leia dados desta interface, como o *Hyperterminal* do Microsoft Windows ou o Minicom para as distribuições baseadas em Linux, para realizar o controle da plataforma. A velocidade de conexão escolhida foi de 115200 bps, com 8 bits de dados, um bit de parada e sem bits de paridade nem controle de fluxo. Na “tela de comandos”, há explicações sobre os possíveis comandos, bem como a identificação da plataforma de software, conforme pode-se verificar na Figura 3.18,

uma simulação da plataforma realizada através do software Proteus, da fabricante Labcenter.

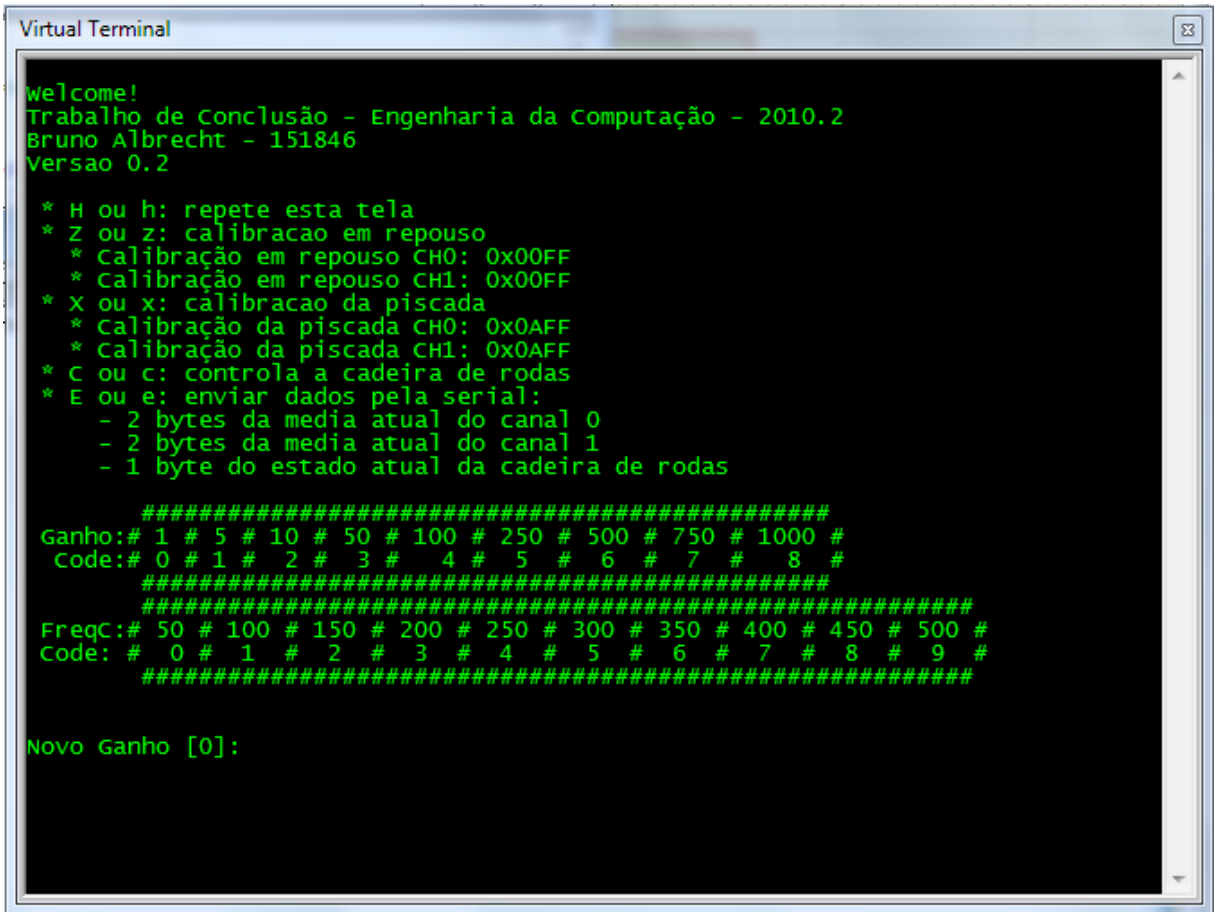



**Figura 3.18 - Simulação da tela de comandos da plataforma de hardware.**

### 3.2.2 Aquisição de dados

A aquisição dos dados mioelétricos dá-se através da comunicação serial do tipo SPI com o conversor analógico-digital, na qual o microcontrolador faz o papel de mestre, enquanto o conversor faz o papel de escravo. A comunicação tem frequência de *clock* de 1250 kHz, os bits são amostrados no final do período de *clock*, a transmissão ocorre nas bordas de subida do *clock*, que possui estado parado em nível de tensão baixo. A aquisição dos dados é realizada para ambos os canais, sendo realizada primeiramente para o canal correspondente ao eletrodo do olho esquerdo. Esses dados são guardados em um *buffer* e outra variável é ativada, de forma a avisar que um dado foi recebido corretamente.

#### 3.2.2.1 Taxa de Amostragem

As frequências do sinal mioelétrico estão compreendidas entre valores próximos do DC até cerca de 500 Hz, apesar da maior parte de sua energia estar compreendida abaixo de 150 Hz (FAVIERO, 2009). Segundo o Teorema de Nyquist, a amostragem de um sinal analógico deve ser realizada em uma taxa no mínimo duas vezes maior do que a maior frequência do sinal adquirido, de forma que possa posteriormente ser reconstruído com a menor perda de informações possível.

De acordo com tais informações, decidiu-se por utilizar para esta aplicação a taxa de amostragem de 1000 Hz, uma vez que é duas vezes maior do que a maior frequência do espectro do sinal mioelétrico. Testes preliminares de medidas do sinal da piscada dos olhos mostraram que esta taxa é ideal ao sistema. O período de um milissegundo contempla, portanto, a rotina de aquisição do sinal, a rotina de tratamento digital do sinal e os algoritmos de detecção de piscadas e controle da cadeira de rodas ou das calibrações. Considerando que o microcontrolador utiliza um cristal de 20 MHz e que as instruções ocorrem a uma taxa 4 vezes menor, portanto com períodos de 200 nanossegundos, o período de amostragem do sinal de um milissegundo corresponde à execução de apenas 5000 instruções.

O período de amostragem foi implementado através de um temporizador de 16 bits interno ao microcontrolador, configurado para gerar uma interrupção a cada milissegundo. A rotina de atendimento dessa interrupção tem por tarefa “avisar” a rotina de aquisição de dados que ela deve ser iniciada.

### 3.2.3 Tratamento digital do sinal

Devido ao baixo poder de processamento do microcontrolador utilizado, o tratamento digital do sinal é reduzido, porém, se mostrou suficientemente robusto para a aplicação. O fluxograma simplificado, criado através do software yEd, do algoritmo de tratamento pode ser visualizado na Figura 3.19.

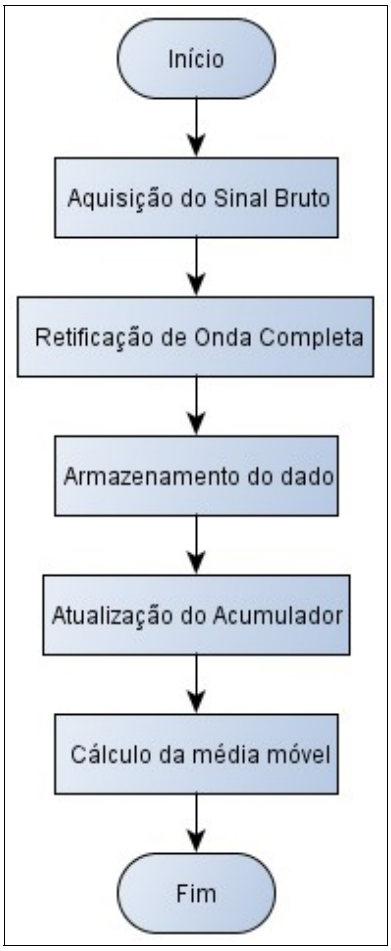



**Figura 3.19 - Fluxograma do tratamento digital dos sinais.**

O sinal digital adquirido do conversor analógico-digital possui 13 bits em codificados no sistema complemento de 2, onde o bit mais significativo indica o sinal do dado e os outros 12 indicam sua magnitude. Quando o número é positivo, o bit de sinal está em zero e a magnitude é um número inteiro simples, porém, quando é negativo, seu bit de sinal está em um e a magnitude está invertida (o maior número indica a menor magnitude). Com isto em vista é necessário converter o valor bruto

para valores positivos, uma vez que somente a magnitude do sinal será utilizada no algoritmo de controle da cadeira de rodas.

O dado digitalizado, já em valores positivos, é então armazenado em um vetor e um acumulador com a soma dos valores guardados no vetor é atualizado, retirando-se o dado mais antigo e inserindo-se o dado novo. A média móvel dos últimos valores é finalmente calculada dividindo-se este acumulador pelo tamanho do vetor.

Uma vez que o microcontrolador possui sua reduzida memória RAM dividida em quatro bancos distintos, três com 96 bytes e um com apenas 80 bytes, não é possível criar vetores de armazenamento de tamanhos elevados. Considerando ainda que os dados provenientes do conversor analógico-digital são armazenados em variáveis de 16 bits, que limitam o tamanho dos vetores em apenas 48 posições, e que não há multiplicadores ou divisores implementados em hardware no PIC16F877A, o que faz com que essas operações tornem-se muito custosas em termos de tempo, decidiu-se por limitar o tamanho em $2^n$, onde n é um número natural positivo. Essa decisão tem por objetivo facilitar o cálculo da média do vetor, que pode ser implementada através de deslocamentos. Devido às limitações da memória RAM, utilizou-se o tamanho de 32 posições, ou 64 bytes, que correspondem a 32 milissegundos de dados.

A primeira versão do algoritmo utilizava um *buffer* simples, no qual quando um dado novo devia ser armazenado, todos os antigos eram deslocados para o início do vetor, o primeiro dado do vetor (o mais antigo) era descartado e o novo era inserido no final do vetor. Além disso, utilizava-se vetores de 45 posições. De acordo com as simulações, a operação de tratamento digital do sinal levava cerca de 900 microssegundos, dos quais cerca de 600 microssegundos eram utilizados somente para o cálculo das médias. Utilizando o *buffer* de 32 posições, esse tempo caiu para 350 microssegundos, dos quais, cerca de 300 microssegundos eram utilizados na atualização dos vetores.

Apesar de suficientemente pequeno, o tempo para atualizar os vetores era excessivo. Dessa forma, decidiu-se implementar um *buffer* circular, no qual uma variável com a posição inicial atual do vetor era guardada. Quando é necessário inserir um dado novo, este é inserido na posição inicial atual e esta é então atualizada para a próxima posição. Desta forma, não é necessário percorrer todo o vetor, diminuindo o número de instruções a serem executadas. Com este algoritmo e

implementando a divisão explicitamente através de deslocamentos, o tempo de execução diminui de 350 microssegundos para apenas 44 microssegundos.

### 3.2.4 Algoritmos de calibração

Devido à grande quantidade de ruídos presente no sistema e às diferenças de amplitudes dos sinais causadas pela má colocação dos eletrodos, os sinais devem ser calibrados. Dessa forma, foram criados dois algoritmos de calibração que devem ser executados antes de se utilizar o sistema.

A calibração em repouso tem por objetivo medir o nível de ruído do sinal, onde piscadas involuntários, que, de acordo com testes preliminares representam amplitudes reduzidas, também são consideradas ruído. O valor da calibração em repouso é único para cada canal de aquisição, uma vez que há diferenças entre os canais, cujas causas variam desde a tolerância dos componentes utilizados até o posicionamento do eletrodos no usuário. Para esta calibração, é pedido ao usuário que mantenha-se em repouso, sem realizar movimentos bruscos. Após dois segundos de dados descartados, realiza-se a aquisição e o tratamento digital do sinal durante outros dois segundos, mas guardando-se o valor de pico da média deste período, que, ao final, será armazenado como valor de calibração.

A calibração das piscadas tem por objetivo medir o valor máximo das piscadas voluntárias do usuário, já que estas podem variar, especialmente devido às diferenças na colocação dos eletrodos. Assim como na calibração em repouso, o valor da calibração das piscadas é único para cada canal, devido às diferenças mecânicas e eletrônicas inerente ao sistema. Para este processo, pede-se que o usuário pisque diversas vezes com ambos os olhos, preferencialmente alternando-os. Após apenas um segundo de dados descartados, realiza-se a aquisição e tratamento digital do sinal durante dois segundos, mas guardando-se novamente o valor de pico da média deste período. Ao final, 25% do valor resultante é descartado, sendo os outro 75% armazenados como valor de calibração. Caso o valor calibrado das piscadas tenha sido menor do que o calculado em repouso, o mesmo é descartado e um erro é mostrado ao usuário, sendo necessário refazer o processo de calibração.

### 3.2.5 Detecção de piscadas

Com o sinal já tratado e as calibrações realizadas, é possível utilizar os dados para detectar piscadas e, então, controlar a cadeira de rodas. A detecção das piscadas levará em conta também o tempo, utilizando como base o próprio período de amostragem do sinal. Na Figura 3.20, pode-se visualizar o fluxograma do algoritmo de detecção, criado através do software yEd.

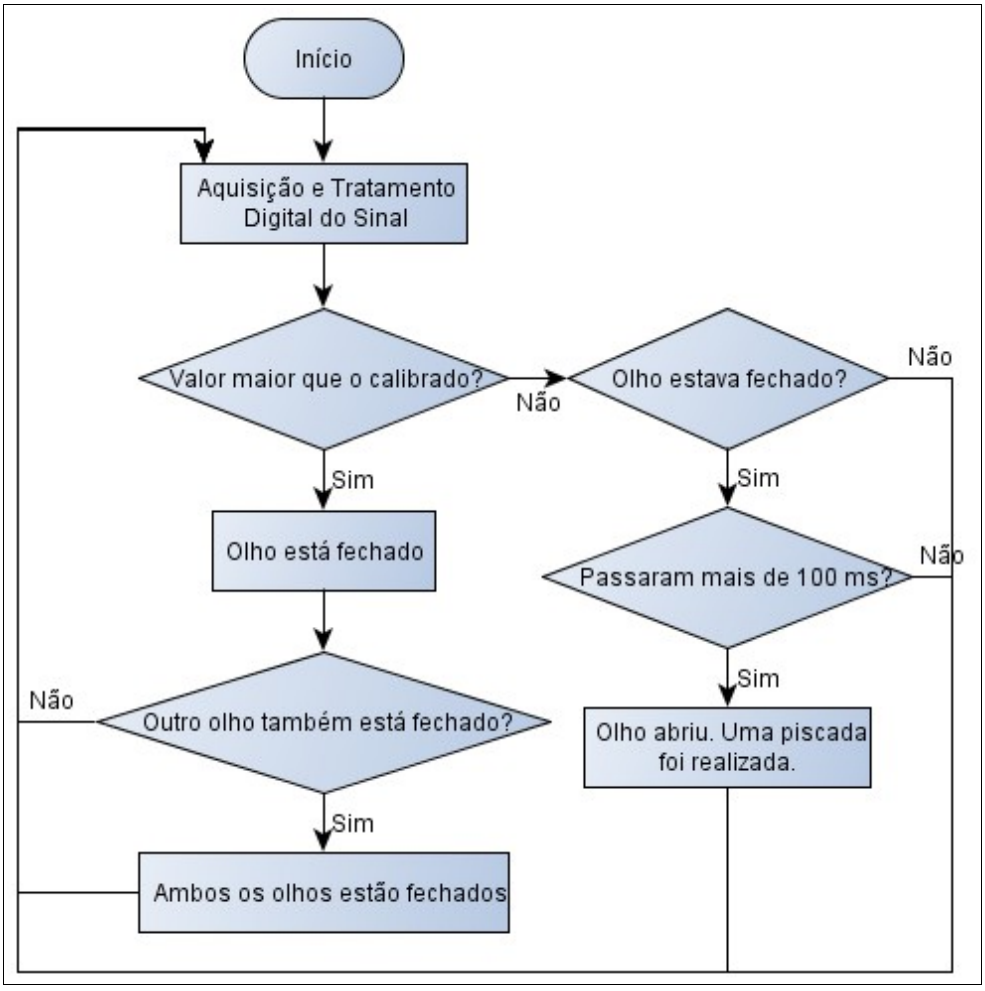



**Figura 3.20 - Fluxograma do algoritmo de detecção de piscadas.**

Para evitar que o algoritmo detecte falsas piscadas, quando os olhos abrem-se por um período muito breve e fecham-se novamente, adicionou-se um tempo de espera de 100 milissegundos até que a piscada seja de fato concluída. Uma vez que o algoritmo de controle da cadeira de rodas baseia-se na quantidade e frequência de piscadas, um temporizador foi adicionado ao algoritmo de detecção de piscadas.

Esse temporizador é zerado ao final da primeira piscada e incrementado a cada aquisição. Além disso, foi implementado um contador de piscadas, que é incrementado ao final de cada piscada e é único para cada olho, de forma que seja possível diferenciar a piscada do olho esquerdo daquela realizada com o olho direito.

### 3.2.6 Controle da cadeira de rodas motorizada

Utilizando-se o número de piscadas de cada olho e um tempo fixo desde o final da primeira piscada, foi implementado um protocolo, segundo o qual, a cadeira será controlada. Primeiramente, definiu-se que quando a cadeira estiver em movimento, somente será possível pará-la, de forma que quaisquer outros comandos serão ignorados. Com a cadeira em repouso, será possível mover-se para frente e para trás e girar sobre o próprio eixo para a esquerda e a direita.

O protocolo foi criado de forma a ser o mais simples possível, uma vez que as habilidades das pessoas diferem quando trata-se de piscar somente um olho. Para mover-se para frente, deve-se piscar duas vezes com ambos os olhos. Para mover-se para trás, é necessário piscar uma vez com cada olho, sem que haja sobreposição, ou seja, primeiro pisca-se com o olho direito e, com os dois olhos abertos novamente, pisca-se com o esquerdo, ou vice-versa. Para movimentar-se para a esquerda, deve-se piscar uma vez com ambos os olhos e uma vez com o olho esquerdo, não sendo levada em conta a sobreposição de piscadas (os dois olhos não precisam piscar ao mesmo tempo). Para movimentar-se para a direita, o protocolo é semelhante ao do movimento para a esquerda, porém, a segunda piscada deve ser realizada com o olho direito. Para parar, pode-se piscar somente uma vez com ambos os olhos ao mesmo tempo ou piscar mais de duas vezes com pelo menos um dos olhos.

O tempo fixo definido desde o final da primeira piscada até a finalização do protocolo foi de dois segundos, o qual, de acordo com os testes realizados previamente, se mostrou suficiente para executar os comandos. O controle da cadeira de rodas foi realizado acionando-se quatro sinais elétricos de nível TTL, conforme foi explicado no Capítulo 3.1.6. Na Figura 3.21, é possível verificar o fluxograma do controle da cadeira de rodas, criado através do software yEd.

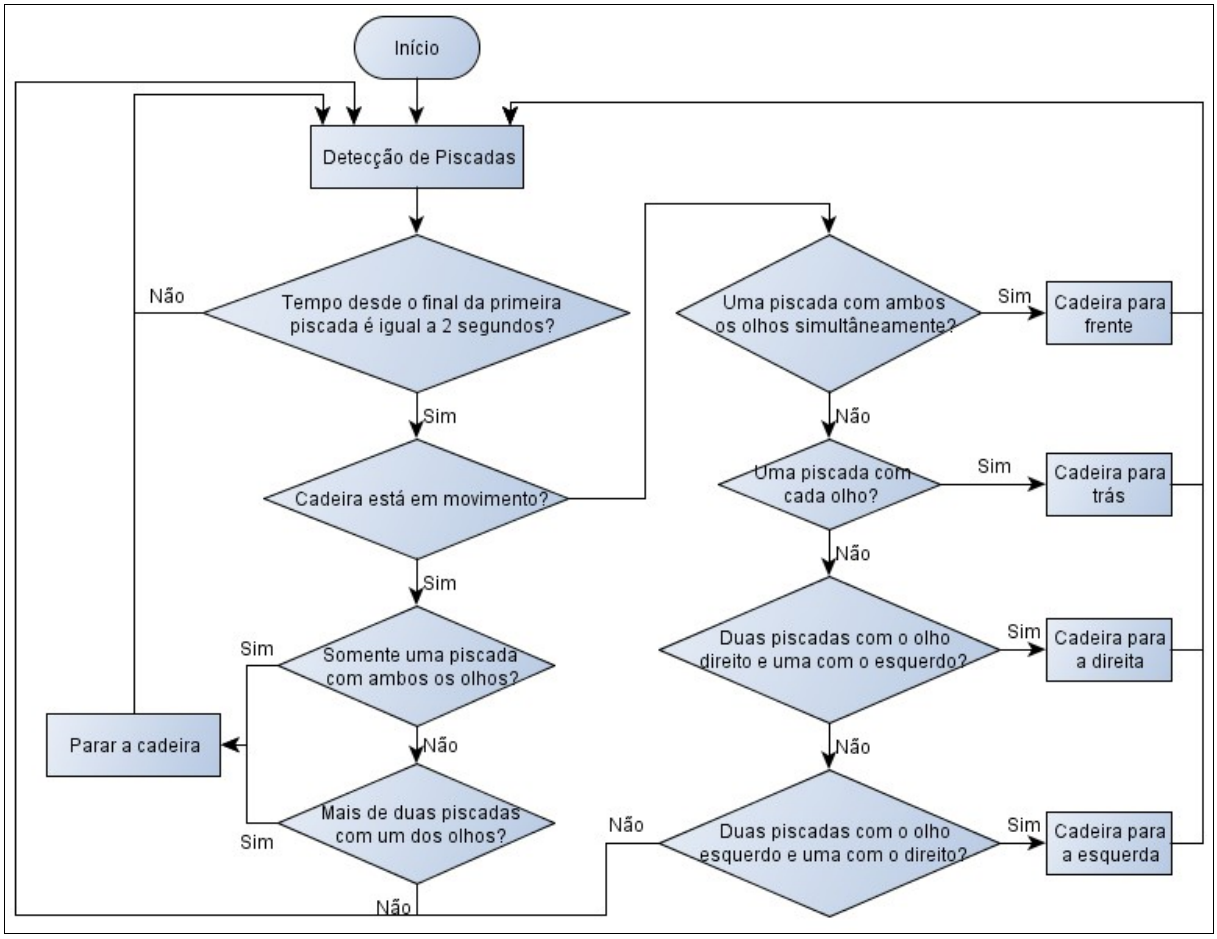



**Figura 3.21 - Fluxograma do controle da cadeira de rodas motorizada.**

### 3.2.7 Envio de dados

O envio dos dados das aquisições tem por propósito guardar um histórico dos comandos realizados pela plataforma de software e é realizado através da conexão serial RS-232C. Os dados são enviados em formato bruto, ou seja, não são convertidos para ASCII, uma vez que o envio deve ser realizado tão rapidamente quanto possível e tal conversão demandaria mais tempo do que é possível utilizar. Os dados são enviados a cada aquisição e estão na seguinte ordem: dois bytes indicando a média atual do sinal do olho esquerdo, dois bytes indicando a média atual do sinal do olho direito e um byte com o código referente ao comando atual da cadeira de rodas.

Considerando que, devido às limitações do conversor analógico-digital, as médias são números de no máximo 12 bits, há quatro bits sem uso nos dois bytes enviados. Dessa forma, implementou-se um sistema para identificar o byte, já que

há o risco de perda de dados na comunicação, conforme pode-se verificar na Tabela 3.3. Na Tabela 3.4, é possível verificar os códigos utilizados para identificar o comando atual da cadeira de rodas.

**Tabela 3.3 - Codificação de cada byte das médias enviadas pela comunicação Serial RS-232C, onde o bit 7 corresponde ao mais significativo.**

| Bit | Significado |
|---|---|
| 7 | Identificação do canal de aquisição (esquerda – 0 – e direita – 1). |
| 6 | Identificação dos dados (mais significativos – 1 – ou menos significativos – 0) |
| 5 a 0 | 6 bits mais ou menos significativos (de acordo com o bit 6) |

**Tabela 3.4 - Codificação do byte correspondente ao estado atual da cadeira de rodas.**

| Bit | Significado |
|---|---|
| 7 | Sempre em 0. |
| 6 | Sempre em 0. |
| 5 | Se cadeira andando para frente, 1, caso contrário, 0. |
| 4 | Se cadeira andando para trás, 1, caso contrário, 0. |
| 3 | Se cadeira andando para direita, 1, caso contrário, 0. |
| 2 | Se cadeira andando para esquerda, 1, caso contrário, 0. |
| 1 | Se cadeira parada, 1, caso contrário, 0. |
| 0 | Sempre em 0. |

# 4 RESULTADOS E DISCUSSÕES

Neste capítulo, será apresentada a validação do sistema completo, realizada através de simulações, e os resultados obtidos a partir da construção e implementação das plataformas de hardware e software. As simulações foram realizadas com o auxílio do software Micro-Cap, da fabricante Spectrum Software, e disponível gratuitamente em versão de avaliação em seu site. A versão de avaliação possui limitações no tamanho do circuito (no máximo 50 componentes), na velocidade de análise (de 0 a 300% mais lento do que a versão profissional), no tamanho da biblioteca de componentes, entre outros recursos (SPECTRUM SOFTWARE, 2010), porém, possui funcionalidades suficientes para a realização da validação do sistema proposto.

## 4.1 Validação da plataforma de hardware

Uma vez que a plataforma de hardware foi construída de forma modular, onde cada placa de circuito impresso representa um estágio do sistema, a validação foi facilitada. Na Figura 4.1 o circuito eletrodo semi ativo foi simulado, utilizando-se como entrada uma senóide de amplitude 40 mV e frequência 100 Hz. Na Figura 4.2, é possível ver uma imagem do osciloscópio realizando uma medição no circuito, no qual uma senóide de amplitude próxima a 20 mV e frequência próxima de 300 Hz foi utilizada como entrada (identificada na imagem como CH2) e a saída exibiu uma amplitude de 180 mV (identificada na imagem como CH4). Considerando os ruídos do sistema de medição, uma vez que o sinal de terra foi referenciado longe do circuito, o que acabou modificando um pouco o sinal de entrada, e as tolerâncias dos componentes utilizados, pode-se dizer que o ganho real foi próximo o suficiente do teórico. Na Figura 4.3, pode-se verificar o eletrodo semi ativo como um todo, incluindo os conectores para eletrodos, o circuito e o conector utilizado na comunicação do circuito com o restante da plataforma de hardware.

