UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE MOUSE UTILIZANDO ACELERÔMETROS

GABRIELE JENNRICH

BLUMENAU
2010

2010/2-15

GABRIELE JENNRICH

PROTÓTIPO DE MOUSE UTILIZANDO ACELERÔMETROS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer, Mestre - Orientador

BLUMENAU
2010

2010/1-15

PROTÓTIPO DE MOUSE UTILIZANDO ACELERÔMETROS

Por

GABRIELE JENNRICH

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
Membro:	Prof. Antonio Carlos Tavares, Mestre – FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB

Blumenau, 07 de dezembro de 2010.

Dedico este trabalho a todos que um dia sonharam, ou perderam o sono com este. Em especial aos amigos que me ajudaram na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que sempre esteve presente, me estimulando para realização deste, e teve total paciência quando comecei a falar sozinha compulsivamente.

A minha mãe e ao meu namorado Charles, por terem percebido e jogado a bóia salva-vidas todas as vezes que ansiava por socorro.

Aos meus amigos, pelos empurrões, cobranças e sátiras.

A professora Janaina Real de Moraes por ter cedido seu tempo e sapiência, me fazendo entender um pouco da fisiologia humana, através de seu trabalho como fisioterapeuta. Por ter autorizado os teste na clínica de fisioterapia da Furb, e me apresentado pessoas incríveis que não se incomodaram em ser os primeiros usuários e talvez cobaias.

Ao meu orientador, Miguel Alexandre Winsintainer, por ter acreditado na conclusão deste trabalho. E ter ficado empolgado com alguns bits quando ninguém o fez.

O problema com o mundo é que os estúpidos são excessivamente confiantes, e os inteligentes são cheios de dúvidas.

Bertrand Russel

RESUMO

Este trabalho apresenta a realização de um protótipo de *mouse* com acelerômetro para pessoas com deficiência motora nos membros superiores. Esse protótipo visa auxiliar a utilização do computador por pessoas portadoras de hemiplegia espástica ou dificuldades motoras semelhantes. No protótipo é utilizado o componente eletrônico acelerômetro, como único meio de interação entre o usuário e o sistema. Para a conversão da aceleração obtida pela inclinação do acelerômetro, foi desenvolvida uma fórmula matemática que possibilita, além da conversão, um deslocamento suave do cursor do *mouse.* Para a realização dos cliques foi necessário o desenvolvimento de uma rotina de configuração e identificação. Almejando a viabilidade do protótipo, ele foi desenvolvido com a linguagem de programação Java por ser multiplataforma e utilizadas as bibliotecas RXTX (para comunicação serial) e Robot (para manipulação do cursor).

Palavras-chave: Java. *Mouse*. Acelerômetro. Deficiência motora. RXTX. Robot.

ABSTRACT

This paper presents the realization of a prototype mouse with accelerometer for people with physical disability in upeer limbs. This prototype aims to help people with spastic hemiplegia or similar motor difficulties, use the computer.. In the prototype is used an accelerometer electronic component as the sole means of interaction between user and system. To the convert of acceleration obained by the tilt of the accelerometer, was developed a mathematical formula that allows, besides convertion, a smooth scrolling of the mouse cursor. For the realization of clicks was necessary to develop a setup routine and na identification routine. By craving the viability of the prototype, it was developed with the Java programming language that is a multiplatform as same as the RXTX libraries (for serial communication) and Robot (for manipulating the cursor).

Key-words: Java. *Mouse*. *Accelerome*. *Motor Difficulties*. RXTX. Robot.

LISTA DE ILUSTRAÇÕES

Figura 1 – Comparação do tamanho do acelerômetro e uma moeda. 16
Figura 2 – SEN-00410 17
Figura 3 – Saída e menu do SEN-00410 18
Figura 4 – Paciente com hemiplegia esquerda 22
Figura 5 – *Mouse* de botões 23
Figura 6 – *Mouse* de botões em régua 24
Figura 7 – *Mouse* por toque 24
Figura 8 – Tracker Pro: Captura do movimenta através de câmera 25
Figura 9 – Óculos-*mouse* 26
Figura 10 – Nintendo Wii 27
Figura 11 – Diagrama de casos de uso do protótipo 29
Quadro 1 – UC01 Movimentar Cursor 30
Quadro 2 – UC02 Realizar cliques 31
Quadro 3 – UC03 Configurar 32
Figura 12 – Diagrama de classes do protótipo de *mouse* 33
Figura 13 – Diagrama de classes do pacote `model` 34
Figura 14 – Diagrama de classes do pacote `view` 34
Figura 15 – Diagrama de classes do pacote `control` 35
Figura 16 – Diagrama de seqüência `Movimentar cursor` 36
Figura 17 – Diagrama de sequência `Realizar clique` 37
Figura 18 – Fluxograma geral do protótipo 38
Figura 19 – Fluxograma da movimentação do cursor do *mouse* 38
Figura 20 – Fluxograma das configurações do protótipo 39
Quadro 4 – Método `findConnect()` da classe `Serial` 41
Quadro 5 – Método `connect(String portName)` da classe `Serial` 42
Quadro 6 – Parte do método `run()` da classe `SerialReader` 42
Quadro 7 – Parte do método `move(double valor, int PositionOriginal)` 43
Quadro 8 – Fórmula do cálculo do deslocamento a partir da aceleração 44
Quadro 9 – Método `controlePosition(double valor, int positionOriginal)` 44
Quadro 10 – Método `move(String Le)` pertencente à classe `Mouse` 45
Quadro 11 – Método `realizaClick(Double[] medias)` pertencente à classe `Mouse` 47

Figura 21 – Tela de reconfiguração do *mouse* 48
Quadro 12 – Método `calibrarEsquedaSimples(Mouse mouse,String le)` pertencente à classe `ControleConfiguracao` 49
Figura 22 – Solicitação de calibração da faixa de neutralidade 49
Figura 23 – Mensagem de sucesso após calibração da faixa de neutralidade 50
Figura 24 – Solicitação de calibração do clique simples 50

LISTA DE SIGLAS

API – *Application Programming Interface*

AWT – *Abstract Window Toolkit*

GNU – *GNU's Not Unix*

JDK – *Java Development Kit*

LGPL – *Lesser General Public License*

RF – Requisito Funcional

RNF – Requisito Não Funcional

TNI – Tabela Nacional de Incapacidades

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS DO TRABALHO 14
1.2 ESTRUTURA DO TRABALHO 14
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 ACELERÔMETROS 16
2.2 SEN-00410 17
2.3 COMUNICAÇÃO ENTRE SISTEMAS OPERACIONAIS E PERIFÉRICOS 19
2.4 TIPOS E GRAUS DE DEFICIÊNCIA MOTORA 19
2.4.1 Hemiplegia espástica 20
2.5 TRABALHOS CORRELATOS 22
2.5.1 *Mouse* com botões 23
2.5.2 *Mouse* por toque 24
2.5.3 Captura de movimento através de câmera 25
2.5.4 Óculos-*mouse* 26
2.5.5 Nintendo Wii 27
3 DESENVOLVIMENTO DO PROTÓTIPO 28
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 28
3.2 ESPECIFICAÇÃO 29
3.2.1 Diagrama de casos de uso 29
3.2.2 Diagramas de classes 32
3.2.3 Diagramas de seqüência 35
3.2.4 Fluxogramas 37
3.3 IMPLEMENTAÇÃO 40
3.3.1 Técnicas e ferramentas utilizadas 40
3.3.2 Desenvolvimento do protótipo 40
3.3.2.1 Etapa 1: comunicação serial com o SEN-00410 41
3.3.2.2 Etapa 2: manipulação do cursor pelos dados obtidos pela serial 42
3.3.2.3 Etapa 3: interface gráfica 48
3.3.3 Operacionalidade da implementação 49
3.3.4 Testes 51
3.4 RESULTADOS E DISCUSSÃO 52

4 CONCLUSÕES 54
4.1 EXTENSÕES 55
REFERÊNCIAS BIBLIOGRÁFICAS 56

1 INTRODUÇÃO

O computador tem evoluído de mera ferramenta de trabalho, para um equipamento, quase imprescindível nos estudos, entretenimento, acesso à informação e socialização. É utilizado por qualquer pessoa em qualquer faixa etária e em qualquer parte do planeta, criando a sensação de liberdade e igualdade. Mas para algumas pessoas, ele ainda apresenta limitações de acessibilidade. Christopher Reeve, famoso ator por interpretar *Superman I, II, III e IV*, e mais tarde, conhecido por sua condição de tetraplegia e seu incansável ativismo em nome da investigação e pesquisa sobre lesão medular; quando questionado sobre a utilidade da internet para pessoas com deficiência, já mencionou seus benefícios, alegando que o uso da informática pode ser decisivo na vida de alguém com limitações (WEB ACCESSIBILITY IN MIND, 2010).

Segundo o censo de 2000 (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2010), existem cerca de sete milhões de brasileiros com algum tipo de deficiência motora, falta de membros ou deficiência causada por doenças genéticas ou cognitivas. Permitir o acesso deste grupo de pessoas, aos recursos da informática, tem sido um desafio abraçado por poucos. Exemplo disso é o fato de grande parte dos *sites* e equipamentos fornecidos pelo próprio governo federal ainda não estarem adequados à demanda destas pessoas, e a legislação vigente, para acessibilidade em *sites* de órgãos públicos brasileiros, data de 2004 (BRASIL, 2004). Para os deficientes motores uma das maiores barreiras está na falta de teclados e *mouses*[1] adaptados ou adaptáveis, e de custo acessível, que atendam as suas limitações. Outra barreira encontrada pelos alunos portadores de deficiência é o fato da legislação os tratar como diferentes e sugerir que as escolas criem uma sala própria para instalar os equipamentos a eles destinados, fazendo com que este aluno não interaja com os demais nas aulas que utilizem o computador.

Tentando resolver parte deste problema físico de acesso à informática, e contribuir para a popularização do acelerômetro, surge a idéia de prototipar um *mouse* que utilize o acelerômetro como principal interface entre o usuário e o computador. Optou-se por desenvolver o protótipo utilizando o acelerômetro encapsulado SEN-00410, desenvolvido pela Sparkfun Electronics. A escolha deste modelo deve-se ao seu baixo custo, versatilidade e

[1] O termo *mouse* é utilizado para descrever dispositivos de apontamento que exerçam as funções realizadas por um *mouse* convencional.

recursos, tais como o microcontrolador[2] *Peripherals Integrated Controller* (PIC) da Microchip Tecnology, acelerômetro de três eixos MMA7260Q e sua interface serial (SPARKFUN ELECTRONICS, 2010).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo de dispositivo para apontamento (*mouse)* que utilize os componentes eletrônicos acelerômetro e o microcontrolador PIC, como principal interface de interação entre o usuário e o sistema.

Os objetivos específicos do trabalho são:

a) traduzir os movimentos obtidos pelo acelerômetro, através de suas coordenadas x, y e z, para o cursor do *mouse*;
b) adequar as tradicionais funções de clique para compatibilizá-las com as limitações motoras do usuário e as do acelerômetro;
c) identificar um tipo de deficiência motora que possua características compatíveis com as do *kit* SEN-00410;
d) permitir ajustes no módulo de funcionamento (movimentação contínua)[3], de forma que possa se adaptar as necessidades ou desejos do usuário.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado em 3 capítulos intitulados respectivamente como: fundamentação teórica, desenvolvimento do protótipo e conclusões.

O capítulo 2 apresenta os aspectos teóricos estudados para o desenvolvimento do trabalho. São abordados temas como acelerômetros, graus de deficiência motora e

---

[2] "Microcontrolador pode ser definido como um 'pequeno' componente eletrônico, dotado de inteligência programável, utilizado no controle de processos lógicos" (SOUZA, 2005, p. 21).

[3] Entende-se por movimentação contínua, a movimentação do cursor do *mouse* segundo a inclinação do acelerômetro, ou seja, o usuário inclina o acelerômetro para a direita então o cursor irá se movimentar para a direita até que a inclinação seja alterada, ou até que o cursor chegue ao limite da tela. O *mouse* irá permanecer parado caso a inclinação esteja entre os valores de neutralidade.

comunicação entre o sistema operacional e periféricos. Também são relacionados alguns trabalhos correlatos.

No capítulo 3 é descrito como foi realizado o desenvolvimento deste trabalho detalhando os requisitos do protótipo, a especificação e a implementação. São apresentados os resultados encontrados com a finalização deste.

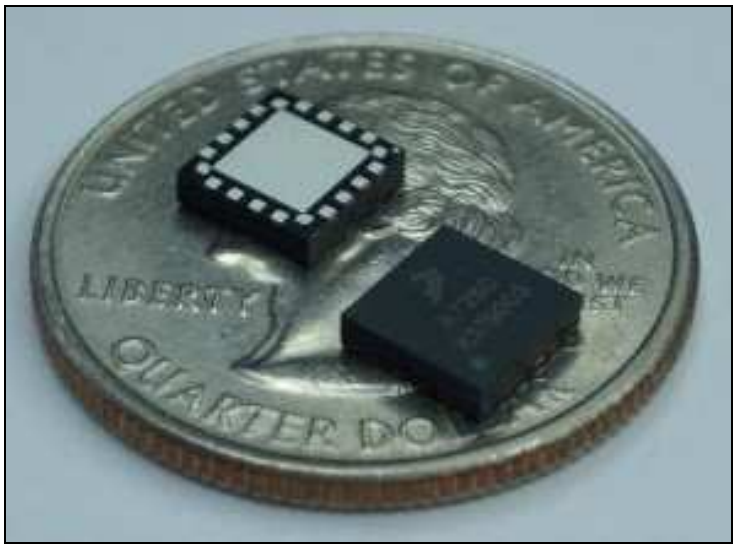
Por fim, o capítulo 4 traz as conclusões deste trabalho, bem como alguns aspectos que poderão ser complementados, servindo de sugestões para futuras extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns dos conceitos relacionados ao trabalho, que servirão como base para o desenvolvimento do mesmo. A seção 2.1 explana, brevemente, sobre o componente eletrônico acelerômetro, seu funcionamento e sua aplicabilidade. Na seção 2.2 é descrito o kit SEN-0410 a ser utilizado como parte física deste protótipo. Na seção 2.3 é explanada a forma de comunicação entre periféricos e sistemas operacionais (Windows e Linux). A seção 2.4 são descritos os diferentes tipos e graus de deficiência motora. Por fim, na seção 2.5 são descritos alguns trabalhos correlatos.

2.1 ACELERÔMETROS

A detecção de movimento em dispositivos é possível graças ao acelerômetro, que é um dispositivo eletro-mecânico com tamanho e custos reduzidos. Segundo United States Navy (2004, p. 140), acelerômetro é definido como um dispositivo que indica tensão proporcionalmente a aceleração (variação de velocidade ou direção) a que este está sendo submetido, e seu funcionamento baseia-se na segunda lei de Newton, que diz que força é igual a massa vezes aceleração.



Fonte:Sparkfun Electronics (2010).

Figura 1 – Comparação do tamanho do acelerômetro e uma moeda.

Os acelerômetros podem ser constituídos de várias formas. Geralmente são compostos

por duas superfícies cerâmicas polarizadas que, submetidas a uma pressão, tensão ou compressão, geram uma carga elétrica proporcional à força aplicada. Este tipo de acelerômetro também é conhecido como acelerômetro tipo piezoelétrico (COELHO, 2007).

