UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE *MOUSE* UTILIZANDO ACELERÔMETROS: VERSÃO 2

LUIZ ROBERTO LEICHT

BLUMENAU
2011

2011/2-16

LUIZ ROBERTO LEICHT

PROTÓTIPO DE *MOUSE* UTILIZANDO ACELERÔMETROS: VERSÃO 2

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
2011

2011/2-16

PROTÓTIPO DE *MOUSE* UTILIZANDO ACELERÔMETROS: VERSÃO 2

Por

LUIZ ROBERTO LEICHT

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. Antônio Carlos Tavares, Mestre – FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB

Blumenau, 12 de dezembro de 2011.

Dedico este trabalho a todos que de alguma forma contribuíram e me apoiaram durante a sua realização.

AGRADECIMENTOS

À minha família, que sempre me apoiou e esteve presente em minha vida.

À minha namorada. Josiane, pela paciência e compreensão nos momentos em que estive dedicado à conclusão deste trabalho.

Ao Marcelo e sua esposa Juliana, que se disponibilizaram e colaboraram com os testes.

Ao meu orientador Miguel A. Wisintainer, que me incentivou e acreditou na conclusão deste trabalho.

A distinção entre passado, presente e futuro é apenas uma ilusão teimosamente persistente.

Albert Einstein

RESUMO

Este trabalho descreve o aperfeiçoamento de um *mouse* utilizando acelerômetro, desenvolvido e adaptado para pessoas portadoras de deficiência motora nos membros superiores. Para isso foi utilizado um acelerômetro capaz de medir acelerações e inclinações e uma placa micro-controladora responsável por interpretar os sinais enviados pelo acelerômetro, o qual aplica algoritmos de estabilização e simulação de cliques. Para a implementação deste protótipo, foi utilizado o ambiente de desenvolvimento Visual Studio 2010 em conjunto com a biblioteca .NET Micro Framework 4.1 na linguagem C# com a versão .NET 4.0.

Palavras-chave: Acelerômetro. *Mouse*. Deficiência motora. Software. Periféricos.

ABSTRACT

This work describes the development of a mouse using accelerometer, developed and adapted for people with physical disability in upper limbs. For this was used an accelerometer capable of measuring accelerations and inclinations and a micro-controller board responsible for interpreting the signals sent by the accelerometer, which applies stabilization algorithms and simulation clicks. To implement this prototype, was used the Visual Studio 2010 in conjunction with the library .NET Micro Framework 4.1 in C# with the .NET 4.0.

Key-words: Accelerometer. Mouse. Motor disabilities. Software. Peripherals.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ARM - *Advanced RISC Machine*

FEZ - *Freakin' Easy*

HID - *Human Interface Device*

IDE - *Integrated Development Environment*

LED - *Light Emitting Diode*

NETMF - *NET Micro Framework*

PCD - Pessoas Com Deficiência

RF - Rádio Frequência

RISC - *Reduced Instruction Set Computer*

SD - *Secure Digital*

SO - Sistema Operacional

UML - *Unified Modeling Language*

USB - *Universal Serial Bus*

XML - *Extensible Markup Language*

W3C - *World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

O mundo tem passado por uma constante evolução tecnológica, criando dispositivos de comunicação cada vez mais ágeis e úteis para seus usuários. Neste processo, o computador evoluiu de uma simples ferramenta de trabalho, para um meio de comunicação, entretenimento e socialização, de grande importância para a sociedade, podendo ser utilizado por todas as faixas etárias, raças e etnias do planeta.

Infelizmente, grande parte das pessoas portadoras de deficiência física não pode utilizar um computador, *mouse* ou teclado, porque estes não foram desenvolvidos ou adaptados para serem utilizados por PCD. Segundo Instituto Brasileiro de Geografia e Estatística (2010), o censo de 2000 indica que aproximadamente 24,5 milhões de pessoas apresentaram algum tipo de incapacidade ou deficiência. São pessoas que possuem ao menos alguma dificuldade de enxergar, ouvir, locomover-se ou com alguma deficiência física ou mental.

Diante do exposto, e com o intuito de contribuir para a inclusão digital para deficientes físicos, foi proposto por Jennrich (2010) um protótipo de *mouse* utilizando um acelerômetro, onde uma pessoa que possua alguma deficiência motora nos membros superiores possa utilizar o *mouse* sem a necessidade de utilizar as mãos. O *mouse* pode ser colocado, por exemplo, na cabeça, onde são feitos leves movimentos pelo usuário que se refletem na movimentação do cursor na tela do computador.

O trabalho aqui apresentado é uma reimplementação baseando-se nos requisitos funcionais do trabalho de Jennrich (2010), com o intuito de desenvolver um protótipo com melhor desempenho, maior compatibilidade para com os SO e maior facilidade de utilização para o usuário final. Para isso a nova implementação utilizou o ambiente de desenvolvimento Visual Studio 2010 em conjunto com a biblioteca .NETMF 4.1 na linguagem C# na versão .NET 4.0, um acelerômetro eZ430-Chronos capaz de medir acelerações e inclinações, e uma placa micro-controladora FEZ Domino responsável por receber os sinais do acelerômetro através de um receptor e interpretá-los aplicando algoritmos desenvolvidos para estabilização e simulação de cliques do *mouse*.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi aperfeiçoar o protótipo de dispositivo para apontamento (*mouse*) desenvolvido por Jennrich (2010).

Os objetivos específicos do trabalho são:

a) montar um novo hardware utilizando a placa FEZ Domino e manter as funcionalidades já desenvolvidas no trabalho anterior;
b) utilizar o acelerômetro eZ430-Chronos para permitir medir acelerações e inclinações;
c) permitir a comunicação do protótipo via conexão USB;
d) permitir a conexão do protótipo a qualquer sistema operacional;
e) permitir a calibração da velocidade de deslocamento do ponteiro do *mouse*;
f) aprimorar a forma de como os cliques do *mouse* são realizados.

1.2 ESTRUTURA DO TRABALHO

A estrutura deste trabalho está dividida em quatro capítulos. O primeiro capítulo apresenta uma introdução ao tema abordado, objetivos e estrutura do trabalho. No segundo capítulo são abordados os conceitos e tecnologias utilizadas. O terceiro capítulo apresenta os requisitos do protótipo assim como implementação, utilização e os resultados obtidos. Por fim, no quarto capítulo são apresentadas as conclusões, limitações e possíveis extensões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são explanados os principais assuntos relacionados ao desenvolvimento do trabalho: problema da deficiência física e suas principais causas, biblioteca .NET Micro Framework 4.1, placa FEZ Domino, XML, acelerômetro eZ430-Chronos, definições para USB-HID, protótipo de *mouse* utilizando acelerômetros (JENNRICH, 2010) e os trabalhos correlatos.

2.1 DEFICIÊNCIA FÍSICA

> [Considera-se deficiência física] alteração completa ou parcial de um ou mais segmentos do corpo humano, acarretando o comprometimento da função física, apresentando-se sob a forma de [...] paralisia cerebral, nanismo, membros com deformidade congênita ou adquirida, exceto as deformidades estéticas e as que não produzam dificuldades para o desempenho de funções. (BRASIL, 2004).

Desta forma, são considerados portadores de deficiência física, todos os indivíduos que apresentam algum comprometimento em sua capacidade motora, de acordo com os padrões considerados normais para a espécie humana (MACIEL, 1998, p. 52).

A deficiência física motora nos membros superiores é observada geralmente nos indivíduos com lesão do sistema nervoso central. Esta lesão pode provocar um distúrbio conhecido como espasticidade[1]. Este distúrbio aparece em diferentes doenças como, por exemplo, paralisia cerebral, lesão medular e a lesão encefálica (ASSOCIAÇÃO BRASILEIRA DE MEDICINA FÍSICA E REABILITAÇÃO, 2006, p. 3).

As causas da deficiência física são muitas e complexas, muitas vezes não existindo uma causa única. Dentre as mais comuns, podem-se citar fatores genéticos, fatores virais ou bacterianos, fatores neonatais e fatores traumáticos (especialmente os medulares) (DEFICIÊNCIA..., 2010).

[1] Espasticidade "é quando ocorre um aumento do tônus muscular, envolvendo hipertonia e hiperreflexia, no momento da contração muscular, causado por uma condição neurológica anormal" (ESPASTICIDADE, 2011).