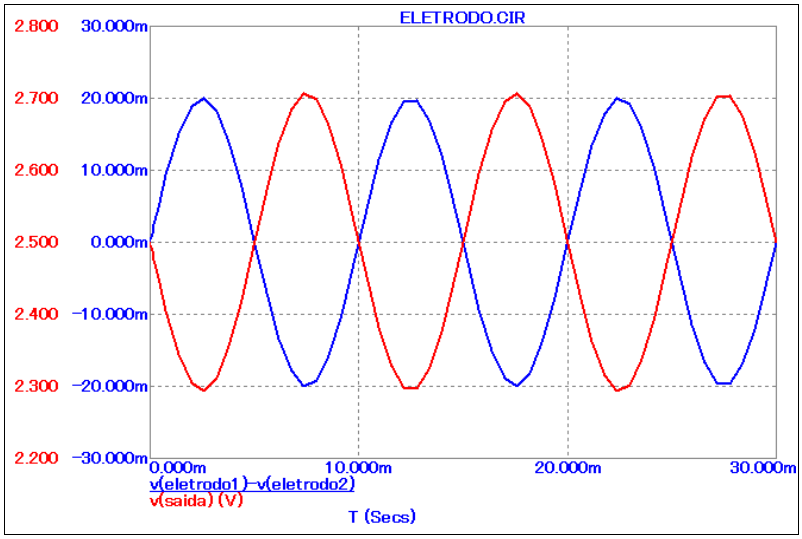



**Figura 4.1 - Simulação do eletrodo semi ativo.**

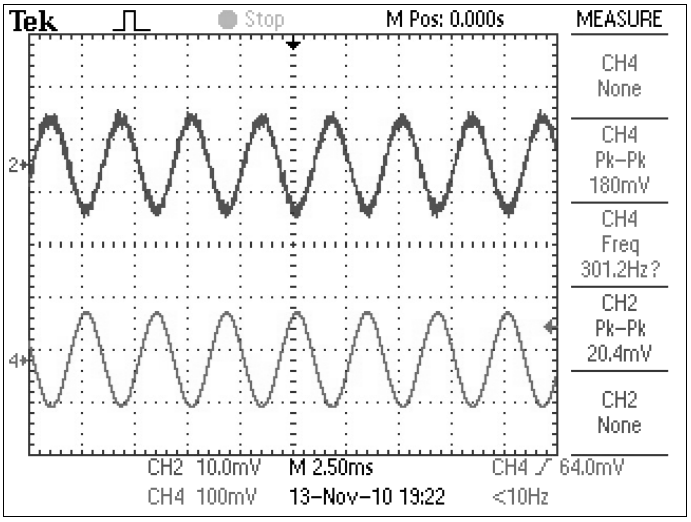



**Figura 4.2 - Imagem do osciloscópio realizando uma medição do circuito do eletrodo semi-ativo, onde CH2 é a entrada e CH4 é a saída.**


**Figura 4.3 - Foto do eletrodo semi ativo utilizado em um dos canais**

A simulação do estágio de entrada, onde há apenas um filtro passa altas e buffers, foi realizada utilizando-se uma senóide de amplitude 200 mV, frequência 100 Hz e centrada em 2,5 V, conforme pode-se verificar na Figura 4.4. Já na Figura 4.5, verifica-se uma imagem do osciloscópio realizando uma medição do circuito, onde o sinal de entrada aplicado possui amplitude 640 mV, frequência 259 Hz e centrado em 2,5 V. A Figura 4.6 é uma foto do módulo de entrada, na qual é possível verificar a entradas para os dois canais de aquisição. O Anexo A.1 apresenta o esquemático utilizado, onde o amplificador operacional utilizado foi o OPA4705, da fabricante Texas Instruments, uma vez que possui um *offset* baixo, aliado a entradas e saídas do tipo *rail-to-rail*. O Anexo A.2 apresenta o layout criado.

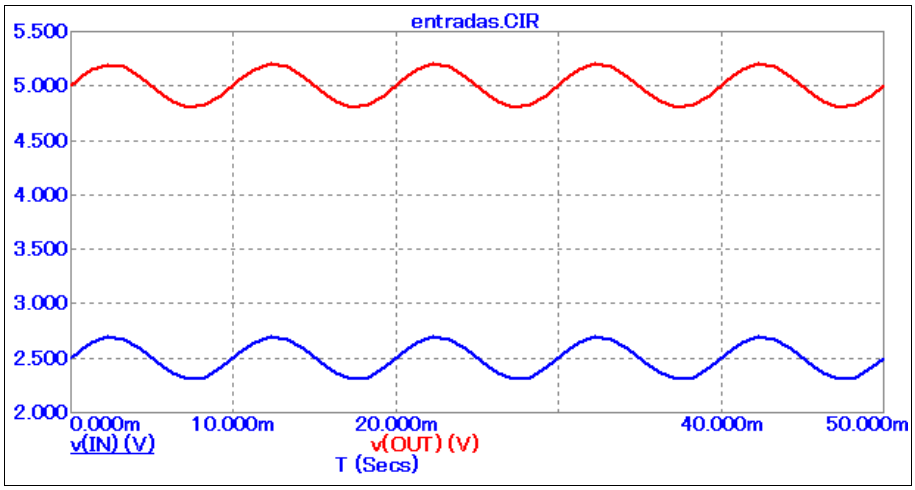



**Figura 4.4 - Simulação do estágio de entrada.**

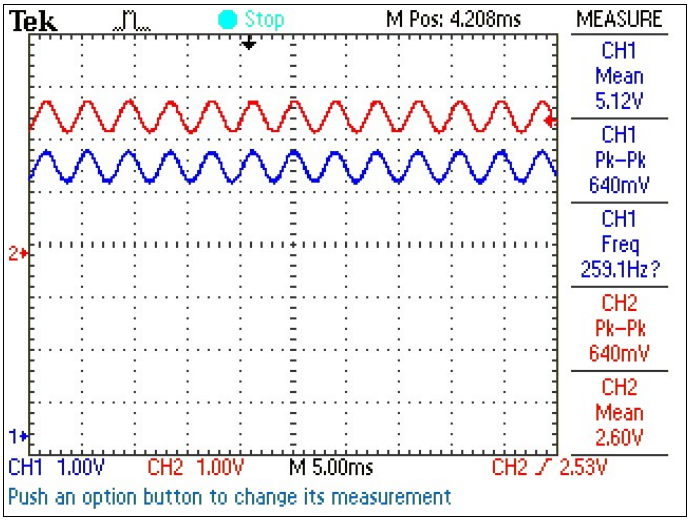



**Figura 4.5 - Imagem do osciloscópio realizando uma medição no circuito de entrada, onde CH2 é a entrada e CH1 é a saída.**

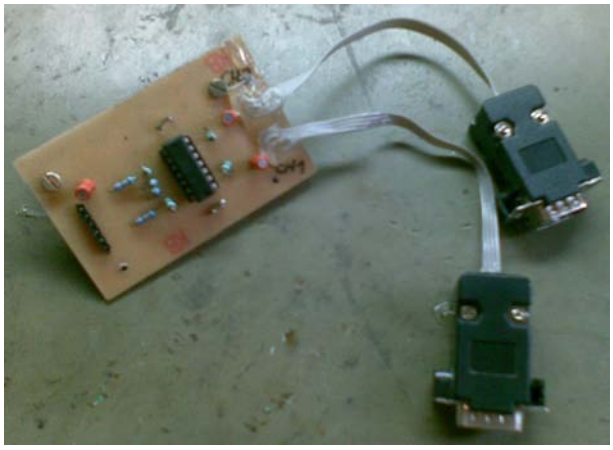


**Figura 4.6 - Foto do estágio de entrada.**

O estágio de filtragem, no que diz respeito ao filtro de capacitores chaveados não pôde ser simulado, já que o componente não está presente nas bibliotecas do Micro-Cap. Na Figura 4.7, pode-se verificar uma imagem do osciloscópio em que percebe-se o efeito do serrilhamento causado pelo CI MF10, conforme explicado no Capítulo 3.1.2. Nas Figuras 4.8(a) e 4.8(b), verifica-se imagens do osciloscópio realizando medidas sobre o circuito de filtragem completo, onde CHX é a entrada e CHY é a saída. Percebe-se que na Figura 4.8(a), um sinal de frequência próxima a 300 Hz e amplitude 720 mV é inserida, resultando em uma saída de amplitude cerca de duas vezes maior, devido ao filtro passa baixas de frequência fixa em 500 Hz e ganho de 6 dB. Já na Figura 4.8(b), o sinal de frequência elevada (cerca de 1600 Hz ) é bastante atenuado. Finalmente, a Figura 4.9 mostra uma foto da placa do estágio de filtragem para um dos canais. O Anexo A.3 apresenta o esquemático completo utilizado, onde o amplificador operacional utilizado foi o OPA4705, da fabricante Texas Instruments, uma vez que possui um *offset* baixo, aliado a entradas e saídas do tipo *rail-to-rail*. Neste esquema, os *buffers* de tensão foram previstos originalmente como uma topologia amplificadora não inversora, porém, como não houve a necessidade de realizar amplificação neste estágio, os resistores utilizados na realimentação do circuito foram substituídos por curto circuitos. O Anexo A.4 apresenta o layout criado.

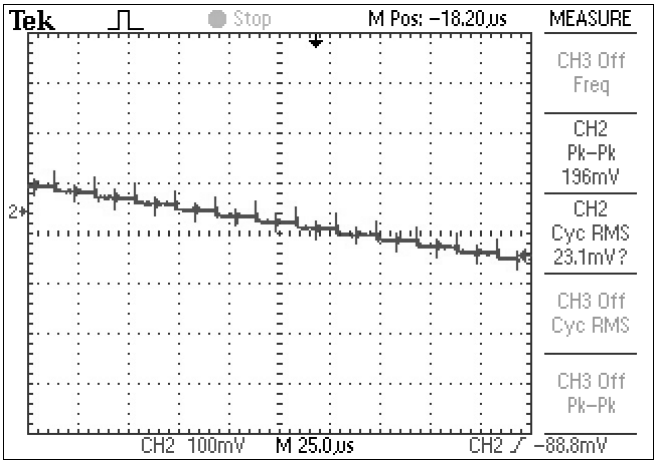



**Figura 4.7 - Imagem do osciloscópio mostrando o efeito de serrilhamento no sinal de saída do CI MF10.**

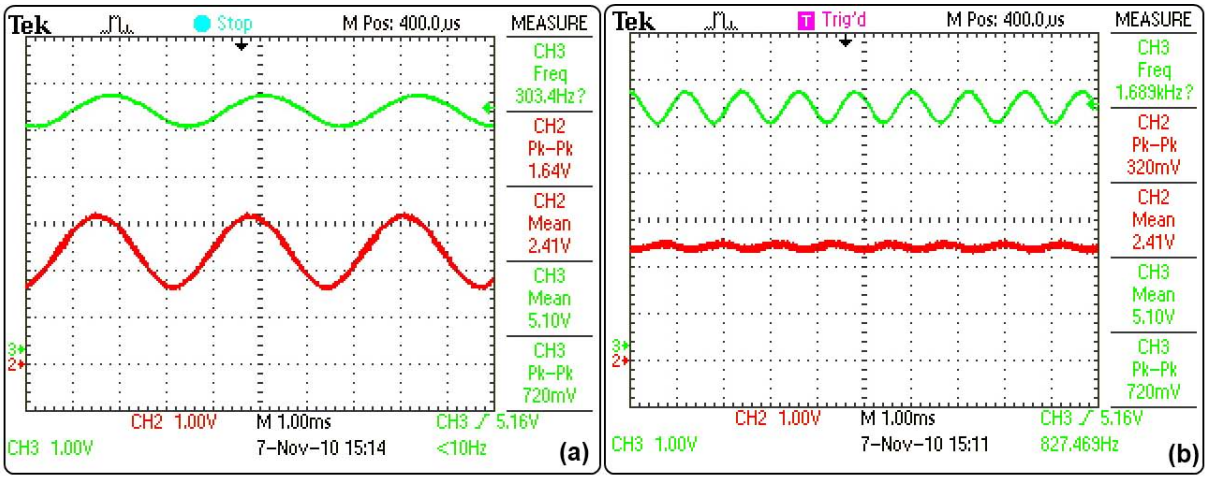



**Figura 4.8 - Medições com o osciloscópio do estágio de filtragem, onde CH2 é a saída e CH3 é a entrada com frequência (a) 303 Hz e (b) 1689 Hz.**

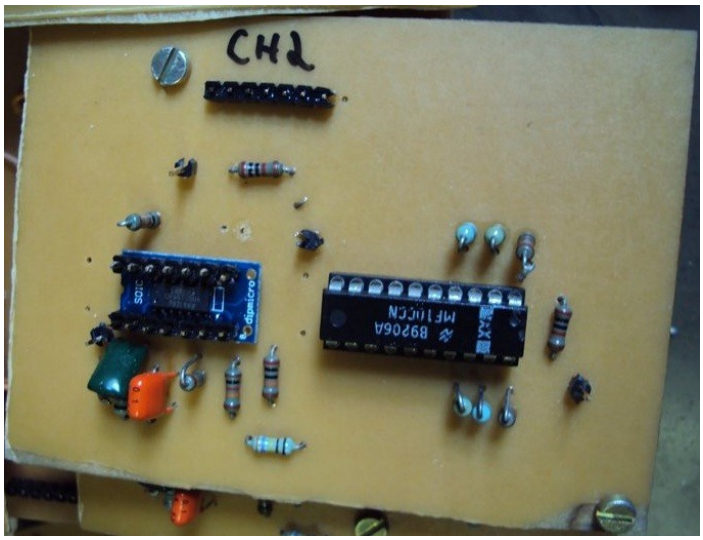



**Figura 4.9 - Foto da placa de circuito impresso criada para o estágio de filtragem.**

O estágio de amplificação completo não foi possível de simular, uma vez que o multiplexador analógico utilizado também não está presente nas bibliotecas do Micro-Cap. Entretanto, pôde-se simular o funcionamento do amplificador de instrumentação INA129, conforme verifica-se na Figura 4.10(a), onde o sinal de entrada possui amplitude 100 mV, frequência 100 Hz e está centrado em 2,5V, e o ganho está configurado para 5,08 vezes, resultando em um sinal de amplitude 510 mV. Na Figura 4.10(b), pode-se verificar a simulação com o mesmo sinal de entrada, mas com o ganho configurado para 10,20 vezes, resultando em uma saída com amplitude próxima a 1 V. A Figura 4.11 possui duas imagens do osciloscópio medindo os sinais do estágio de amplificação com ganhos configurados em 5 – Figura 4.11(a) – e 10 vezes – Figura 4.11(b) –, onde, em ambas, CH2 é o sinal de entrada, com amplitude 360 mV, frequência próxima de 120 Hz e centrado em 2,5V, e CH1 é a saída. Finalmente, na Figura 4.12, é possível verificar a placa criada para o estágio de amplificação.

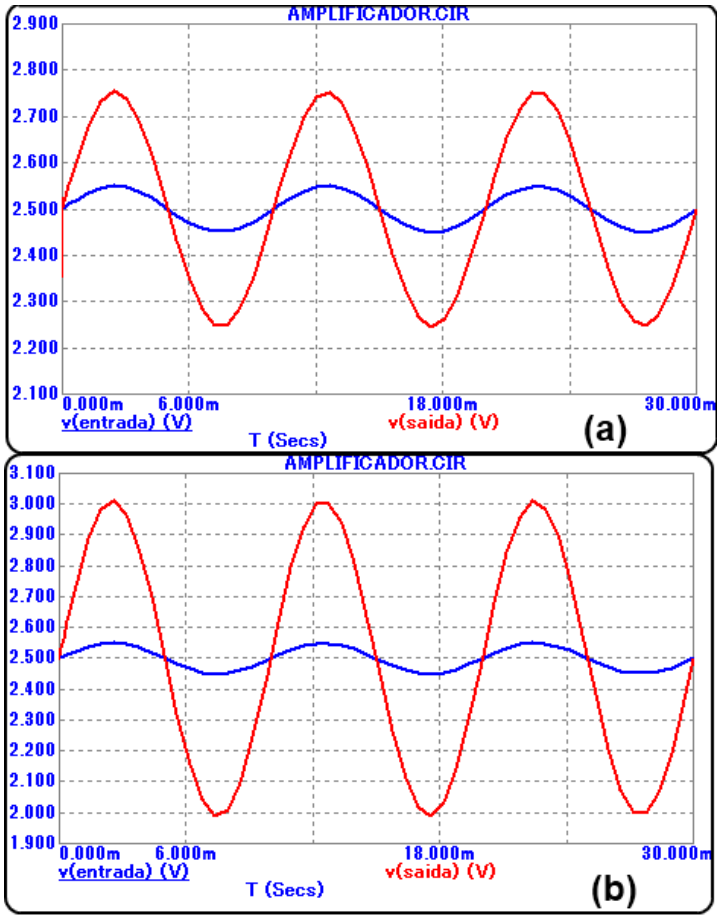



**Figura 4.10 - (a) Simulação do circuito de amplificação com ganho configurado em 5 vezes e (b ) em 10 vezes.**

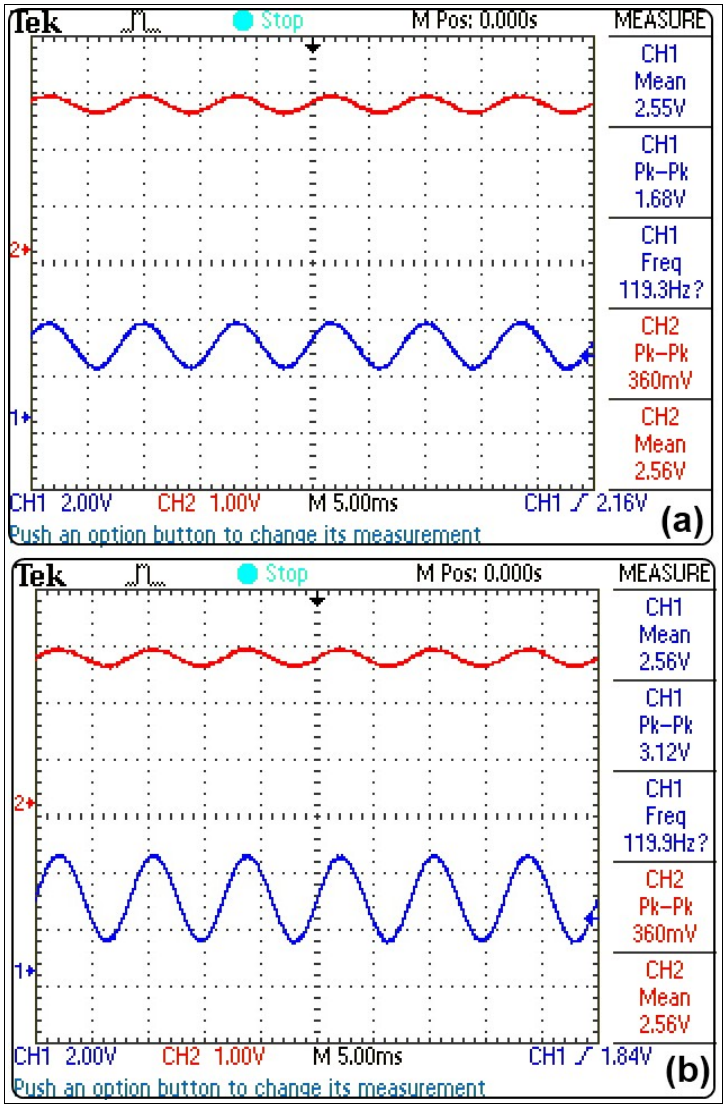



**Figura 4.11 - (a) Imagem do osciloscópio realizando uma medição no circuito de amplificação com ganho configurado em 5 vezes e (b) em 10 vezes.**

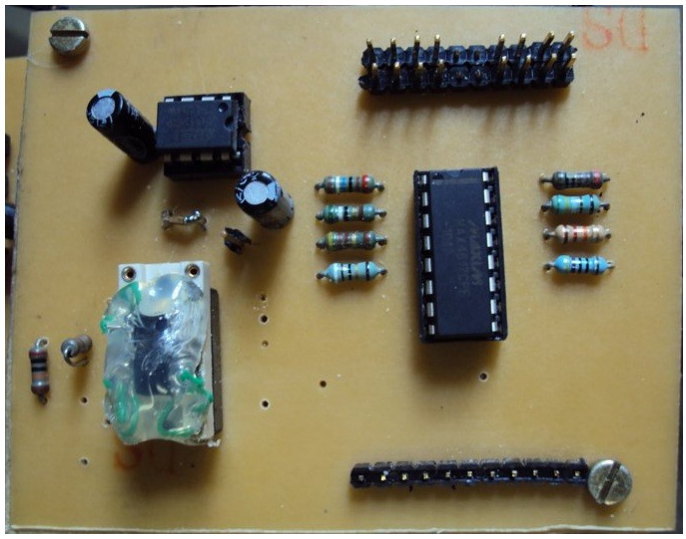


**Figura 4.12 - Foto da placa do estágio de amplificação.**

O Anexo A.5 apresenta o esquemático completo do estágio de amplificação utilizado, onde o amplificador operacional utilizado foi o OPA4705, da fabricante Texas Instruments, uma vez que possui um *offset* baixo, aliado a entradas e saídas do tipo *rail-to-rail*. Neste esquema, os *buffers* de tensão foram previstos originalmente como uma topologia amplificadora não inversora, porém, como não houve a necessidade de realizar amplificação neste estágio, os resistores utilizados na realimentação do circuito foram substituídos por curto circuitos. O Anexo A.6 apresenta o layout criado.

A parte digital da plataforma de hardware foi criada em uma placa à parte da parte analógica e é composta pelo microcontrolador PIC16F877A, o CI MAX232, um conector para a comunicação Serial RS-232C, o conversor analógico-digital MCP3304, uma barra de pinos fêmea para a conexão com a parte analógica do circuito e um conector para a comunicação com o circuito de acionamento da cadeira de rodas. Na Figura 4.13, pode-se verificar uma foto da parte digital da plataforma de hardware.

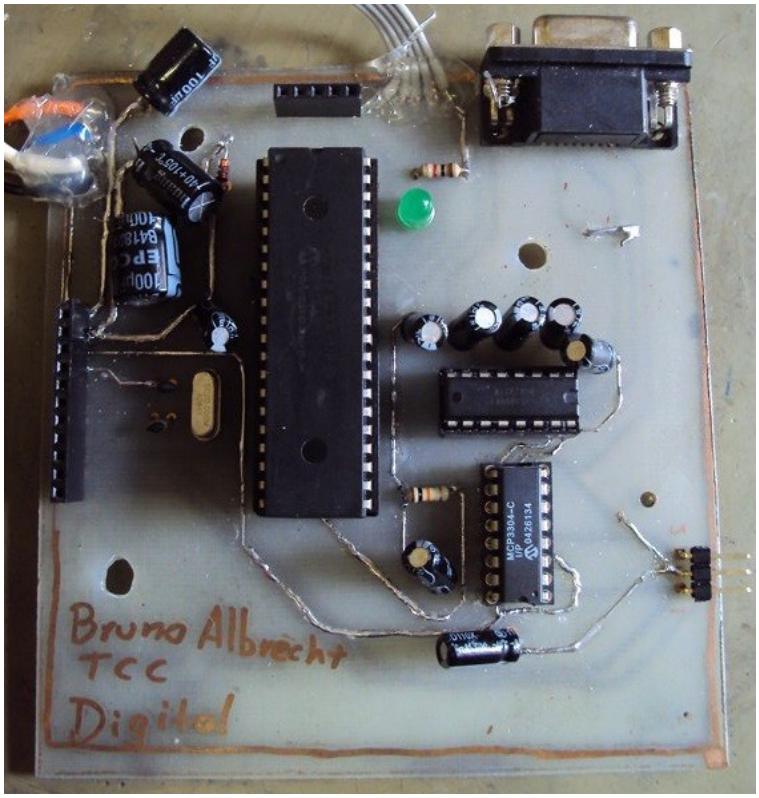



**Figura 4.13 - Foto da parte digital da plataforma de hardware.**

O Anexo A.7 apresenta o esquemático completo do estágio chamado de “Digital” da plataforma de hardware. Deve destacar que, para proteger o restante do circuito durante a gravação do microcontrolador, o resistor de *pull-up* do pino 1 (resistor R2) foi trocado por um diodo simples de sinal com o anodo conectado à tensão de 5 V. O Anexo A.8 apresenta o layout criado.

A conexão da parte digital com a parte analógica do sistema foi realizada através de uma placa intermediária, responsável por receber alguns sinais da parte digital, como, por exemplo, a alimentação dos circuitos, o sinal de configuração da frequência de corte do estágio de filtragem e os sinais responsáveis pela escolha do ganho do estágio de amplificação, e distribuí-los pelas outras placas, de acordo com a necessidade. Na Figura 4.14(a), é possível verificar a face superior desta placa intermediária, onde os estágios da parte analógica conectam-se, e, na Figura 4.14(b ), é possível verificar a face inferior da mesma. Nos Anexos A.9 e A.10, é possível verificar o esquemático e o layout utilizados na criação desta placa, respectivamente.

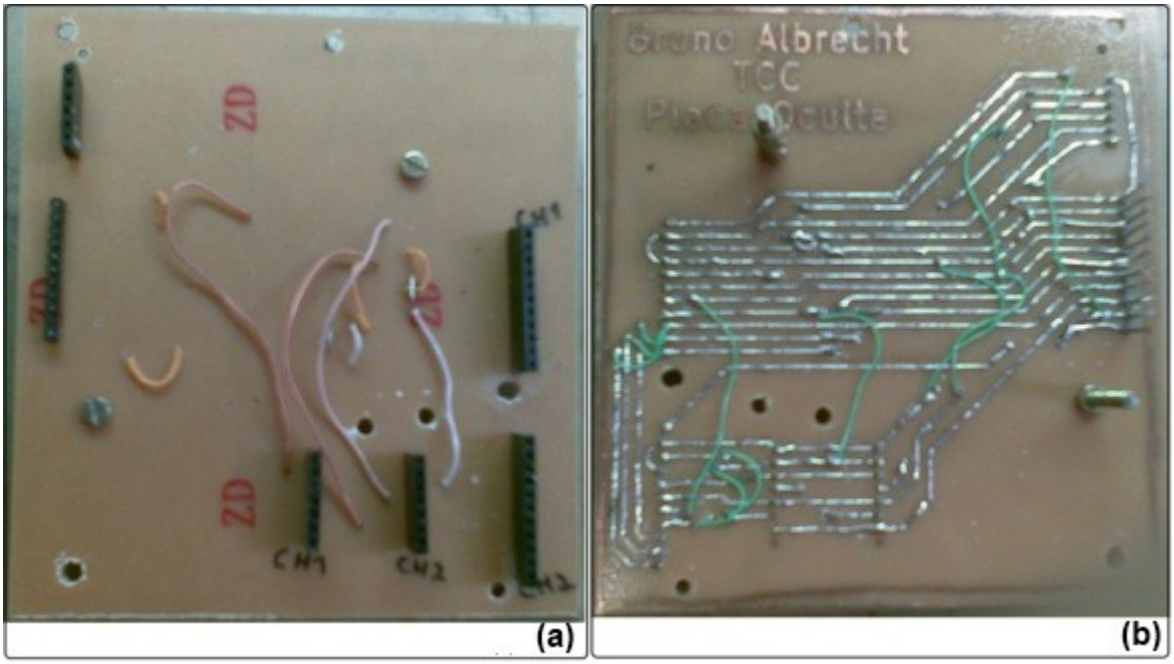



**Figura 4.14 - (a) Foto da face superior da placa de distribuição dos sinais e (b) foto da face inferior da mesma.**

Para a alimentação do circuito, foi criada uma placa que recebe como alimentação a tensão da bateria e provê ao circuito as tensões de 10 V, 5 V e 2,5 V. Na Figura 4.15, é possível verificar uma foto da placa de alimentação do circuito e

nos Anexos A.11 e A.12, é possível verificar o esquemático e o layout da placa criada.