Atualmente o acelerômetro é amplamente aplicado em estudos de impacto, movimentação de placas tectônicas, análises do movimento do corpo humano, além de também estar presente em telefones celulares, controles de videogame e em automóveis no controle do disparo de *airbag*.

2.2 SEN-00410

Segundo Sparkfun Electronics (2010), o SEN-00410 é um acelerômetro encapsulado. Ele é composto por um acelerômetro de três eixos, modelo MMA7260Q desenvolvido pela Freescale, um PIC16LF88 e uma interface serial do tipo RS-232. Sua comunicação e alimentação são realizadas através da interface serial, não necessitando de outras alimentações externas. O SEN-00410 possui taxa de transmissão de dados variável, comando de *reset* que restaura as configurações originais de fábrica e um software já incluído no PIC16LF88. Este software pode ser configurado para detectar movimentos em determinada faixa de freqüência, sendo também possível escolher o tipo de saída do acelerômetro.

Fonte:Sparkfun Electronics (2010).

Figura 2 – SEN-00410

As saídas do *kit* podem ser do tipo valor real de aceleração (entre -5 e +5), valor bruto

(a ser calculado pelo computador) e binário (onde cada bit tem um significado específico), esses valores são calculados a partir do ângulo de inclinação do acelerômetro em relação ao eixo e o valor da força gravitacional sobre o acelerômetro. É indicado que para cada tipo de saída seja utilizada uma freqüência de transmissão próxima a: 125Hz, 195Hz ou 595Hz, respectivamente (SPARKFUN ELECTRONICS, 2005 p. 3-4). A Figura 3 exemplifica a saída do tipo valor real para as coordenadas x, y, z e o menu de configurações.

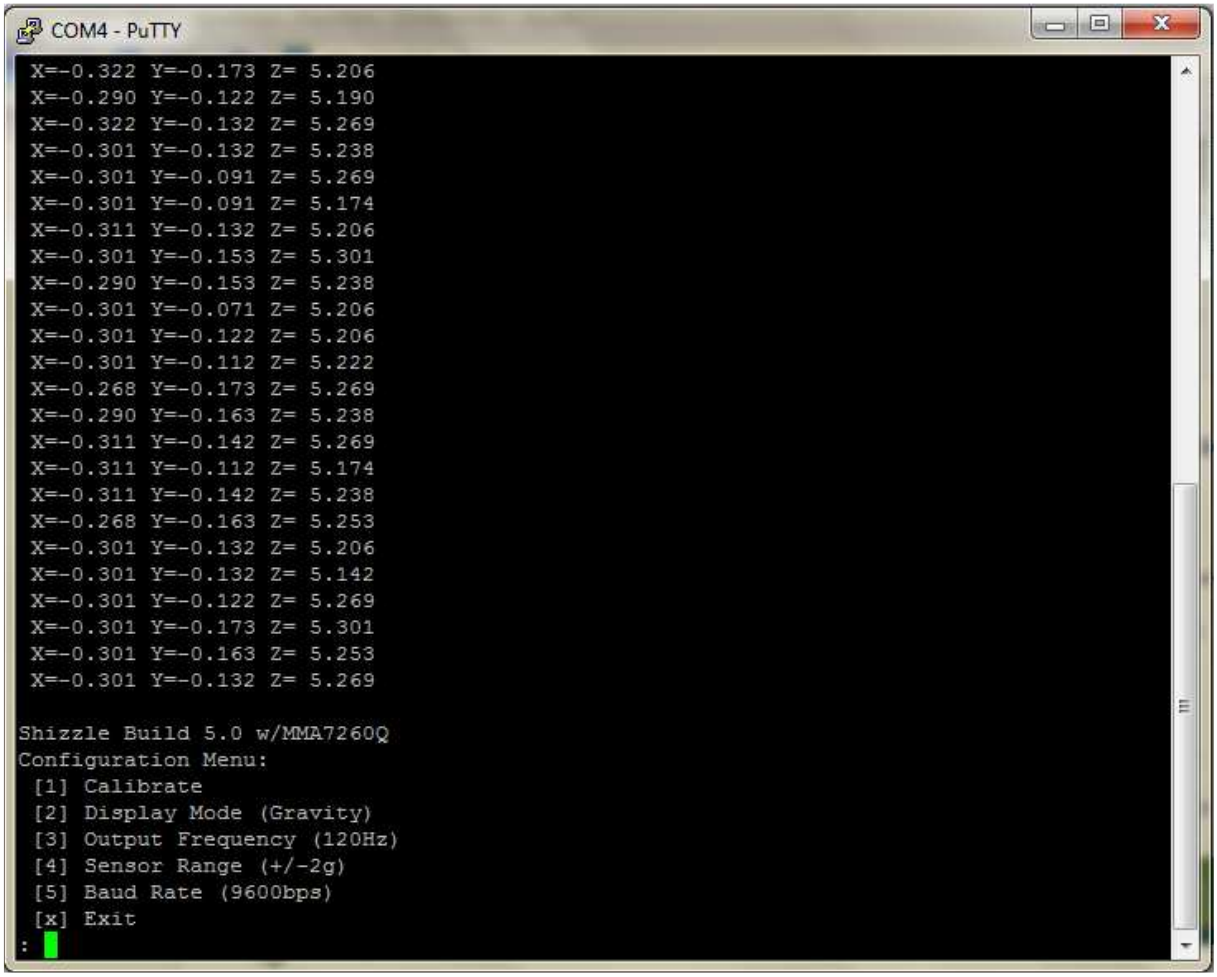


Figura 3 – Saída e menu do SEN-00410

A Sparkfun Electronics (2005, p. 2), cita que o SEN-00410 vem calibrado de fábrica, com valores de máximo e mínimo de rotação tendendo a 900 e 0, respectivamente, que são armazenadas numa estrutura não volátil. Mas recomenda-se a recalibração, buscando estes valores, na primeira vez que ele for utilizado.

2.3 COMUNICAÇÃO ENTRE SISTEMAS OPERACIONAIS E PERIFÉRICOS

Segundo Ferreira e Ramos (2006, p. 4-6), no sistema operacional Linux os periféricos são considerados arquivos. Cada periférico é representado por um arquivo, que contém informações sobre ele mesmo, seu proprietário, permissões de acesso... Como em Linux praticamente tudo é arquivo, os periféricos foram classificados como arquivos de caractere ou de blocos. Esta classificação permite que seu acesso seja direto ao arquivo que corresponde ao dispositivo, não necessitando de transmissão de mensagens ou protocolos.

Já, no sistema operacional Windows, os periféricos não podem ser acessados diretamente. Cada periférico é acessado através da interface que o representa. Para comunicar-se com as interfaces, o Windows utiliza-se de troca de mensagens (MURRAY; PAPPAS, 1992, p. 29-34).

Na linguagem Java existe uma biblioteca nativa, a *Abstract Window Toolkit* (AWT), que é responsável por gerenciar os componentes gráficos envolvidos nas aplicações. Nesta biblioteca existe a classe Robot que é responsável por manipular os eventos que envolvam o *mouse* e ou o teclado (ORACLE, 2010).

Encontra-se também, em Java, a *Application Programming Interface* (API) chamada *Java Communication* (Javacomm), que é responsável por gerenciar a comunicação do software com as portas seriais e paralelas. Baseando-se nesta biblioteca foi desenvolvida, pela comunidade de desenvolvedores, a RXTX, que é um melhoramento da Javacomm. A RXTX é multiplataforma, e além de realizar a comunicação com as portas serial e paralela, ela consegue se comunicar com adaptadores USB – serial (JARVI, 1998).

2.4 TIPOS E GRAUS DE DEFICIÊNCIA MOTORA

Segundo Freitas (2009), deficiência motora é a disfunção, congênita ou adquirida, dos membros superiores ou inferiores que acarrete em dificuldade de locomoção, coordenação motora ou fala. Para ser considerado um deficiente motor, a pessoa deve apresentar um grau de deficiência permanente, nos membros superiores ou inferiores, em grau igual ou superior a 60% avaliada pela Tabela Nacional de Incapacidades (TNI).

Souza (1994, p. 3-4) classifica a deficiência motora segundo o grau de

comprometimento em paresia ou paralisia. O termo paralisia refere-se à perda da capacidade de controle muscular voluntário, já paresia refere-se ao movimento limitado ou fraco, ou seja um comprometimento parcial.

Outra forma de classificação da deficiência dá-se através do agente causador da lesão e é subdivido em mais grupos. Este tipo de classificação é adotado por Duarte e Gorla, (2009, p. 33-37), que divide a deficiência físico/motora segundo os três grupos que apresentam maior incidência: paralisia cerebral, lesão medular e acidente vascular cerebral.

Segundo Bobath e Bobath (1989, p. 26), o modelo mais comum classifica a deficiência segundo a quantidade de membros afetados e sua localização, como:

a) monoplegia/monoparesia: perda total ou parcial das funções motoras de um só membro inferior ou superior;
b) hemiplegia/hemiparesia: perda das funções motoras de um hemisfério do corpo;
c) paraplegia/paraparesia: perda das funções motoras dos membros inferiores;
d) triplegia/triparesia: perda das funções motoras de três membros;
e) tetraplegia/tetraparesia: perda das funções motoras dos membros inferiores e superiores.

Segundo Duarte e Gorla (2009, p. 42-43), um mesmo tipo de deficiência pode apresentar diversos graus de limitação motora, isso devido a altura em que a lesão medular ocorreu, a profundidade da mesma, a demora no início do tratamento ou reabilitação e por características individuais. Também enfatiza, que só após uma avaliação médica completa pode-se definir o grau de comprometimento motor do indivíduo.

2.4.1 Hemiplegia espástica

"A hemiplegia, qualquer que seja sua causa, é caracterizada pela perda do controle motor de em lado do corpo" (DAVES, 1996a, p. 33).

Segundo a Associação Brasileira de Medicina Física e Reabilitação (2006, p. 3), "a espasticidade é uma alteração motora caracterizada pela hipertonia e hiper-reflexia muscular." A hipertonia se manifesta pelo aumento da resistência do músculo ao estiramento, e hiper-reflexia pelo "aumento quantitativo da resposta do músculo estimulado à diminuição do limiar de estimulação e o aumento da área reflexogêna".

A espasticidade é um dos distúrbios motores e de incapacitantes mais freqüentes observados nos indivíduos com lesão do sistema nervoso central. Aparece em diferentes

doenças, destacadas com maior frequência, na paralisia cerebral, lesão medular, lesão encefálica, adquiridas por diferentes causas: traumáticas, tumorais, vasculares, infecciosas e degenerativas (ASSOCIAÇÃO BRASILEIRA DE MEDICINA FÍSICA E REABILITAÇÃO, 2006, p. 3).

As principais características da hemiplegia espástica nos membros superiores e tronco são (DAVES, 1996b, p.) :

a) a escápula[4] está retraída e a cintura escapular[5] em depressão (ombro caído);
b) o cotovelo esta fletido com pronação do ante braço, em alguns casos a suspinação domina;
c) o punho está flexionado com algum desvio ulnar (punho dobrado apontando para baixo ou para cima);
d) os dedos estão fletidos e aduzidos (mão fechada);
e) o tronco está rodado para trás no lado hemiplégico com flexão lateral do mesmo lado.

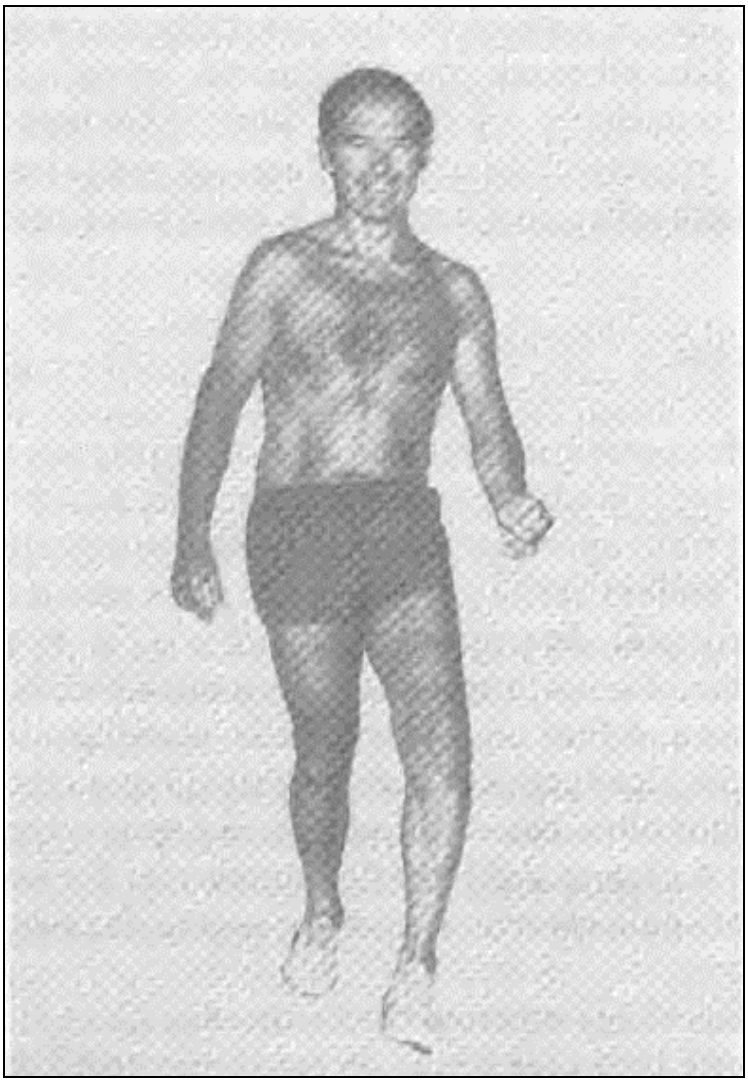
Segundo Daves (1996a, p. 12, 24 e 33) as escápulas estão estreitamente relacionadas às mãos, e tornam possível que elas explorem e experimentem o ambiente a partir do início da infância. A fim de possibilitar uma amplitude mais livre do movimento do ombro, a cintura escapular, não possui nenhuma articulação direta com a coluna vertebral, é portanto dependente de uma atividade muscular complexa a fim de fornecer uma base estável para o braço em movimento. Com a hemiplegia, existe uma perda da atividade seletiva dos músculos que controlam o tronco, particularmente nos músculos responsáveis pela rotação, flexão frontal e lateral.

Daves (1996a, p. 64) também afirma que, o braço e a mão somente podem ser usados funcionalmente se a escápula e o ombro puderem ser ativamente controlados de tal maneira a trazê-los e mantê-los firmes na posição desejada. E que a espátula somente pode ser estabilizada dinamicamente se a coluna torácica e as costelas forem capazes de proporcionar o alicerce adequado aos grupos de músculos relevantes. E afirma que muitos pacientes ao tentarem movimentar o seu braço hemiplégico tendem a fixar a espástula, o que acaba por dificultar o uso normal dos braços para executar tarefas.

---

[4] "A espástula consiste num osso par, chato e fino, de forma triangular, podendo evidenciar-se em certos pontos. (...) Situa-se na região dorsal do ombro, em altura entre a segunda e a sétima costela e articula-se com dois ossos: (...) A escápula não irá articular-se diretamente com o tronco, mas sim através da clavícula. Mesmo assim, a movimentação dos braços está relacionada à musculatura que parte desse osso de grande importância funcional."(ANDRADE, 2010)

[5] Cintura escapular é formada pela espástula e clavícula. Formando a região dos ombros.