2.2 .NET MICRO FRAMEWORK 4.1

Macoratti (2010) define .NETMF como “[...] uma versão inicializável do .NET Framework que permite criar projetos para dispositivos embutidos usando a linguagem C#, [...]". Roda no Microsoft Visual Studio 2010 Express, o qual é grátis e possui um ambiente integrado de desenvolvimento de alto desempenho.

Segundo GHI Electronics (2011, p. 4), desenvolvedores já experientes em .NET e Visual Studio podem tirar proveito de suas habilidades, reduzindo a curva de aprendizagem para uma linguagem de programação de micro-controladores.

Aplicações desenvolvidas nesta plataforma requerem poucos recursos do hardware e um baixo consumo de energia, o que a torna ideal para o desenvolvimento de aplicações para pequenos dispositivos, como por exemplo, sensores, visores, controles remotos, periféricos, sistemas multimídias, *displays* eletrônicos e sistemas de posicionamento global.

Para facilitar os testes durante o desenvolvimento de uma aplicação, a biblioteca .NETMF 4.1 inclui um emulador que permite a execução da aplicação sem precisar do hardware. Este emulador possui alguns recursos, como por exemplo, botões, *display* de cristal líquido e simulação de entradas e saídas.

De acordo com Issa (2010, p. 8), o .NETMF já possui uma qualidade garantida no mercado, pois tem sido testado em diversos produtos comerciais.

O Quadro 1 apresenta um código exemplo utilizando a biblioteca .NETMF, para acender um LED.

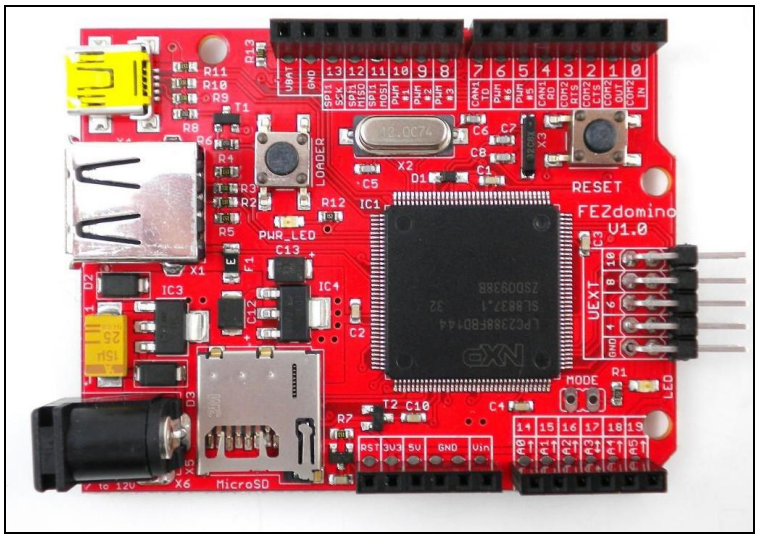
```
using System.Threading;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.FEZ;

namespace MFConsoleApplication1
{
   public class Program
   {
       public static void Main()
       {
              OutputPort LED;
              LED = new OutputPort((Cpu.Pin)FEZ_Pin.Digital.LED, true);
              Thread.Sleep(Timeout.Infinite);
       }
   }
}
```

Quadro 1 - Código exemplo utilizando .NETMF

2.3 FEZ DOMINO

Segundo GHI Eletronics (2011), FEZ Domino (Figura 1) é uma pequena placa *open source*[2] que executa o .NETMF e é baseada no *chipset* USBizi[3]. Oferece vários recursos a periféricos[4], como por exemplo, USB *host* e interface SD. Executa em um processador ARM da NXP de 72Mhz, suporta modo *debug* em tempo real e pode-se utilizar o Visual Studio 2010 Express Edition como ferramenta de desenvolvimento.



Fonte: Issa (2010, p. 13).

Figura 1 - Placa FEZ Domino

Esta placa, segundo Issa (2010, p. 13), é ideal para iniciantes, mas também serve como ponto de partida de baixo custo para profissionais que pretendem explorar os recursos do .NETMF. Sua utilização pode ser aplicada a diversas áreas, como por exemplo, robótica e periféricos.

---

[2] O termo *open source* é "um conceito de distribuição de software, que estabelece como fundamentais, os princípios de desenvolvimento compartilhado, distribuição na forma de código fonte e licenciamento gratuito" (LEITE, 2005).

[3] Chipset USBizi, pronunciado como USB *easy*, é um micro-controlador com um processador ARM7 da NXP, com um *firmware* desenvolvido especialmente para hospedar .NETMF com vários drivers Hardware Abstract Layer (HAL) (GHI ELECTRONICS, 2011, p. 5).

[4] Segundo Ferreira (2007, p. 3), considera-se periférico todo e qualquer componente que envie ou receba alguma informação de um computador.

2.4 XML

Segundo Furtado Júnior (2010), XML é uma linguagem de marcação de dados que provê um formato para descrever dados estruturados. Foi desenvolvida pela W3C em meados de 1990, com o princípio de desenvolver uma linguagem que pudesse ser lida por software e integrar-se com as demais linguagens (XML, 2011).

Heitlinger (2001, p. 3) descreve XML como sendo um formato universal para compartilhar dados entre aplicações. Documentos em formato XML podem conter diversos tipos de informações, entre os quais citam-se: bases de dados; transações comerciais; equações matemáticas; dados bibliográficos e configuração de sistemas.