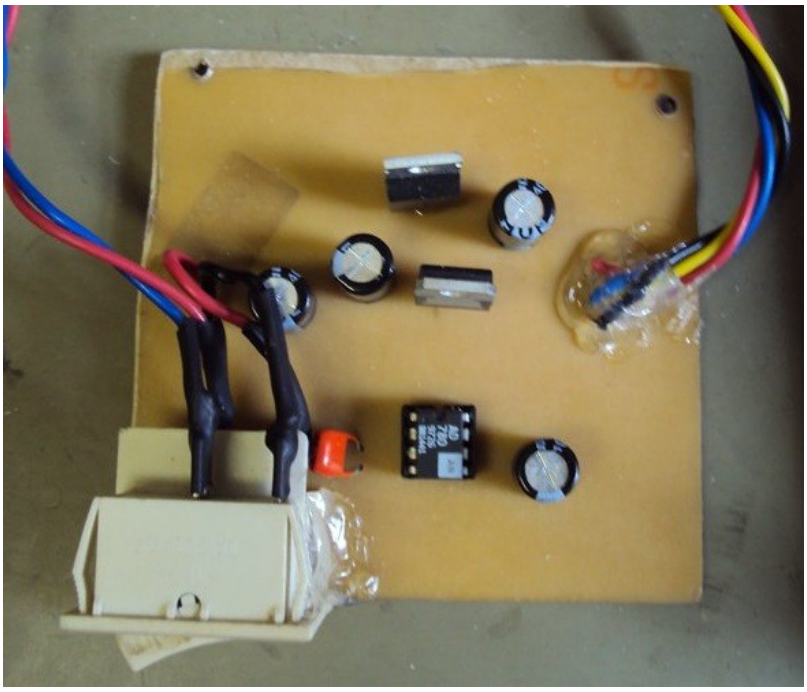


**Figura 4.15 - Foto do circuito de alimentação da plataforma de hardware.**

## 4.2 Validação da plataforma de software

Utilizando o software de desenvolvimento MPLAB em conjunto com o software de simulação Proteus, foi possível simular o funcionamento da plataforma de software durante o desenvolvimento. Na Figura 4.16, pode-se verificar uma imagem do Terminal do Proteus, que mostra os dados provenientes da comunicação Serial do microcontrolador, e, na Figura 4.18, há os sinais de controle da frequência de corte do estágio filtragem quando esta está em 300 e 500 Hz. O código fonte completo do software implementado no microcontrolador PIC16F877A pode ser verificado nos Anexos A.13, A.14, A.15, A.16 e A.17.

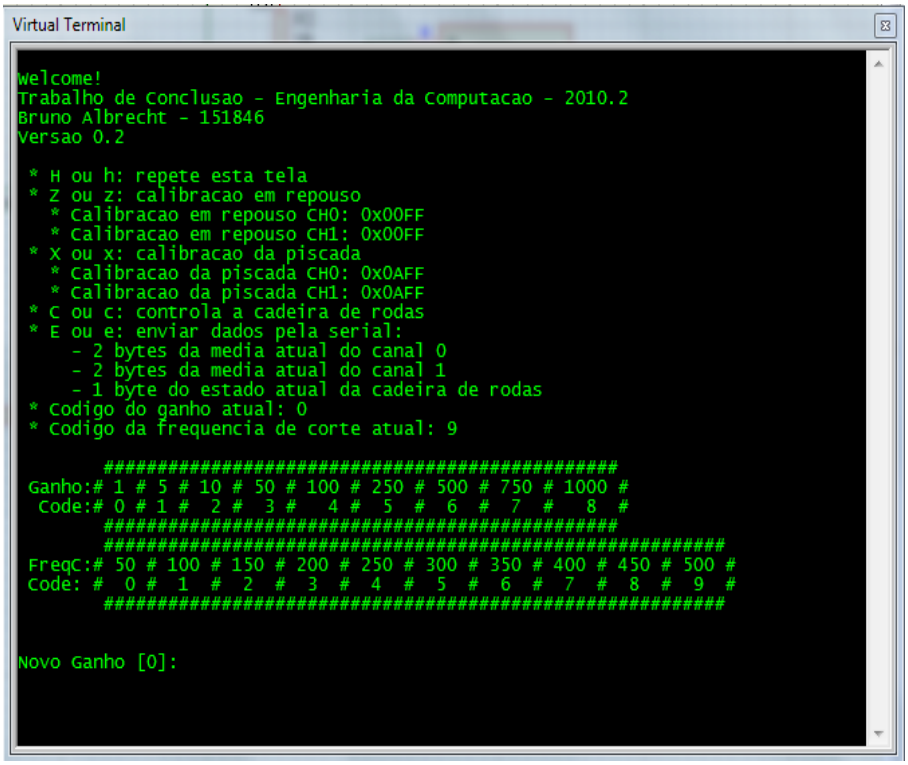



**Figura 4.16 - Simulação da comunicação Serial do microcontrolador.**

Na Figura 4.17, é possível verificar uma imagem do programa Realterm, um software gratuito para leitura de dados da porta Serial do computador, realizando a comunicação entre o computador e o microcontrolador. Já na Figura 4.19, há os sinais de controle da frequência de corte do estágio de filtragem medidos através do osciloscópio.

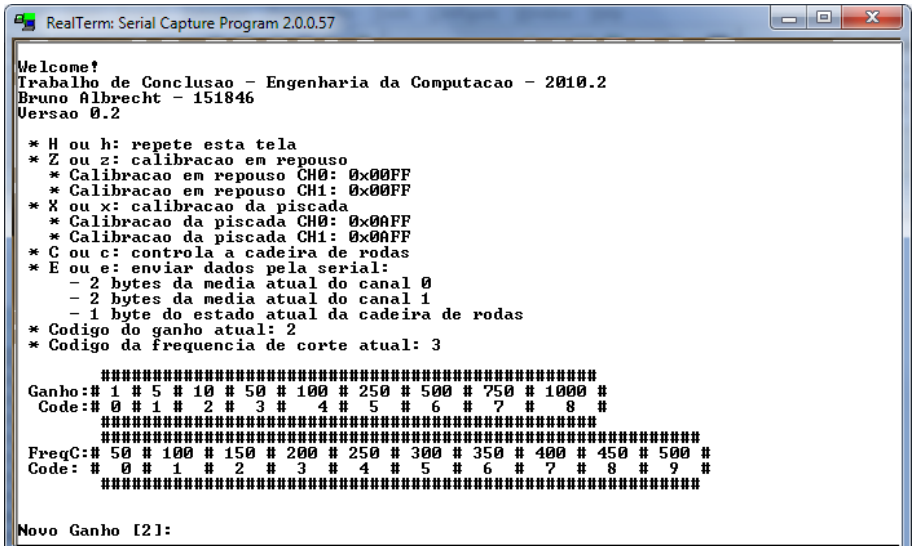



**Figura 4.17 - Imagem do software Realterm realizando a comunicação Serial com o microcontrolador.**

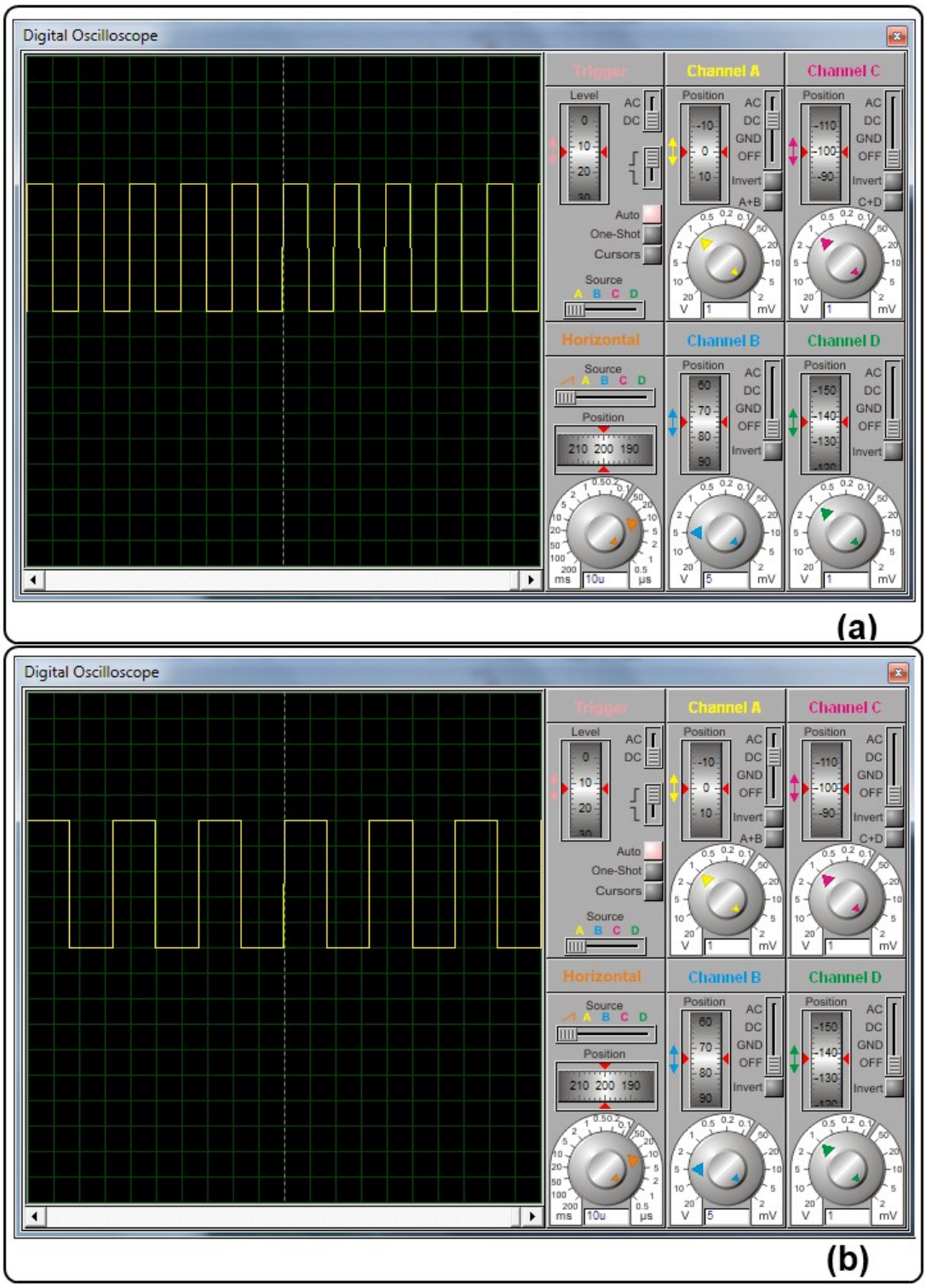



**Figura 4.18 - (a) Imagem do osciloscópio simulado realizando medições do sinal de controle da frequência de corte do estágio de filtragem em 500 Hz e (b) em 300 Hz.**

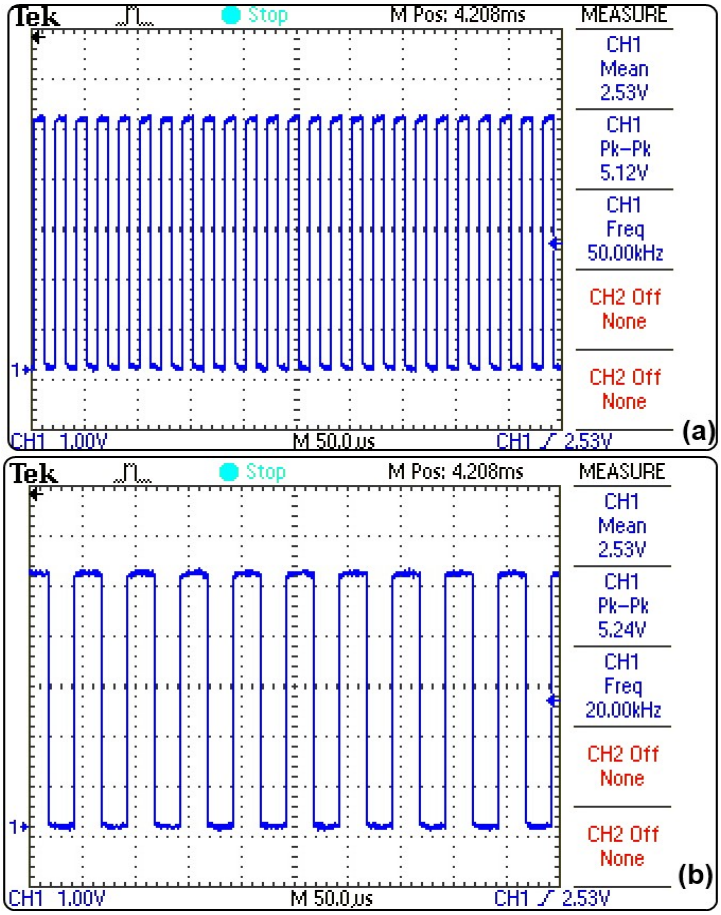



**Figura 4.19 - (a) Imagem do osciloscópio realizando medições do sinal de controle da frequência de corte do estágio de filtragem em 500 Hz e (b) em 200 Hz.**

## 4.3 Validação com sinais reais

A validação inicial do sistema integrado utilizou sinais mioelétricos do músculo bíceps para as medições, devido ao seu comportamento mais expressivo, com sinais de maior amplitude. Os eletrodos utilizados são originalmente destinados a eletrocardiogramas infantis, porém seu único diferencial é o tamanho reduzido. Eles são embebidos em gel condutivo e feitos de uma liga metálica de prata e cloreto de prata. Para a fixação dos eletrodos, primeiramente limpou-se a área da pele onde seriam colados, de forma a retirar as impurezas presentes que diminuiriam a relação sinal ruído. O eletrodo de referência foi fixado no lóbulo da orelha, uma região neutra, onde não há atividade muscular. Para a captura dos sinais, realizou-se a calibração em repouso e durante as piscadas do sistema, caso não especificado. A criação dos gráficos foi realizada no software BrOffice Calc, com os dados enviados pela plataforma de hardware através da conexão Serial.

A Figura 4.20 mostra um gráfico do valor bruto do recebido do conversor analógico digital pelo tempo, onde o canal 1 é o equivalente ao canal do olho esquerdo e seus eletrodos estão conectados, e o canal 2 é o equivalente ao do olho direito e está desconectado. Já na Figura 4.21, é possível verificar uma situação semelhante, porém utilizando-se o canal 2 de aquisição, enquanto o canal 1 permanece desconectado. Em ambas as figuras, o músculo bíceps cujo sinal mioelétrico foi analisado exercitou-se com um peso de 5 kgf, sendo facilmente identificáveis os momentos de maior esforço. Nota-se ainda que o fato do canal inativo estar desconectado causa um sinal com ruído muito acentuado, chegando a um quarto do *span* do conversor analógico-digital. Em ambas as aquisições, foram utilizados ganhos de 100 vezes ou 40 dB no estágio de amplificação e frequência de corte do estágio de filtragem em 300 Hz, uma vez, após alguns testes, essa configuração se mostrou mais eficaz, exibindo uma relação sinal ruído mais elevada.

Deve-se destacar que o ganho total do sistema, incluindo o eletrodo semi-ativo, cujo ganho é de cerca de 10 vezes ou 20 dB, e o estágio de filtragem, cujo ganho é de 6 dB, é de 2000 vezes ou 66 dB, o que faz com que sinais de amplitude 1 mV na entrada do sistema sejam amplificados para 2 V. Devido a essa relação, ganhos maiores do que 100 vezes no estágio de amplificação não foram utilizados, já que os valores de *offset* do circuito faziam com que os sinais de saída saturassem, dando à saída um comportamento não linear.

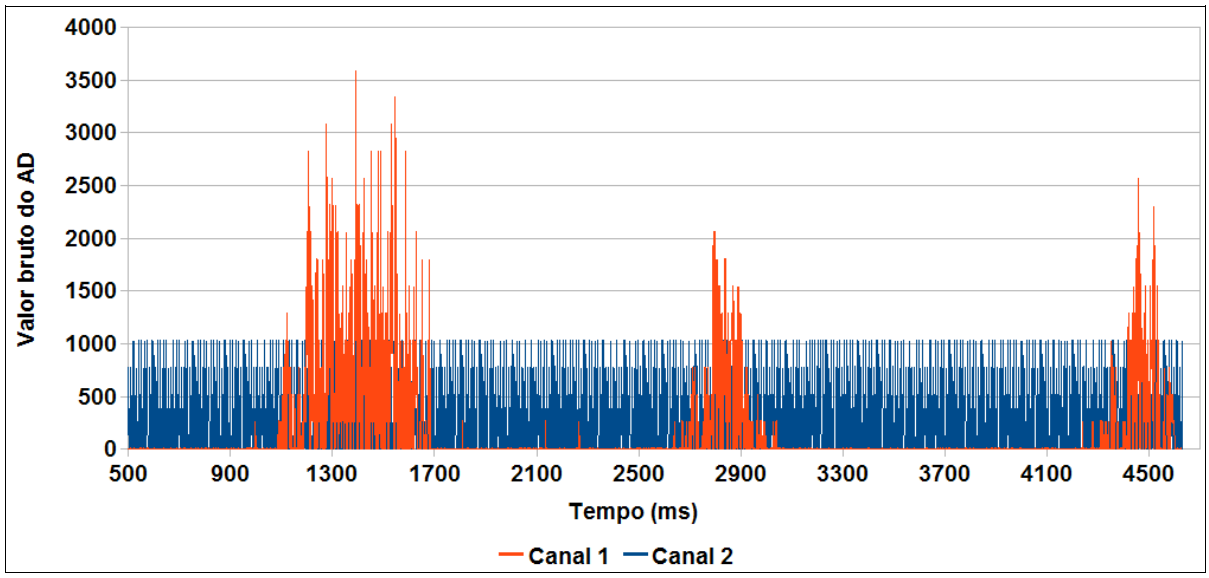



**Figura 4.20 - Validação do canal 1 de aquisição com sinal mioelétrico do músculo bíceps exercitando-se com peso de 5kgf.**

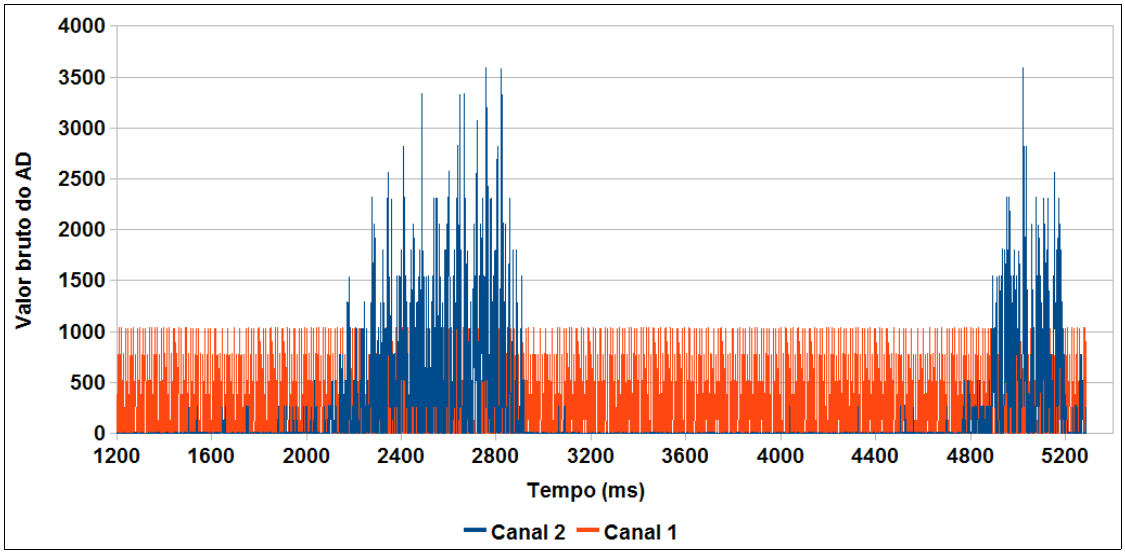



**Figura 4.21 - Validação do canal 2 de aquisição com sinal mioelétrico do músculo bíceps exercitando-se com peso de 5kgf.**

### 4.3.1 Ruídos devido à alimentação

Em um primeiro momento, utilizou-se o eletromiógrafo com sua alimentação provida por uma fonte de tensão de bancada, que, por sua vez, alimenta-se da rede elétrica. As primeiras medições mostraram um nível de ruído muito grande, reduzindo a relação sinal ruído drasticamente. Uma análise posterior do sinal adquirido em repouso mostrou a presença expressiva de sinais de frequências próximas a 60 Hz. Tais sinais são claramente de origem da rede elétrica e, apesar dos capacitores de desacoplamento utilizados no estágio de alimentação do circuito, não foram evitados corretamente. A solução para este problema foi realizar os testes utilizando como alimentação a própria bateria da cadeira de rodas, de forma que a relação SNR obteve uma grande melhoria, mas ainda não satisfatória.

Uma vez que a comunicação Serial com o computador foi utilizada para a captação e armazenamento dos sinais mioelétricos, foi necessária a utilização de um notebook. Percebeu-se, porém, que, quando o circuito da plataforma de hardware era alimentado pela bateria da cadeira de rodas e o notebook permanecia conectado à tomada, os ruídos causados pela rede elétrica eram bastante expressivos, estando esta tomada aterrada ou não. Desconectando-se o computador da rede elétrica, a relação SNR obteve um desempenho muito satisfatório, sendo esta metodologia a utilizada no decorrer dos testes. Para ilustrar esta situação, verifica-se na Figura 4.22 o sinal do músculo bíceps em repouso

quando o notebook estava conectado à rede elétrica e, na Figura 4.23, a mesma situação de repouso, porém com o notebook operando através de sua bateria interna. Percebe-se uma redução na ordem de dez vezes no nível de ruídos do sinal, que foi adquirido sem realizar a calibração em repouso.

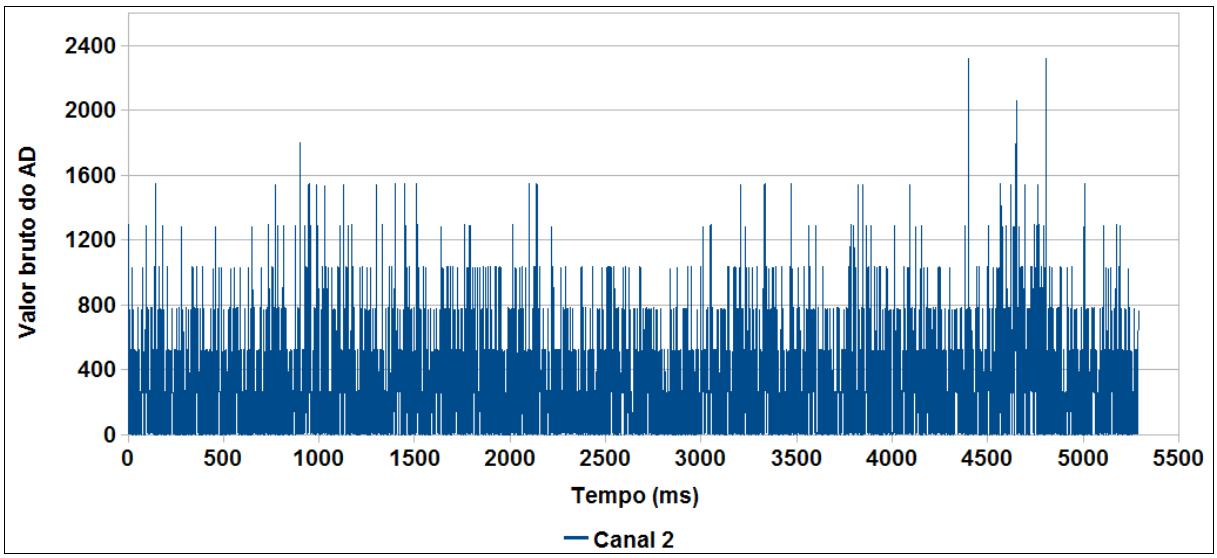



**Figura 4.22 - Sinal do músculo bíceps em repouso com a aquisição dos dados em um notebook conectado à rede elétrica.**

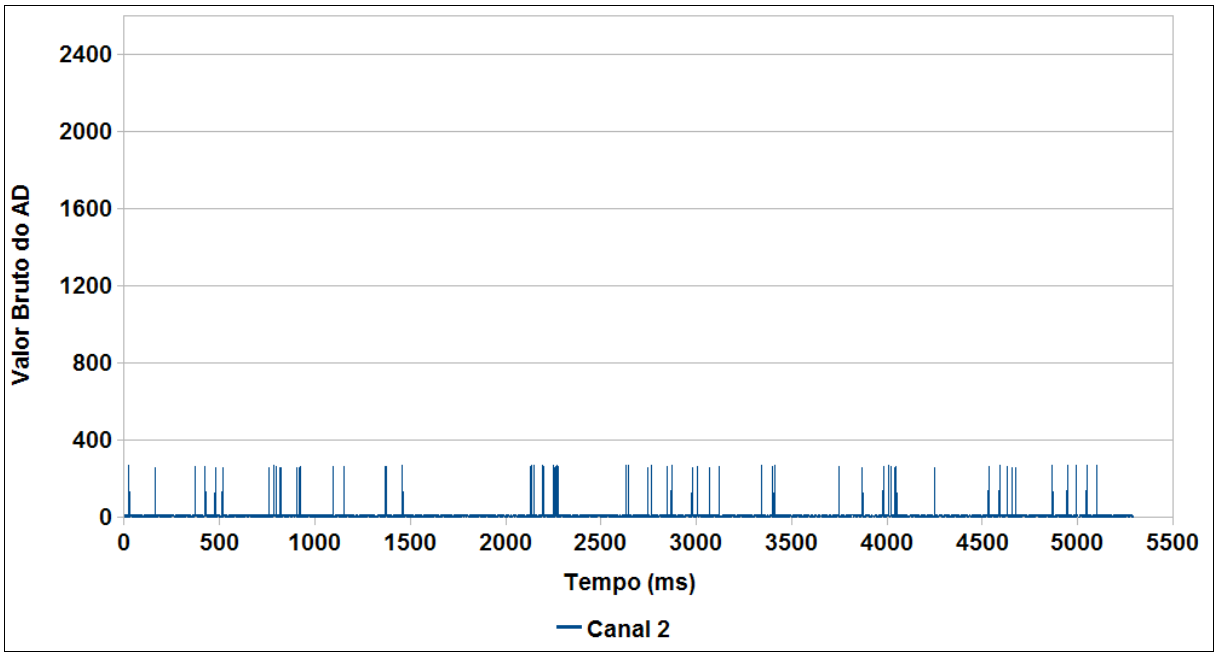



**Figura 4.23 - Sinal do músculo bíceps em repouso com a aquisição dos dados em um notebook operando através de sua bateria interna.**

## 4.4 Controle da cadeira de rodas

A partir do sistema validado e funcional, pôde-se realizar testes utilizando os sinais mioelétricos das piscadas dos olhos, de forma que foi possível controlar a cadeira de rodas. Os eletrodos de medição foram dispostos na face de acordo com a Figura 4.24, ficando abaixo dos olhos, e o eletrodo de referência fixado ao meio da testa. Deve-se atentar ao fato de que os comandos realizados com as piscadas só são atendidos dois segundos após a finalização da primeira piscada e no mínimo 100 milissegundos após o fim da última piscada. Para os testes, foram realizadas as calibrações em repouso e das piscadas, sendo a última realizada piscando-se diversas vezes durante o período necessário.