Fonte: Daves (1996b).

Figura 4 – Paciente com hemiplegia esquerda

Para Daves (1996a, p. 154) se o braço e a mão não forem incorporados no movimento e nas atividades diárias, eles não terão quase nenhuma experiência, e os movimentos ativos podem permanecer adormecidos. E a mão irá se tornar inútil.

2.5 TRABALHOS CORRELATOS

Existem vários modelos de *mouse* adaptados, alguns em escala comercial, outros em fase de teste. Todos fornecem alternativas às pessoas portadoras de dificuldades motoras, e desempenham papel semelhante ao proposto. Dentre eles, foram selecionados *mouse* com botões, *mouse* por toque, captura do movimento através de câmera e óculos-*mouse*.

Quanto às aplicações do acelerômetro, foi escolhida o Wii Remote, pertencente ao

videogame Nintendo Wii (REZINI, 2008, p. 14), devido as semelhanças de funcionamento com o protótipo sugerido .

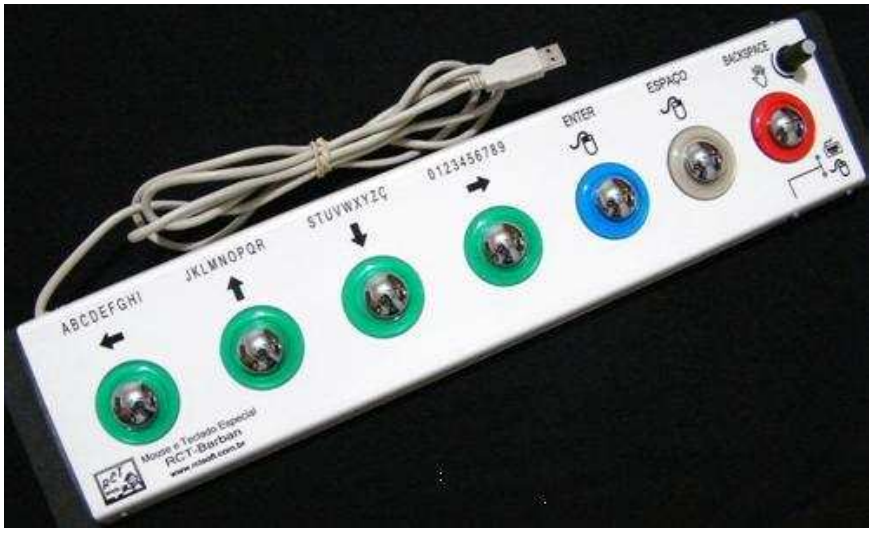
2.5.1 *Mouse* com botões

Consiste de uma régua ou plataforma com várias teclas que ao serem pressionadas realizam uma das ações do *mouse* convencional, tais como movimentação ou cliques (CLICK TECNOLOGIA ASSISTIVA, 2010). Existem diversos modelos, com mais ou menos teclas, e de tamanhos variados. Os mais simples são compostos por seis botões, quatro deles destinados à movimentação (horizontal e vertical) e os demais a cliques (seleção e duplo clique). Os modelos mais completos, como o citado por Alves (2003), contêm até doze botões, permitindo a movimentação do cursor na diagonal, cliques com o tradicional botão direito do *mouse* e arrasto de componentes; e ainda, permitem acoplar um *mouse* convencional.

Fonte: Alves (2010).

Figura 5 – *Mouse* de botões

As teclas são normalmente circulares e tem um diâmetro de aproximadamente três centímetros. Elas podem ser facilmente pressionadas tanto com os dedos, os pés, os cotovelos ou o nariz. Como cada ação do *mouse* é realizada por um botão, usuários com níveis de deficiência motora mais intensa podem não conseguir utilizá-lo ou apresentar fadiga após o seu emprego.



Fonte: Click Tecnologia Assistiva (2010).

Figura 6 – *Mouse* de botões em régua

2.5.2 *Mouse* por toque

Semelhante ao *mouse touch screen*, contido em *notebooks*, o *mouse* por toque é constituído por uma superfície lisa dividida em cinco regiões, cada qual executando uma ação específica. A região maior e central é destinada a movimentar o cursor, conforme a movimentação do dedo sobre ela. Outras duas regiões, localizadas lateralmente em relação à área central, são responsáveis por executar o clique simples do *mouse* nos botões direito e esquerdo. As outras duas áreas, localizadas na parte superior, são responsáveis pelo clique duplo do *mouse* e pela seleção de objetos (ABLENET, 2008).

Fonte: Ablenet (2009).

Figura 7 – *Mouse* por toque

Qualquer uma das regiões é sensível ao toque em toda a sua extensão. Elas não medem mais de dois centímetros e localizam-se muito próximas umas das outras, o que torna este *mouse* ideal para pessoas com baixa mobilidade nas mãos ou no pulso, visto que apenas movimentando um dedo pode-se atingir toda a sua extensão. Mas, ao mesmo tempo torna-se inadequado a pessoas com baixa coordenação motora ou que sofram espasmos musculares.

2.5.3 Captura de movimento através de câmera

Este modelo de *mouse* é composto por duas partes. Uma constituída de um adesivo sinalizador e a outra por uma câmera responsável por captar a movimentação do sinalizador. O usuário cola o adesivo em sua testa, mais ou menos sobre a linha do nariz, e posiciona a câmera sobre o monitor, para que esta possa visualizar o sinalizador. Ao inclinar a cabeça o cursor irá se movimentar na direção correspondente a inclinação. Mas, para realizar as funções de cliques e seleção, faz-se necessário a instalação e configuração de um software adicional, não inclusos no produto, ou a inclusão de dispositivos físicos, tais como botão, régua bucal (através da pressão sobre ela realiza a ação) (MADENTEC, 2009).

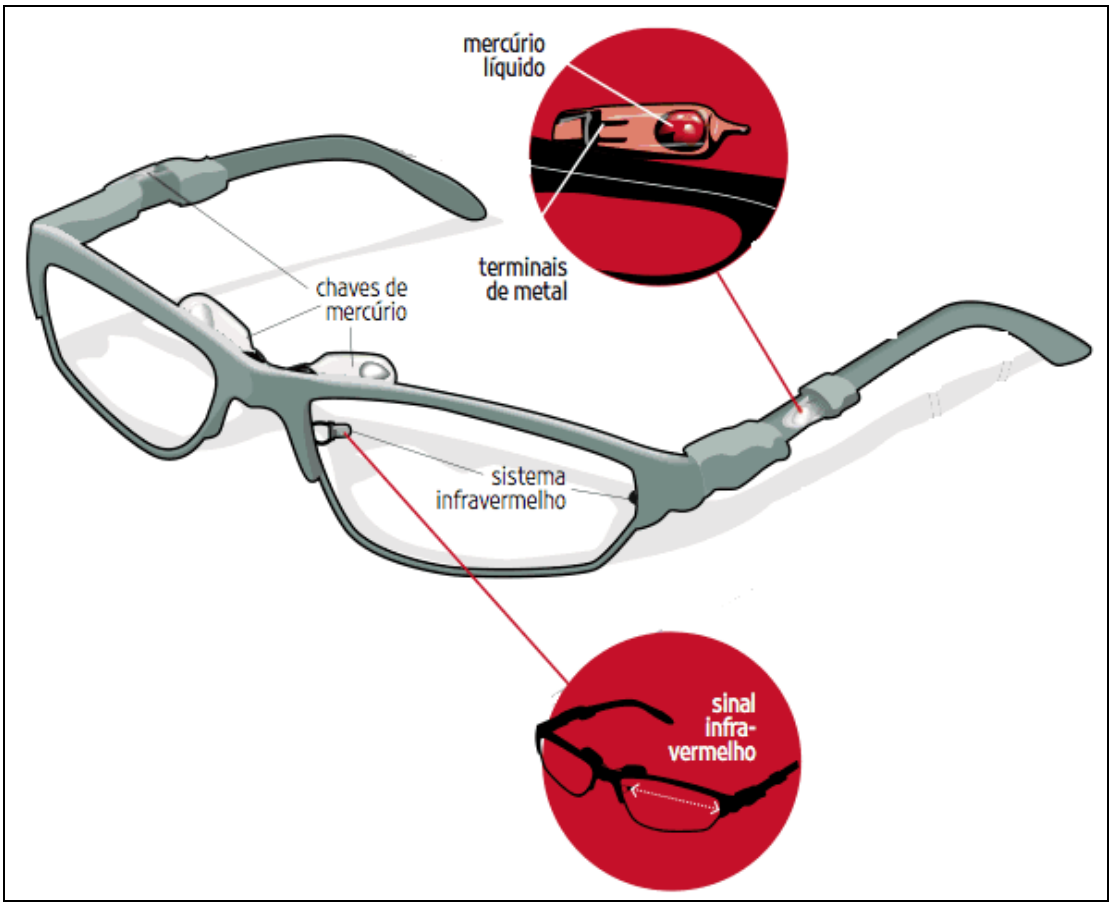
Fonte: Madentec (2009).

Figura 8 – Tracker Pro: Captura do movimenta através de câmera

É indicado para pessoas sem os membros superiores ou com baixíssima mobilidade, mas não atende pessoas que tenham estas características e ainda cabeça pendida para um dos lados. Outra desvantagem é o fato de não atender a todas as funções do *mouse* tradicional sem a inclusão de outros dispositivos.

2.5.4 Óculos-*mouse*

De todos os modelos citados este ainda não foi disponibilizado comercialmente, e atualmente encontra-se como protótipo desenvolvido por alunos do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense (IFSul). É constituído por um óculos convencional, sem as lentes, onde foram adicionados sensores infravermelhos, para captação de cliques (piscada voluntária dos olhos), e circuitos de sensores para detectar a inclinação da cabeça (QUADROS; SAMPAIO; CARVALHO, 2009).



Fonte: adaptado de Bercito (2010).

Figura 9 – Óculos-*mouse*

A movimentação do cursor é feita através da inclinação da cabeça, já os comandos de cliques são executados através do piscar, voluntário, dos olhos (BERCITO, 2010). Exemplificando, caso deseja-se realizar o duplo clique o usuário pisca duas vezes com o olho esquerdo. Este piscar dos olhos é notado pelos sensores infra-vermelhos através da obstrução de passagem de luz. Este modelo é indicado para pessoas sem os braços ou com baixa mobilidade.

2.5.5 Nintendo Wii

Nintendo Wii é um *videogame* que, segundo Rezini (2008, p. 13), “possui o Wii Remote, um *joystick* diferente de todos já utilizados nos demais consoles. Ele possui um acelerômetro que capta os movimentos dos jogadores em qualquer direção com grande sensibilidade, aumentando a interação entre o jogador e o jogo”.

O Wii Remote, também conhecido como Wiimote, possui um acelerômetro tri-axial e sensores infra-vermelhos, além de dispositivos para vibração do controle, criando assim a sensação de resistência durante o jogo. Segundo Rezini (2008, p. 14), o acelerômetro detecta os movimentos e inclinações da mão do jogador, convertendo em ações imediatas no jogo. Também são utilizados os sensores infra-vermelhos para a captação da orientação do controle independente da posição da televisão.

Fonte: Rezini (2008).

Figura 10 – Nintendo Wii

Rezini (2008, p. 22) também afirma que para realmente entender como o Wii Remote revoluciona a maneira de jogar, deve-se experimentá-lo. “E que sua facilidade de uso e interatividade permite uma única experiência para todos os usuários”.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta o desenvolvimento do protótipo de *mouse*, sua utilização, testes realizados e os resultados alcançados.

Serão detalhadas também as técnicas utilizadas, as ferramentas de desenvolvimento adotadas, as dificuldades encontradas durante o processo de desenvolvimento, assim como o hardware escolhido, a linguagem e o ambiente de desenvolvimento e suas limitações.

O público a que este protótipo se destina são pessoas que apresentem as mesmas características das pessoas com hemiplegia espástica. Estas pessoas utilizaram o protótipo preso ao pulso ou a cabeça.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo proposto deverá possuir os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

a) traduzir a inclinação do acelerômetro em movimentos no cursor do *mouse*, através das coordenadas x, y e z do mesmo (RF);
b) interpretar movimentos específicos que representam os atuais cliques de botão direito e esquerdo e comando de seleção do *mouse* tradicional (RF);
c) possuir uma interfase para configuração do *mouse*, onde o usuário poderá calibrar a inclinação para realização de cada um dos cliques e calibrar uma faixa de valores para a qual não exista movimentação (RF);
d) ser implementado utilizando o ambiente de programação Eclipse Galileo e Netbeans (RNF);
e) ser compatível com os sistemas operacionais Windows XP, Vista e Seven ou Linux (RNF).

3.2 ESPECIFICAÇÃO

Na sequência é apresentada a especificação do protótipo de *mouse*, que foi modelada na ferramenta *Entreprise Architect*. Foram utilizados conceitos da orientação a objetos e da *Unified Modeling Language* (UML) para a criação dos diagramas de casos de uso, de classe, de sequência e fluxogramas.

3.2.1 Diagrama de casos de uso

O protótipo possui três casos de uso: `Movimentar cursor`, `Realizar cliques` e `Configurar`, como pode ser observado na Figura 11. Como pode ser observado, nos quadros 1 e 2, os casos de uso `Movimentar cursor` e `Realizar cliques` necessitam que o protótipo já esteja configurado com as características escolhidas pelo usuário. E que para o caso de uso `Configurar`, o SEN-00410 esteja conectado a porta serial que sua transmissão esteja sendo identificada pelo protótipo.

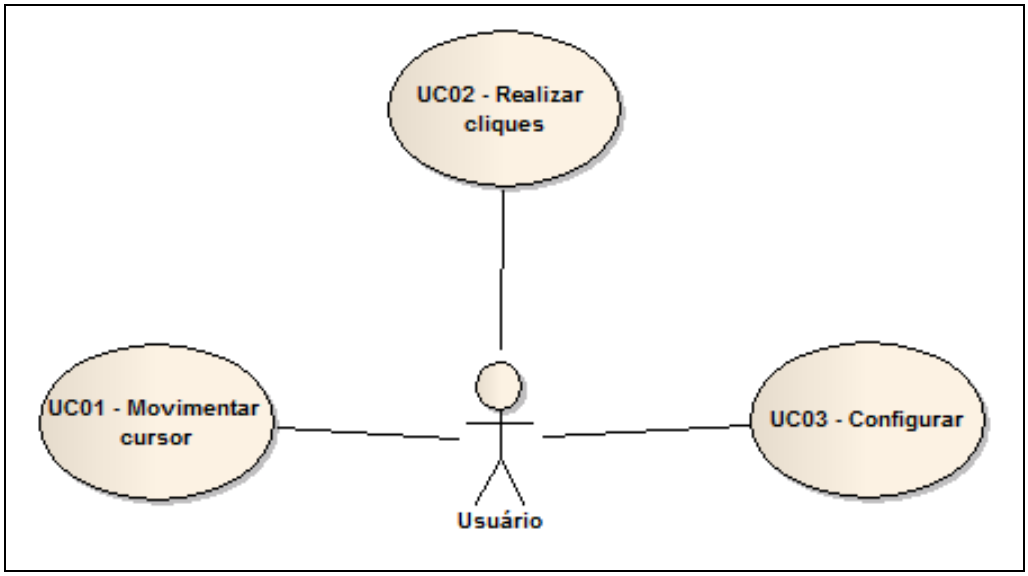


Figura 11 – Diagrama de casos de uso do protótipo

O primeiro caso de uso (Quadro 1), designado `Movimentar cursor`, é o que apresenta quais as ações realizadas pelo usuário e pelo protótipo são necessárias para a movimentação do cursor do *mouse*. Pode-se considerar este caso como o principal caso de uso deste protótipo. Este caso, além do cenário principal, é composto de um cenário de exceção. Nota-se que o cenário de exceção é o caso de uso `Realizar cliques`.