No Quadro 2 é apresentado um código exemplo seguindo as definições para formatação de arquivos XML.

```
<?xml version="1.0"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Fonte: W3Schools (2011).

Quadro 2 - Código exemplo em formato XML

2.5 ACELERÔMETRO EZ430-CHRONOS

O acelerômetro eZ430-Chronos (Figura 2) é um sistema de desenvolvimento integrado e portátil baseado no CC430[5], desenvolvido pela empresa americana Texas Instruments. Este dispositivo em formato de relógio possui diversos recursos, dos quais se citam: acelerômetro de três eixos; sensor de pressão, sensor de temperatura; sensor de voltagem da bateria e *display* LCD (TEXAS..., 2010, p. 11);

[5] O CC430 é um chip único de alta performance e com baixo consumo de energia de RF para solução de microcontroladores baseados em aplicações. (TEXAS INSTRUMENTS, 2010)

Figura 2 - Acelerômetro eZ430-Chronos

Este dispositivo possui uma interface *wireless* que permite a comunicação via RF de 1GHz com outros dispositivos receptores de RF como, por exemplo, o CC430 ou a série CC11xx. É possível ainda customizar e reprogramar o eZ430-Chronos utilizando o eZ430-RF USB, o qual liga o dispositivo eZ430-Chronos a um computador para programação e depuração em tempo real (TEXAS..., 2010, p. 11).

Segundo Texas Instruments (2010, p. 16), a transmissão dos dados de aceleração é feita selecionando-se a opção `ACC` no dispositivo eZ430-Chronos e para a obtenção dos dados de aceleração é utilizado o dispositivo *access point* CC1111 USB RF (Figura 3).

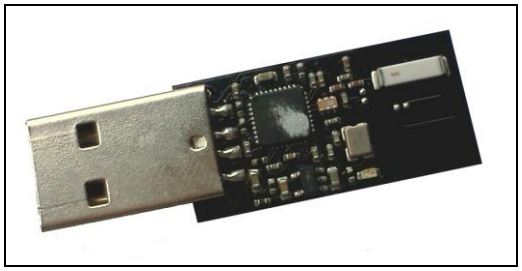


Figura 3 - Access point CC1111 USB RF

2.6 USB-HID

O HID é um tipo de dispositivo de computador que interage diretamente com o ser humano e que na maioria das vezes tem como entrada de dados, uma ação efetuada pelo próprio ser humano através de algum periférico, como por exemplo, teclado e *mouse* (HUMAN..., 2011).

A classe USB-HID possui diversas funções predefinidas que são utilizadas por

diversos fabricantes de hardware. Desta forma, os hardwares desenvolvidos podem comunicar-se com o computador sem a necessidade de instalação de um *driver* específico, desde que o hardware desenvolvido esteja dentro das especificações do HID. Somente há a necessidade de instalação de um *driver*, quando o periférico possui alguma função estendida e que foi personalizada pelo fabricante.

2.7 PROTÓTIPO DE *MOUSE* UTILIZANDO ACELERÔMETROS

Utilizando-se da linguagem de programação Java e do componente eletrônico SEN-00410, Jennrich (2010) desenvolveu um protótipo de *mouse* utilizando acelerômetro, voltado especialmente para pessoas portadoras de deficiência motora nos membros superiores. Segundo Jennrich (2010, p. 51), o protótipo obteve os resultados esperados e pode ser uma alternativa viável para o desenvolvimento de *mouses* adaptados.

Jennrich (2010, p. 51) comenta que encontrou algumas dificuldades no desenvolvimento do seu protótipo, como por exemplo, a necessidade de instalação de um software para o reconhecimento do *mouse* nos sistemas operacionais Windows e Linux.