Na Figura 4.25, é possível verificar o comando “andar para frente”, seguido do “parar” e as mudanças do estado da cadeira de rodas, onde os estados “parado” e “andando para frente” são representados pelos valores 0 e 2500, respectivamente. Ao canal referente ao olho direito, foi adicionado um valor de *offset* de 5000 unidades apenas para facilitar a visualização dos dados, devido às sobreposições dos sinais.

**Figura 4.24 - Posicionamento dos eletrodos para as medições.**

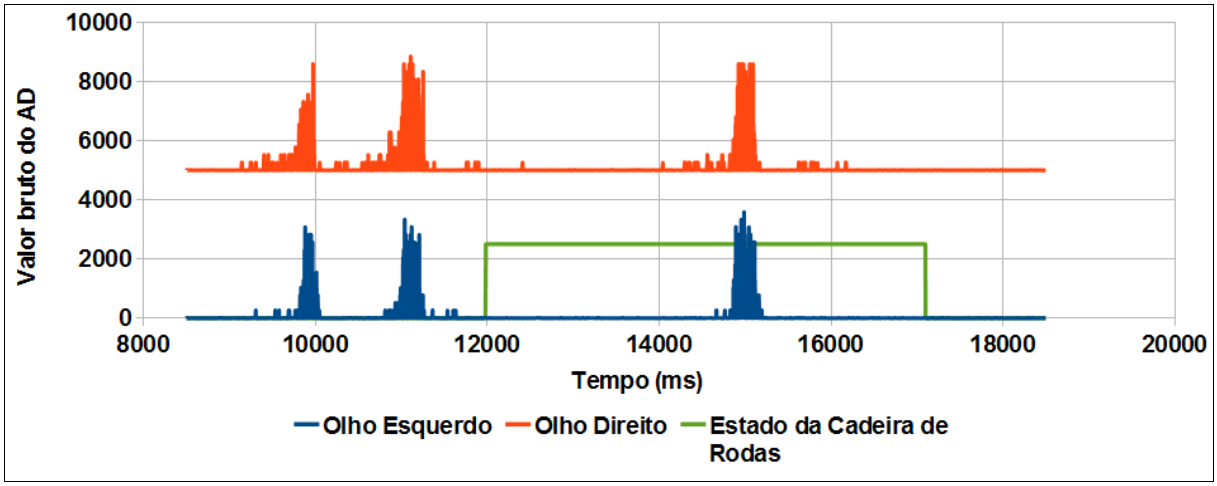



**Figura 4.25 - Comandos “andar para frente” e “parar” ativados por sinais da piscada dos olhos.**

Na Figura 4.26, pode-se verificar a ativação do comando “andar para trás”, seguido do comando “parar”, e as mudanças de estado da cadeira de rodas, onde os estados “parado” e “andando para trás” estão representados pelos valores 0 e 5000, respectivamente. Novamente, foi adicionado um *offset* ao valor bruto do conversor analógico digital referente aos sinais do olho direito.

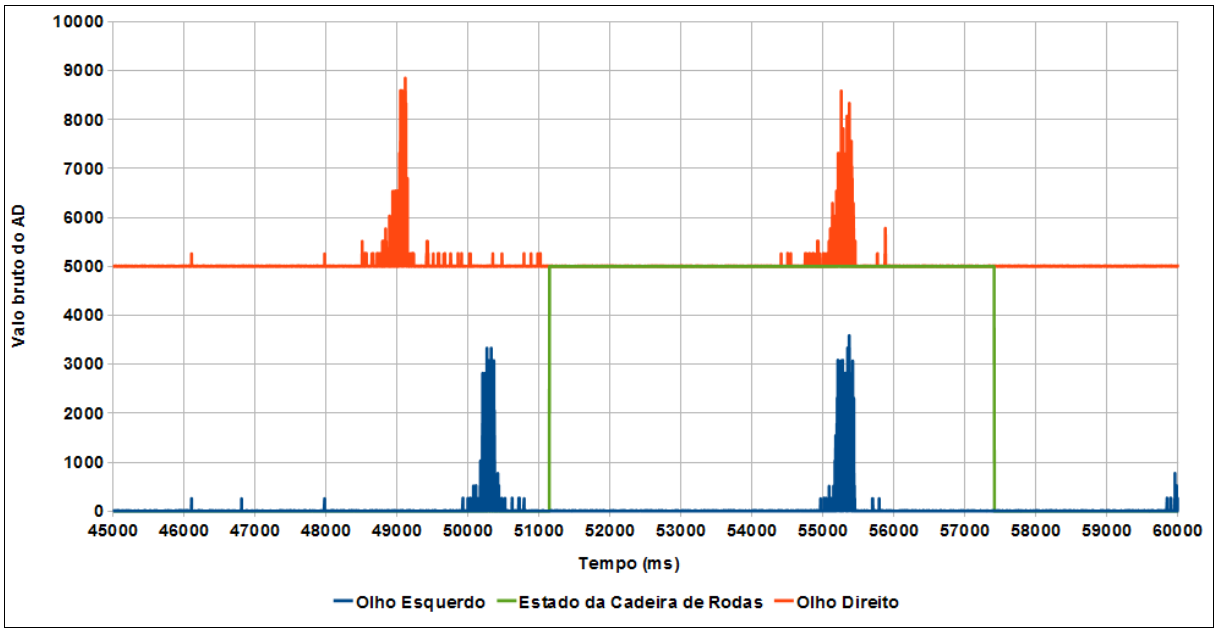



**Figura 4.26 - Comandos “andar para trás” e “parar” ativados por sinais da piscada dos olhos.**

Para verificar o funcionamento de comandos inválidos durante o movimento, quando somente é permitido parar, foi ativado primeiramente o comando “andar para

a direita", tenta-se ativar o comando "andar para frente" e ativa-se o "parar", como pode-se verificar na Figura 4.27. Nesta imagem, os estados "parado" e "andando para a direita" são representados pelos valores brutos do conversor analógico digital 0 e 10000, e ao sinal "Olho Direito" foi adicionado o *offset* de 5000 unidades, assim como realizado nas análises anteriores.

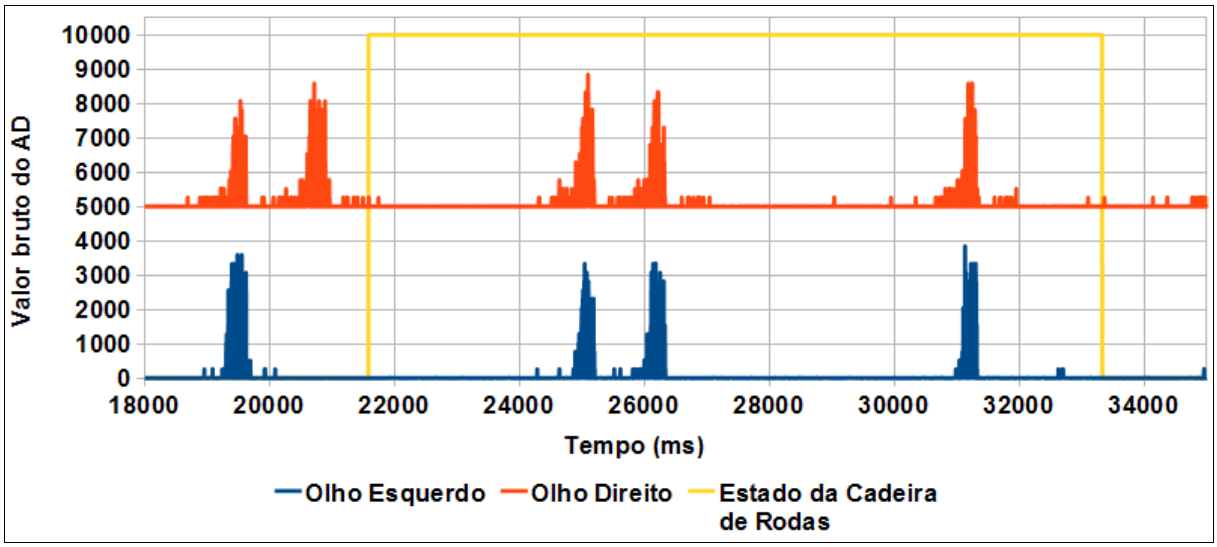



**Figura 4.27 - Comandos "andar para a direita", tentativa de ativação do comando "andar para frente" e comando "parar" ativados por sinais da piscada dos olhos.**

Percebe-se, portanto, que o funcionamento dos algoritmos de tratamento do sinal, de detecção de piscadas e de controle da cadeira de rodas foi satisfatório, uma vez que, apesar de estar sob um nível significante de ruídos, foi robusto o suficiente para reconhecer os sinais das piscadas e, desta forma, realizar o controle da cadeira de rodas. Percebe-se também que a plataforma de hardware, quando operando sob as condições corretas (sem conexões à rede elétrica), cumpriu com os objetivos de filtragem e amplificação dos sinais, tornando-os legíveis para a plataforma de software.

# 5 CONCLUSÕES

O objetivo deste trabalho foi projetar um sistema de aquisição de sinais mioelétricos, especificamente para aqueles relacionados ao movimento das pálpebras, bem como de um software de tratamento para os dados recebidos. Como ferramenta para a análise dos dados, utilizou-se o próprio microcontrolador responsável também por configurar a frequência de corte do estágio de filtragem e o ganho do estágio de amplificação, de forma que não fosse necessário utilizar um computador em conjunto com o sistema.

Como pôde-se perceber, a plataforma de hardware projetada possui a intenção de ser bastante flexível, de forma que possa ser configurado por meio de uma comunicação Serial com um computador, o que permite a um software modificá-la ainda em tempo de execução. Já a plataforma de software possuiu como objetivos principais a simplicidade associada à robustez, ainda que o código deva ser reduzido para que seja implementado em um microcontrolador.

Portanto, os principais objetivos deste trabalho foram alcançados: projeto de um sistema de hardware para aquisição de sinais analógicos configurável digitalmente, a implementação de rotinas de tratamento digital de sinais mioelétricos de forma simples e o estudo das ferramentas disponíveis para a implementação destas tarefas.

# 6 SUGESTÕES PARA TRABALHOS FUTUROS

## 6.1 Melhorias da plataforma de hardware

A fabricação caseira das placas de circuito impresso não possui uma precisão boa, de forma que é muito comum encontrar-se falhas em trilhas, as quais fazem com que o circuito não funcione ou funcione de forma errática. Além disso, essa forma de fabricação exige que as trilhas tenham uma largura grande, limitando e dificultando o roteamento das trilhas, além de não permitir a utilização de componentes de tamanho muito reduzido, o que requisita um tamanho de placa maior. Uma sugestão para solucionar este problema é industrializar as placas de circuito impresso, o que possibilitaria uma maior confiabilidade quanto a continuidade de trilhas, porém, o custo teria um grande aumento. Com a utilização de uma placa industrializada, seria possível ainda modificar os tipos dos encapsulamentos dos circuitos integrados, atualmente DIP, para tamanhos bastante menores do tipo SMD, como SOIC ou, ainda, SSOP. Isso reduziria o tamanho total da placa, reduzindo bastante os ruídos dos sinais.

Relativo ao circuito implementado, sugere-se utilizar filtros de capacitores chaveados específicos para a implementação de filtros passa baixas, tais como o MAX7410, MAX7480 ou MAX7408, da fabricante Maxim Integrated Products, que são filtros passa baixas do tipo Butterworth de 5ª ordem, Elíptico de 5ª ordem e Butterworth de 8ª ordem, respectivamente. Esses CIs utilizam menos componentes discretos externos, evitando, desta forma, não linearidades causadas pelas tolerâncias dos mesmos. Ainda poderia-se trocar os reguladores de tensão lineares por conversores DC/DC chaveados, cujo consumo é bastante baixo, o que economizaria energia da bateria, ou, ainda, permitiria a utilização da plataforma com pilhas, isolando a alimentação do circuito da alimentação dos motores, reduzindo ruídos. Sugere-se ainda implementar um isolamento elétrico dos eletrodos para evitar possíveis danos à saúde do usuário do sistema.

O estágio de amplificação possui a possibilidade de ser configurado com ganhos na ordem de centenas de vezes, o que faz com que sinais de baixa amplitude apresentem uma resposta expressiva na saída do sistema. Devido às tensões de *offset* presentes no sistema, ganhos nessa ordem fazem com que essas tensões também sejam amplificadas, causando a saturação prematura do sinais.

Uma forma de minimizar esse efeito é realizar diversas pequenas amplificações, de apenas dezenas de vezes, seguidas de filtragens que corrijam o efeito das tensões de *offset*.

Quanto ao microcontrolador utilizado, sugere-se trocá-lo por um com maior poder de processamento, como aqueles baseado em arquitetura ARM, ou, ainda, modelos superiores da linha PIC da fabricante Microchip, tais como as famílias PIC18, PIC24, PIC32 ou ainda os dsPICs. A utilização de microcontroladores melhores permitiria a implementação de algoritmos de processamento digital de sinais mais robustos, bem como algoritmos de detecção de piscadas mais precisos.

À cadeira de rodas, sugere-se implementar o sensoriamento de posição e velocidade, de forma que o controle possa ter maior precisão quanto à direção de movimentação desejada. Com tal característica, seria possível obter uma realimentação do estado real da cadeira de rodas e, caso um comportamento incorreto seja detectado, pode-se corrigi-lo.

Outra abordagem que poderia ser utilizada no controle da cadeira de rodas é a utilização de eletrooculografia, ou seja, a medição do sinal elétrico relativo à movimentação do globo ocular. Este sinal possui uma amplitude mais baixa do que a utilizada neste trabalho, sendo captada na ordem de poucos microvolts. Tais sinais, apesar de mais complexos para serem adquiridos, permitem a usuários cujas habilidades de piscar olhos sejam limitadas utilizarem a cadeira de rodas sem necessidade de auxílio.

## 6.2 Melhorias da plataforma de software

A criação de uma plataforma de hardware programável permite sua reconfiguração em tempo real, de acordo com a possível necessidade da plataforma de software, característica que não foi abordada neste trabalho. Sugere-se, portanto, a implementação do controle do ganho do estágio de amplificação e frequência de corte do estágio de filtragem, permitindo a operação do sistema mesmo com sinais degradados. Sugere-se ainda realizar a calibração dinâmica do sinal adquirido, uma vez que o uso contínuo dos eletrodos de superfície faz com que a relação sinal ruído tenha um decremento após um certo tempo, devido a fatores como o suor ou mesmo a exaustão do músculo em análise.

Utilizando-se uma plataforma embarcada com maior desempenho, seria possível implementar algoritmos mais robustos para a detecção de piscadas, permitindo a utilização da informação do padrão elétrico do sinal adquirido como mais uma entrada ao sistema. A utilização destes dados em conjunto com os dados de um sensoriamento da cadeira de rodas, permitiria, por exemplo, efetuar o controle da velocidade de deslocamento da cadeira.

## REFERÊNCIAS BIBLIOGRÁFICAS


- ALONSO, A. A.; DE LA ROSA, R.; DEL VAL, L.; JIMÉNEZ, M. I.; FRANCO, S. (2009). "**A Robot Controlled by Blinking for Ambient Assisted Living**". Em: Lecture Notes In Computer Science, v. 5518, p. 839-842.
- BALBINOT, A.; TOMASZEWSKI, J. R.; BADARACO, F. R. S.; RADTKE, C. (2006). "**Desenvolvimento de uma prótese experimental controlada por eletromiografia**". Em: IV Congresso Íbero-Americano sobre tecnologias de apoio a portadores de deficiência, v.1, MA-3 - MA-6.
- BAREA, R.; BOQUETE, L.; BERGASA, L. M.; LÓPEZ, E.; MAZO, M. (2003). "**Electro-Oculographic Guidance of a Wheelchair Using Eye Movements Codification**". Em: The International Journal of Robotics Research, v. 22, n. 7-8, p.641-652.
- BASTOS FILHO, T. F.; BUENO, L.; FRIZERA NETO, A.; FERREIRA, A.; CELESTE, W. C.; MARTINS, V. R. (2006). "**Desenvolvimento de Protótipos de Tecnologia Assistiva para Pessoas com Deficiência**". Em: Anais do I Fórum de Tecnologia Assistiva e Inclusão Social da Pessoa Deficiente. Belém, Brasil. p. 71-90.
- BASTOS FILHO, T. F.; CELESTE, W. C.; FERREIRA, A.; FRIZERA NETO, A.; MARTINS, V. R. (2006). "**Proposta de Adaptação de uma Cadeira de Rodas Motorizada para Pessoas Portadoras de Mobilidade Muito Reduzida ou Nula**". Em: Anais do I Fórum de Tecnologia Assistiva e Inclusão Social da Pessoa Deficiente. Belém, Brasil. p. 91-110.
- BELTRAMINI, L. M. (1997) . "**Elementos de Histologia e Anatomo-Fisiologia Humana**". 1.ed. São Carlos, Serviço Gráfico do Instituto de Física de São Carlos - USP, v. 1, 270p.
- BERSCH, R. (2008). "**Introdução à Tecnologia Assistiva**", disponível em: <http://www.assistiva.com.br/Introducao%20TA%20Rita%20Bersch.pdf>, acessado em: novembro de 2010.
- BRONZINO, J. D. (1995). "**The Biomedical Engineering Handbook**". 1. ed., Boca Raton, CRC Press, cap. 3, 14, 48 e 52. Estado Unidos da América.
- CHOI, K.; SATO, M.; KOIKE, Y. (2006). "**A new, human-centered wheelchair system controlled by the EMG signal**". Em: International joint Conference on Nueral Networks, p. 4664-4671.

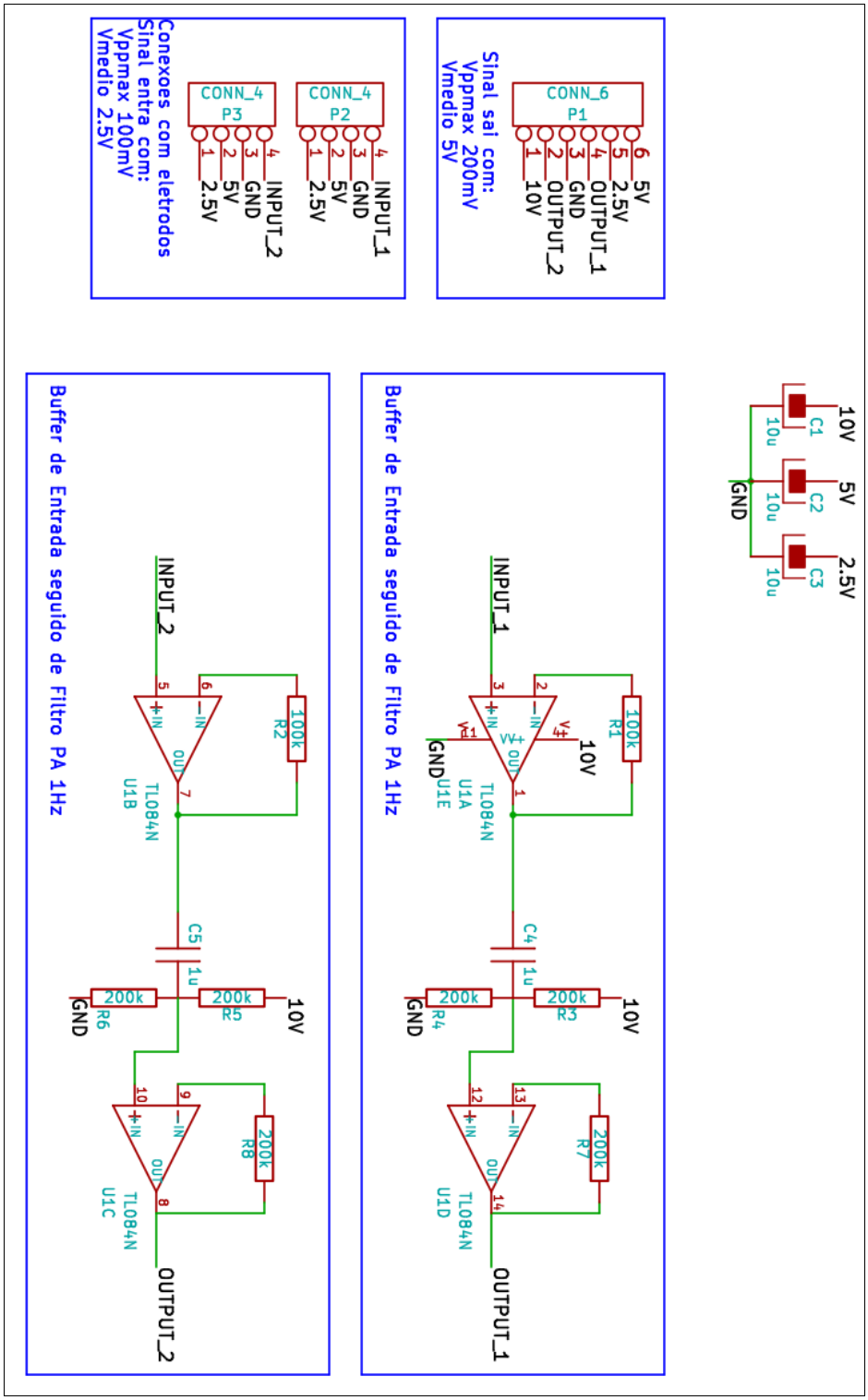
- CORDE (2007), Comitê de Ajudas Técnicas, **Portarias e Atas** disponíveis em: <http://portal.mj.gov.br/corde/comite.asp>, Acessados em: novembro de 2010.
- CORTÉS, U.; URDIALES, C.; ANNICCHIARICO, R.; BARRUÉ, C.; MARTINEX, A. B.; CALTAGIRONE, C. (2007). "**Assistive Wheelchair Navigation: A Cognitive View**". Em: Advanced Computational Intelligence Paradigms In Healthcare – 1, Studies In Computational Intelligence, v. 48, p. 165–187.
- CLARK, L. L. J. (1997). "**Design and Testing of a Quick-Connect Wheelchair Power Add-On Unit**". Dissertação de doutorado de filosofia em Engenharia Industrial e de Sistemas, Faculdade do Instituto Politécnico e Universidade Estadual da Virginia, Blacksburg - Estados Unidos da América.
- ESTADOS UNIDOS DA AMÉRICA. **Constituição (2006 Suplemento III)**. The Code of Laws of The United States of America. Título 29, Capítulo 31.
- EVINGER C., MANNING K. A., SIBONY P. A. (1991). "**Eyelid movements. Mechanisms and normal data**". Em: Investigative Ophthalmology & Visual Science, v. 32, n. 2, p. 387–400.
- FAVIEIRO, G. W. (2009). "**Controle de uma prótese experimental do segmento mão-braço por sinais mioelétricos e redes neurais artificiais**". Trabalho de conclusão do curso de Engenharia da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.
- FUSCO, D. A. (2010). "**Acionamento de uma cadeira de rodas através de um acelerômetro bi-axial como inclinômetro**". Trabalho de conclusão do curso de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.
- GALVÃO FILHO, T. A.; DAMASCENO, L. L. (2008). "**Programa InfoEsp: Premio Reina Sofia 2007 de Rehabilitación y de Integración**". Em: Boletin del Real Patronato Sobre Discapacidad, Ministerio de Educación, Política Social y Deporte, n. 63, p. 14–23, Madri, Espanha.
- GUYTON, A. C. (2006). "**Textbook of medical physiology**". 11. ed. Philadelphia, Elsevier Saunders, Estados Unidos da América. 1116p.
- HAN, J.; BIEN, Z. Z.,; KIM, D.; LEE, H.; KIM, J. (2003). "**Human-Machine Interface for wheelchair control with EMG and Its Evaluation**". Em: Proceedings of the 25th Annual International Conference of the IRR EMBS, p. 1602-1605.

- KHARE, V.; SANTHOSH, J.; ANAND, S.; BHATIA, M. (2010). "**Controlling Wheelchair Using Electroencephalogram**". Em: International Journal of Computer Science and Information Security, v. 8, n. 2, p. 181-187.
- KUO, C.; CHAN, Y.; CHOU, H.; SIAO, J. (2009). "**Eyeglasses Based Electrooculography Human-wheelchair Interface**". Em: Systems, Man and Cybernetics. IEEE International Conference, p. 4746-4751.
- LABCENTER (2010). Web site. Disponível em: <http://www.labcenter.com/download/prodemo_download.cfm#professional>. Acessado em: novembro de 2010.
- MACE, M.; ABDULLAH-AL-MAMUN, K.; VAIDYANATHAN, R.; WANG, S.; GUPTA, L. (2010). "**Real-time Implementation of a Non-invasive Tongue-based Human-Robot Interface**". Em: The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 5486-5491.
- MARIEB, E. N.; HOEHN, K. (2006). "**Human Anatomy & Physiology**". 7. ed. San Francisco, Banjamin Cummings Publishing Company Inc., Estados Unidos da América, 1159p.
- MUSEUM OF DISABILITY (2010), **Web Site**. Disponível em: <http://www.museumofdisability.org/>. Acessado em: novembro de 2010.
- MICROCHIP (2008). "**MCP3302/04 – 13-Bit Differential Input, Low Power A/D Converter with SPI Serial Interface**". Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/21697e.pdf>. Acessado em: novembro de 2010.
- MICROCHIP (2010). **Web Site**. Disponível em: <http://www.microchip.com>. Acessado em: novembro de 2010.
- NATIONAL SEMICONDUCTOR (1999). "**LMF100 - High Performance Dual Switched Capacitor Filter**". Disponível em: <http://www.national.com/ds/LM/LMF100.pdf>. Acessado em: novembro de 2010.
- NATIONAL SEMICONDUCTOR (2005). "**Designing Active High Speed Filters**". Disponível em: <http://www.national.com/an/OA/OA-26.pdf>. Acessado em: novembro de 2010.
- NIST (2010). National Institute of Standards and Technology. "**Assistive Technology".** Disponível em: <http://standards.gov/standards_gov/assistiveTechnology.cfm>. Acessado em: novembro de 2010.