UC01 – Movimentar Cursor	
Pré-condição;	O *mouse* deve estar configurado.
Cenário principal:	1) O usuário inclina o SEN-00410 na direção em que gostaria que o cursor do *mouse* deslocasse; 2) O SEN-00410 envia os valores da aceleração obtidos pela movimentação; 3) O sistema converte os valores de aceleração em valores de deslocamento; 4) O sistema interpreta os valores de deslocamento, verificando se este não representa um comando de clique; 5) O sistema transmite os valores de movimentação para o cursor do *mouse*.
Cenário de exceção:	No passo 4, caso os valores de deslocamento representem cliques, o sistema irá interpretá-lo como tal, e o transmitirá para o *mouse*.
Pós-condição:	O cursor do *mouse* deve ter sido movimentado.

Quadro 1 – UC01 Movimentar Cursor

O segundo caso de uso (Quadro 2), denominado `Realizar cliques`, descreve os passos necessários para realizar os cliques do *mouse*. Ele possui, além do cenário principal, um cenário de exceção, que como nota-se é o caso de uso `Movimentar Cursor`.

UC02 – Realizar cliques	
Pré-condição;	O *mouse* deve estar configurado.
Cenário principal:	1) O usuário inclina o SEN-00410 de maneira semelhante a que foi configurada como clique; 2) O SEN-00410 envia os valores da aceleração obtidos pela movimentação; 3) O sistema converte os valores de aceleração em valores de deslocamento; 4) O sistema interpreta os valores de deslocamento, verificando se este representa um comando de clique; 5) O sistema identifica qual clique o comando representa e o converte no clique respectivo do *mouse*.
Cenário de exceção:	No passo 4, caso os valores de deslocamento não representem cliques, o sistema irá converter os valores obtidos para deslocamento, e o transmitirá para o *mouse*.
Pós-condição:	O *mouse* deve ter realizado o clique respectivo à movimentação realizada.

Quadro 2 – UC02 Realizar cliques

O terceiro caso de uso (Quadro 3), designado `Configurar`, descreve os passos executados pelo usuário para realizar as configurações do módulo de operação do protótipo. Ele é constituído de um cenário principal, um cenário alternativo e um cenário de exceção.

UC03 – Configurar	
Pré-condição;	*Mouse* ter sido conectado ao computador e identificado pelo sistema.
Cenário principal:	1) O usuário solicita a reconfiguração do *mouse*; 2) O sistema apresenta uma tela para a escolha do dado a ser configurado; 3) O usuário segue os passos solicitados pelo sistema; 4) O sistema armazena as novas configurações; 5) O sistema configura o *mouse*.
Cenário alternativo:	Caso o *mouse* seja identificado pelo sistema, mas não possua nenhuma configuração: 1) O sistema solicitará que o usuário realize a configuração total do *mouse*; 2) Retorna ao passo 2 do cenário principal.
Cenário de exceção:	No passo 4, caso algum dos valores obtidos na configuração sejam muito próximos a um dos valores já cadastrados, o sistema apresentará uma mensagem de erro e solicitará que o usuário tente configurar novamente o *mouse*.
Pós-condição:	O *mouse* deve ter sido configurado.

Quadro 3 – UC03 Configurar

3.2.2 Diagramas de classes

O diagrama de classes fornece uma visão de como as classes estão estruturadas e relacionadas, de forma a facilitar a estruturação e a relação entre elas. São apresentados quatro diagramas de classes.

O primeiro diagrama, na Figura 12, fornece a visão geral do *mouse*, mostrando todas as classes e seus relacionamentos. O sistema é dividido em três pacote: `model`, `control` e `view`. O pacote `view` contém as classes responsáveis pela interação gráfica com o usuário no momento da reconfiguração. As classes pertencentes ao pacote `model`, são responsáveis pela modelagem do *mouse*. O pacote `control`, tem por finalidade controlar a comunicação do protótipo com a porta serial e a comunicação com o arquivo de configuração, é também responsável por identificar e executar as movimentações no cursor do *mouse*.

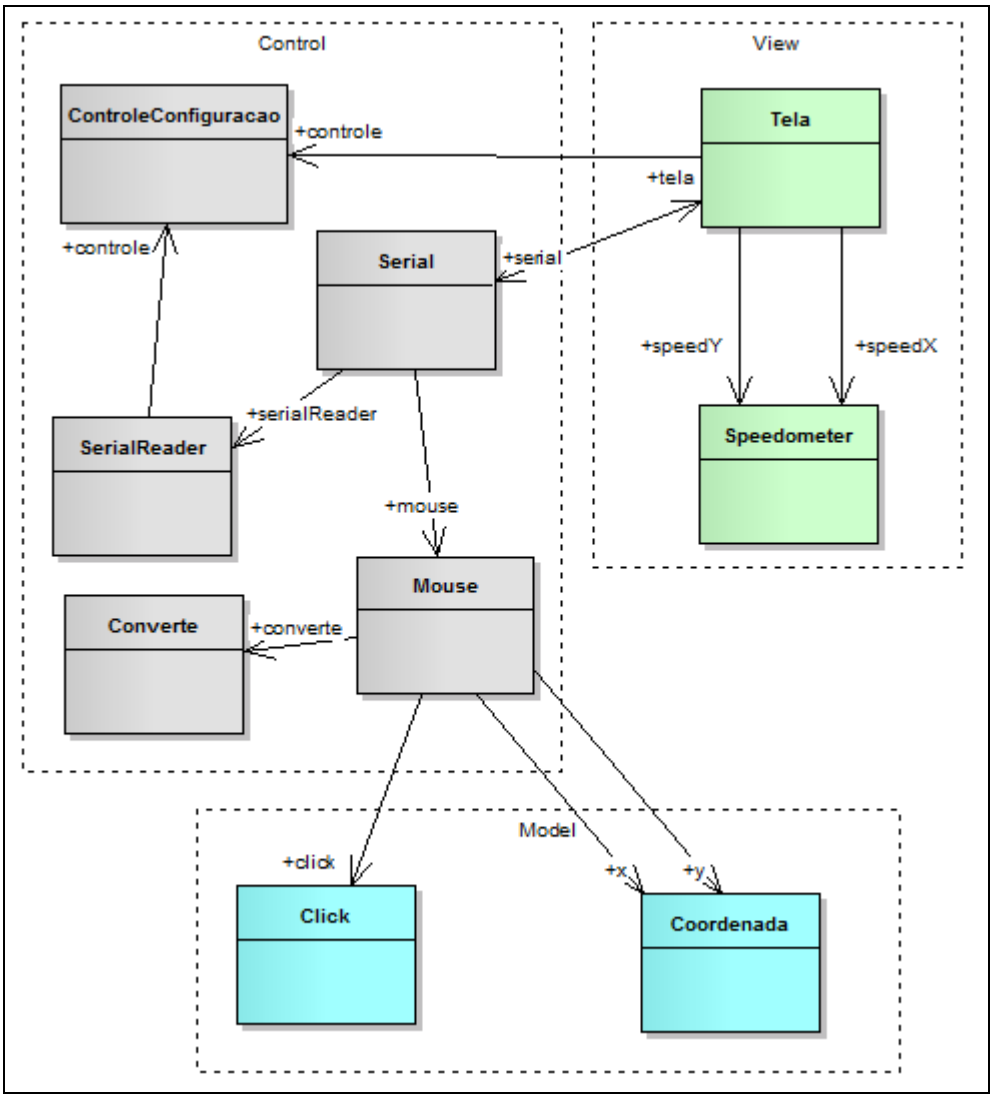


Figura 12 – Diagrama de classes do protótipo de *mouse*

No pacote `model` possui duas classes, a `Click` e a `Coordenada`. Como pode ser observado na Figura 13, a classe `Click` é responsável por guardar os valores que representam cada um dos quatro tipos de cliques existentes (clique simples, clique duplo, clique simples para seleção e clique do botão direito). Já a classe `Coordenada` guarda diversas informações sobre uma determinada coordenada cartesiana, tais como: posição máxima e mínima do cursor na tela, posição atual e anterior do cursor, alguns fatores de ajuste e a posição onde foi identificado o desejo de realizar o clique.

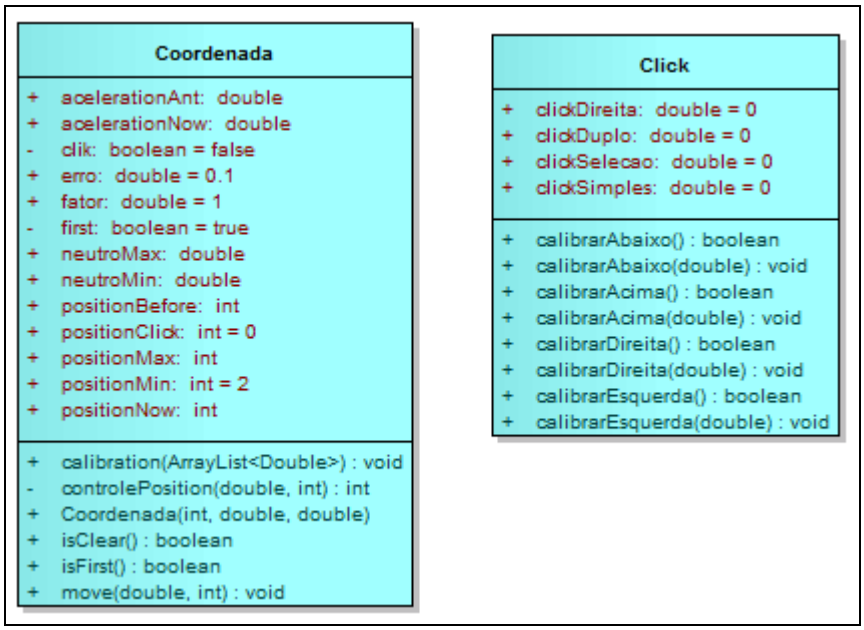


Figura 13 – Diagrama de classes do pacote `model`

O pacote `view`, Figura 14, contém as classes de interação com o usuário e que são executadas quando o usuário deseja reconfigurar o *mouse*. Nesse pacote existem duas classes que possuem como único objetivo tornar o processo de reconfiguração mais ilustrativo.

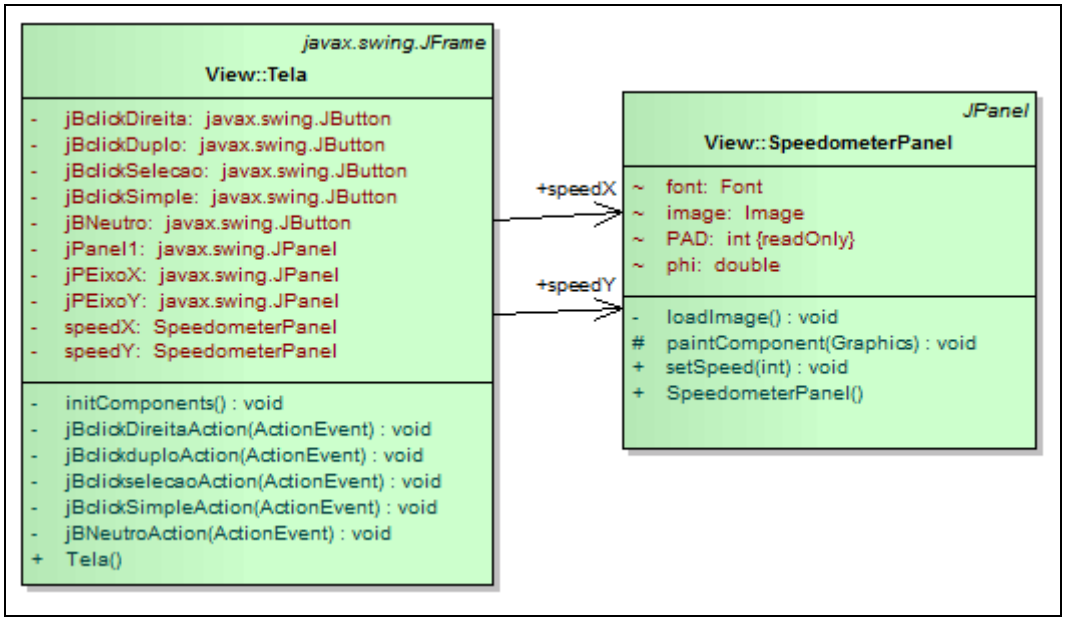


Figura 14 – Diagrama de classes do pacote `view`

Na Figura 15, é apresentado o diagrama do pacote `control`, como já mencionado este pacote é responsável por duas grandes atividades, controle das comunicações e movimentação do cursor do *mouse*. Toda a parte de comunicação entre o software e porta serial ou arquivo de configuração é realizada pelas classes `Serial` e `ControleConfiguração`, é também através dessas classes que ocorrem a interação entre os pacotes `control` e `view`. A identificação de qual movimento deve ser realizado no *mouse* e a realização deste movimento

é gerenciada pelas classes `Mouse` e `Converte`. A classe `Converte` é responsável por converter os valores lidos pela serial em valores numéricos que possam ser interpretados e convertidos pela classe `Mouse` em movimentos do cursor ou cliques.