Referente às extensões, foram dadas algumas sugestões como, por exemplo, a calibração de velocidades de deslocamento independentes para cada uma das inclinações do acelerômetro; a substituição do componente acelerômetro SEN-00410 por outro que seja capaz de medir a força de aceleração do movimento; a inclusão de outro componente eletrônico no protótipo, para que este fique responsável por identificar os cliques; a realização de testes do protótipo em portadores de outras deficiências motoras, identificando assim se ele atenderia ou não à sua necessidade; e a utilização de um hardware que dispense a instalação de qualquer software no sistema operacional e que se conecte via USB (JENNRICH, 2010, p. 52).

2.8 TRABALHOS CORRELATOS

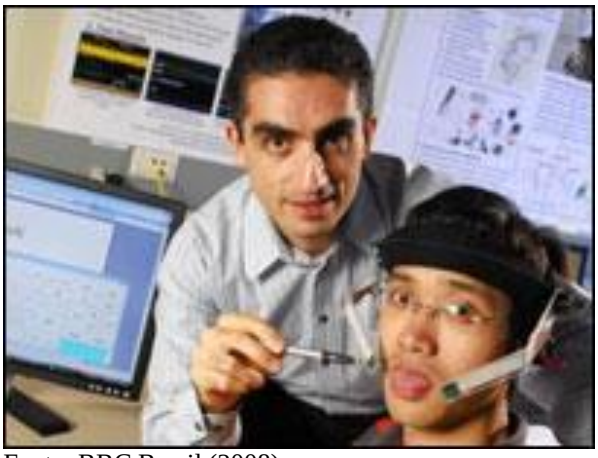
É possível encontrar diversos trabalhos relacionados ao proposto, alguns já comercializados e outros ainda em fase de testes, todos com o objetivo de auxiliar pessoas

com alguma deficiência motora. Dentre os quais, foram selecionados três trabalhos, sendo eles: Tongue Drive (BBC BRASIL, 2008), *Mouse* Visual (BANCO BRADESCO S.A., 2010) e o capacete EPOC (EMOTIV, 2011).

2.8.1 Tongue Drive

Desenvolvido por um grupo de cientistas do Georgia Institute of Technology nos Estados Unidos, Tongue Drive é um dispositivo que consiste em um ímã do tamanho de um grão de arroz, que é implantado na ponta da língua do paciente. Seu objetivo é substituir o *mouse* e até mesmo o *joystick* de uma cadeira de rodas (BBC BRASIL, 2008).

Ao movimentar a língua, sensores acoplados a um capacete ou a um aparelho ortodôntico bocal, detectam a movimentação do dispositivo e transmitem esta informação a um computador portátil. Na Figura 4 é apresentado o dispositivo Tongue Drive implantado na língua de um paciente.



Fonte: BBC Brasil (2008).

Figura 4 - Dispositivo Tongue Drive implantado na língua de um paciente

De acordo com BBC Brasil (2008), o aparelho foi testado em 18 pessoas saudáveis, as quais executaram alguns comandos, incluindo movimentos, clique único e clique duplo. O resultado apontou quase 100% de precisão nos movimentos.

2.8.2 *Mouse* Visual

Desenvolvido para seus clientes, o Banco Bradesco lançou um *mouse* virtual especialmente para pessoas portadoras de deficiência motora nos membros superiores. Este software consiste na captura de movimentos faciais efetuados pelo usuário através de uma câmera *webcam*, os quais são processados pelo software que por sua vez efetua o movimento do cursor. Para simular um clique de *mouse*, o usuário deve efetuar um leve movimento de abertura da boca. A Figura 5 apresenta o software *mouse* virtual Bradesco.

Fonte: Veloso (2010).

Figura 5 - Software *mouse* visual Bradesco

Segundo Banco Bradesco S.A. (2010), este software permite aos usuários utilizar o computador para efetuar transações bancárias, navegar na internet, efetuar a digitação de textos e enviar e receber *e-mails*.

2.8.3 Capacete EPOC

A idéia de controlar um computador apenas com o pensamento é algo que parecia impossível alguns anos atrás e somente era visto em filmes de ficção científica. Recentemente, diversas pesquisas nesta área estão sendo realizadas e muitas delas com excelentes resultados.