- OONISHI, Y.; OH, S.; HORI, Y. (2008). "**New control Method for Power-assisted Wheelchair Based on Upper Extremity Movement Using Surface Myoelectric Signal**". Em: 10th IEEE International Workshop on Advanced Motion Control, p. 498-503.
- ORTOLAN, R. L. (2002). "**Estudo e Avaliação de Técnicas de Processamento do Sinal Mioelétrico para o Controle de Sistemas de Reabilitação**". Tese de Mestrado em Engenharia Elétrica Programa de Pós-Graduação em Engenharia Elétrica, Universidade de São Paulo, São Carlos, Brasil.
- PIRES, G.; CASTELO-BRANCO, M.; NUNES, U. (2008). "**Visual P300-based BCI to steer a Wheelchair: a Bayesian Approach**". Em: Engineering in Medicine and Biology Society, 2008. 30th Annual International Conference of the IEEE, p. 658-661.
- PLOTKIN, A.; SELA, L; WEISSBROD, A.; KAHANA, R.; HAVIV, L.; YESHURUN, Y.; SOROKER, N.; SOBEL, N. (2010). "**Sniffing enables communication and environmental control for the severely disabled**". Em: Proceeding of the National Academy of Sciences of the United States of America, p. 14413-14418.
- PRIDE MOBILITY (2010). **Web Site**. Disponível em: <http://www.pridemobility.com/>. Acessado em: novembro de 2010.
- RADTKE, C. A. M. (2007). "**Protótipo de um sistema de aquisição e processamento de sinais mioelétricos para caracterização da fadiga muscular**". Trabalho de conclusão do curso de Engenharia Elétrica, Universidade Luterana do Brasil, Canoas, Brasil.
- REBSAMEN, B.; BURDET, E.; GUAN, C.; ZHANG, H.; TEO, C. L.; ZENG, Q.; LAUGIER, C.; ANG JR., M. H. (2007). "**Controlling a Wheelchair Indoors Using Thought**". Em: Intelligent Systems, IEEE, v. 22, n. 2, p. 18-24.
- SPECTRUM SOFTWARE (2010). Web Site. Disponível em: <http://www.spectrum-soft.com/demoform.shtm>. Acessado em: novembro de 2010.
- STERN JR, J. T. (2003). "**Essentials of Gross Anatomy**". New York, Stony Brook University, 650p. Disponível em: <http://www.anat.stonybrook.edu/HBA531/EGA/EGA2003.html>. Acessado em: novembro de 2010.

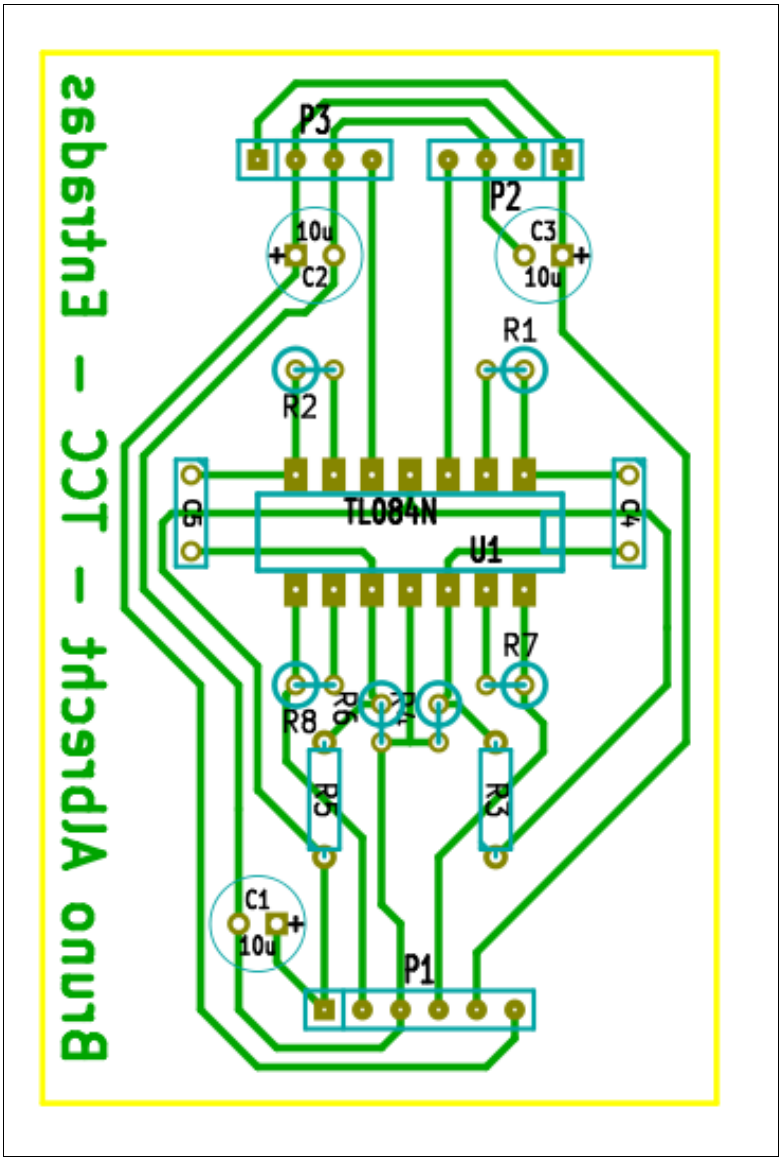
- SUNRISE MEDICAL, 2010. **Web Site**. Disponível em: <http://www.sunrisemedical.com/> Acessado em: novembro de 2010.
- TSUI, C. S. L.; JIA, P.; GAN, J. Q.; HU, H.; YUAN, K. (2007). "**EMG-based Hands-Free Wheelchair Control with EOG Attention Shift Detection**". Em: Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics, p. 1266-1271.
- VANDERWERF, F.; BRASSINGA, P.; REITS, D.; ARAMIDEH, M.; DE VISSER, B. O. (2002). "**Eyelid Movements: Behavioral Studies of Blinking in Humans Under Different Stimulus Conditions**". Em: Journal of Neurophysiology, v. 89, p. 2784-2796.
- WEI, L.; HU, H.; YUAN, K. (2009). "**Use of Forehead Bio-signals for Controlling an Intelligent Wheelchair**". Em: Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetrics, p. 108-113.
- WHEELCHAIR NETWORK, 2010. **Web Site**. Disponível em: <http://www.wheelchairnetwork.com/>. Acessado em: novembro de 2010.

# ANEXO A.1



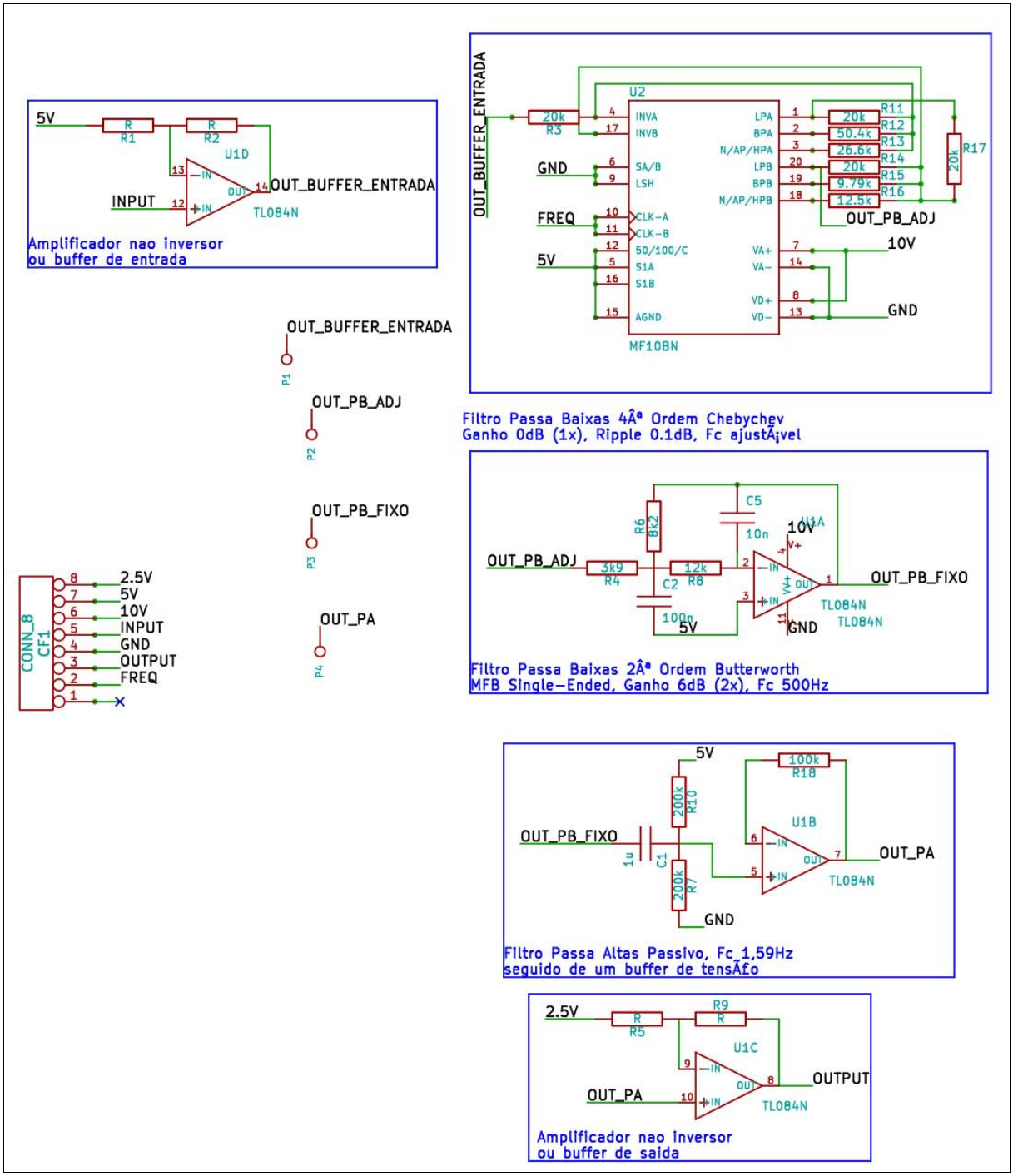

Esquemático do estágio de entrada.

**ANEXO A.2**



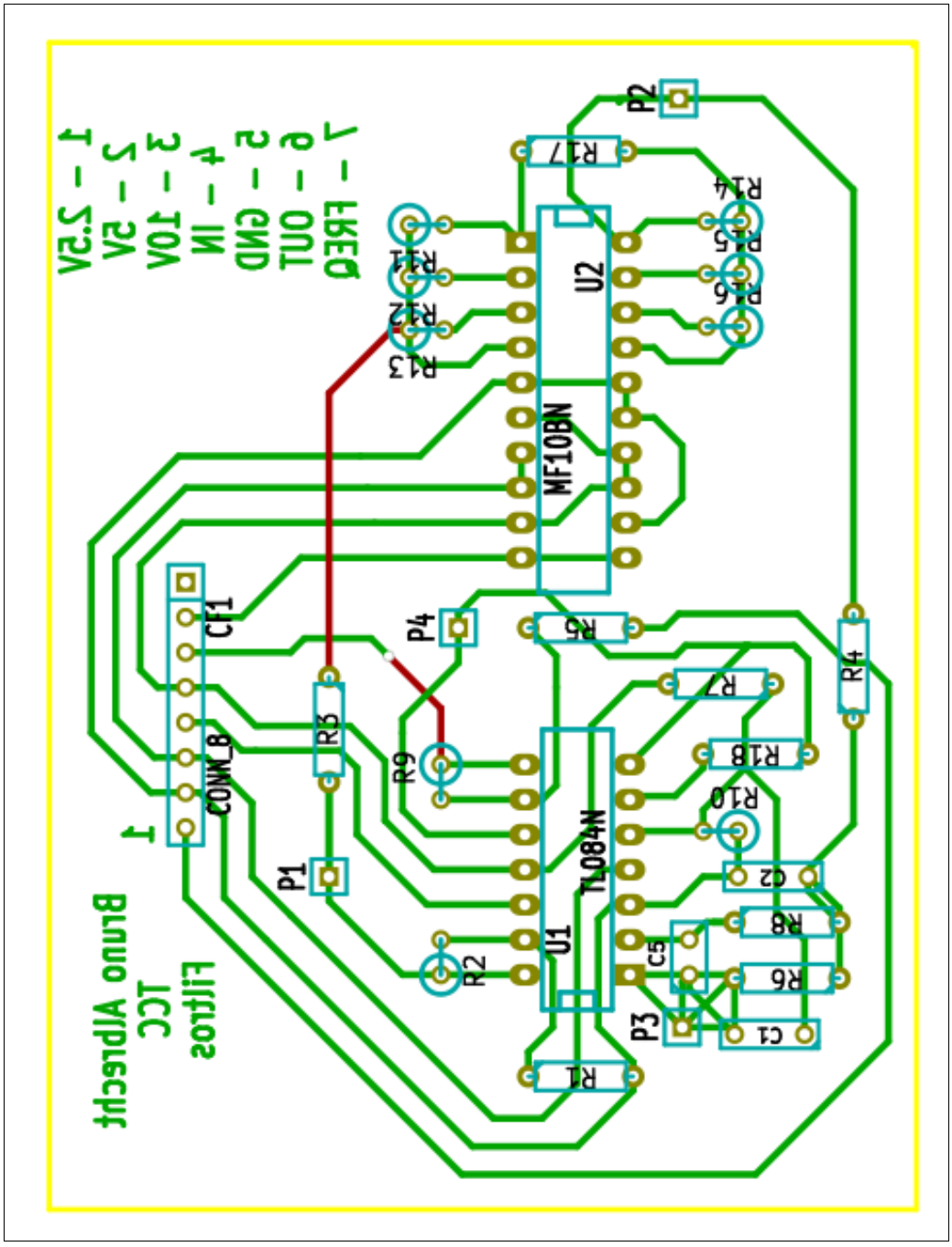

Layout do estágio de entrada.

# ANEXO A.3



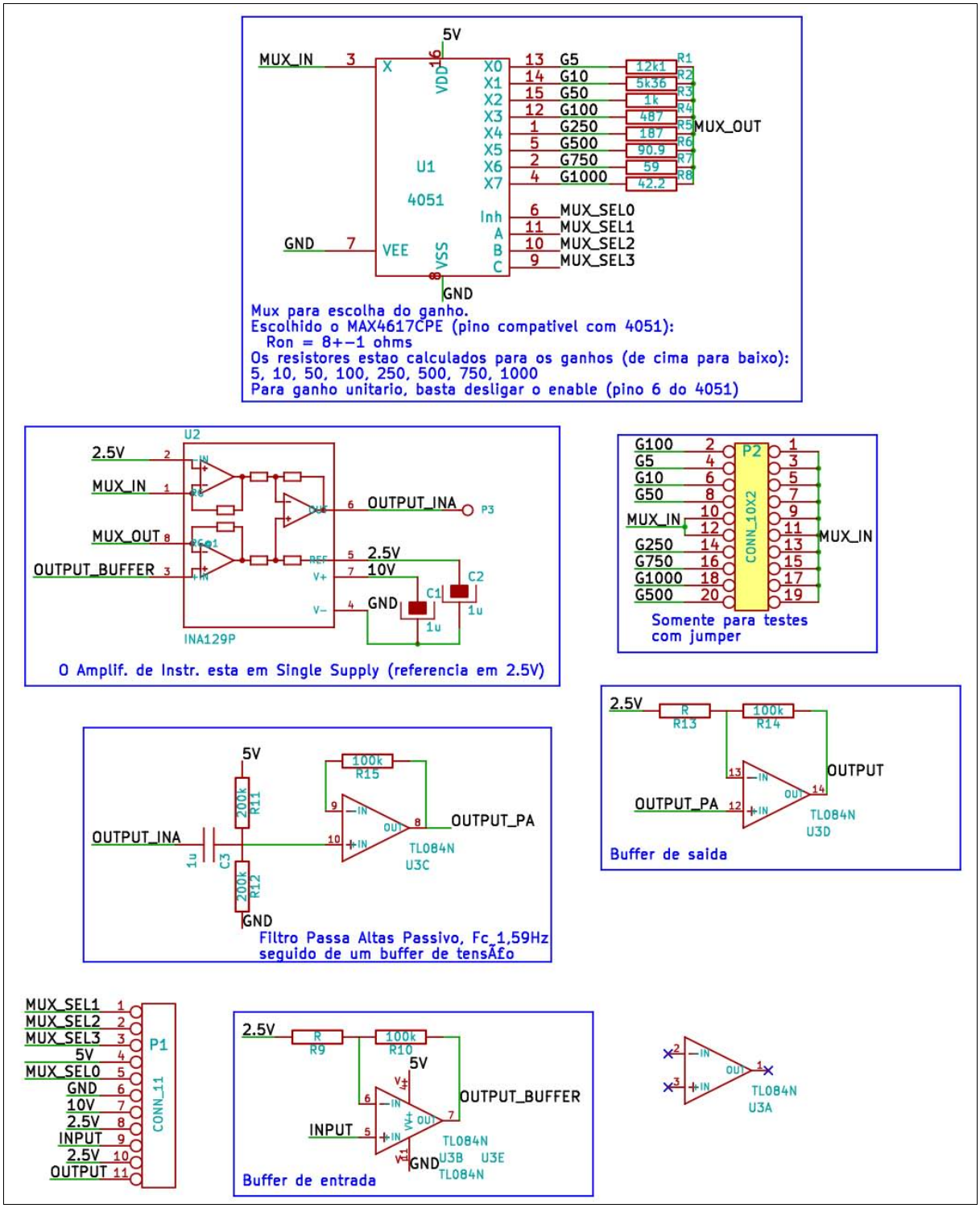

Esquemático do estágio de filtragem.

**ANEXO A.4**



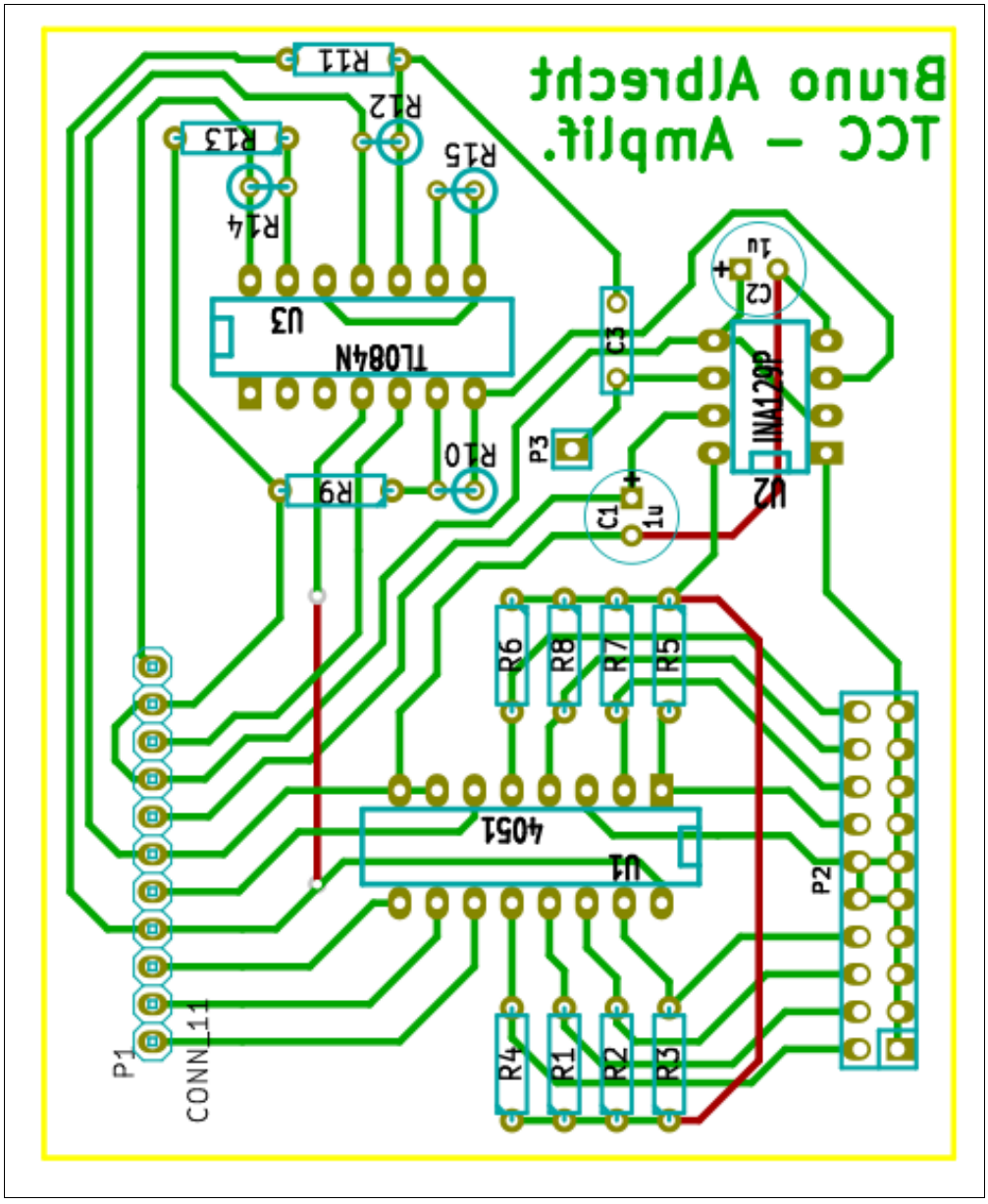

Layout do estágio de filtragem.

# ANEXO A.5



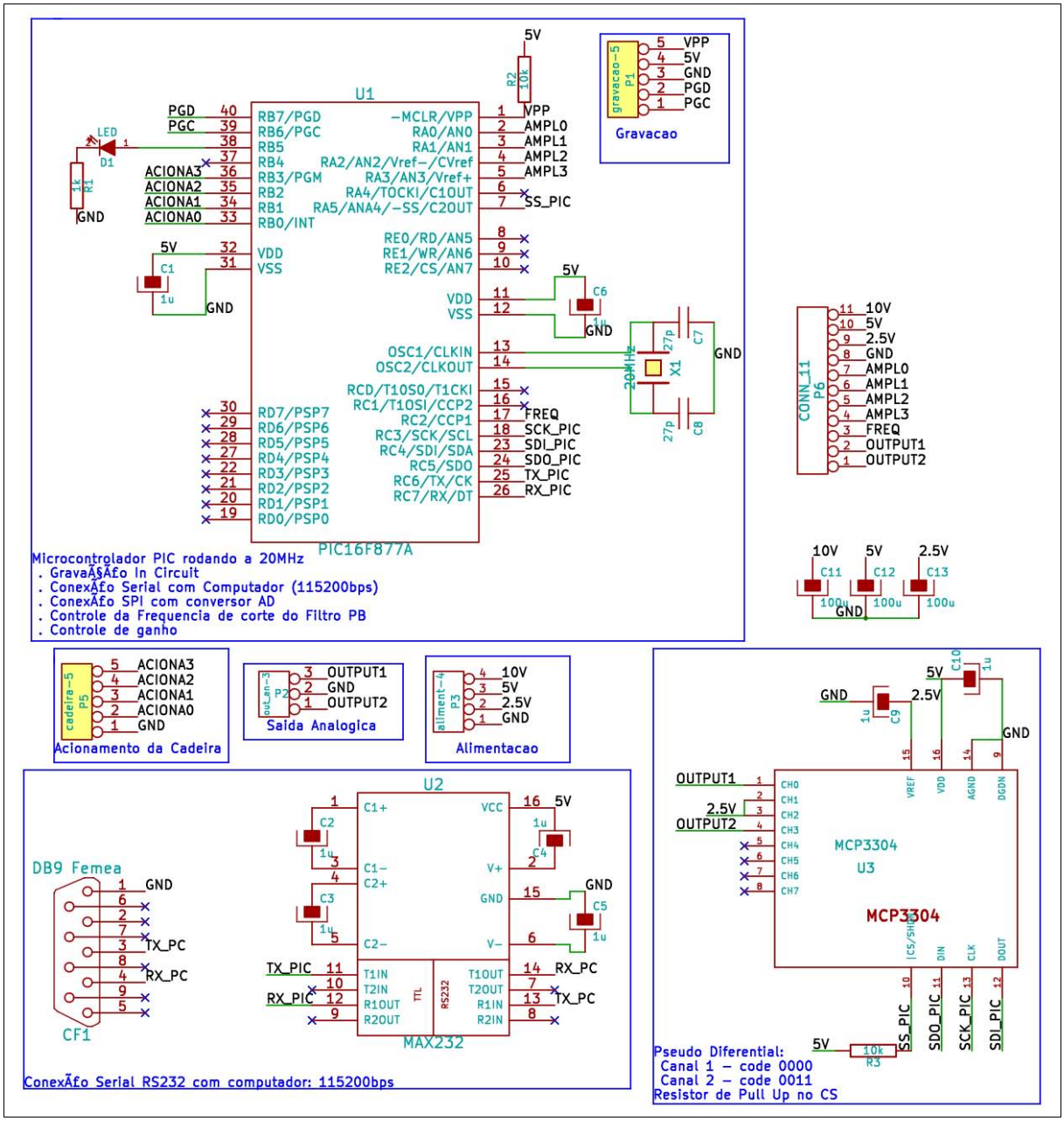

Esquemático do estágio de amplificação.