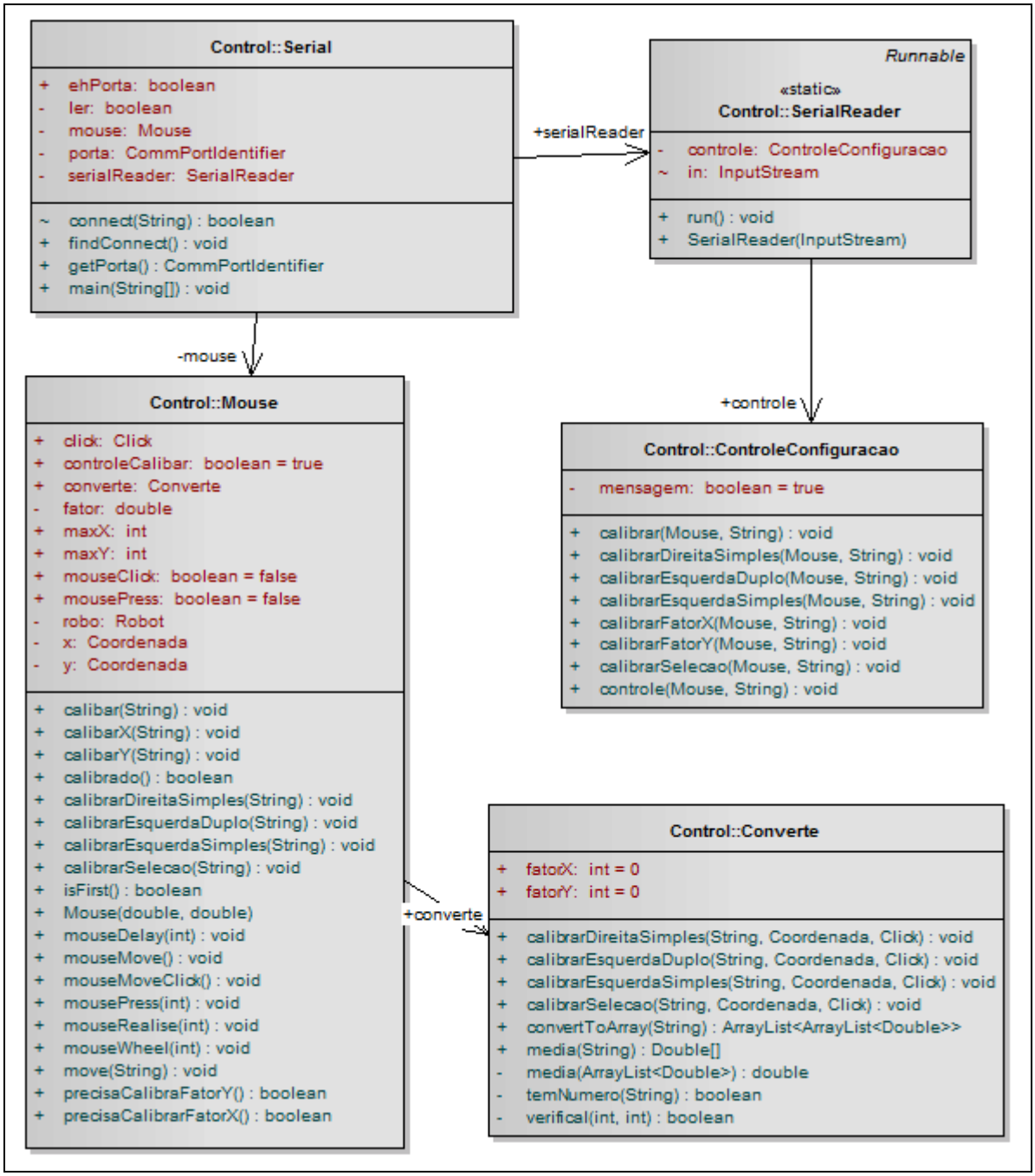


Figura 15 – Diagrama de classes do pacote `control`

3.2.3 Diagramas de seqüência

Diagramas de seqüência representam o conjunto de passos que o programa executa

para realizar determinada tarefa, com base nas ações do usuário. Esta seção apresenta o diagrama de seqüência do caso de uso `Movimentar cursor` e do `Realizar clique`.

A Figura 16 apresenta a seqüência da chamada de métodos executada pelo sistema para realizar a movimentação do cursor do *mouse*. Este diagrama corresponde ao caso de uso `Movimentar cursor`.

Todo o processo de comunicação do protótipo com o sistema é realizado através de uma *thread* criada no método `connect(String portName)`. Essa *thread* executa a classe `SerialReader` que é responsável por capturar os dados vindos da porta serial e identificar se esses dados possuem um determinado padrão. Ao identificar esse padrão é chamado o método `controle(Mouse mouse, String leitura)`, que por sua vez chama o método `move(String le)` da classe `Mouse`. Este último é responsável por realizar a movimentação do cursor, através da alteração dos valores das coordenada x e y.

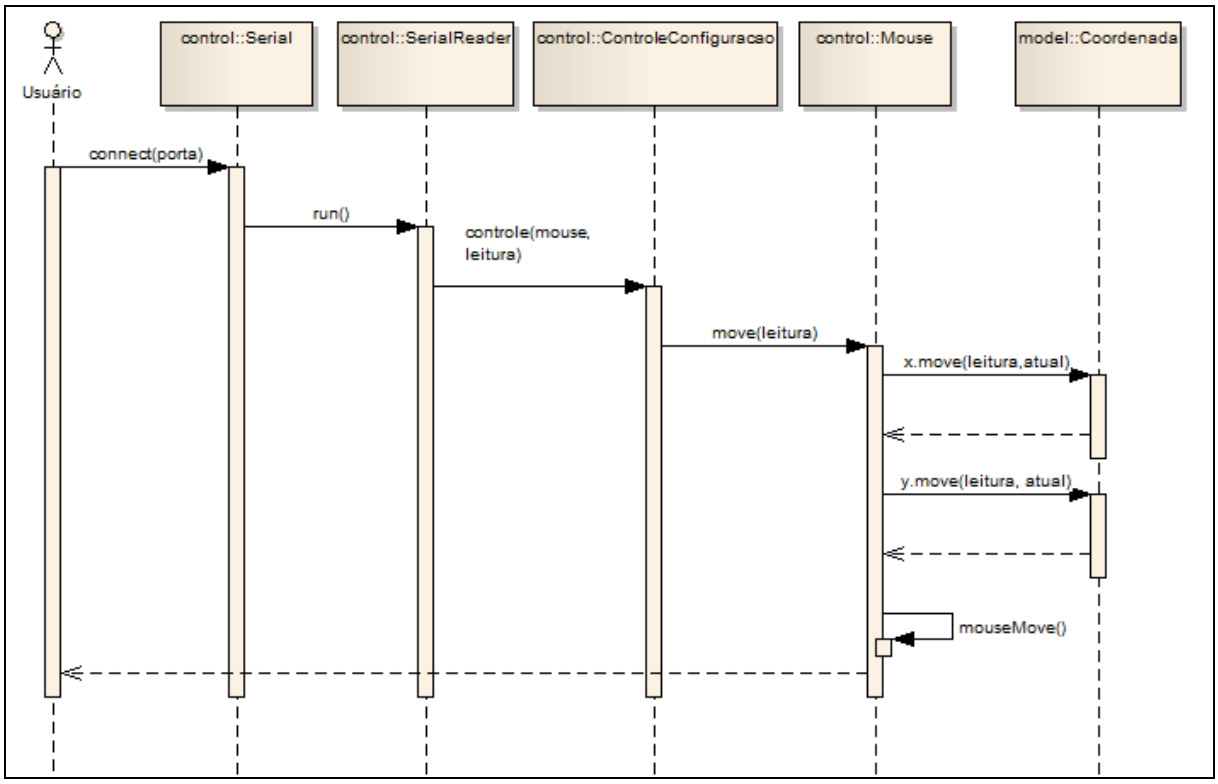


Figura 16 – Diagrama de seqüência `Movimentar cursor`

A Figura 17 apresenta o diagrama de seqüência correspondente ao caso de uso `Realizar clique`. O caso de uso em questão envolve quatro opções de cliques (clique simples, duplo, direita e de seleção), para simplificar o diagrama, nele é apresentado apenas o clique simples. Para a realização dos outros cliques são alteradas as chamadas de métodos `click.clickSimples()` para o nome do clique desejado. E nos métodos `mousePress` e `mouseRelease` é alterado o valor do botão que é passado como parâmetro.

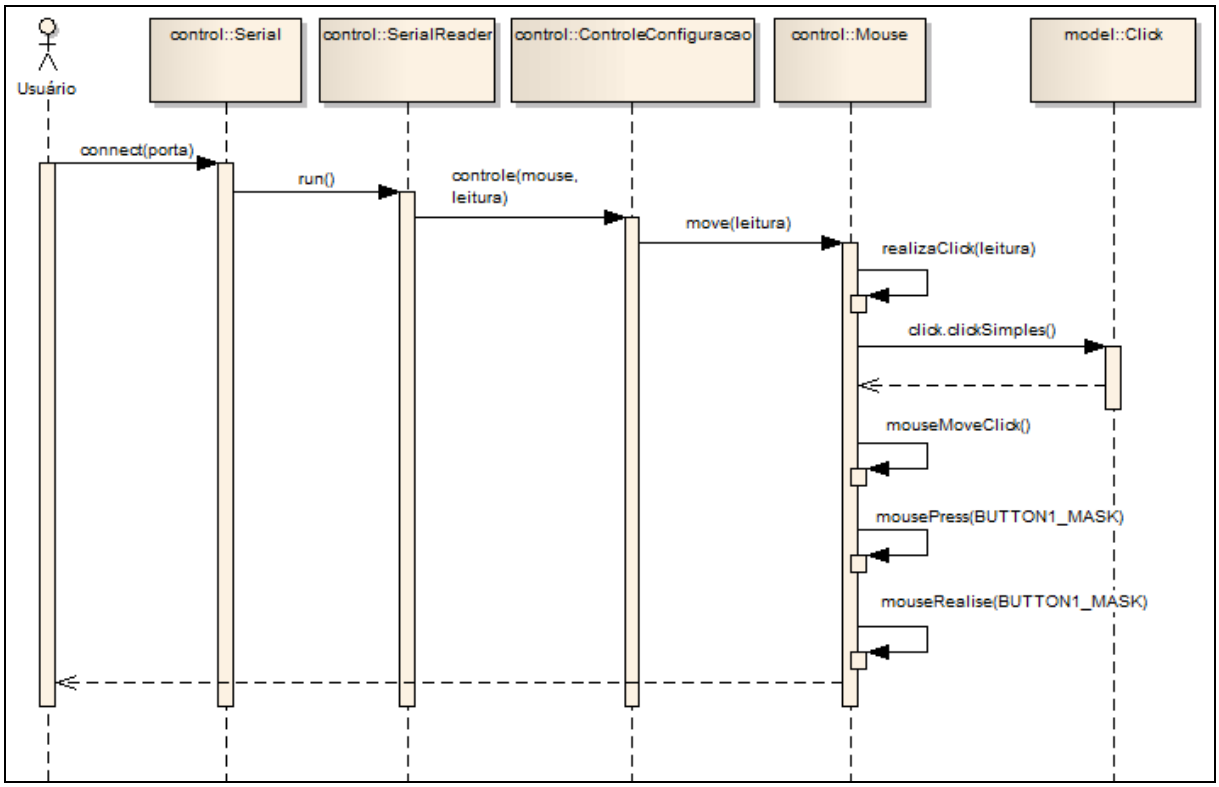


Figura 17 – Diagrama de sequência `Realizar clique`

3.2.4 Fluxogramas

Os fluxogramas fornecem uma visão esquemática de um processo e vem complementar os diagramas de classes e de sequência apresentados. A Figura 18, fornece uma visão geral do funcionamento do protótipo, suas tomadas de decisões e seus processos. Como pode ser notado, o protótipo funciona como um *loop* infinito, onde o *mouse* será executado enquanto estiver conectado a porta serial.

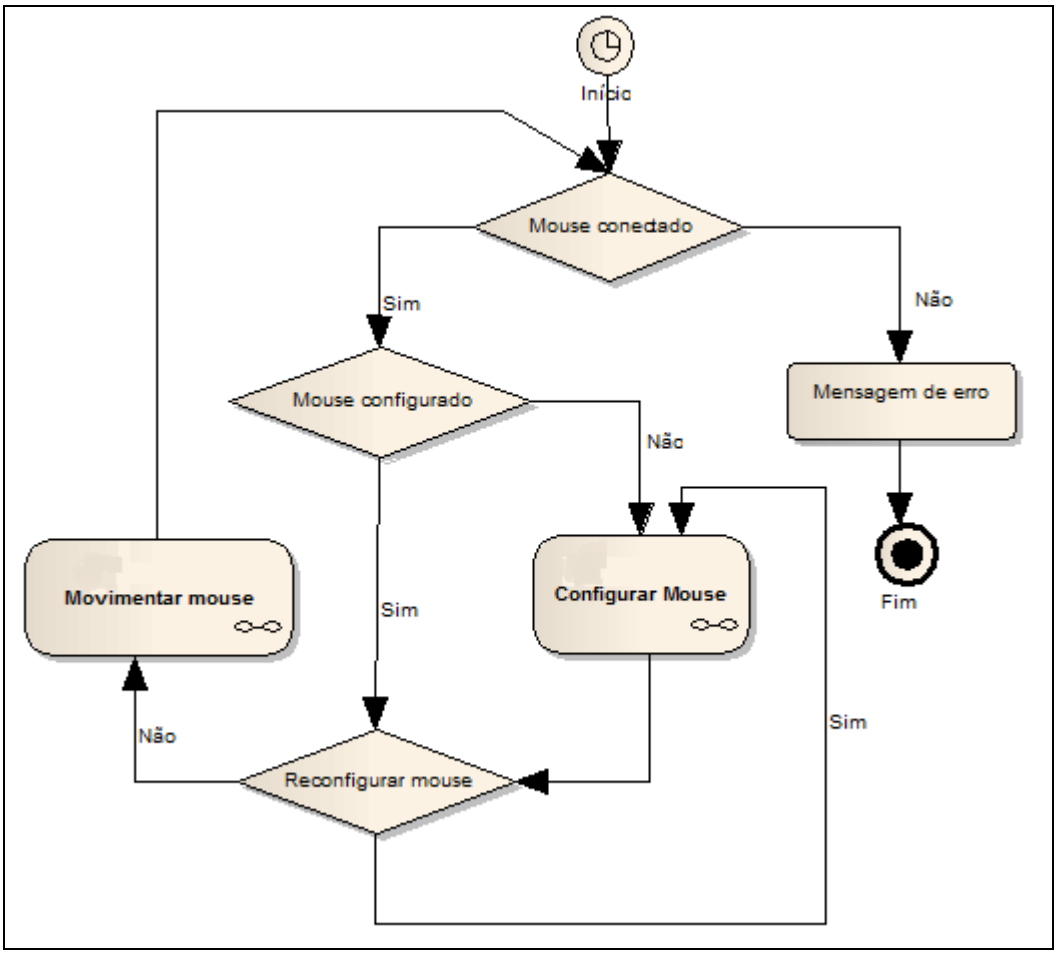


Figura 18 – Fluxograma geral do protótipo

Na Figura 19, esta representada a principal funcionalidade do protótipo: movimentar e realizar os cliques do *mouse*.

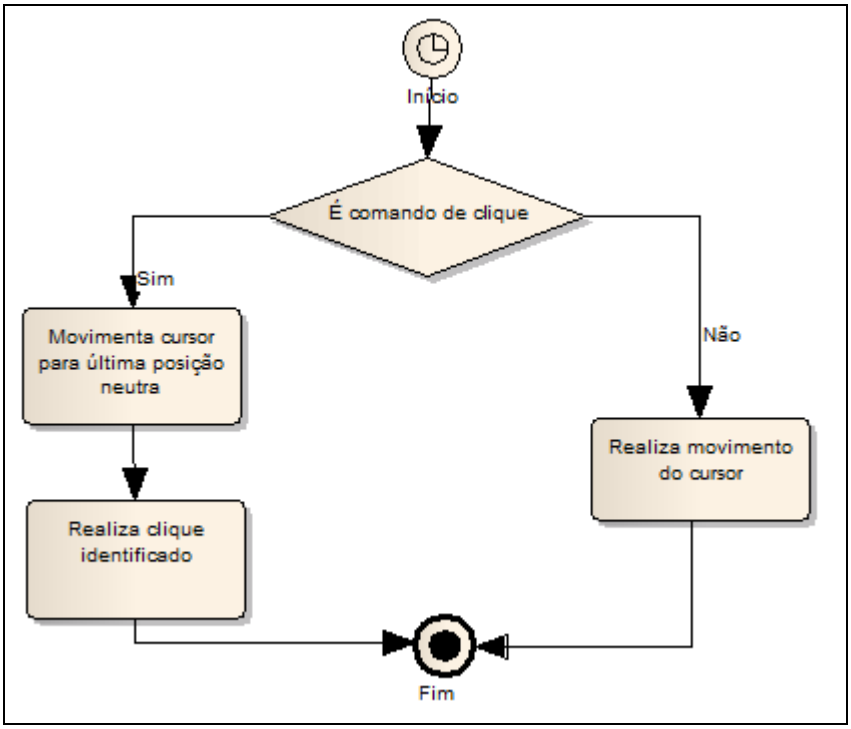


Figura 19 – Fluxograma da movimentação do cursor do *mouse*

Já na Figura 20, é apresentado de forma mais detalhada o processo de configuração do *mouse*, como pode ser notado através da Figura 18, o protótipo pode ser configurado no início do processo (caso não haja um arquivo de configurações) ou durante a sua execução.