A exemplo disso, o capacete Emotiv EPOC (Figura 6), criado pela empresa australiana

Emotiv, permite controlar o computador apenas com o poder do pensamento. Este capacete consiste em 14 sensores neurais com a função de escanear regiões do cérebro responsáveis por sensações como alegria e tristeza, permitindo assim executar comandos com base nestas sensações. O dispositivo possui ainda um giroscópio, que permite identificar o posicionamento do mesmo (EMOTIV, 2011).

Fonte: Emotiv (2011).

Figura 6 - Capacete Emotiv EPOC

Sua utilização pode ser aplicada para auxiliar pessoas portadoras de alguma deficiência física, como por exemplo, controlar uma cadeira de rodas utilizando apenas o pensamento, ou até mesmo em jogos que são desenvolvidos especialmente para o EPOC.

3 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento do protótipo construído. São apresentados requisitos, especificação e implementação do protótipo. Para finalizar são apresentados os resultados obtidos e uma comparação com os trabalhos correlatos e com o trabalho desenvolvido por Jennrich (2010).

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Analisando a proposta para o desenvolvimento do protótipo utilizando acelerômetros, foram resgatados do trabalho de Jennrich (2010) os seguintes Requisitos Funcionais (RF):

a) interpretar a movimentação, aceleração e inclinação do acelerômetro em movimentos no cursor do *mouse*, através das coordenadas x, y e z do mesmo (RF);
b) permitir movimentos específicos que representem os atuais cliques de botão direito e esquerdo e comando de seleção do *mouse* tradicional (RF);
c) possuir uma interface para configuração do *mouse*, onde o usuário poderá efetuar a calibração da velocidade de deslocamento do ponteiro (RF);
d) ser implementado utilizando o ambiente de programação Microsoft Visual Studio 2010 Express (Requisito Não Funcional - RNF);
e) utilizar a biblioteca .NET Micro Framework 4.1 para implementação das rotinas do protótipo (RNF);
f) utilizar a placa micro-controladora FEZ Domino para controlar e manipular os movimentos do *mouse* (RNF);
g) permitir a comunicação do protótipo com o computador via conexão USB (RNF);
h) ser independente de sistema operacional ou software (RNF);
i) rodar em plataforma x86 (RNF).

3.2 ESPECIFICAÇÃO

A especificação foi realizada a partir da ferramenta Enterprise Architect, utilizando a linguagem de modelagem UML. Para a modelagem foram utilizados os diagramas de casos de uso, classes, atividades e estados.

3.2.1 Diagrama de casos de uso

Os diagramas de casos de uso estão divididos em módulo configuração e módulo *mouse*. O módulo configuração traz como ator um usuário, responsável pela configuração do *mouse*. O módulo *mouse* exibe a utilização do protótipo em um computador.

A Figura 7 ilustra o diagrama de casos de uso configuração.

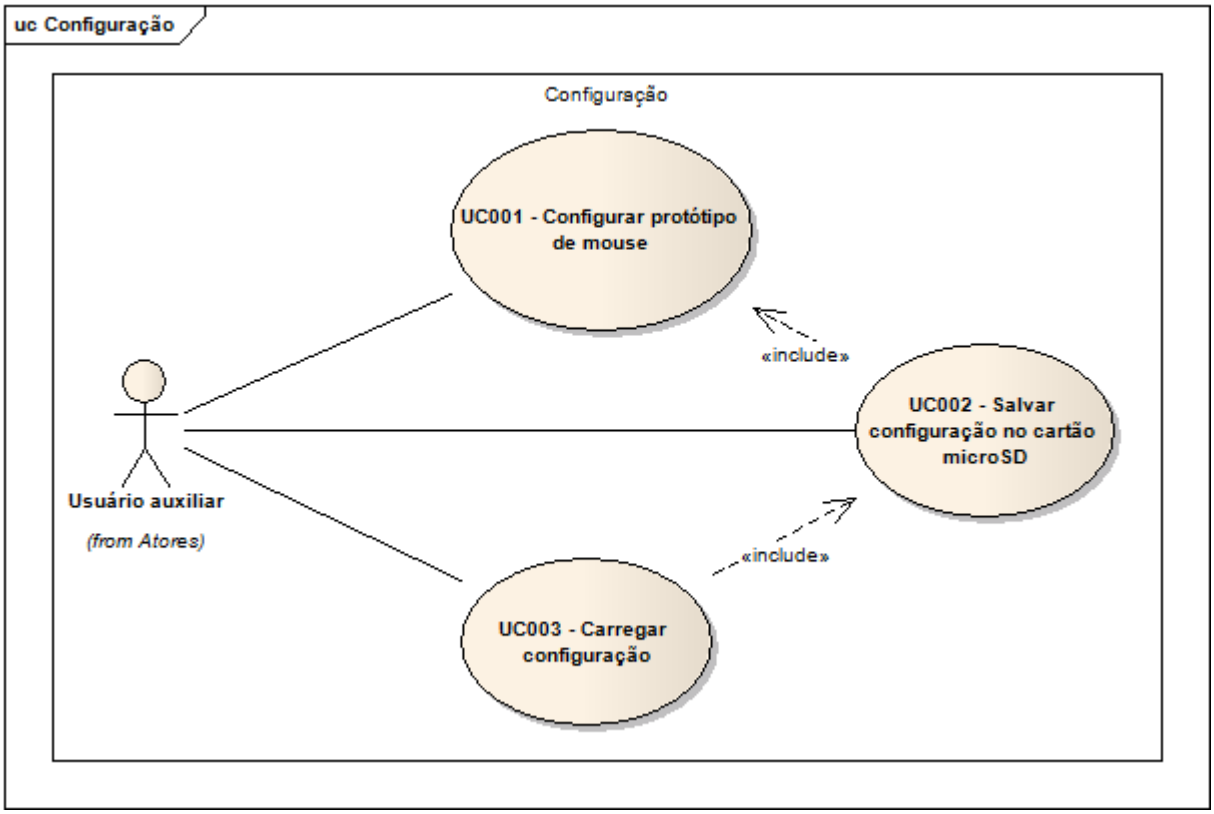


Figura 7 - Diagrama de casos de uso configuração

O detalhamento dos principais casos de uso apresentados na Figura 7 estão descritos no Apêndice A.

A Figura 8 exibe o diagrama de casos de uso protótipo de *mouse*.

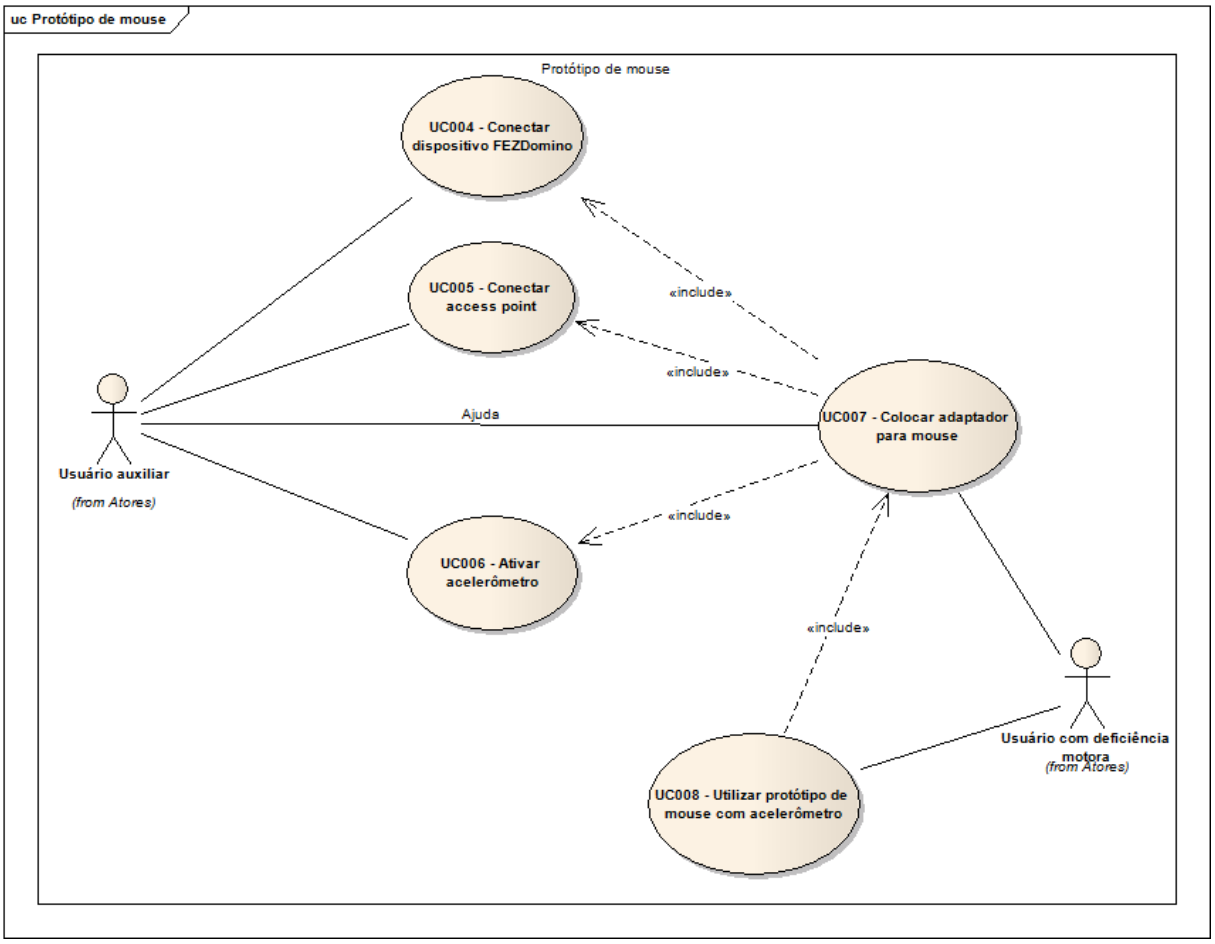


Figura 8 - Diagrama de casos de uso protótipo de *mouse*