**ANEXO A.6**



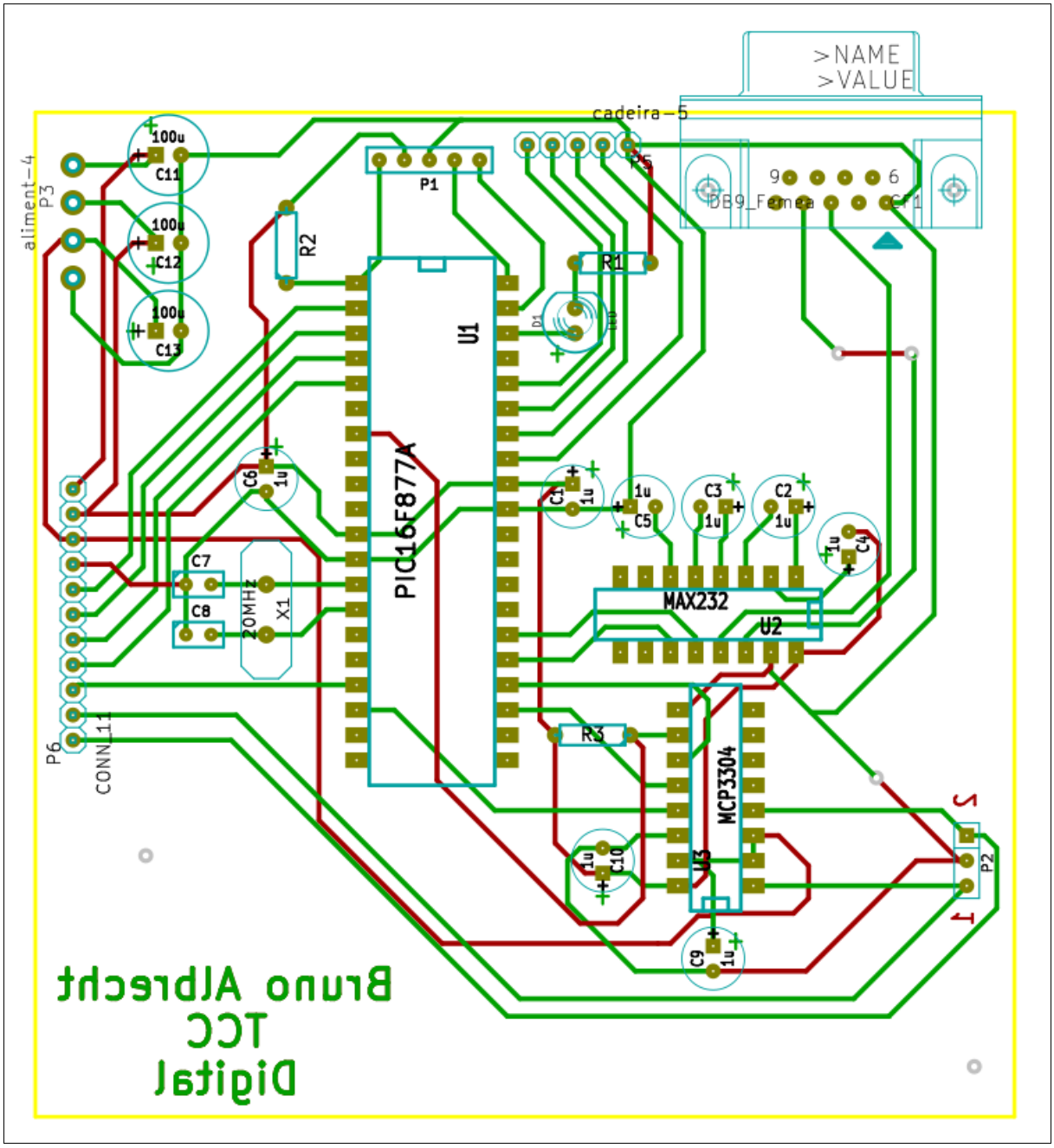

Layout do estágio de amplificação.

# ANEXO A.7



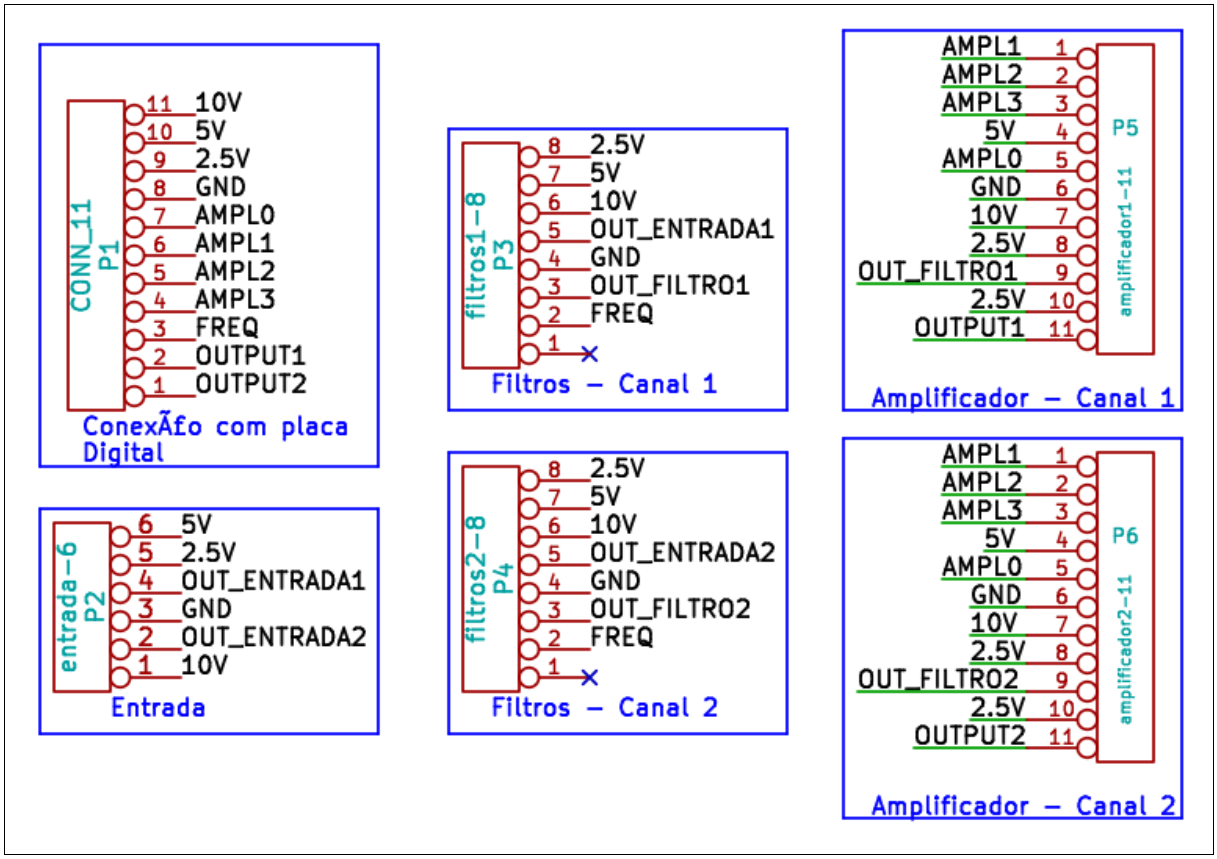

Esquemático do estágio digital.

**ANEXO A.8**



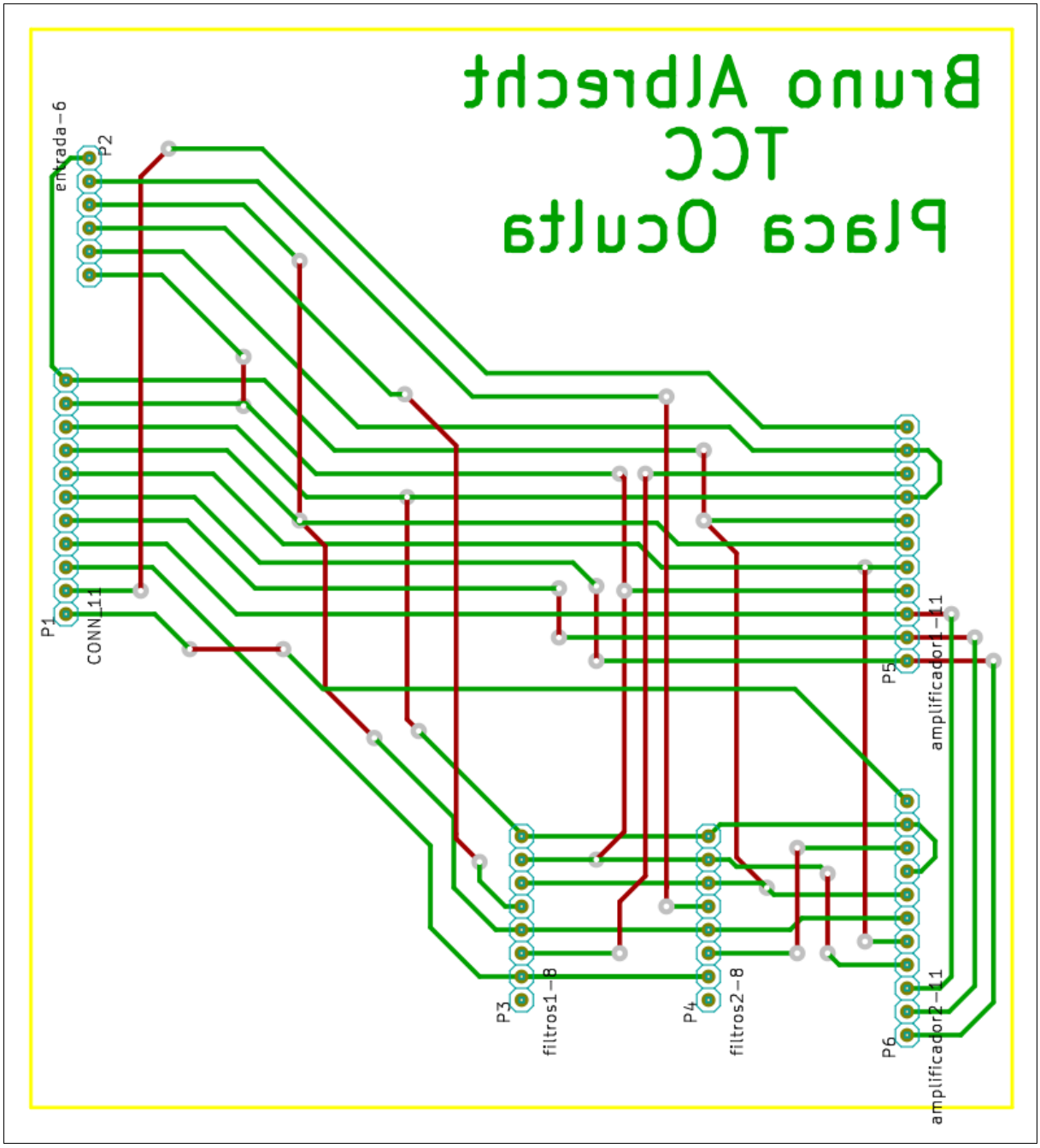

Layout do estágio digital.

## ANEXO A.9



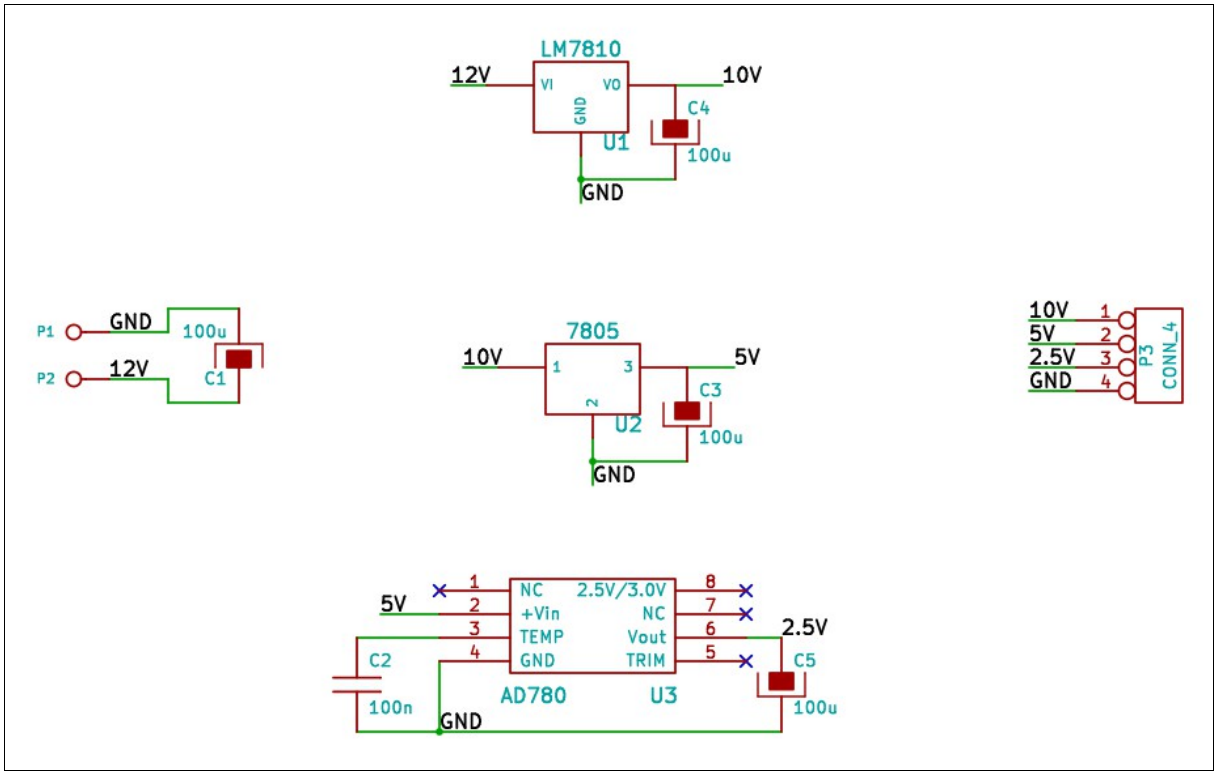

Esquemático da placa oculta.

**ANEXO A.10**



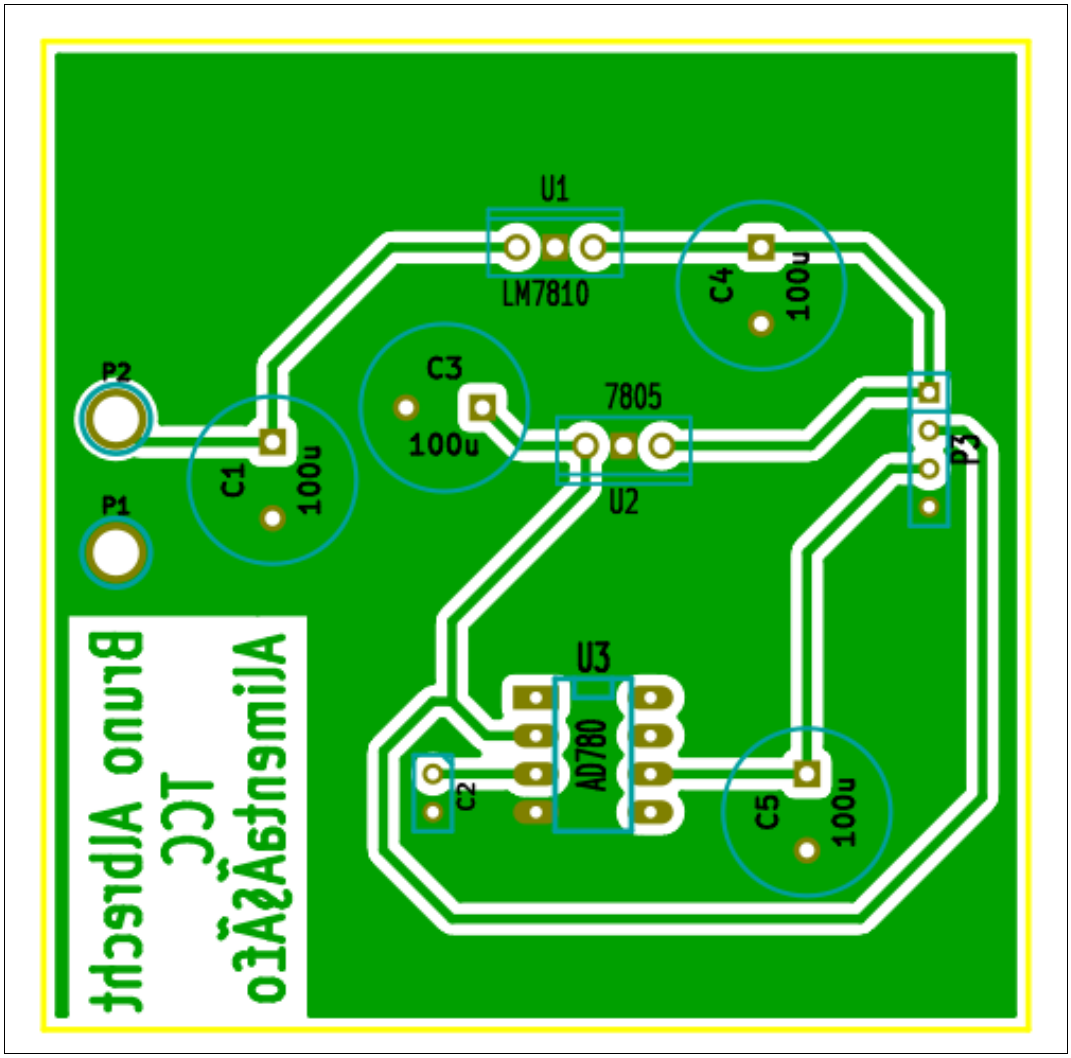

Layout da placa oculta.

## ANEXO A.11




Esquemático da placa de alimentação.

**ANEXO A.12**




Layout da placa de alimentação.