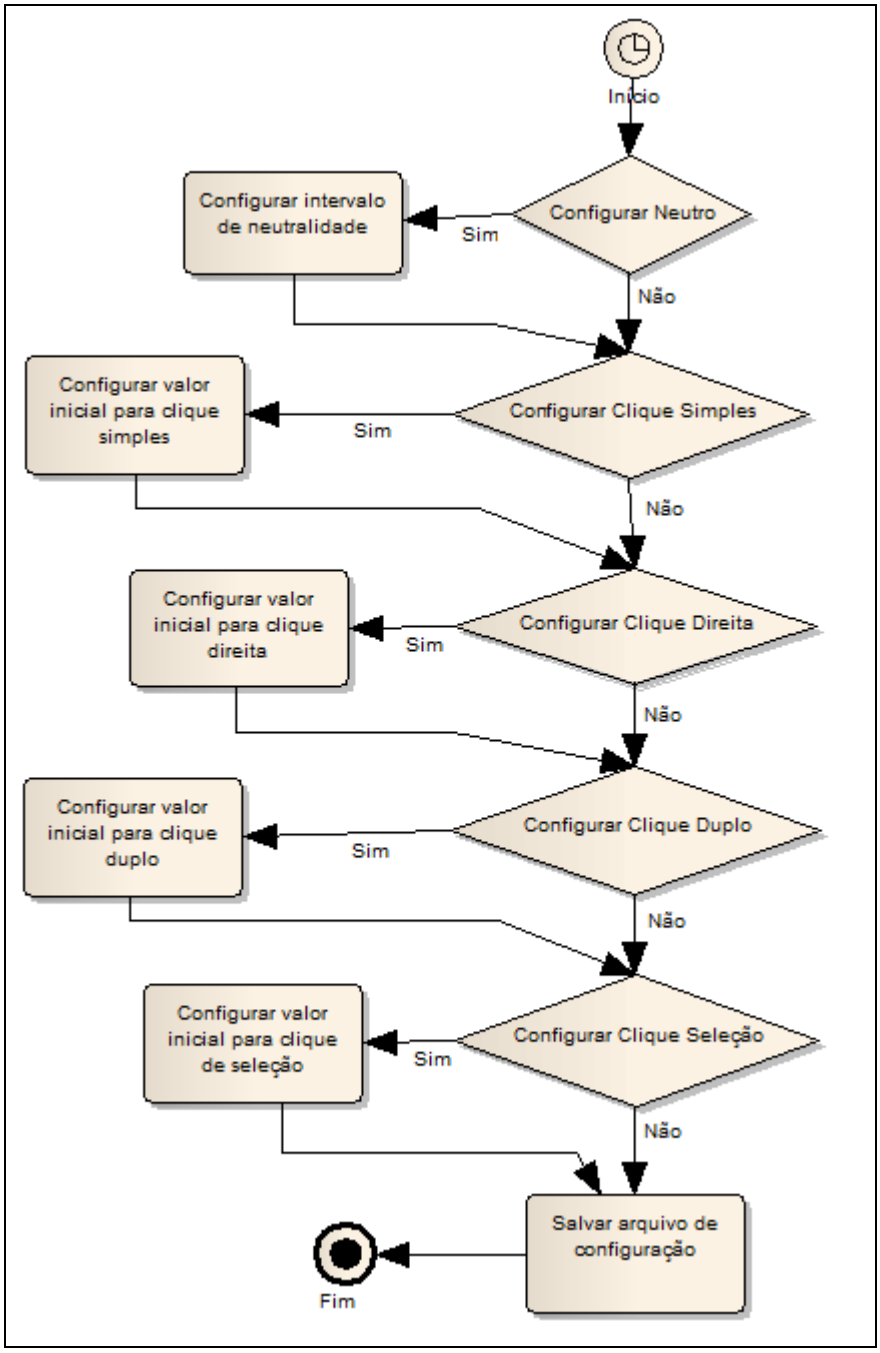


Figura 20 – Fluxograma das configurações do protótipo

3.3 IMPLEMENTAÇÃO

Nesta seção serão apresentadas informações sobre as técnicas e ferramentas utilizadas para a desenvolvimento do protótipo de *mouse*, bem como a sua implementação.

3.3.1 Técnicas e ferramentas utilizadas

O protótipo foi implementado na linguagem Java, seguindo o paradigma da orientação a objetos. Para a codificação foram utilizados dois ambientes de desenvolvimento (Eclipse e NetBeans) e duas bibliotecas (RXTX e AWT).

O Eclipse (versão Galileo) foi utilizado para a implementação de toda a camada de modelo e controle. Já o NetBeans (versão 6.8) foi utilizado na implementação da tela de configuração, por permitir uma ágil prototipação de janelas.

De acordo com Jarvi (1998), a *Application Programming Interface* (API) RXTX é desenvolvida pela comunidade de desenvolvedores baseando-se na biblioteca *Java Communication* (Javacomm), que é disponibilizada pela Sun. Ela é disponibilizada em forma de software livre, e licenciada pela *GNU's Not Unix* (GNU) *Lesser General Public License* (LGPL) e é multiplataforma. Atualmente encontra-se na versão 2.1.

A biblioteca *Abstract Window Toolkit (*AWT) é nativa do *Java Development* (JDK), além de conter um conjunto de recursos gráficos comumente conhecidos pelos sistemas de interface usando janelas, é também responsável por gerenciar esses componentes. Dentro desta biblioteca foi utilizada principalmente a classe Robot para realizar as manipulações dos eventos envolvendo o cursor do *mouse* (ORACLE, 2010).

3.3.2 Desenvolvimento do protótipo

O desenvolvimento do *mouse* foi dividido em três etapas: a comunicação serial com o SEN-00410, a manipulação do cursor pelos dados obtidos pela serial e a interface de configuração.

3.3.2.1 Etapa 1: comunicação serial com o SEN-00410

O protótipo foi desenvolvido utilizando-se de um notebook (que não possui portas seriais) fazendo-se necessário uso de um conversor USB – serial e da biblioteca RXTX para realizar as comunicações entre software e porta serial. Ao utilizar o SEN-00410, observou-se que o mesmo envia dados constantemente à porta serial, não necessitando que o software solicite os dados.

Considerando que o usuário pode conectar o protótipo em qualquer uma das portas seriais ou USBs, o código desenvolvido realiza uma busca por todas as portas seriais, listadas pelo sistema operacional, identificando as que estejam enviando informação com determinadas características. Com isso verificou-se a necessidade da utilização de uma *thread*, destinada a escutar cada uma das portas seriais, em busca da correta.

Como pode ser observado no Quadro 4, o método `findConnect()`, da classe `Serial`, é o responsável por percorrer todas as portas conectadas, identificando quais são do tipo serial. O Quadro 5 apresenta o método `connect(String portName)`, também da classe `Serial`, que conecta-se a cada uma das portas criando a *thread* para escuta-la. E na classe `SerialReader` o método `run()`. No Quadro 6 é apresentado o método que verifica se a informação recebida é a esperada e marca a porta como a correta.

```
1. public void findConnect() throws Exception {
2.   //cria uma lista com as portas identificadas pelo sistema operacional
3.    Enumeration<?> portList = CommPortIdentifier.getPortIdentifiers();
4.    ler = true;
5.   //percorre as portas
6.    while (portList.hasMoreElements()) {
7.       CommPortIdentifier portId = (CommPortIdentifier) portList
         .nextElement();
8.        //identifica se a porta é do tipo serial
9.       if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
10.         if (!ehPorta) {
11.              // chama método para conectar-se a porta
12.               if (connect(portId.getName())) {
13.                     porta = portId;
14.               } else {
15.                     porta = null;
16.               }
17.         }
18.      }
19.   }
20.}
```

Quadro 4 – Método `findConnect()` da classe `Serial`

```
1. boolean connect(String portName) throws Exception {
2.    CommPortIdentifier portIdentifier = CommPortIdentifier
            .getPortIdentifier(portName);
3.    // identifica se a porta esta disponivel
4.    if (portIdentifier.isCurrentlyOwned()) {
5.          return false;
6.    } else {
7.          CommPort commPort = portIdentifier.open(portName, 57600);
8.          // verifica se é porta do tipo serial
9.          if (!(commPort instanceof SerialPort)) {
10.               return false;
11.         }
12.         SerialPort serialPort = (SerialPort) commPort;
13.         serialPort.setSerialPortParams(9600,
                  SerialPort.DATABITS_8,
                  SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
14.        // cria uma thread para leitura da porta
15.         InputStream in = serialPort.getInputStream();
16.         serialReader = new SerialReader(in);
17.         Thread thread = new Thread(serialReader);
18          thread.start();
19.        // pausa a thread para dar tempo ao processo de identificacao
20.         thread.sleep(1000);
21.         if(!ehPorta){
22.               thread = null;
23.         }
24.         return ehPorta;
25.   }
26.}
```

Quadro 5 – Método `connect(String portName)` da classe `Serial`

```
1. public void run() {
      (...)
6.    // enquanto houver buffer vindo da porta verifica se é o esperado
7.           while ((len = this.in.read(buffer)) > -1) {
8.               le = (new String(buffer, 0, len));
9.              // verifica se contem o padrão 'X='
10.              if (!le.contains("X=")) {
11.                    ehPorta = false;
12.                    continue;
13.              }
14.              // e porta esperada inicia configuracao e movimentacao
15.              ControleConfiguracao.controle(mouse, le);
16.              ehPorta = true;
      (...)
25. }
```

Quadro 6 – Parte do método `run()` da classe `SerialReader`

3.3.2.2 Etapa 2: manipulação do cursor pelos dados obtidos pela serial

Ao estudar a classe `Robot` da biblioteca AWT, notou-se que para movimentar o *mouse* é utilizada a posição, em pixel, da tela onde se deseja posicionar o cursor. Sabe-se que os valores obtidos pelo acelerômetro referem-se à aceleração, por isso foi adaptada uma das fórmulas da cinemática, para realizar a conversão da aceleração e pixel, vide quadro .

A fórmula de conversão está dividida nos métodos `move(double valor, int positionOriginal)` e `controlePosition(double valor, int positionOriginal)` ambos pertencentes a classe `Coordenada`. O método `move`, Quadro 7, é responsável por identificar se a aceleração obtida, para a eixo em questão, está na faixa de neutralidade[6] (linha 6) ou não. E por tratar qual a direção da movimentação (linhas 18 e 26) e qual a diferença de aceleração entre a aceleração atual e a anterior. Quando o valor de aceleração é um valor dentro da faixa de neutralidade, a posição onde o cursor encontra-se, é armazenada para ser utilizada durante a realização do clique.

```
1. public void move(double valor, int positionOriginal) {
2.    (...)
5.    // verifica se o valor esta dentro da faixa de neutralidade
6.    if (valor >= neutroMin && valor <= neutroMax) {
7.         // grada posicao e habilita click
8.          positionBefore = positionNow;
9.          positionNow = positionOriginal;
10.         positionClick = positionOriginal;
11.         clik = true;
12.         return;
13.   }
14.
15.   double dif = 0;
16.   // verifica se valor acima da faixa
17.   // move na direção positiva do eixo
18.   if (valor > neutroMax) {
19.         dif = valor - neutroMax;
20.         dif = Math.abs(dif);
21.        // se click habilitado o desabilita
22.         if (clik) {
23.               clik = false;
24.         }
25.   }
26.   if (valor < neutroMin) {
            (...)
31.   }
32.   // guarda valor da aceleracao e da posicao
33.   acelerationAnt = acelerationNow;
34.   acelerationNow = dif;
35.   positionBefore = positionNow;
36.   positionNow = controlePosition(dif, positionOriginal);
37.}
```

Quadro 7 – Parte do método `move(double valor, int PositionOriginal)`

O método `controlePosition`, Quadro 9, realiza a conversão e controla para que a nova posição não seja maior ou menor que os limites da tela (linhas 5 e 7). A fórmula de conversão foi desenvolvida com base na fórmula física de cálculo de deslocamento a partir da aceleração, Quadro 8. Considerando que o deslocamento inicial e a aceleração inicial para o protótipo são zero, a fórmula adotada resumiu-se a $S = at^2/2$.

[6] Faixa de neutralidade são os valores de aceleração onde não deseja-se movimentar o cursor. Estes valores são obtidos durante a calibração.

```
S = So + Vot + at²/2
        Onde S = deslocamento final
             So = deslocamento inical
             Vo = velocidade inicial
             t = tempo
             a = aceleração
```

Quadro 8 – Fórmula do cálculo do deslocamento a partir da aceleração

Na fórmula adaptada (Quadro 9, linha 2) a constante `erro` guarda o valor em segundos do tempo entre cada uma das transmissões do acelerômetro. O argumento `valor` guarda o valor da aceleração obtida. Após vários testes concluiu-se que o resultado da fórmula deveria ser multiplicado por cinco mil, a fim de tornar o caminhamento do *mouse* rápido e ao mesmo tempo suave e controlável para pessoas com deficiência motora. Este valor, também, é baseado tempo entre cada uma das transmissões do *kit*.

```
1. private int controlePosition(double valor, int positionOriginal) {
2.    double s = ((erro * erro * valor) / 2) * 5000;
3.    int newPosition = positionOriginal + (int) s;
4.
5.    if (newPosition >= positionMax) {
6.          return positionMax;
7.    } else if (newPosition <= positionMin) {
8.          return positionMin;
9.    } else {
10.         return newPosition;
11.   }
12.}
```

Quadro 9 – Método `controlePosition(double valor, int positionOriginal)`

Para a realização de cliques, a classe `Robot` precisa ser informada sobre qual dos botões do *mouse* deve ser pressionado e liberado. Para que fosse possível realizar cliques utilizando-se apenas do acelerômetro sem demais componentes eletrônicos auxiliares, estipulou-se que durante o processo de configuração seriam definidos valores para cada um dos cliques. Inicialmente estipulou-se que esses valores pertenceriam ao eixo z. Mas ao longo do desenvolvimento notou-se que os valores obtidos para esse eixo possuem pouca variância, o que tornaria complexo a sua calibração e utilização. Optou-se então por valores pertencentes ao eixo x e y. Estes valores são o máximo da inclinação positiva e negativa que o usuário consegue atingir. Durante o processo de desenvolvimento e o processo de testes notou-se ser complexo para o usuário atingir sem ultrapassar e permanecer alguns instantes parado, com o acelerômetro posicionado na mesmo ponto do valor configurado. Alterou-se então, o protótipo de tal forma a permitir que um clique seja realizado quando o valor configurado fosse ultrapassado.

A partir dessas configurações notou-se que ao inclinar o protótipo até obter o valor do clique, ele realiza a movimentação respectiva a essa inclinação. Devido a essa movimentação, os cliques acabavam não sendo realizados sobre o ponto desejado. Para permitir que o clique seja realizado sobre o ponto de desejo do usuário, criou-se uma estrutura para armazenar a posição do cursor, toda vez que o valor obtido pertença à faixa de neutralidade. Assim para realizar um clique é necessário parar o *mouse* sobre o local e então inclinar o protótipo até obter o valor configurado para clique. Mesmo com o deslocamento do cursor, o clique será realizado sobre o último ponto onde o *mouse* esteve parado.