O detalhamento dos principais casos de uso apresentados na Figura 8 estão descritos no Apêndice B.

3.2.2 Diagrama de classes

Os diagramas de classes estão divididos em modulo aplicativo de configuração e módulo protótipo de *mouse*. Na Figura 9 é exibido o diagrama de classes conceitual do aplicativo de configuração do protótipo, e o Quadro 3 apresenta a responsabilidade de cada classe.

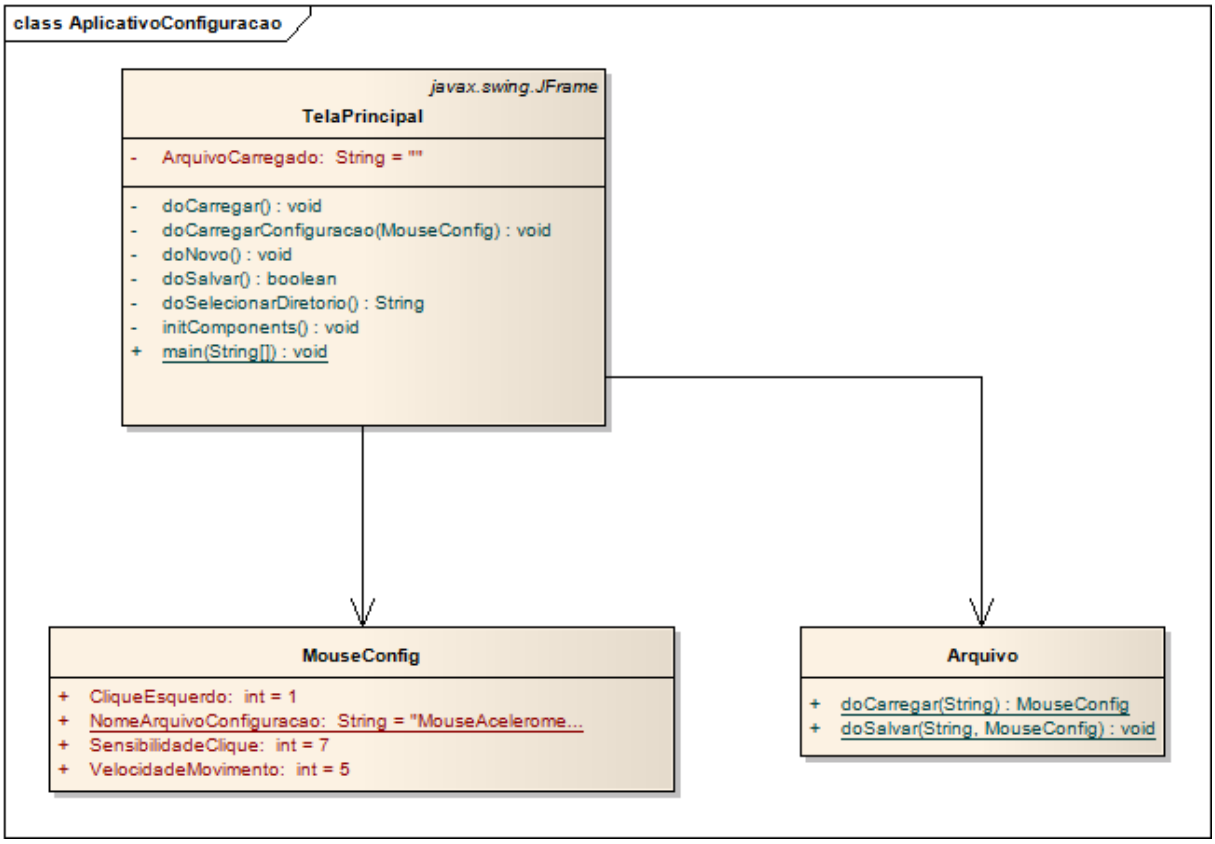


Figura 9 - Diagrama de classes do aplicativo de configuração do protótipo

Classe	Descrição
`TelaPrincipal`	Classe utilizada para integração entre a camada de controle e a camada de interação com o usuário.
`MouseConfig`	Classe utilizada para encapsular dados de uma configuração de mouse.
`Arquivo`	Classe utilizada para manipulação de arquivos físicos.

Quadro 3 - Descrição das classes do aplicativo de configuração do protótipo

Na Figura 10 são apresentadas as classes do protótipo de *mouse*, e o Quadro 4 exibe a responsabilidade de cada classe.

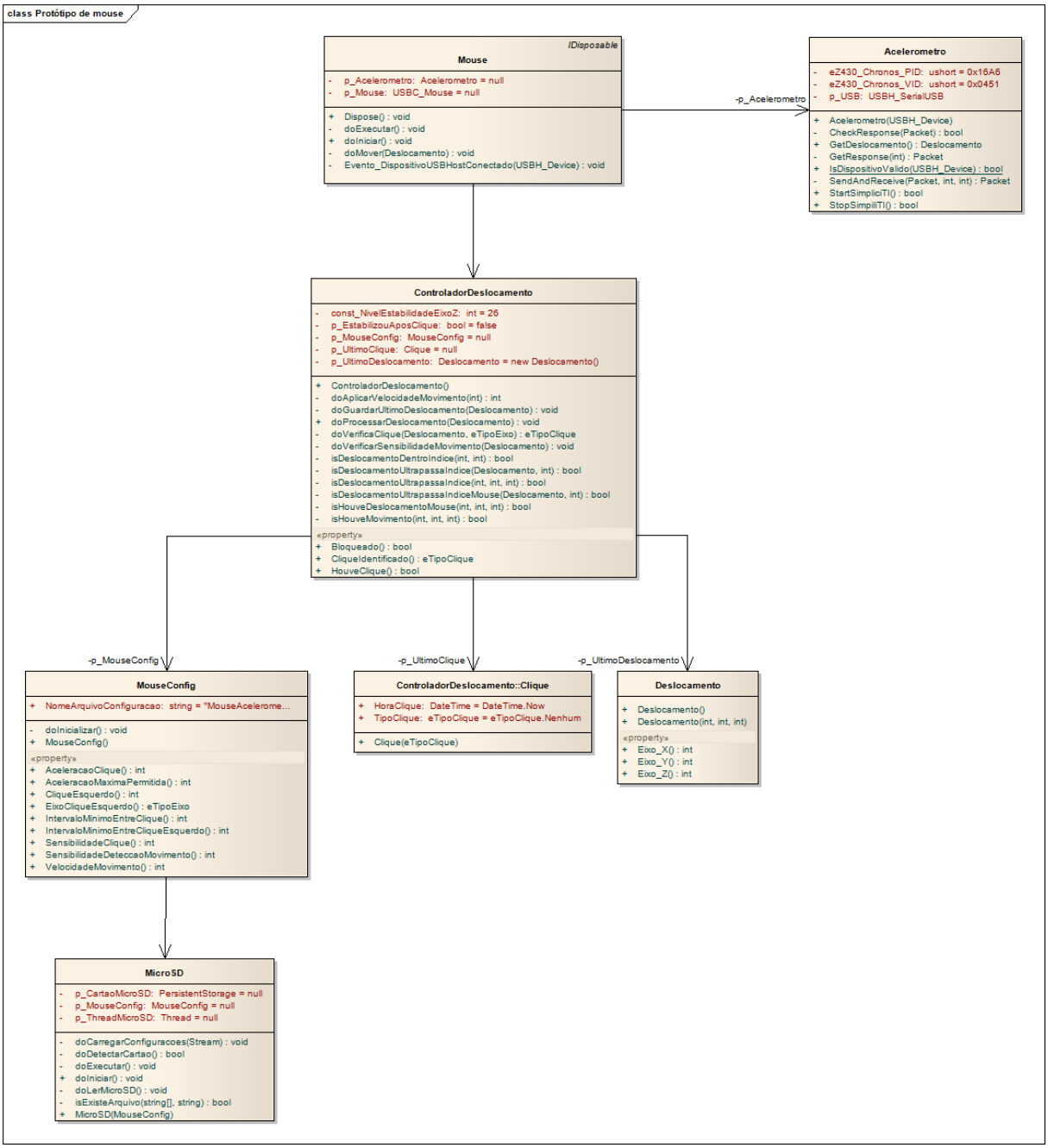


Figura 10 - Diagrama de classes do protótipo de *mouse*

Classe	Descrição
`Mouse`	Classe utilizada para realizar a comunicação entre o protótipo de mouse e o computador.
`Acelerometro`	Classe utilizada para realizar a integração entre o acelerômetro eZ430-Chronos e o protótipo de mouse.
`ControladorDeslocamento`	Classe utilizada para realizar a interpretação e manipulação dos dados recebidos pelo acelerômetro.
`Deslocamento`	Classe utilizada para encapsular os dados referentes ao deslocamento dos eixos X, Y e Z.
`Clique`	Classe utilizada para encapsular os dados de um clique em relação ao clique efetuado.
`MouseConfig`	Classe utilizada para encapsular os dados de uma configuração do protótipo de mouse em relação à configuração carregada.
`MicroSD`	Classe utilizada para realizar a integração entre o cartão micro SD e o protótipo de mouse.

Quadro 4 - Descrição das classes do protótipo de *mouse*

3.2.3 Diagrama de atividades

Na Figura 11 é apresentado o diagrama de atividades para indicar a sequência de configuração do protótipo de *mouse*. O usuário auxiliar começa iniciando a aplicação de configuração do protótipo, onde ele pode optar pelas ações:

a) nova configuração;

b) ou carregar configuração já existente.

No caso de criar uma nova configuração, o usuário auxiliar deverá preencher os dados, após isso, o sistema irá solicitar um diretório, o usuário irá informar um diretório, e após isso, o sistema irá criar um arquivo no formato XML contendo os dados da configuração. No caso de carregar uma configuração já existente, o sistema irá solicitar o arquivo que será alterado, o usuário auxiliar seleciona o arquivo de configuração e o sistema carrega os dados do arquivo de configuração, após isso, o usuário auxiliar altera os dados, e após isso, o sistema atualiza o arquivo de configuração com os novos dados informados.

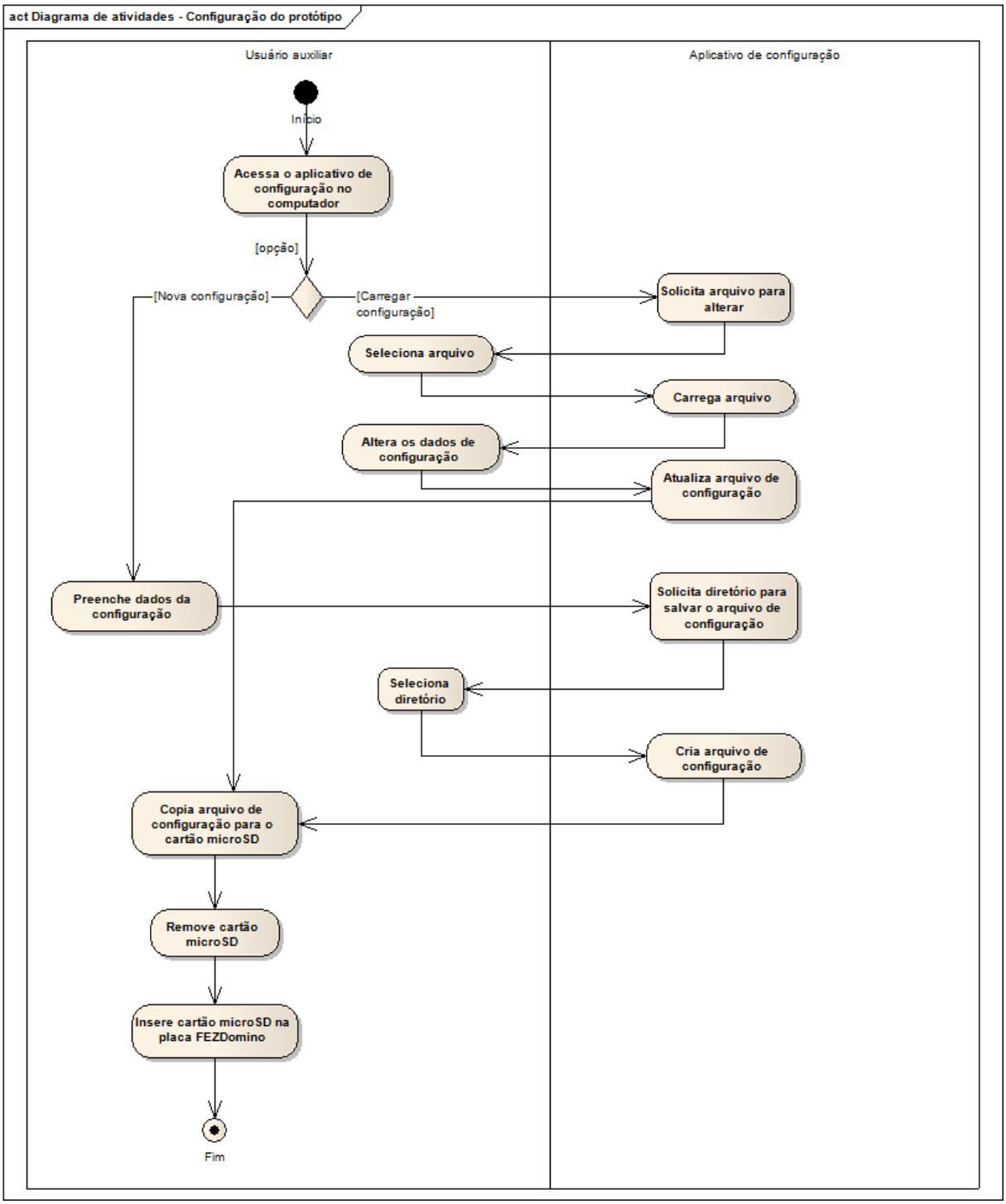


Figura 11 - Diagrama de atividades do aplicativo de configuração do protótipo de *mouse*