**ANEXO A.13**

Código fonte do software implementado: arquivo code.c.

```
#include "code.h"

//===== Inicializações
================================================================
==========//
void init (void) {
	// ajustar interrupções
	INTCON=0;
	PIE1=0;
	PIR1=0; //zera flags
	PIE2=0;
	PIR2=0;

	//ajustar gpio
	TRISA=0;
	TRISB=0;
	TRISC=0;
	TRISD=0;
	TRISE=0;
	PORTA=0;
	PORTB=0;
	PORTC=0;
	PORTD=0;
	PORTE=0;

	//ajustar timers
	OPTION=0;   //TMR0
	T1CON=0;    //TMR1
	TMR1L=0;
	TMR1H=0;
	T2CON=0;    //TMR2
	PR2=0xFF;
	TMR2=0;

	//ajustar Capture/Compare/PWM (por enquanto, tudo desligado)
	CCP1CON=0;
	CCPR1L=0;
	CCPR1H=0;
	CCP2CON=0;
	CCPR2L=0;
	CCPR2H=0;

	//MSSP - SPI e I2C
	SSPSTAT=0;
	SSPCON=0;
```

```
SSPCON2=0;
SSPBUF=0;
SSPADD=0;

//USART (por enquanto, tudo desligado)
TXSTA=0;
RCSTA=0;
SPBRG=0;
TXREG=0;

//Conversor A/D interno
ADCON0=0;
ADCON1=0b00000110;

//Comparador Analógico - desligado
CMCON=0b00000111;
CVRCON=0;

//variaveis globais
maq_est_le_comandos=0;
maq_est_serial_para=0;
freqc_atual=DEFAULT_FREQ;
ganho_atual=DEFAULT_GANHO;

controla_cadeira=0;
adquire_dados=0;
maq_est_spi_para=0;
maq_est_le_ad=0;
ad_dado_lido=0;
ponteiro_buf=0;
acum_ch0=0;
acum_ch1=0;
media_ch0=0;
media_ch1=0;
canal0.word=0;
canal1.word=0;
tempo=0;
piscou_ch0=PISCOU_0X;
piscou_ch1=PISCOU_0X;
limite_up_ch0=LIMITE_UP_DEFAULT;
limite_up_ch1=LIMITE_UP_DEFAULT;
limite_down_ch0=LIMITE_DOWN_DEFAULT;
limite_down_ch1=LIMITE_DOWN_DEFAULT;
tempo_ch0=0;
tempo_ch1=0;
calibracao_repouso=0;
calibracao_piscada=0;
estado_cadeira=CADEIRA_PARADA;

envia_serial=0;
envia_serial_ok=0;
maq_est_envia_serial=0;
```

```
	//tudo inicializado (desligado)
	//agora liga o que precisa:

	//serial
	initUsart();

	//Freq de corte
	initFreq(freqc_atual);

	//Ganho
	initGanho(ganho_atual);

	//conversor AD externo:
		//SPI
//	initSPI(); //é feito no initAD
		//frequencia de amostragem
//	initAD(AMOSTR_DEFAULT); //é feito somente na maq_est da serial

	//LED piscante
	TRISB &= 0b11011111;
	LED=1;
	cont_led=CONT_LED;
	OPTION &= 0b11000000; //timer0 - clk: interno (Fosc/4), prescaler: timer0
	OPTION |= 0b00000111; //timer0 - precaler: 1:256
	TMR0=TMR0_LED;
	TMR0IF=0;
	TMR0IE=1;

	//controle da cadeira de rodas
	TRISB &= 0b11110000;
	COMANDO_CADEIRA_FRENTE=0;
	COMANDO_CADEIRA_TRAS=0;
	COMANDO_CADEIRA_DIREITA=0;
	COMANDO_CADEIRA_ESQUERDA=0;

	//ganho e SS
	TRISA=0;


	//liga interrupções gerais
	PEIE=1;
	GIE=1;
}

//===== Configuração da USART
================================================================
===//

void initUsart(void) {
// Configuração geral da USART
    // Setar shift clock
        BRGH=1;      // High Baud Rate Select bit: 0 = Low speed, 1 = High speed
```

```
        SPBRG=DEFAULT_SPBRG;       //115200 bps

        // habilitando transmissao assincrona
        SYNC = 0;          // USART Mode Select bit: '0 = Assincrona', 1 = Sincrona
        TRISC&=0b00111111;
        TRISC|=0b10000000;
        // comunicacao habilitada
        SPEN = 1;          // Serial Port Enable bit: Configura RC7 e RC6 como
comunicação serial


// Configuração da transmiçao
        // Interrupção? Nao
        TXIE = 0;          // interrupcao de envio desabilitada
        TX9 = 0;                        // 8 bits
        TXEN = 1;          // Transmit Enable bit -> isso seta o TXIF (indica que o buffer
de envio esta vazio)


// Configuração da recepção
        RCIE = 1;          // interrupcao de recebimento habilitada
        RX9  = 0;          // 8 bits
        CREN = 1;          // Continuous Receive Enable bit
        RCIF=0;
}

//===== Configuração da SPI
================================================================
===//

void initSPI(void) {
//configuração geral da interface SPI

        SSPCON &= 0b11110000;
        //SSPCON |= 0b00000010; //SPI Master Mode, clock = Fosc/64 = 312,5 kHz
        SSPCON |= 0b00000001; //SPI Master Mode, clock = Fosc/16 = 1250 kHz

        SMP = 1;
        CKE = 0;
        BF=0;
        WCOL=0;

        TRISC &= 0b11010111; //SDO, SCK == output
        TRISC |= 0b00010000; //SDI == input
        TRISA = 0; //SS == output
        STATUS &= 0b10011111;
        PORTA=PORTA|LIGA_SS; //liga o SS (RA5)

        SSPIF = 0;
        SSPIE = 1;

        SSPEN = 1;
}
```

```
void stopSPI(void) {
	SSPEN=0;
	SSPCON =0;
	SSPCON2=0;
	SSPSTAT=0;
	SSPIE=0;
	SSPIF=0;
	PORTA&=DESLIGA_SS;
	RC3=0;
	RC5=0;
}

//===== Configuração da Frequencia de Corte
==========================================================
===//

void initFreq(unsigned char FREQ) {
//ajusta a frequencia

	CCP1CON=0; //desliga o PWM

switch (FREQ) {
	case FREQ_50:
		PR2=PR2_50;
		T2CKPS0=T2CKPS0_50;
		T2CKPS1=T2CKPS1_50;
		CCPR1L=CCPR1L_50;
		CCP1X=CCP1X_50;
		CCP1Y=CCP1Y_50;
		break;
	case FREQ_100:
		PR2=PR2_100;
		T2CKPS0=T2CKPS0_100;
		T2CKPS1=T2CKPS1_100;
		CCPR1L=CCPR1L_100;
		CCP1X=CCP1X_100;
		CCP1Y=CCP1Y_100;
		break;
	case FREQ_150:
		PR2=PR2_150;
		T2CKPS0=T2CKPS0_150;
		T2CKPS1=T2CKPS1_150;
		CCPR1L=CCPR1L_150;
		CCP1X=CCP1X_150;
		CCP1Y=CCP1Y_150;
		break;
	case FREQ_200:
		PR2=PR2_200;
		T2CKPS0=T2CKPS0_200;
		T2CKPS1=T2CKPS1_200;
		CCPR1L=CCPR1L_200;
```

```
        CCP1X=CCP1X_200;
        CCP1Y=CCP1Y_200;
        break;
case FREQ_250:
        PR2=PR2_250;
        T2CKPS0=T2CKPS0_250;
        T2CKPS1=T2CKPS1_250;
        CCPR1L=CCPR1L_250;
        CCP1X=CCP1X_250;
        CCP1Y=CCP1Y_250;
        break;
case FREQ_300:
        PR2=PR2_300;
        T2CKPS0=T2CKPS0_300;
        T2CKPS1=T2CKPS1_300;
        CCPR1L=CCPR1L_300;
        CCP1X=CCP1X_300;
        CCP1Y=CCP1Y_300;
        break;
case FREQ_350:
        PR2=PR2_350;
        T2CKPS0=T2CKPS0_350;
        T2CKPS1=T2CKPS1_350;
        CCPR1L=CCPR1L_350;
        CCP1X=CCP1X_350;
        CCP1Y=CCP1Y_350;
        break;
case FREQ_400:
        PR2=PR2_400;
        T2CKPS0=T2CKPS0_400;
        T2CKPS1=T2CKPS1_400;
        CCPR1L=CCPR1L_400;
        CCP1X=CCP1X_400;
        CCP1Y=CCP1Y_400;
        break;
case FREQ_450:
        PR2=PR2_450;
        T2CKPS0=T2CKPS0_450;
        T2CKPS1=T2CKPS1_450;
        CCPR1L=CCPR1L_450;
        CCP1X=CCP1X_450;
        CCP1Y=CCP1Y_450;
        break;
case FREQ_500:
        PR2=PR2_500;
        T2CKPS0=T2CKPS0_500;
        T2CKPS1=T2CKPS1_500;
        CCPR1L=CCPR1L_500;
        CCP1X=CCP1X_500;
        CCP1Y=CCP1Y_500;
        break;
default:
```

```
                PR2=PR2_DEFAULT;
                T2CKPS0=T2CKPS0_DEFAULT;
                T2CKPS1=T2CKPS1_DEFAULT;
                CCPR1L=CCPR1L_DEFAULT;
                CCP1X=CCP1X_DEFAULT;
                CCP1Y=CCP1Y_DEFAULT;
                break;
}
//ajusta o TRISC<2>
        TRISC&=0b11111011;

//liga o TMR2
        TMR2ON=1;

//liga o módulo CCP1 em PWM
        CCP1M3=1;
        CCP1M2=1;

//atualiza variavel global
        freqc_atual=FREQ;
}

//===== Configuração do ganho
================================================================
===//

void initGanho (unsigned char GANHO){
        unsigned char aux;

        aux=GANHO;
        if (aux>8) {
                aux=DEFAULT_GANHO;
                GANHO=DEFAULT_GANHO;
        }
        if (!aux) //ganho unitário (desliga o enable do mux)
                aux=1;
        else {
                aux--;
                aux<<=1;
        }
        PORTA&=0xF0;
        PORTA|=(aux&0x0F);
        ganho_atual=GANHO;
}

//===== Configuração do Timer para o AD
================================================================
==//

void initAD (unsigned char freq_amostragem) {

        T1CON=CONFIG_T1CON;
```

```
	switch (freq_amostragem) {
		case AMOSTR_1KHZ:
			TMR1H=TMR1H_1KHZ;
			TMR1L=TMR1L_1KHZ;
			tmr1h=TMR1H_1KHZ;
			tmr1l=TMR1L_1KHZ;
			break;
		case AMOSTR_2KHZ:
			TMR1H=TMR1H_2KHZ;
			TMR1L=TMR1L_2KHZ;
			tmr1h=TMR1H_2KHZ;
			tmr1l=TMR1L_2KHZ;
			break;
		case AMOSTR_2_5KHZ:
			TMR1H=TMR1H_2_5KHZ;
			TMR1L=TMR1L_2_5KHZ;
			tmr1h=TMR1H_2_5KHZ;
			tmr1l=TMR1L_2_5KHZ;
			break;
		case AMOSTR_4KHZ:
			TMR1H=TMR1H_4KHZ;
			TMR1L=TMR1L_4KHZ;
			tmr1h=TMR1H_4KHZ;
			tmr1l=TMR1L_4KHZ;
			break;
		default:
			TMR1H=TMR1H_DEFAULT;
			TMR1L=TMR1L_DEFAULT;
			tmr1h=TMR1H_DEFAULT;
			tmr1l=TMR1L_DEFAULT;
			break;
	}
	TMR1IF=0;
	TMR1IE=1;
	TMR1ON=1;
	initSPI();
	maq_est_spi_para=1; //para esperar a primeira interrupt do timer...
}

void stopAD(void) {
	stopSPI();
	TMR1ON=0;
	TMR1IE=0;
	TMR1IF=0;
	T1CON=0;
	TMR1H=0;
	TMR1L=0;
}

//===== Mostra tela de boas vindas
================================================================
==//
```

```
void ShowWelcome(void){
	unsigned char tam_tela;

//	TXIE=1;
	tam_tela=TAM_TELA_WELCOME1+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome1[TAM_TELA_WELCOME1-tam_tela
];TXREG=aux;
	}
	tam_tela=TAM_TELA_WELCOME1_1+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome1_1[TAM_TELA_WELCOME1_1-tam_tela
];TXREG=aux;
	}
	tam_tela=TAM_TELA_WELCOME1_2+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome1_2[TAM_TELA_WELCOME1_2-tam_tela
];TXREG=aux;
	}
	tam_tela=TAM_TELA_WELCOME1_3+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome1_3[TAM_TELA_WELCOME1_3-tam_tela
];TXREG=aux;
	}

	tam_tela=TAM_TELA_WELCOME2+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome2[TAM_TELA_WELCOME2-tam_tela
];TXREG=aux;
	}
	tam_tela=TAM_TELA_WELCOME2_1+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome2_1[TAM_TELA_WELCOME2_1-tam_tela
];TXREG=aux;
	}
		tam_tela=TAM_TELA_AD2+1;
		while (tam_tela--) {
			while (!TXIF);aux=ad2[TAM_TELA_AD2-tam_tela];TXREG=aux;
		}
		//mostra o limite_down_ch0 em hexa (12bits, 3 nibbles)
		aux=(unsigned char)(limite_down_ch0>>8);
		aux&=0x0F;
		if (aux<10) aux=aux + '0';
		else aux=aux - 10 + 'A';
		while(!TXIF); TXREG=aux;
		aux=(unsigned char)(limite_down_ch0>>4);
		aux&=0x0F;
		if (aux<10) aux=aux + '0';
		else aux=aux - 10 + 'A';
		while(!TXIF); TXREG=aux;
		aux=(unsigned char)limite_down_ch0;
		aux&=0x0F;
```

```
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;

                tam_tela=TAM_TELA_AD3+1;
                while (tam_tela--) {
                        while (!TXIF);aux=ad3[TAM_TELA_AD3-tam_tela];TXREG=aux;
                }
                //mostra o limite_down_ch1 em hexa (12bits, 3 nibbles)
                aux=(unsigned char)(limite_down_ch1>>8);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)(limite_down_ch1>>4);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)limite_down_ch1;
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;

                tam_tela=TAM_TELA_AD6+1;
                while (tam_tela--) {
                        while (!TXIF);aux=ad6[TAM_TELA_AD6-tam_tela];TXREG=aux;
                }
        tam_tela=TAM_TELA_WELCOME2_2+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_2[TAM_TELA_WELCOME2_2-tam_tela
];TXREG=aux;
        }
                tam_tela=TAM_TELA_AD4+1;
                while (tam_tela--) {
                        while (!TXIF);aux=ad4[TAM_TELA_AD4-tam_tela];TXREG=aux;
                }
                //mostra o limite_up_ch0 em hexa (12bits, 3 nibbles)
                aux=(unsigned char)(limite_up_ch0>>8);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)(limite_up_ch0>>4);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)limite_up_ch0;
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
```

```
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;

                tam_tela=TAM_TELA_AD5+1;
                while (tam_tela--) {
                        while (!TXIF);aux=ad5[TAM_TELA_AD5-tam_tela];TXREG=aux;
                }
                //mostra o limite_up_ch1 em hexa (12bits, 3 nibbles)
                aux=(unsigned char)(limite_up_ch1>>8);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)(limite_up_ch1>>4);
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;
                aux=(unsigned char)limite_up_ch1;
                aux&=0x0F;
                if (aux<10) aux=aux + '0';
                else aux=aux - 10 + 'A';
                while(!TXIF); TXREG=aux;

                tam_tela=TAM_TELA_AD6+1;
                while (tam_tela--) {
                        while (!TXIF);aux=ad6[TAM_TELA_AD6-tam_tela];TXREG=aux;
                }
        tam_tela=TAM_TELA_WELCOME2_3+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_3[TAM_TELA_WELCOME2_3-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME2_4+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_4[TAM_TELA_WELCOME2_4-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME2_5+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_5[TAM_TELA_WELCOME2_5-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME2_6+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_6[TAM_TELA_WELCOME2_6-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME2_7+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_7[TAM_TELA_WELCOME2_7-tam_tela
];TXREG=aux;
```

```
        }
        tam_tela=TAM_TELA_WELCOME2_8+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_8[TAM_TELA_WELCOME2_8-tam_tela
];TXREG=aux;
        }
        //mostra o código do ganho atual
        while (!TXIF);TXREG=ganho_atual+'0';
        tam_tela=TAM_TELA_WELCOME2_9+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_9[TAM_TELA_WELCOME2_9-tam_tela
];TXREG=aux;
        }
        //mostra o código da freq de corte atual
        while (!TXIF);TXREG=freqc_atual+'0';
        tam_tela=TAM_TELA_WELCOME2_10+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome2_10[TAM_TELA_WELCOME2_10-
tam_tela];TXREG=aux;
        }

        tam_tela=TAM_TELA_WELCOME3+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome3[TAM_TELA_WELCOME3-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME3_1+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome3_1[TAM_TELA_WELCOME3_1-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME3_2+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome3_2[TAM_TELA_WELCOME3_2-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME3_3+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome3_3[TAM_TELA_WELCOME3_3-tam_tela
];TXREG=aux;
        }

        tam_tela=TAM_TELA_WELCOME4+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome4[TAM_TELA_WELCOME4-tam_tela
];TXREG=aux;
        }
        tam_tela=TAM_TELA_WELCOME4_1+1;
        while (tam_tela--) {
                while (!TXIF);aux=welcome4_1[TAM_TELA_WELCOME4_1-tam_tela
];TXREG=aux;
        }
```

```
	tam_tela=TAM_TELA_WELCOME4_2+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome4_2[TAM_TELA_WELCOME4_2-tam_tela
];TXREG=aux;
	}
	tam_tela=TAM_TELA_WELCOME4_3+1;
	while (tam_tela--) {
		while (!TXIF);aux=welcome4_3[TAM_TELA_WELCOME4_3-tam_tela
];TXREG=aux;
	}

	return;
}

//===== Mostra dados que serão usados no controle da cadeira
==============================//
void ShowAD(void) {
	unsigned char tam_tela;

	tam_tela=TAM_TELA_AD1+1;
	while (tam_tela--) {
		while (!TXIF);aux=ad1[TAM_TELA_AD1-tam_tela];TXREG=aux;
	}
	tam_tela=TAM_TELA_AD2+1;
	while (tam_tela--) {
		while (!TXIF);aux=ad2[TAM_TELA_AD2-tam_tela];TXREG=aux;
	}
	//mostra o limite_down em hexa (12bits, 3 nibbles)
	aux=(unsigned char)(limite_down_ch0>>8);
	aux&=0x0F;
	if (aux<10) aux=aux + '0';
	else aux=aux - 10 + 'A';
	while(!TXIF); TXREG=aux;
	aux=(unsigned char)(limite_down_ch0>>4);
	aux&=0x0F;
	if (aux<10) aux=aux + '0';
	else aux=aux - 10 + 'A';
	while(!TXIF); TXREG=aux;
	aux=(unsigned char)limite_down_ch0;
	aux&=0x0F;
	if (aux<10) aux=aux + '0';
	else aux=aux - 10 + 'A';
	while(!TXIF); TXREG=aux;

	tam_tela=TAM_TELA_AD3+1;
	while (tam_tela--) {
		while (!TXIF);aux=ad3[TAM_TELA_AD3-tam_tela];TXREG=aux;
	}
	//mostra o limite_down em hexa (12bits, 3 nibbles)
	aux=(unsigned char)(limite_down_ch0>>8);
	aux&=0x0F;
	if (aux<10) aux=aux + '0';
```

```
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;
aux=(unsigned char)(limite_down_ch0>>4);
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;
aux=(unsigned char)limite_down_ch0;
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;

tam_tela=TAM_TELA_AD6+1;
while (tam_tela--) {
        while (!TXIF);aux=ad6[TAM_TELA_AD6-tam_tela];TXREG=aux;
}
tam_tela=TAM_TELA_AD4+1;
while (tam_tela--) {
        while (!TXIF);aux=ad4[TAM_TELA_AD4-tam_tela];TXREG=aux;
}
//mostra o limite_up em hexa (12bits, 3 nibbles)
aux=(unsigned char)(limite_up_ch0>>8);
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;
aux=(unsigned char)(limite_up_ch0>>4);
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;
aux=(unsigned char)limite_up_ch0;
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;

tam_tela=TAM_TELA_AD5+1;
while (tam_tela--) {
        while (!TXIF);aux=ad5[TAM_TELA_AD5-tam_tela];TXREG=aux;
}
//mostra o limite_up em hexa (12bits, 3 nibbles)
aux=(unsigned char)(limite_up_ch1>>8);
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
while(!TXIF); TXREG=aux;
aux=(unsigned char)(limite_up_ch1>>4);
aux&=0x0F;
if (aux<10) aux=aux + '0';
else aux=aux - 10 + 'A';
```

```
        while(!TXIF); TXREG=aux;
        aux=(unsigned char)limite_up_ch1;
        aux&=0x0F;
        if (aux<10) aux=aux + '0';
        else aux=aux - 10 + 'A';
        while(!TXIF); TXREG=aux;

        tam_tela=TAM_TELA_AD6+1;
        while (tam_tela--) {
                while (!TXIF);aux=ad6[TAM_TELA_AD6-tam_tela];TXREG=aux;
        }
}

//===== Funções
================================================================
================//

void MaqEstLeComandos(void) {
        unsigned char index_tela;
        unsigned char novo;

if (!maq_est_serial_para) {
        switch (maq_est_le_comandos) {
                case 0:
                        index_tela=TAM_TELA_GANHO+1;
                        while (index_tela--) {
                                while (!TXIF);
                                aux=novo_ganho[TAM_TELA_GANHO-index_tela];
                                TXREG=aux;
                        }
                        while(!TXIF);
                        TXREG=ganho_atual+'0';
                        index_tela=TAM_TELA_FECHA+1;
                        while (index_tela--) {
                                while (!TXIF);
                                aux=fecha[TAM_TELA_FECHA-index_tela];
                                TXREG=aux;
                        }
                        maq_est_le_comandos++;
                        maq_est_serial_para=1;
                        break;

                case 1:
                        if ((buff>='0')&&(buff<='9')) {
                                while (!TXIF);
                                TXREG=buff;
                                novo=buff-'0';
                                if (novo!=ganho_atual)
                                        initGanho(novo);
                                index_tela=TAM_TELA_FREQC+1;
                                while (index_tela--) {
                                        while (!TXIF);
```

```
                        aux=nova_freqc[TAM_TELA_FREQC-index_tela];
                        TXREG=aux;
                }
                while(!TXIF);
                TXREG=freqc_atual+'0';
                index_tela=TAM_TELA_FECHA+1;
                while (index_tela--) {
                        while (!TXIF);
                        aux=fecha[TAM_TELA_FECHA-index_tela];
                        TXREG=aux;
                }
                maq_est_le_comandos++;
                maq_est_serial_para=1;
        } else maq_est_le_comandos=3; //é algum comando
        break;

case 2:
        if ((buff>='0')&&(buff<='9')) {
                while (!TXIF);
                TXREG=buff;
                novo=buff-'0';
                if (novo!=freqc_atual)
                        initFreq(novo);
                index_tela=TAM_TELA_GANHO_ATUAL+1;
                while (index_tela--) {
                        while (!TXIF);

aux=tela_ganho_atual[TAM_TELA_GANHO_ATUAL-index_tela];
                        TXREG=aux;
                }
                while (!TXIF);
                TXREG=ganho_atual+'0';
                index_tela=TAM_TELA_FREQC_ATUAL+1;
                while (index_tela--) {
                        while (!TXIF);

aux=tela_freqc_atual[TAM_TELA_FREQC_ATUAL-index_tela];
                        TXREG=aux;
                }
                while (!TXIF);
                TXREG=freqc_atual+'0';
                while (!TXIF);
                TXREG='\r';
                while (!TXIF);
                TXREG='\n';
                maq_est_le_comandos=0;
        } else maq_est_le_comandos=3;
        break;

case 3:
        if (((buff == 'h')||(buff == 'H'))&&(!controla_cadeira)) {
                while (!TXIF); TXREG='\r';
```

```
                        while (!TXIF); TXREG='\n';
                        ShowWelcome();
                        maq_est_le_comandos=0;
                } else if ((buff == 'c')||(buff == 'C')) {
                        if (!controla_cadeira) {
                                controla_cadeira=1;
                                ShowAD();
                                initAD(AMOSTR_DEFAULT);
                                maq_est_serial_para=1;
                        } else {
                                stopAD();
                                controla_cadeira=0;
                                envia_serial_ok=0;
                                while (!TXIF); TXREG='\r';
                                while (!TXIF); TXREG='\n';
                                ShowWelcome();
                                maq_est_le_comandos=0;
                        }
                } else if ((buff == 'e')||(buff == 'E')) {
                        if (!envia_serial_ok) {
                                envia_serial_ok=1;
                                if (!controla_cadeira) {
                                        controla_cadeira=1;
                                        ShowAD();
                                        initAD(AMOSTR_DEFAULT);
                                } else ;
                                maq_est_serial_para=1;
                        } else {
                                stopAD();
                                envia_serial_ok=0;
                                controla_cadeira=0;
                                while (!TXIF); TXREG='\r';
                                while (!TXIF); TXREG='\n';
                                ShowWelcome();
                                maq_est_le_comandos=0;
                        }
                } else if ((buff == 'z')||(buff == 'Z')) {
                        calibracao_repouso=1;
                        maq_est_serial_para=1;
                } else if ((buff == 'x')||(buff == 'X')) {
                        calibracao_piscada=1;
                        maq_est_serial_para=1;
                }
                break;

        default:
                maq_est_le_comandos=0;
                break;
        }
}
}
```

```
void MaqEstCalibracaoRepouso(void) {
        unsigned char tam_tela;
if (calibracao_repouso) {
        switch (maq_est_calib) {
                case 0:
                        tam_tela=TAM_TELA_CALIB_REPOUSO+1;
                        while (tam_tela--) {
                                while (!TXIF
);aux=calib_repouso[TAM_TELA_CALIB_REPOUSO-tam_tela];TXREG=aux;
                        }
                        initAD(AMOSTR_DEFAULT);
                        adquire_dados=1;
                        tempo=0;
                        limite_down_ch0=0;
                        limite_down_ch1=0;
                        limite_ch0_aux=0;
                        limite_ch1_aux=0;
                        maq_est_calib++;
                        break;
                case 1:
                        if (ad_dado_lido==2) {
                                tempo++;
                                if (tempo > TEMPO_CALIBRACAO_REPOUSO) {
                                        tempo=0;
                                        maq_est_calib++;
                                } else ;
                                ad_dado_lido=0;
                        } else ;
                        break;
                case 2:
                        if (ad_dado_lido==2) {
                                if (media_ch0>limite_ch0_aux)
                                        limite_ch0_aux=media_ch0;
                                if (media_ch1>limite_ch1_aux)
                                        limite_ch1_aux=media_ch1;
                                tempo++;
                                if (tempo>TEMPO_CALIBRACAO_REPOUSO) {
                                        adquire_dados=0;
                                        stopAD();
                                        maq_est_calib++;
                                }
                                ad_dado_lido=0;
                        }
                        break;
                case 3:
                        limite_down_ch0=limite_ch0_aux;
                        limite_down_ch1=limite_ch1_aux;
                        ShowWelcome();
                        maq_est_le_comandos=0;
                        maq_est_serial_para=0;
                        maq_est_calib=0;
                        calibracao_repouso=0;
```

```
                        break;
                default:
                        maq_est_calib=0;
                        break;
        }



}
}

void MaqEstCalibracaoPiscada(void) {
        unsigned char tam_tela;
if (calibracao_piscada) {
        switch (maq_est_calib) {
                case 0:
                        tam_tela=TAM_TELA_CALIB_PISCADA+1;
                        while (tam_tela--) {
                                while (!TXIF
);aux=calib_piscada[TAM_TELA_CALIB_PISCADA-tam_tela];TXREG=aux;
                        }
                        initAD(AMOSTR_DEFAULT);
                        adquire_dados=1;
                        tempo=0;
                        limite_up_ch0=0;
                        limite_up_ch1=0;
                        limite_ch0_aux=0;
                        limite_ch1_aux=0;
                        maq_est_calib++;
                        break;
                case 1:
                        if (ad_dado_lido==2) {
                                tempo++;
                                if (tempo > TEMPO_CALIBRACAO_REPOUSO) {
                                        tempo=0;
                                        maq_est_calib++;
                                } else ;
                                ad_dado_lido=0;
                        } else ;
                        break;
                case 2:
                        if (ad_dado_lido==2) {
                                if (media_ch0>limite_ch0_aux)
                                        limite_ch0_aux=media_ch0;
                                if (media_ch1>limite_ch1_aux)
                                        limite_ch1_aux=media_ch1;
                                tempo++;
                                if (tempo>TEMPO_CALIBRACAO_PISCADA) {
                                        adquire_dados=0;
                                        stopAD();
                                        maq_est_calib++;
                                }
```

```
                        ad_dado_lido=0;
                }
                break;

        case 3:
                limite_ch0_aux/=100;
                limite_ch0_aux*=FATOR_CALIBRACAO_PISCADA;
                limite_ch1_aux/=100;
                limite_ch1_aux*=FATOR_CALIBRACAO_PISCADA;
                limite_up_ch0=limite_ch0_aux;
                limite_up_ch1=limite_ch1_aux;
                ShowWelcome();
                maq_est_le_comandos=0;
                maq_est_serial_para=0;
                maq_est_calib=0;
                calibracao_piscada=0;
                break;
        default:
                maq_est_calib=0;
                break;
        }
}
}

void MaqEstLeAD(void) {
        unsigned char dummy;

if ((!maq_est_spi_para)&&((controla_cadeira)||(adquire_dados))) {
        switch (maq_est_le_ad) {
                case 0: //primeiro, baixa o SS, e manda o primeiro byte de config do
canal0
                        PORTA&=DESLIGA_SS;
                        NOP();
                        SSPBUF=CANAL0_A;
                        maq_est_le_ad++;
                        maq_est_spi_para=1;
                        break;


                case 1: //terminou de enviar o primeiro byte, recebeu um dummy byte,
envia o proximo byte de config do canal0
                        if (BF) {
                                dummy=SSPBUF;
                                SSPBUF=CANAL0_B;
                                maq_est_le_ad++;
                                maq_est_spi_para=1;
                        }
                        break;

                case 2: //terminou de enviar a config do canal0, deve receber: 0b??
0SDDDD, deve enviar byte dummy
                // ?=dummy, 0=0, S=sinal do dado, D=dado
```

```
                    if (BF) {
                        canal0.byte[1]=SSPBUF;
                        //canal0.byte[1]&=MASCARA_DADOS;
                        SSPBUF=DUMMY;
                        maq_est_le_ad++;
                        maq_est_spi_para=1;
                    }
                    break;

                case 3: //terminou de enviar o byte dummy, deve receber:
0bDDDDDDDD, deve subir o SS, esperar 475 ns, baixar e mandar primeiro byte de
config do canal1
                    if (BF) {
                        canal0.byte[0]=SSPBUF;
                        PORTA|=LIGA_SS;
                        //cada instrução é 200ns, então espera 3 instruções pra
baixar o SS
                        NOP();
                        NOP();
                        NOP();
                        PORTA&=DESLIGA_SS;
                        NOP();
                        SSPBUF=CANAL1_A;
                        maq_est_le_ad++;
                        maq_est_spi_para=1;
                    }
                    break;

                case 4: //terminou de enviar o primeiro byte, recebeu um dummy byte,
envia o proximo byte de config do canal1
                    if (BF) {
                        dummy=SSPBUF;
                        SSPBUF=CANAL1_B;
                        maq_est_le_ad++;
                        maq_est_spi_para=1;
                    }
                    break;

                case 5: //terminou de enviar a config do canal0, deve receber: 0b??
0SDDDD, deve enviar byte dummy
                // ?=dummy, 0=0, S=sinal do dado, D=dado
                    if (BF) {
                        canal1.byte[1]=SSPBUF;
                        //canal1.byte[1]&=MASCARA_DADOS;
                        SSPBUF=DUMMY;
                        maq_est_le_ad++;
                        maq_est_spi_para=1;
                    }
                    break;
```

```
                case 6: //terminou de enviar o byte dummy, deve receber:
0bDDDDDDDD, deve subir o SS, avisar que recebeu o dado e aguardar o tempo
necessário para a próxima conversão.
                        if (BF) {
                                canal1.byte[0]=SSPBUF;
                                PORTA|=LIGA_SS;
                                ad_dado_lido=1;;
                                maq_est_spi_para=1;
                                maq_est_le_ad=0;
                        }
                        break;

        }
}


}

void MaqEstEnviaSerial(void) {
if ((envia_serial)&&(envia_serial_ok)) {
        switch (maq_est_envia_serial) {
                case 0:
                        if (TXIF) {
                                aux=(unsigned char)(dado_ch0>>8);
                                aux=aux&0x0F;
                                while(!TXIF);TXREG=aux;
                                aux=(unsigned char)(dado_ch0);
                                aux=aux&0x0F;
                                while(!TXIF); TXREG=aux;
                                maq_est_envia_serial++;
                        }
                        break;
                case 1:
                        if (TXIF) {
                                aux=(unsigned char)(dado_ch1>>8);
                                aux=aux&0x0F;
                                while(!TXIF);TXREG=aux;
                                aux=(unsigned char)(dado_ch1);
                                aux=aux&0x0F;
                                while(!TXIF); TXREG=aux;
                                maq_est_envia_serial++;
                        }
                        break;
                case 2:
                        if (TXIF) {
                                while(!TXIF);TXREG=estado_cadeira+'0';
                                maq_est_envia_serial=0;
                                while(!TXIF);TXREG=' ';
                                envia_serial=0;
                        }
                        break;
```

```
		default:
			maq_est_envia_serial=0;
			envia_serial=0;
			break;
	}
}
}

void MaqEstComandaCadeira(void) {
if ((controla_cadeira)&&(ad_dado_lido==2)) {
		//agora vem a magica...
		tempo++;
		tempo_ch0++;
		tempo_ch1++;
		if (media_ch0>limite_up_ch0) { //está piscando!
			tempo_ch0=0;
			if ((piscou_ch0==PISCOU_0X)||
(piscou_ch0==PISCOU_1X_OLHO_FECHADO)) { //nao tinha piscado ainda ou
ainda está piscando pela primeira vez...zera o tempo
				piscou_ch0=PISCOU_1X_OLHO_FECHADO;
				if (piscou_ch1==PISCOU_1X_OLHO_FECHADO) {
					sobreposicao=1; //se o outro olho tb está fechado,
entao os dois estão piscando juntos
					tempo=0; //se os dois estao piscando junto e é a
primeira piscada, entao pode zerar o tempo
				} else if (piscou_ch1==PISCOU_0X) { //o outro olho nao
piscou ainda
					sobreposicao=0;
					tempo=0;
				} else sobreposicao=0;
			} else if (piscou_ch0==PISCOU_1X) { // ja tinha piscado uma
vez...
				piscou_ch0=PISCOU_2X_OLHO_FECHADO;
				//tempo++;
			} else if (piscou_ch0==PISCOU_2X) { //ja tinha piscado 2X

	piscou_ch0=PISCOU_MUITAS_VEZES_OLHO_FECHADO;
				//tempo++;
			} else { //ja piscou mais de 2X ou ainda está piscando...
				//tempo++;
			}
		} else {
			if ((piscou_ch0==PISCOU_1X_OLHO_FECHADO)
&&(tempo_ch0>=DELAY_OLHO_FECHADO)) piscou_ch0=PISCOU_1X;
			else if ((piscou_ch0==PISCOU_2X_OLHO_FECHADO)
&&(tempo_ch0>=DELAY_OLHO_FECHADO)) piscou_ch0=PISCOU_2X;
			else if
((piscou_ch0==PISCOU_MUITAS_VEZES_OLHO_FECHADO)
&&(tempo_ch0>=DELAY_OLHO_FECHADO))
piscou_ch0=PISCOU_MUITAS_VEZES;
			//tempo++;
		}
```

```
                if (media_ch1>limite_up_ch1) { //está piscando!
                        tempo_ch1=0;
                        if ((piscou_ch1==PISCOU_0X)||
(piscou_ch1==PISCOU_1X_OLHO_FECHADO)) { //nao tinha piscado ainda ou
ainda está piscando pela primeira vez...zera o tempo
                                piscou_ch1=PISCOU_1X_OLHO_FECHADO;
                                if (piscou_ch1==PISCOU_1X_OLHO_FECHADO) {
                                        sobreposicao=1; //se o outro olho tb está fechado,
entao os dois estão piscando juntos
                                        tempo=0; //se os dois estao piscando junto e é a
primeira piscada, entao pode zerar o tempo
                                } else if (piscou_ch1==PISCOU_0X) { //o outro olho nao
piscou ainda
                                        sobreposicao=0;
                                        tempo=0;
                                } else sobreposicao=0;
                        } else if (piscou_ch1==PISCOU_1X) { // ja tinha piscado uma
vez...
                                piscou_ch1=PISCOU_2X_OLHO_FECHADO;
                                //tempo++;
                        } else if (piscou_ch1==PISCOU_2X) { //ja tinha piscado 2X

        piscou_ch1=PISCOU_MUITAS_VEZES_OLHO_FECHADO;
                                //tempo++;
                        } else { //ja piscou mais de 2X ou ainda está piscando...nao
precisa fazer nada
                                //tempo++;
                        }
                } else {
                        if ((piscou_ch1==PISCOU_1X_OLHO_FECHADO)
&&(tempo_ch1>=DELAY_OLHO_FECHADO)) piscou_ch1=PISCOU_1X;
                        else if ((piscou_ch1==PISCOU_2X_OLHO_FECHADO)
&&(tempo_ch1>=DELAY_OLHO_FECHADO)) piscou_ch1=PISCOU_2X;
                        else if
((piscou_ch1==PISCOU_MUITAS_VEZES_OLHO_FECHADO)
&&(tempo_ch1>=DELAY_OLHO_FECHADO))
piscou_ch1=PISCOU_MUITAS_VEZES;
                        //tempo++;
                }

                if (tempo==TEMPO_ENTRE_PISCADAS) {
                        //deve fazer algo com a cadeira...nem q seja nada...
                        if ((piscou_ch0!=PISCOU_1X_OLHO_FECHADO)
&&(piscou_ch0!=PISCOU_2X_OLHO_FECHADO)&&(piscou_ch0!
=PISCOU_MUITAS_VEZES_OLHO_FECHADO)&&
                                (piscou_ch1!=PISCOU_1X_OLHO_FECHADO)
&&(piscou_ch1!=PISCOU_2X_OLHO_FECHADO)&&(piscou_ch1!
=PISCOU_MUITAS_VEZES_OLHO_FECHADO)) {
                                if (estado_cadeira!=CADEIRA_PARADA){
                                        if ((piscou_ch0==PISCOU_MUITAS_VEZES)||
(piscou_ch1==PISCOU_MUITAS_VEZES)) {
                                                estado_cadeira=CADEIRA_PARADA;
```

```
                                        COMANDO_CADEIRA_FRENTE=0;
                                        COMANDO_CADEIRA_TRAS=0;
                                        COMANDO_CADEIRA_DIREITA=0;
                                        COMANDO_CADEIRA_ESQUERDA=0;
                                    } else if ((piscou_ch0==PISCOU_1X)
&&(piscou_ch1==PISCOU_1X)&&(sobreposicao)) {
                                        estado_cadeira=CADEIRA_PARADA;
                                        COMANDO_CADEIRA_FRENTE=0;
                                        COMANDO_CADEIRA_TRAS=0;
                                        COMANDO_CADEIRA_DIREITA=0;
                                        COMANDO_CADEIRA_ESQUERDA=0;
                                    } else ;
                                } else {
                                    if ((piscou_ch0==PISCOU_2X)
&&(piscou_ch1==PISCOU_2X)) {
                                        estado_cadeira=CADEIRA_FRENTE;
                                        COMANDO_CADEIRA_FRENTE=1;
                                        COMANDO_CADEIRA_TRAS=0;
                                        COMANDO_CADEIRA_DIREITA=0;
                                        COMANDO_CADEIRA_ESQUERDA=0;
                                    } else if ((piscou_ch0==PISCOU_1X)
&&(piscou_ch1==PISCOU_2X)) {
                                        estado_cadeira=CADEIRA_DIREITA;
                                        COMANDO_CADEIRA_FRENTE=0;
                                        COMANDO_CADEIRA_TRAS=0;
                                        COMANDO_CADEIRA_DIREITA=1;
                                        COMANDO_CADEIRA_ESQUERDA=0;
                                    } else if ((piscou_ch0==PISCOU_2X)
&&(piscou_ch1==PISCOU_1X)) {
                                        estado_cadeira=CADEIRA_ESQUERDA;
                                        COMANDO_CADEIRA_FRENTE=0;
                                        COMANDO_CADEIRA_TRAS=0;
                                        COMANDO_CADEIRA_DIREITA=0;
                                        COMANDO_CADEIRA_ESQUERDA=1;
                                    } else if ((piscou_ch0==PISCOU_1X)
&&(piscou_ch1==PISCOU_1X)&&(!sobreposicao)) {
                                        estado_cadeira=CADEIRA_TRAS;
                                        COMANDO_CADEIRA_FRENTE=0;
                                        COMANDO_CADEIRA_TRAS=1;
                                        COMANDO_CADEIRA_DIREITA=0;
                                        COMANDO_CADEIRA_ESQUERDA=0;
                                    } else ;
                                }
                                piscou_ch0=PISCOU_0X;
                                piscou_ch1=PISCOU_0X;
                                sobreposicao=0;
                                tempo_ch0=0;
                                tempo_ch1=0;
                                tempo=TEMPO_10SEG; //qualquer valor grande...
                            } else { //esta com pelo menos um olho fechado...volta o tempo
1ms para esperar abrir o olho
                                tempo--;
```

```
                }
            } else if (tempo>TEMPO_10SEG) tempo=TEMPO_10SEG; //para nao
dar overflow


            envia_serial=1;
            ad_dado_lido=3;
        }

}

void MaqEstAdquireDados(void) {
        unsigned char i;

if ((controla_cadeira)||(adquire_dados)) {
        if (ad_dado_lido==1) { //tem dado novo!
            acum_ch0-=(unsigned long)buf_ch0[ponteiro_buf];
            acum_ch1-=(unsigned long)buf_ch1[ponteiro_buf];
            /*for (i=1;i<BUF_SIZE;i++) { //desloca todo mundo e vai calculando a
média atual
                buf_ch0[i-1]=buf_ch0[i];
                buf_ch1[i-1]=buf_ch1[i];
            }*/

            //converte os dados de complemento de 2 para inteiros sem sinal
            if (canal0.byte[1]&MASCARA_DADO_TESTE) { //se o dado era
negativo
                canal0.byte[1]&=MASCARA_DADO;       //tira o sinal
                dado_ch0=AJUSTA_NEGATIVO-canal0.word;//ajusta o dado
(original estava em complemento de 2
                dado_ch0&=MASCARA_DADO_INT;     //mascara novamente
            } else { //se o dado era positivo
                canal0.byte[1]&=MASCARA_DADO;       //tira o ruido
                dado_ch0=canal0.word; //apenas mascara o dado para tirar
ruido
            }
            if (canal1.byte[1]&MASCARA_DADO_TESTE) { //se o dado era
negativo
                canal1.byte[1]&=MASCARA_DADO;       //tira o sinal
                dado_ch1=AJUSTA_NEGATIVO-canal1.word;//ajusta o dado
(original estava em complemento de 2
                dado_ch1&=MASCARA_DADO_INT;     //mascara novamente
            } else { //se o dado era positivo
                canal1.byte[1]&=MASCARA_DADO;       //tira o ruido
                dado_ch1=canal1.word; //apenas mascara o dado para tirar
ruido
            }
            //tira ruido
            if (dado_ch0>limite_down_ch0) dado_ch0-=limite_down_ch0;
            else dado_ch0=0;
            if (dado_ch1>limite_down_ch1) dado_ch1-=limite_down_ch1;
            else dado_ch1=0;
```

```
                //coloca o dado no vetor
                buf_ch0[ponteiro_buf]=dado_ch0;
                acum_ch0+=(unsigned long)dado_ch0;
                buf_ch1[ponteiro_buf]=dado_ch1;
                acum_ch1+=(unsigned long)dado_ch1;
                //ajusta ponteiro pro final do buff circular
                ponteiro_buf++;
                if (ponteiro_buf>=BUF_SIZE)
                        ponteiro_buf=0;
                //acumulou todas as medições, calcula a média
                media_ch0=acum_ch0>>5;
                media_ch1=acum_ch1>>5;
                //media_ch1=(unsigned int)(acum_ch1/(unsigned long)BUF_SIZE);
                canal1.word=0;
                canal0.word=0;
                ad_dado_lido=2;

        } else {
        }
}



}
//===== Interrupção
============================================================
=============//

static void interrupt intrr(void) {
        if (TMR1IF) { //hora de ler uma nova medição
                TMR1H=tmr1h;
                TMR1L=tmr1l;
                maq_est_spi_para=0;
                //LED=!LED;
                TMR1IF=0;
        } else
        if (RCIF) {
                maq_est_serial_para=0;
                buff=RCREG;
                RCIF=0;
        } else
        if (SSPIF) {
                maq_est_spi_para=0;
                SSPIF=0;
        } else
        if (TMR0IF) {
                TMR0=TMR0_LED;
                TMR0IF=0;
                cont_led--;
                if (!cont_led) {
                        cont_led=CONT_LED;
                        LED=!LED;
```

```
                }
        }
}

//===== Programa principal
================================================================
======//

void main (void) {
        init();
        //envia tela de boas vindas
        ShowWelcome();
        while (1) {
                MaqEstLeComandos();
                MaqEstLeAD();
                MaqEstAdquireDados();
                MaqEstComandaCadeira();
                MaqEstEnviaSerial();
                MaqEstCalibracaoRepouso();
                MaqEstCalibracaoPiscada();
        }
}
```