A classe responsável por controlar e identificar se os valores obtidos pelo acelerômetro correspondem a cliques denomina-se `Mouse`. Nesta classe existe o método `move(String le)`, Quadro 10, que é responsável por identificar se o valor lido corresponde à movimentação e realizá-la. Observando-se as linhas 23 a 25 nota-se que o processo de movimentação é muito simples e independente para as coordenadas x e y.

```
1. public void move(String le) {
2.    //Converte valor lido para valores reais separando por eixo
3.    Double[] medias = converte.media(le);
4.    // acha a posição atual do mouse
5.    Point p = java.awt.MouseInfo.getPointerInfo().getLocation();
6.
7.    // identifica se mouse nao pressionado - realizando seleção
8.    if (!mousePress) {
9.          realizaClick(medias);
10.   }
11.
12.   // se click para seleção - solta botao
13.   if (medias[1] <= click.clickSelecao) {
14.         if (!mouseClick) {
15.               mouseMoveClick();
16.               mouseRealise(InputEvent.BUTTON1_MASK);
17.               mouseClick = true;
18.               mousePress = false;
19.         }
20.
21.   } else {
22.         //movimenta
23.         x.move(medias[0], p.x);
24.         y.move(medias[1], p.y);
25.         mouseMove();
26.         mouseClick = false;
27.   }
28.}
```

Quadro 10 – Método `move(String Le)` pertencente à classe `Mouse`

Como pode ser observado na linha 9, este método chama o método `realizaClik(Double[] medias)` o qual identifica e realiza o cliques. Para identificar se é o *mouse* que está realizando o clique de seleção criou-se a variável de controle `mousePress` da classe `Control`. Enquanto esta variável estiver atribuída como `true`, o *mouse* só poderá

movimentar-se ou terminar a seleção. O método `move(String le)` também é responsável por realizar a liberação, soltar botão, quando o *mouse* estiver marcado com seleção e o valor lido corresponder a função de seleção (linha 13 a 19).

No Quadro 11, método `realizaClik(Double[] medias)`, nota-se a existência da variável de controle `mouseClick` da classe `Control`. Esta variável é responsável por não permitir que duas funções de clique sejam executadas em seguida, sem que haja uma movimentação entre elas. É este método que compara o valor lido com os valores calibrados, para cada um dos cliques, e caso o valor lido corresponda a algum clique o realiza (linhas 2, 14, 26 e 41).

```
1. void realizaClick(Double[] medias) {
2.    if (medias[0] <= click.clickSimples) {
3.          //valor representa clique simples
4.          if (!mouseClick) {
5.                // move para posição do clique
6.                mouseMoveClick();
7.                //clique simples
8.                mousePress(InputEvent.BUTTON1_MASK);
9.                mouseDelay(10);
10.               mouseRealise(InputEvent.BUTTON1_MASK);
11.               mouseClick = true;
12.         }
13.
14.   } else if (medias[0] >= click.clickDireita) {
15.         //valor representa clique direita
16.         if (!mouseClick) {
17.               // move para posição do clique
18.               mouseMoveClick();
19.               // clique direita
20.               mousePress(InputEvent.BUTTON3_MASK);
21.               mouseDelay(10);
22.               mouseRealise(InputEvent.BUTTON3_MASK);
23.               mouseClick = true;
24.         }
25.
26.   } else if (medias[1] >= click.clickDuplo) {
27.         //valor representa clique duplo
28.         if (!mouseClick) {
29.               // move para posição do clique
30.               mouseMoveClick();
31.               // clique duplo
32.               mousePress(InputEvent.BUTTON1_MASK);
33.               mouseDelay(5);
34.               mouseRealise(InputEvent.BUTTON1_MASK);
35.               mousePress(InputEvent.BUTTON1_MASK);
36.               mouseDelay(5);
37.               mouseRealise(InputEvent.BUTTON1_MASK);
38.               mouseClick = true;
39.         }
40.
41.   } else if (medias[1] <= click.clickSelecao) {
42.         //valor representa clique de seleção (inicial)
43.         if (!mouseClick) {
44.               // move para posição do clique
45.               mouseMoveClick();
46.               //inicial seleção
47.               mousePress(InputEvent.BUTTON1_MASK);
48.               mouseClick = true;
49.               mousePress = true;
50.         }
51.   }
52.}
```

Quadro 11 – Método `realizaClick(Double[] medias)` pertencente à classe `Mouse`

3.3.2.3 Etapa 3: interface gráfica

Como já mencionado, o processo de configuração do protótipo pode ocorrer de duas formas: logo no início quando o *mouse* for detectado ou durante a utilização. Para a reconfiguração durante o processo de utilização foi desenvolvida uma tela com botões e dois *speedometer* para tornar a configuração mais intuitiva.

A Figura 21 apresenta a interface gráfica gerada. Esta tela foi desenvolvida na classe `TelaConf` e na classe `Speedometer`. Para a realização das ações dos botões a classe `TelaConf` chama a ação respectiva na classe ControleConfiguração.

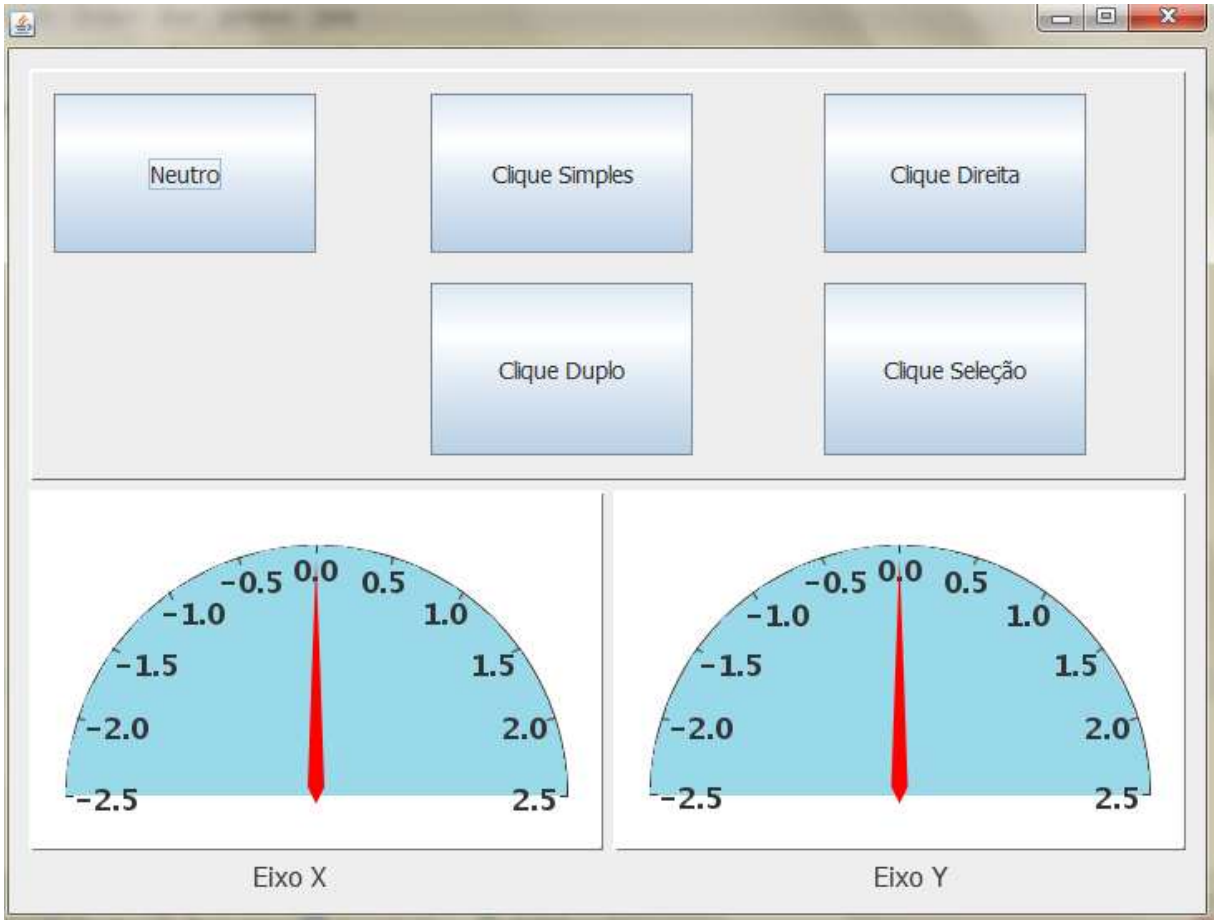


Figura 21 – Tela de reconfiguração do *mouse*

O Quadro 12 apresenta o método `calibrarEsquerdaSimples(Mouse mouse, String le)`, que é responsável por calibrar o valor a partir do qual deve ser realizado o clique simples. Esse método está definido na classe `ControleConfiguracao`. O processo de calibração do clique é realizado em duas etapas. Na primeira etapa, o sistema envia uma mensagem ao usuário instruindo-o a realizar determinada movimentação (linha 3). Já na segunda etapa é realizada a calibração (linha 6) e verificado se o valor calibrado é válido.

```
1. public static void calibrarEsquerdaSimples(Mouse mouse, String le) {
2.    if (mensagem) {
3.          JOptionPane.showMessageDialog(null,
                  "Incline o mouse para o máximo da esquerda. Este será seu
click simples");
4.          mensagem = false;
6.          return;
7.    }
8.
9.    mouse.calibrarEsquerdaSimples(le);
10.   if(mouse.click.calibrarEsquerda()){
11.         JOptionPane.showMessageDialog(null, "Não foi possível calibrar
            o clique simples. Aguarde estamos tentando novamente." );
12.   }else{
13.         JOptionPane.showMessageDialog(null, "O clique simples foi
            calibrado com sucesso." );
12.   }
13.   mensagem = true;
14.   return;
15.}
```

Quadro 12 – Método `calibrarEsquedaSimples(Mouse mouse,String le)` pertencente à classe `ControleConfiguracao`

3.3.3 Operacionalidade da implementação

Ao iniciar a utilização do protótipo será solicitada a configuração do mesmo, caso não seja identificado o arquivo de configurações. Este processo resume-se a movimentar o protótipo como solicitado pelo sistema. Para exemplificar é apresentado o processo de configuração da faixa de neutralidade e do clique de simples da esquerda:

a) o sistema apresenta a mensagem da Figura 22;

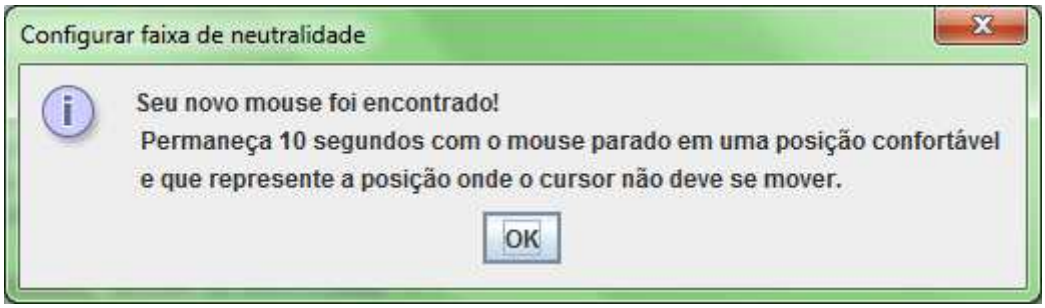


Figura 22 – Solicitação de calibração da faixa de neutralidade

b) o usuário se posiciona de forma que esta seja a posição na qual o *mouse* não deve realizar nenhum movimento. Esta posição deve ser confortável para o usuário;

c) o sistema identifica os valores obtidos nesta posição e apresenta a mensagem de sucesso apresentada na Figura 23;

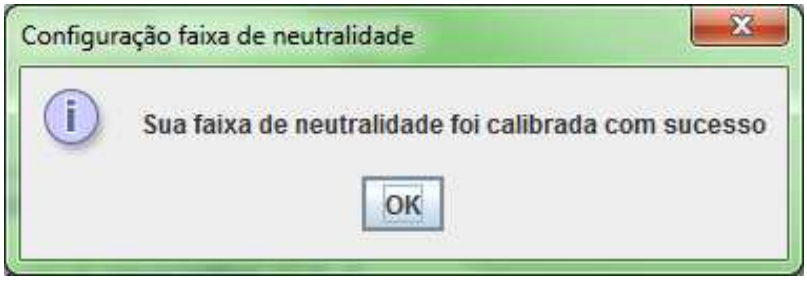


Figura 23 – Mensagem de sucesso após calibração da faixa de neutralidade

d) o sistema apresenta a mensagem da Figura 24 solicitando a inclinação do protótipo para a esquerda;

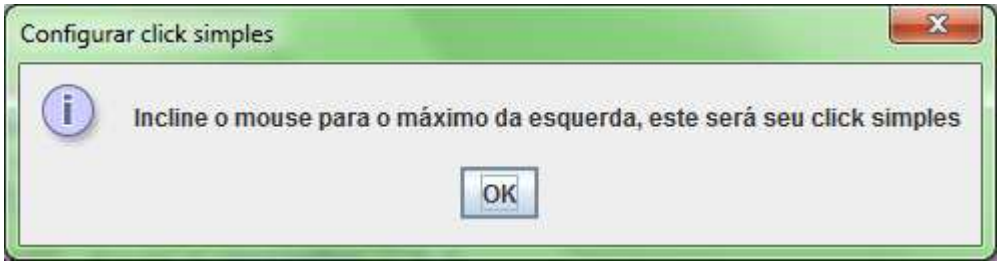


Figura 24 – Solicitação de calibração do clique simples

e) o usuário irá inclinar o protótipo para a esquerda, visando atingir o máximo. Essa inclinação irá representar o clique simples da esquerda, por isso a inclinação não deve ser muito próxima a faixa de neutralidade e nem difícil de ser repetida pelo usuário;

f) o sistema identifica o valor e apresenta mensagem de sucesso, semelhante a apresentada na Figura 23.

Para as demais calibrações o processo é semelhante alterando-se apenas as mensagens apresentadas pelo sistema e o tipo de movimentação que o usuário deve realizar. O clique da direita é calibrado se inclinado o protótipo para a direita, o clique duplo inclinado para cima e o clique de seleção inclinado para baixo. Assim que o *mouse* estiver totalmente calibrado ele irá movimentar-se conforme a inclinação lida.

Para movimentar o cursor para a esquerda é necessário que o protótipo seja inclinado para a esquerda sem alcançar o valor cadastrado para o clique simples da esquerda. Para movimentar o *mouse* para a direita o inclina-se para direita sem atingir o valor cadastrado para o clique da direita. E assim sucessivamente.

Desejando-se abrir uma pasta situada no *desktop,* por exemplo, o *mouse* deve ser posicionado sobre essa pasta. Assim que o cursor estiver parado sobre a pasta, inclina-se a protótipo para cima buscando o valor configurado como clique duplo. Até ser alcançado esse valor o *mouse* irá deslocar-se para cima. Assim que o valor for alcançado o cursor se posiciona sobre a pasta e realiza o clique duplo, abrindo-a.