A Figura 12 apresenta o diagrama de atividades onde o usuário com deficiência física motora utiliza o protótipo de *mouse*. O usuário auxiliar inicia conectando a placa FEZ Domino ao computador utilizando-se de um cabo USB. Em seguida, conecta o dispositivo *access point* na placa FEZ Domino, e após isso, ativa o acelerômetro eZ430-Chronos.

O usuário com deficiência física, com ajuda do usuário auxiliar, coloca o acelerômetro na cabeça. O usuário auxiliar pode optar entre as ações:

a) utilizar configuração padrão;

b) utilizar configuração customizada.

No caso de utilizar a configuração padrão, o protótipo utilizará os valores pré-definidos para inicialização das variáveis de controle de movimento e cliques. No caso de utilizar uma configuração customizada, o usuário auxiliar insere na placa FEZ Domino o cartão micro SD que contém o arquivo de configuração do protótipo, após isso, o dispositivo FEZ Domino carrega as configurações do cartão.

O usuário com deficiência física motora efetua movimentos com a cabeça, o acelerômetro capta estes movimentos e envia os dados de movimento e aceleração ao dispositivo *access point*, após isso, o dispositivo FEZ Domino executa algoritmos para interpretar os movimentos recebidos, e após isso, o dispositivo identifica quais comandos serão executados. Estes comandos podem ser:

a) clique esquerdo;
b) clique direito;
c) deslocamento do cursor.

O dispositivo envia o comando identificado para o computador e este executa o comando. O usuário com deficiência física motora verifica se os movimentos e cliques estão bons ou se precisam de alguma alteração na configuração. No caso de precisar de alguma alteração, o usuário auxiliar remove o cartão micro SD, altera a configuração, e após isso, insere o cartão micro SD na placa FEZ Domino. O dispositivo carrega as configurações, e após isso, o usuário com deficiência física realiza novos movimentos com a cabeça. Caso os movimentos estejam bons, o usuário com deficiência física pode optar entre continuar utilizando o protótipo, ou, encerrar sua utilização.

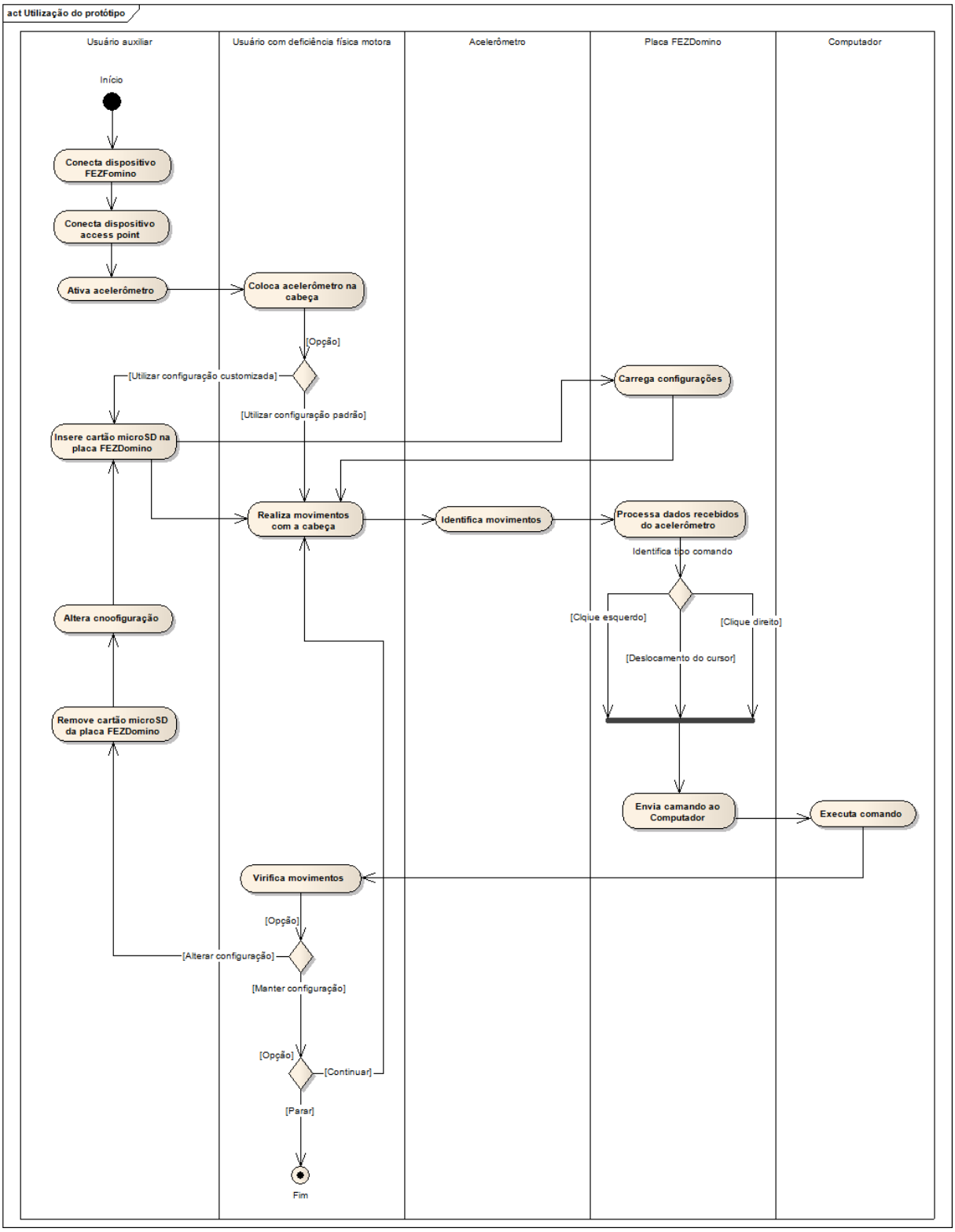


Figura 12 - Diagrama de atividades da utilização do protótipo de *mouse*

3.2.4 Diagrama de estados

Na Figura 13 é apresentado o diagrama de estados da utilização do protótipo.

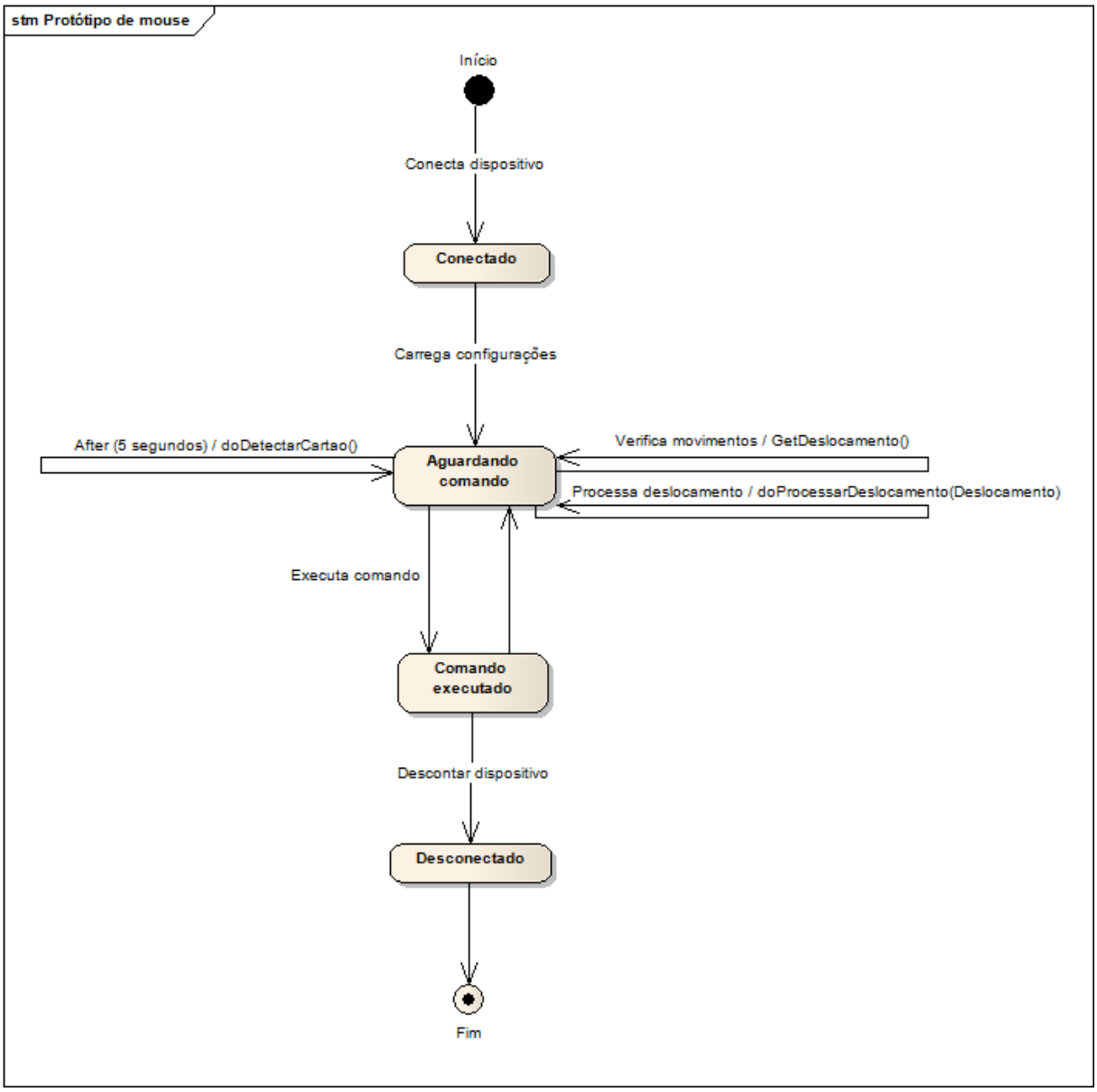


Figura 13 - Diagrama de estados do protótipo de *mouse*

3.3 IMPLEMENTAÇÃO

Esta seção irá apresentar inicialmente as técnicas e ferramentas utilizadas para implementação, e a seguir será apresentada a operacionalidade do protótipo.

3.3.1 Técnicas e ferramentas utilizadas

Esta seção está dividida em três seções. A primeira apresenta a preparação e montagem do hardware utilizado no protótipo de *mouse*. Na segunda é apresentada a implementação do aplicativo de configuração do protótipo. E por último é apresentado a implementação das rotinas do protótipo de *mouse*.

3.3.1.1 Preparação do hardware

Para preparação do hardware do protótipo foram utilizadas uma placa FEZ Domino, um acelerômetro modelo eZ430-Chronos, um *access point* modelo CC1111 USB RF, uma placa RS232 *Shield* para Arduino, um cabo USB/miniUSB, um cabo Serial RS232, um cabo RS232 com conversor para USB e um cartão micro SD. A Figura 14 exibe todos os itens utilizados para preparação do hardware.