**ANEXO A.14**

Código fonte do software implementado: arquivo code.h.

```
#ifndef CODES_H
#define CODES_H

#include <pic.h>
#include "defines.h"
#include "telas.h"
#include "inits.h"

//===== Configuração geral do pic
=============================================================//

__CONFIG (HS & WDTDIS & PWRTDIS & BORDIS & LVPDIS & DEBUGDIS &
UNPROTECT & DUNPROT);

//===== Declaração das variáveis globais
=============================================================//
unsigned char cont_led;
unsigned char maq_est_le_comandos;
unsigned char maq_est_serial_para;
unsigned char buff;
unsigned char freqc_atual;
unsigned char ganho_atual;
unsigned char tmr1l;
unsigned char tmr1h;
unsigned char controla_cadeira;
unsigned char adquire_dados;
unsigned char maq_est_spi_para;
unsigned char maq_est_le_ad;
unsigned char ad_dado_lido;
union {
        unsigned char byte[2];
        unsigned int word;
} canal0;
union {
        unsigned char byte[2];
        unsigned int word;
} canal1;

bank2 unsigned int buf_ch0[BUF_SIZE];
unsigned long acum_ch0;
unsigned int media_ch0;
unsigned char piscou_ch0;
unsigned char tempo_ch0;
unsigned int dado_ch0;

bank3 unsigned int buf_ch1[BUF_SIZE];
```

```
unsigned long acum_ch1;
unsigned int media_ch1;
unsigned char piscou_ch1;
unsigned char tempo_ch1;
unsigned int dado_ch1;

unsigned char ponteiro_buf;

unsigned char envia_serial;
unsigned char envia_serial_ok;
unsigned char maq_est_envia_serial;

unsigned char sobreposicao;
unsigned int tempo;                 //tempo desde a primeira piscada em 2 segundos
                                                //cada valor equivale ao período de
amostragem -> 1 == 500us, 2 == 1ms, etc
unsigned int limite_up_ch0;                 //acima desse valor, é uma piscada válida
unsigned int limite_up_ch1;                 //acima desse valor, é uma piscada válida
unsigned int limite_down_ch0;     //abaixo desse valor, não é uma piscada (valor
base de ruido do sistema)
unsigned int limite_down_ch1;     //abaixo desse valor, não é uma piscada (valor
base de ruido do sistema)
unsigned int limite_ch0_aux;      //abaixo desse valor, não é uma piscada (valor
base de ruido do sistema)
unsigned int limite_ch1_aux;      //abaixo desse valor, não é uma piscada (valor
base de ruido do sistema)
unsigned char calibracao_repouso;
unsigned char calibracao_piscada;
unsigned char maq_est_calib;
unsigned char estado_cadeira;

char aux;


        //aplicação
void MaqEstLeComandos(void);
void MaqEstLeAD(void);
void MaqEstComandaCadeira(void);

#endif
```

## ANEXO A.15

Código fonte do software implementado: arquivo defines.h.

```
#ifndef DEFINES_H
#define DEFINES_H

#define FOSC        20000000 //20 MHz

//pisca led:
#define LED RB5
        //timer0:
//      incremento a cada 200*256 ns = 51,2us
//      interrupção a cada 12,8ms: 250 * 51,2us
//pisca led a cada 250ms: ~20 * 12,8ms = 256ms
#define TMR0_LED 12
#define CONT_LED 20

//cadeira de rodas
#define COMANDO_CADEIRA_FRENTE              RB3
#define COMANDO_CADEIRA_TRAS            RB2
#define COMANDO_CADEIRA_DIREITA             RB1
#define COMANDO_CADEIRA_ESQUERDA    RB0


//serial
        //SPBRG = FOSC/(16*baud_rate) - 1 = 20000000/16*115200 - 1 =
20000000/1843200 - 1 = 9.8507 ~ 10
#define BAUD_115200     10 //113636 bps ~ 115200bps
#define BAUD_57600      20
#define         DEFAULT_SPBRG 10

//configs do ADC - MCP3304
        //configs gerais
#define BUF_SIZE                    32
#define MASCARA_DADO_TESTE      0b00010000//(unsigned int)
0b0001000000000000
#define MASCARA_DADO        0b00001111//(unsigned int)0b0000111111111111
#define MASCARA_DADO_INT    0b0000111111111111//(unsigned int)
0b0000111111111111
#define AJUSTA_NEGATIVO             (unsigned int)0x0FFF
//#define SS RA5
#define LIGA_SS             0b00100000
#define DESLIGA_SS      0b11011111
#define CANAL0_A 0b00001000 //4 zeros, START BIT, Differential, D2, D1
#define CANAL0_B 0b00000000 //D0, 7 zeros
#define CANAL1_A 0b00001000//0b00001001 //4 zeros, START BIT, Differential,
D2, D1
#define CANAL1_B 0b10000000//0b10000000 //D0, 7 zeros
```

```
#define DUMMY              0b10101010
#define MASCARA_DADOS      0b00111111
#define AMOSTR_1KHZ        0
#define AMOSTR_2KHZ        1
#define AMOSTR_2_5KHZ      2
#define AMOSTR_4KHZ        3
#define AMOSTR_DEFAULT     AMOSTR_1KHZ
        //timer1 - 16 bits
//prescaler - 1:1 - incrementa a cada instrução (200 ns)
//prescaler - 1:2 - incrementa a cada 2 instruções (400ns)
#define CONFIG_T1CON  0b00010000 //prescaler: 1:2, osc: off, clock source:
fosc/4, enable: off
//4kHz:
        //7,2us a mais devido a tratamento de interrupção
        //7,2us = 36 * 200ns = 18 * 400ns
        //1 interrupt a cada 250us = (1250-36) * 200ns =  (635-18) * 400ns
#define TMR1H_4KHZ         0xFD
#define TMR1L_4KHZ         0x97
//2.5kHz:       1 interrupt a cada 400us = (2000-16) * 200ns = (1000-8) * 400ns
#define TMR1H_2_5KHZ   0xFC
#define TMR1L_2_5KHZ   0x20
//2kHz:               1 interrupt a cada 500us = (2500-16) * 200ns = (1250-8) * 400ns
        //3,2us a mais devido ao tratamento de interrupção
        //3,2us = 16 * 200ns = 8 * 400ns
#define TMR1H_2KHZ         0xFB
#define TMR1L_2KHZ         0x26
//1kHz:               1 interrupt a cada 1ms        = (5000-16) * 200ns = (2500-8) *
400ns
#define TMR1H_1KHZ         0xF6
#define TMR1L_1KHZ         0x44
//default:
#define TMR1H_DEFAULT      TMR1H_2KHZ
#define TMR1L_DEFAULTTMR1L_2KHZ


//ajuste da frequencia
        //a frequencia gerada deve ser 100 vezes maior do que a frequencia de corte
do filtro
        //PWM_freq = FOSC/((PR2+1)*4*TMR2_prescaler)
        // 50Hz: PR2=249, TMR2PS=4, Duty=500
#define FREQ_50            0
#define PR2_50             249
#define T2CKPS0_50         1
#define T2CKPS1_50         0
#define CCPR1L_50          0b01111101
#define CCP1X_50  0
#define CCP1Y_50  0
        //100Hz: PR2=124, TMR2PS=4, Duty=250
#define FREQ_100  1
#define PR2_100            124
#define T2CKPS0_100        1
#define T2CKPS1_100        0
```

```
#define CCPR1L_100      0b00111110
#define CCP1X_100       1
#define CCP1Y_100       0
        //150Hz: PR2=82 , TMR2PS=4, Duty=167 *
#define FREQ_150 2
#define PR2_150         82
#define T2CKPS0_150     1
#define T2CKPS1_150     0
#define CCPR1L_150      0b00101001
#define CCP1X_150       1
#define CCP1Y_150       1
        //200Hz: PR2=249, TMR2PS=1, Duty=500
#define FREQ_200 3
#define PR2_200         249
#define T2CKPS0_200     0
#define T2CKPS1_200     0
#define CCPR1L_200      0b01111101
#define CCP1X_200       0
#define CCP1Y_200       0
        //250Hz: PR2=199, TMR2PS=1, Duty=400
#define FREQ_250 4
#define PR2_250         199
#define T2CKPS0_250     0
#define T2CKPS1_250     0
#define CCPR1L_250      0b01100100
#define CCP1X_250       0
#define CCP1Y_250       0
        //300Hz: PR2=166, TMR2PS=1, Duty=333 *
#define FREQ_300 5
#define PR2_300         166
#define T2CKPS0_300     0
#define T2CKPS1_300     0
#define CCPR1L_300      0b01010011
#define CCP1X_300       0
#define CCP1Y_300       1
        //350Hz: PR2=142, TMR2PS=1, Duty=286 *
#define FREQ_350 6
#define PR2_350         142
#define T2CKPS0_350     0
#define T2CKPS1_350     0
#define CCPR1L_350      0b01000111
#define CCP1X_350       1
#define CCP1Y_350       0
        //400Hz: PR2=124, TMR2PS=1, Duty=250
#define FREQ_400 7
#define PR2_400         124
#define T2CKPS0_400     0
#define T2CKPS1_400     0
#define CCPR1L_400      0b00111110
#define CCP1X_400       1
#define CCP1Y_400       0
        //450Hz: PR2=110, TMR2PS=1, Duty=222 *
```

```
#define FREQ_450  8
#define PR2_450             110
#define T2CKPS0_450     0
#define T2CKPS1_450     0
#define CCPR1L_450      0b00110111
#define CCP1X_450       1
#define CCP1Y_450       0
        //500Hz: PR2=99 , TMR2PS=1, Duty=200
#define FREQ_500  9
#define PR2_500             99
#define T2CKPS0_500     0
#define T2CKPS1_500     0
#define CCPR1L_500      0b00110010
#define CCP1X_500       0
#define CCP1Y_500       0
        //default
#define         DEFAULT_FREQ    FREQ_300
#define PR2_DEFAULT             PR2_500
#define T2CKPS0_DEFAULT     T2CKPS0_500
#define T2CKPS1_DEFAULT     T2CKPS1_500
#define CCPR1L_DEFAULT      CCPR1L_500
#define CCP1X_DEFAULT       CCP1X_500
#define CCP1Y_DEFAULT       CCP1Y_500

//ajuste do ganho
#define GANHO_1         0
#define GANHO_5         1
#define GANHO_10 2
#define GANHO_50 3
#define GANHO_100       4
#define GANHO_250       5
#define GANHO_500       6
#define GANHO_750       7
#define GANHO_1000      8
#define DEFAULT_GANHO       GANHO_100

//controle da cadeira de rodas
#define LIMITE_UP_DEFAULT   0x01FF //0x01FF
#define LIMITE_DOWN_DEFAULT     0x0000
#define FATOR_CALIBRACAO_PISCADA    75 //quantos % do valor máximo de
uma piscada representa de fato uma piscada

#define CADEIRA_PARADA              0
#define CADEIRA_FRENTE              1
#define CADEIRA_TRAS        2
#define CADEIRA_ESQUERDA    3
#define CADEIRA_DIREITA             4

#define PISCOU_0X                                   0
#define PISCOU_1X_OLHO_FECHADO                      1
#define PISCOU_1X                                   2
#define PISCOU_2X_OLHO_FECHADO                      3
```

```
#define PISCOU_2X                                   4
#define PISCOU_MUITAS_VEZES_OLHO_FECHADO    5
#define PISCOU_MUITAS_VEZES                         6
//tempo maximo entre piscadas (em multiplos do periodo de amostragem - 1ms/1kHz
):
//2 segundos: 2000 * 1 ms
#define TEMPO_ENTRE_PISCADAS     (unsigned int)2000 //2.000 ms
#define TEMPO_MINIMO_SOBREPOSICAO (unsigned int)50 //50 ms
#define DELAY_OLHO_FECHADO                  (unsigned char)100 //100ms
#define TEMPO_10SEG                         (unsigned int)10000 //10.000 ms
#define TEMPO_CALIBRACAO_REPOUSO     (unsigned int)5000 //5.000ms
#define TEMPO_CALIBRACAO_PISCADA     (unsigned int)5000 //5.000ms


#endif
```

## ANEXO A.16

Código fonte do software implementado: arquivo inits.h.

```
#ifndef INITS_H
#define INITS_H

#include "code.h"

        //inicializadoras
void init(void);
void initUsart(void);
void initSPI(void);
void stopSPI(void);
void initFreq(unsigned char FREQ);
void initGanho(unsigned char GANHO);
void initAD(unsigned char freq_amostragem);
void stopAD(void);

#endif
```

**ANEXO A.17**

Código fonte do software implementado: arquivo telas.h.

```
#ifndef TELAS_H
#define TELAS_H

//telas
#define     TAM_TELA_WELCOME1  12
#define     TAM_TELA_WELCOME1_1     59
#define     TAM_TELA_WELCOME1_2     25
#define     TAM_TELA_WELCOME1_3     14

#define TAM_TELA_WELCOME2       29
#define TAM_TELA_WELCOME2_1     34
#define TAM_TELA_WELCOME2_2     34
#define TAM_TELA_WELCOME2_3     40
#define TAM_TELA_WELCOME2_4     38
#define TAM_TELA_WELCOME2_5     42
#define TAM_TELA_WELCOME2_6     42
#define TAM_TELA_WELCOME2_7     51
#define TAM_TELA_WELCOME2_8     26
#define TAM_TELA_WELCOME2_9     42
#define TAM_TELA_WELCOME2_10    4

#define     TAM_TELA_WELCOME3  58
#define     TAM_TELA_WELCOME3_1     59
#define     TAM_TELA_WELCOME3_2     59
#define     TAM_TELA_WELCOME3_3     58

#define     TAM_TELA_WELCOME4  68
#define     TAM_TELA_WELCOME4_1     69
#define     TAM_TELA_WELCOME4_2     69
#define     TAM_TELA_WELCOME4_3     70

#define TAM_TELA_GANHO          14
#define TAM_TELA_FREQC          14
#define TAM_TELA_FECHA          3
#define TAM_TELA_GANHO_ATUAL    15
#define TAM_TELA_FREQC_ATUAL    15

#define TAM_TELA_AD1   42
#define TAM_TELA_AD2   35
#define TAM_TELA_AD3   37
#define TAM_TELA_AD4   35
#define TAM_TELA_AD5   37
#define TAM_TELA_AD6   2

#define TAM_TELA_CALIB_REPOUSO 47
```

```
#define TAM_TELA_CALIB_PISCADA   47

//===== Declaração das Telas
================================================================
====//
        //boas vindas
const char welcome1[] =   "\r\nWelcome!\r\n"; //12
const char welcome1_1[] = "Trabalho de Conclusao - Engenharia da Computacao -
2010.2\r\n"; //59
const char welcome1_2[] = "Bruno Albrecht - 151846\r\n"; //25
const char welcome1_3[] = "Versao 0.2\r\n\r\n"; //14

const char welcome2[] =   " * H ou h: repete esta tela\r\n"; //29
const char welcome2_1[] = " * Z ou z: calibracao em repouso\r\n"; //34
const char welcome2_2[] = " * X ou x: calibracao da piscada\r\n"; //34
const char welcome2_3[] = " * C ou c: controla a cadeira de rodas\r\n"; //40
const char welcome2_4[] = " * E ou e: enviar dados pela serial:\r\n"; //38
const char welcome2_5[] = "     - 2 bytes da media atual do canal 0\r\n"; //42
const char welcome2_6[] = "     - 2 bytes da media atual do canal 1\r\n"; //42
const char welcome2_7[] = "     - 1 byte do estado atual da cadeira de rodas\r\n"; //51
const char welcome2_8[] = " * Codigo do ganho atual: ";
const char welcome2_9[] = "\r\n * Codigo da frequencia de corte atual: ";
const char welcome2_10[] = "\r\n\r\n";


const char welcome3[] =   "
##################################################\r\n"; //58
const char welcome3_1[] = " Ganho:# 1 # 5 # 10 # 50 # 100 # 250 # 500 # 750 #
1000 #\r\n"; //59
const char welcome3_2[] = "  Code:# 0 # 1 #  2 #  3 #   4 #  5  #  6  #  7  #   8
#\r\n"; //59
const char welcome3_3[] = "
##################################################\r\n"; //58
const char welcome4[] =   "
############################################################\r\n"; //68
const char welcome4_1[] = " FreqC:# 50 # 100 # 150 # 200 # 250 # 300 # 350 # 400
# 450 # 500 #\r\n"; //69
const char welcome4_2[] = " Code: #  0 #  1  #  2  #  3  #  4  #  5  #  6  #  7  #  8  #  9
#\r\n"; //69
const char welcome4_3[] = "
############################################################\r\n\r\n"; //70

        //
const char novo_ganho[] = "\r\nNovo Ganho ["; //14
const char nova_freqc[] = "\r\nNova FreqC ["; //14
const char fecha[] = "]: "; //3
const char tela_ganho_atual[] = "\r\nGanho Atual: "; //15
const char tela_freqc_atual[] = "\r\nFreqC Atual: "; //15

const char ad1[] = "\r\n--- Controlando a cadeira de rodas ---\r\n"; //42
const char ad2[] = "   * Calibracao em repouso CH0: 0x0"; //35
const char ad3[] = "\r\n   * Calibracao em repouso CH1: 0x0"; //37
```

```
const char ad4[] = "   * Calibracao da piscada CH0: 0x0"; //35
const char ad5[] = "\r\n   * Calibracao da piscada CH1: 0x0"; //37
const char ad6[] = "\r\n"; //2

const char calib_repouso[] = "\r\n Realizando calibracao em repouso:
repouse!\r\n"; //47
const char calib_piscada[] = "\r\n Realizando calibracao de piscadas:
pisque!\r\n"; //47

        //telas
void ShowWelcome(void);
void ShowAD(void);


#endif
```