Desejando-se recalibrar o valor do clique simples durante a utilização do *mouse*, o usuário deve abrir o arquivo denominado `MouseAcelerometro`[7] situado no *desktop* e apertar o botão correspondente `Clique Simples`. Ao pressionar o botão o sistema irá apresentar a mensagem da Figura 24 e são repetidos os itens **e** e **f**.

3.3.4 Testes

Durante o desenvolvimento do *mouse* foram desenvolvidos alguns testes com cinco pacientes da clínica de fisioterapia da FURB. Os pacientes escolhidos para participar desse processo são portadores de hemiplegia espástica, com diferentes graus de comprometimento. Dois desses pacientes tiveram a oportunidade de utilizarem o protótipo durante mais de uma hora e em mais de uma oportunidade. Observa-se que realização dos testes não seguiu nenhuma metodologia específica apenas baseou-se na observação dos pacientes.

O teste inicia-se com uma breve explicação do funcionamento do protótipo e de seu objetivo, em seguida foi realizada a calibração do *mouse*.

Na utilização do protótipo, foi sugerido ao paciente que movimentasse o cursor pela tela, sem realizar nenhum clique, para tentar adaptar-se a funcionamento do mesmo. Após cinco minutos sugeriu-se que o paciente tenta-se abrir uma pasta ou arquivo localizado no *desktop*. Conforme o desempenho apresentado pelo usuário foram sugeridas outras atividades como: trocar o papel de parede, abrir um programa localizado no menu iniciar; fechar um programa; selecionar partes de um texto.

Cada paciente foi submetido ao teste primeiramente com a mão onde os sintomas da hemiplegia estão presente, em seguida com a mão sem os sintomas. E novamente na mão com sintomas.

Ao utilizar o protótipo na mão sem os sintomas da hemiplegia, pode-se notar que os pacientes precisaram de mais tempo para adaptar-se ao funcionamento do protótipo visto que estes utilizam o *mouse* tradicional com esta mão. Também notou-se que a realização dos cliques foi mais fácil. Já na mão com os sintomas, o tempo de adaptação foi menor e a movimentação foi mais dinâmica. Quanto a realização de cliques observou-se uma grande dificuldade, e que em muitos momentos o clique não era realizado. Essa dificuldade é divida à vários fatores tais como: o cursor movimentar-se antes de realizar cliques, e do paciente ter

[7] Esse arquivo é a interface de configuração apresentada na Figura 21.

calibrado o clique em uma inclinação difícil dele atingir normalmente. Na segunda vez em que o protótipo foi utilizado nesta mão o paciente já estava acostumado com o funcionamento e conseguiu realizar mais cliques de uma forma menos dificultosa.

Após os testes e em conversas durante os testes os pacientes sugeriram que os cliques fossem controlados por outro componente como um botão ou régua de pressão. E também afirmaram ser fácil a movimentação do cursor com o acelerômetro.

3.4 RESULTADOS E DISCUSSÃO

Os resultados obtidos com a realização deste trabalho são satisfatórios no que tange a utilização do acelerômetro para a realização da movimentação do *mouse*.

Ao iniciar o desenvolvimento deste protótipo almejava-se que a movimentação do cursor fosse semelhante à movimentação do *mouse* tradicional e a movimentação realiza em controle de vídeo games tipo *joystick*. No decorrer do desenvolvimento notou-se que o SEN-00410 não é capaz de calcular a aceleração aplicada sobre ele. Ele apenas calcula o valor proporcional da aceleração da gravidade aplicada sobre sua inclinação. Devido a isso limitou a movimentação do *mouse* apenas a inclinação do protótipo, o que resulta na necessidade de treinamento ao utilizá-lo.

Nos testes realizados com pacientes, foi observado certa dificuldade inicial de adaptação ao controle dos movimentos através da inclinação do protótipo. Após certo tempo de uso nota-se a total adaptação do usuário ao protótipo. A dificuldade inicial, deve-se em parte ao fato dos pacientes estarem a costumados ao uso do *mouse* tradicional, onde para realizar um movimento é necessário movimentar o aparelho e não incliná-lo.

Durante os testes pode-se notar uma dificuldade constante, mesmo após treinamento e certo tempo de uso, em todos os pacientes na realização dos cliques. Essa dificuldade é mais visível nos cliques da direita e da esquerda. Observando-se os processos de calibração e a utilização, notou-se que o usuário confundia-se com a movimentação realizada até atingir-se o valor do clique e poucas vezes conseguiam atingir o valor calibrado.

Pela observação e entrevista com os pacientes verificou-se a necessidade de utilizar algum outro dispositivo para a realização dos cliques, o que facilitaria e simplificaria a utilização do protótipo. Observou-se também, a possibilidade de configurar valores de velocidades independentes para cada uma das direções, visto que alguns paciente possuem

graus de dificuldades diferentes para cada uma das inclinações. Outra observação que deve ser mencionada é o fato de que quanto maior o grau da lesão no paciente, maior a dificuldade de movimentar o protótipo utilizando-o afixado ao pulso. Além disso o tempo de uso provoca certo cansaço e dor no membro.

Em relação aos *mouses* adaptados para pessoas com deficiência, nota-se que o protótipo é altamente sensível durante a movimentação, o que pode ser considerado uma vantagem, mas a realização dos cliques é muito mais complexa independente de qual *mouse* for usado como referencia.

4 CONCLUSÕES

O computador evoluiu de mera ferramenta de trabalho, para uma ferramenta completa, usada nos estudos, no lazer e na socialização. Sendo quase imprescindível em nossa rotina. Mas o computador não está adaptado a atender pessoas com algum tipo de deficiência motora nos membros superiores como é o caso das pessoas com hemiplegia espástica. Nestas pessoas a mobilidade do corpo é limitada, o que torna a utilização do *mouse* tradicional complexa e algumas vezes cansativa. Pensando-se em auxiliar estas pessoas foi desenvolvido um protótipo de *mouse* utilizando o acelerômetro como único meio de interações entre o usuário e o sistema.

No início do desenvolvimento do protótipo foram elencadas algumas funcionalidades que ele deveria ter, como: realizar a movimentação do cursor do *mouse*, realizar os comandos de cliques e possuir uma interface de configuração. Ao término pode-se observar que as funcionalidades foram criadas, atingindo os objetivos.

Durante o desenvolvimento deste foram enfrentadas algumas dificuldades. A primeira foi constatar que a biblioteca Javacomm, não é capaz de realizar a comunicação do software com a porta serial através de um conversor USB - serial, e conseqüente busca por outra biblioteca que fosse baseada na Javacomm, mas permitisse a utilização deste conversor. Ao encontrar a API RXTX necessitou-se configurar os sistemas operacionais (Windows e Linux) para utilizá-la de forma correta.

Outra dificuldade foi a incapacidade do SEN-00410 de calcular a aceleração aplicada sobre ele. O que resultou na reformulação do padrão adotado para a movimentação, impedindo que o protótipo realizasse a movimentação referencial[8].

Houve também dificuldade na implementação dos cliques. Inicialmente almejava-se a realização deles através da coordenada z. Mas ao notar-se que esta possui uma pequena variância optou-se por utilizar as coordenadas x e y. Essa escolha acabou resultando em outra dificuldade, como reconhecer que um determinado valor identifica um clique e não uma movimentação. Para isso foi definido que os valores máximo e mínimo da inclinação alcançada por cada usuário representem os cliques. E que estes valores possam ser calibrados.

Durante os testes identificou-se que após uma dificuldade inicial de adaptação, os usuários conseguem realizar a movimentação do cursor com certa facilidade. E que apesar das

[8] Entende-se por referencial, a movimentação do cursor semelhante à movimentação realizada no acelerômetro.

dificuldades encontradas em relação à realização dos cliques, a utilização do acelerômetro pode ser uma alternativa viável. Bastando a utilização de algum outro dispositivo para auxiliar na realização dos cliques, deixando o acelerômetro unicamente responsável por controlar a movimentação do *mouse.*

O protótipo se mostrou uma alternativa viável para o desenvolvimento de *mouses* adaptados. No entanto, existem limitações no protótipo, como a necessidade de instalação de um software para reconhecimento e utilização do *mouse*, a movimentação ser apenas contínua e ser difícil a realização dos cliques para a pessoa com deficiência.

4.1 EXTENSÕES

Existem pontos que podem ser agregados ou melhorados no protótipo. Como sugestão pode-se citar:

a) substituir o SEN-00410 por outro kit que possua, além da leitura da aceleração gravitacional aplicada sobre o kit, a leitura das demais acelerações aplicadas sobre ele ou o resultado da somas das acelerações;
b) adicionar ao protótipo físico outro componente eletrônico que fique responsável por identificar cliques, deixando o acelerômetro responsável apenas pela movimentação;
c) testar o protótipo em portadores de outras deficiências motoras para identificar se ele não atenderia suas necessidades;
d) montar um hardware que dispense a instalação de qualquer software no sistema operacional e que se conecte a USB;
e) permitir a calibração de velocidades de deslocamento independentes para cada uma das inclinações.

REFERÊNCIAS BIBLIOGRÁFICAS

ABLENET. **Cruise adapted trackpad.** Roseville, 2008. Disponível em: <http://www.ablenetinc.com/Store/tabid/205/Default.aspx?CategoryCode=113>. Acesso em: 20 mar. 2010.

ALVES, Manoel F. Alunos desenvolvem mouse para deficientes. **Jornal da Unicamp**, Campinas, 1 a 7, dez. 2003. p. 4.

ANDRADE, Ana C. C. **Póstero-depressão escapular.** [S. l.]. Disponível em: <http://br.monografias.com/trabalhos/postero/postero.shtml#ESCAPULA>. Acesso em: 25 ago. 2010.

ASSOCIAÇÃO BRASILEIRA DE MEDICINA FÍSICA E REABILITAÇÃO. Espasticidade: avaliação clínica. In: ASSOCIAÇÃO MÉDICA BRASILEIRA, CONSELHO FEDERAL DE MEDICINA. **Projetos Diretrizes**. São Paulo: Associação Médica Brasileira, 2006. 5 v., cap. 18. Disponível em: <http://www.projetodiretrizes.org.br/5_volume/18-Espasticid.pdf>. Acesso em: 20 ago. 2010.

BERCITO, Diogo. Mouse em forma de óculos ajuda deficientes físicos. **Folha online,** São Paulo, 15 mar. 2010. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u707065.shtml >. Acesso em: 26 mar. 2010.

BRASIL. Decreto-lei n 5.296, de 02 de dezembro de 2004. Regulamenta prioridade de atendimento, estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida. Brasília, 2004. Disponível em: <http://www.planalto.gov.br/ccivil/_ato2004-2006/2004/Decreto/D5296.htm>. Acesso em: 20 mar. 2010.

BOBATH, Berth; BOBATH, Karel. **Desenvolvimento motor nos diferentes tipos de paralisia cerebral**. Tradução Elaine Elizabetsky. São Paulo: Manole, 1989.

CLICK TECNOLOGIA ASSISTIVA. **Produtos para acessibilidade ao computador**. Porto Alegre, 2010. Disponível em: <http://www.clik.com.br/clik_01.html>. Acesso em: 19 mar. 2010.

COELHO, Guilherme. **Acelerômetros**: o truque da detecção dos movimentos na nova geração de celulares. [S.l.], 2007. Disponível em: <http://webinsider.uol.com.br/2007/11/29/acelerometros/>. Acesso em: 20 mar. 2010.

DAVES, Patricia M. **Exatamente no centro**. Tradução Nelson Gomes de Oliveira. São Paulo: Manole, 1996.

DAVES, Patricia M. **Passos a seguir**. Tradução Nelson Gomes de Oliveira. São Paulo: Manole, 1996.

DUARTE, Edison; GORLA, José I. Pessoas com deficiência. In: GORLA, José I.; CAMAPANA, Matheus B.; OLIVEIRA, Luciana Z. (Org.). **Teste e avaliação em esportes adaptados**. São Paulo: Phorte, 2009. cap. 1, p. 23-27.

FERREIRA, Edenilson L.; RAMOS, Larissa F. **Sistema de arquivo no Linux**. Cuiaba, 2006. Trabalho não publicado. Disponivél em: <http://www.getec.cefetmt.br/~ruy/pos-graduacao/SO/trabalhos_alunos/Sistema_de_Arquivos_no_Linux-Edenilson-Larissa.pdf>. Acesso em: 17 maio 2010.

FREITAS, Kelly P. **Deficiência motora**. Santos, 2009. Disponível em: <http://comunidadeaacd.ning.com/profiles/blog/show?id=2406047%3ABlogPost%3A44305> . Acesso em: 23 mar. 2010.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Censos demográficos**. [Brasília], 2010. Disponível em: <http://www.ibge.gov.br/home/estatistica/populacao/censo2000/default_prim_resultados.shtm >. Acesso em: 23 mar. 2010.

JARVI, Keane. **RXTX**: the prescription form transmission. [S.l.], 1998. Disponível em: <http://www.rxtx.org>. Acesso em: 30 maio 2010.

MADENTEC. **TrackerPro**. Edmonton, Canada, 2009. Disponível em: <http://www.madentec.com/products/tracker-pro.php>. Acesso em: 20 mar. 2010.

MURRAY, William H.; PAPPAS, Chris H. **Programação para Windows 3.0**. Tradução: Nilson Martinho. São Paulo: Makron Books, 1992.

ORACLE. **Class Robot**: javadoc. [S.l.], 2010. Disponível em <http://download.oracle.com/javase/1.4.2/docs/api/java/awt/Robot.html>. Acesso em: 27 jul. 2010.

QUADROS, Cleber L. S. M.; SAMPAIO, Alexandre. O.; CARVALHO, Filipe. **Óculos-mouse**: projeto de criação de um mouse para pessoas com deficiência físico-motora. [São Paulo], 2009. Disponível em: <http://febrace.org.br/virtual/ENG/104/>. Acesso em: 20 mar. 2010.

REZINI, Robson V. **Protótipo de um game embarcado em plataforma ARM dotado de acelerômetro**. 2008. 63 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SPARKFUN ELECTRONICS. **SerAccel**: a self contained triple-axis acclerometer. Boulder, 2005. Disponível em <http://www.sparkfun.com/datasheets/Accelerometers/SerAccel-v5.pdf>. Acesso em: 01 jun. 2010.

SPARKFUN ELECTRONICS. **Serial accelerometer tri-axial**: enclosed. Boulder, [2010]. Disponível em: <http://www.sparkfun.com/commerce/product_info.php?products_id=410>. Acesso em: 10 fev. 2010.

SOUZA, David J. **Desbravando o PIC**: ampliado e atualizado para PIC 1F628A. 9. ed. São Paulo: Érica, 2005.

SOUZA, Pedro A. **O esporte na paraplegia e tetraplegia**. Rio de Janeiro: Guanabara Koogan, 1994.

UNITED STATES NAVY. **Sincros servomecanismos e fundamentos de giros**. Tradução Centro de Instrumentação Almirante Wandenlolk. [São Paulo]: Hemus, 2004. Disponível em: <http://books.google.com.br/books?id=YXAyD7pUCSUC&printsec=frontcover&dq=Sincros+servomecanismos+e+fundamentos+de+giros&hl=pt-BR&ei=oZqrS4e_EMKVtge7xNG-Dw&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCsQ6AEwAA#v=onepage&q=&f=false>. Acesso em: 18 mar.2010.

WEB ACCESSIBILITY IN MIND. **Motor disabilites**. Logan, Utah, USA, 2010. Disponível em: <http://www.webaim.org/articles/motor/>. Acesso em: 24 mar. 2010.