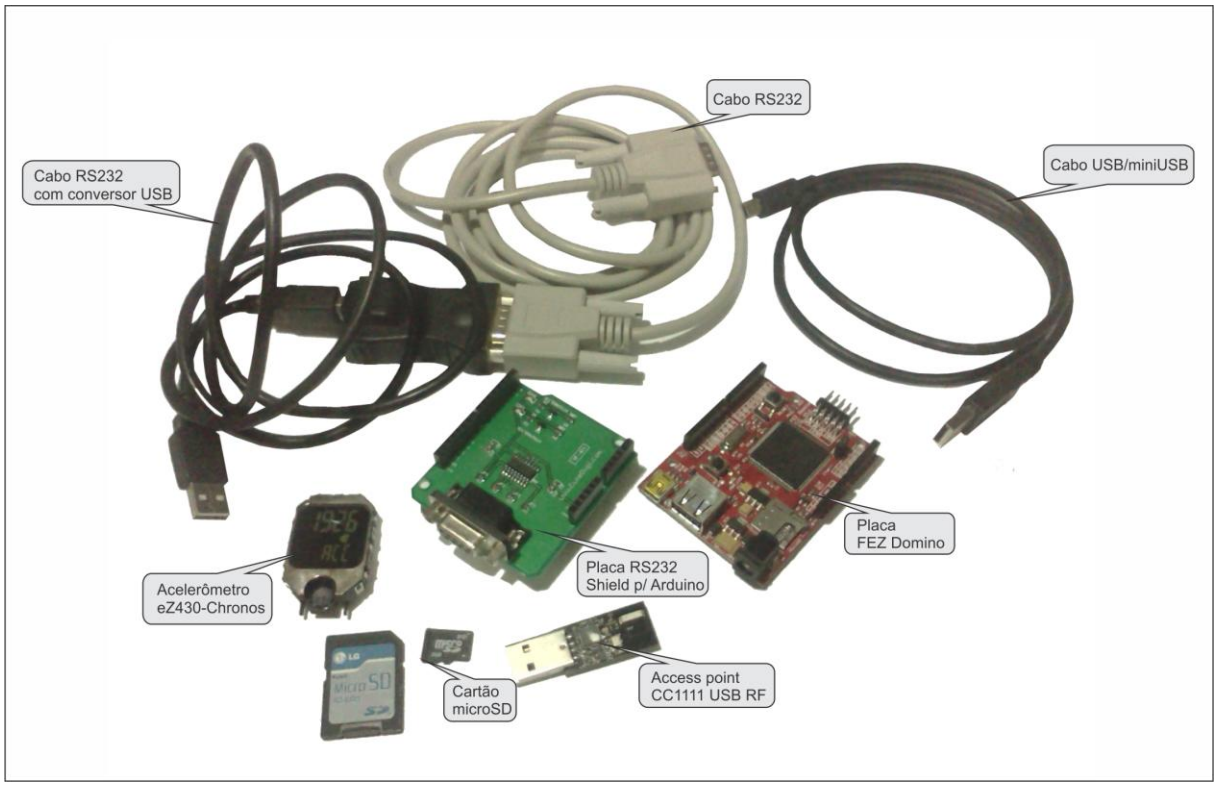


Figura 14 - Itens utilizados para preparação do hardware

Os cabos serial RS232 e RS232 com conversor para USB e a placa RS232 *Shield* para arduino foram utilizados somente durante a implementação das rotinas do protótipo. O código fonte foi compilado no ambiente de desenvolvimento Visual Studio 2010 e automaticamente

era convertido para código de máquina e inserido na placa FEZ Domino quando a aplicação era executada pelo Visual Studio. Para isto foi necessário conectar a placa RS232 Shield na placa FEZ Domino e ao computador através dos cabos RS232. A placa FEZ Domino foi conectada ao computador pelo cabo USB/miniUSB e o *access point* CC1111 USB RF foi conectado na placa FEZ Domino para receber os dados transmitidos pelo acelerômetro.

O ambiente de desenvolvimento Visual Studio 2010 foi configurado para compilar o código fonte na placa FEZ Domino através de uma porta serial, conforme mostra Figura 15.

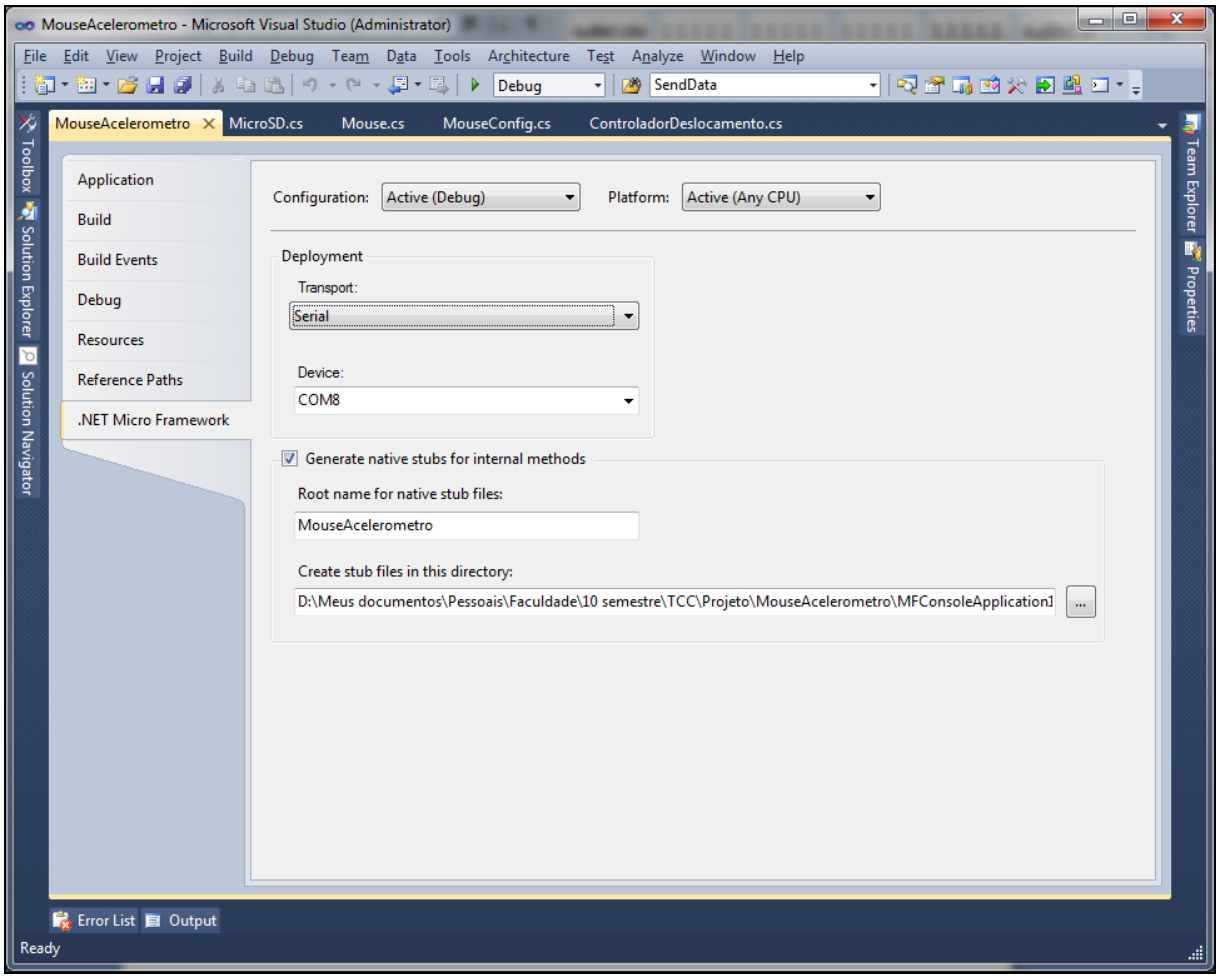


Figura 15 - Porta serial configurada no ambiente de desenvolvimento Visual Studio 2010

Para utilização e testes do protótipo, foi desenvolvido um adaptador (Figura 16) para ser fixado na cabeça do usuário com deficiência física motora. Este adaptador permite que o acelerômetro eZ430-Chronos fique preso à cabeça do usuário durante a utilização do protótipo, permitindo efetuar movimentos em todas as direções com o acelerômetro.

Figura 16 - Adaptador para acelerômetro

O adaptador desenvolvido consiste em um elástico circular preso a um pedaço de isopor, o qual possui um orifício para alojar o acelerômetro eZ430-Chronos. Nas duas extremidades do elástico foi costurada uma tira de velcro, permitindo ajustar o tamanho do elástico para cada pessoa.

3.3.1.2 Aplicativo de configuração

O aplicativo de configuração do protótipo foi desenvolvido utilizando-se da linguagem de programação Java através da IDE Netbeans 7.0.1. A Figura 17 apresenta a tela principal do aplicativo.

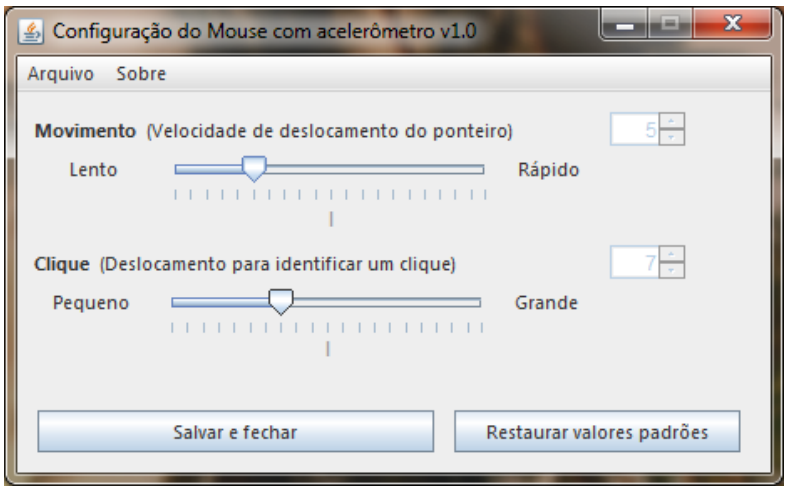


Figura 17 - Tela do aplicativo de configuração do protótipo de *mouse*

Essa aplicação fornece a possibilidade de criar um arquivo no formato XML, contendo parâmetros para serem utilizados pelo protótipo de *mouse*. São disponibilizados dois parâmetros para customização do usuário, dos quais se citam:

a) `Movimento`;

b) `Clique`.

O parâmetro `Movimento` permite informar um valor no formato inteiro, através de uma barra deslizante. Este parâmetro é utilizado para calcular a velocidade de deslocamento do ponteiro do *mouse* com relação à velocidade de deslocamento do acelerômetro. A barra deslizante permite informar um valor inicial de zero até vinte. Quanto maior o valor estabelecido neste campo, mais rápido será o movimento do cursor na tela do computador.

O parâmetro `Clique` é similar ao parâmetro `Movimento`, contendo inclusive as mesmas propriedades e valores da barra deslizante. Porém, o valor estabelecido neste parâmetro é utilizado para identificar o clique nos movimentos efetuados com o acelerômetro eZ430-Chronos. Quanto menor o valor estabelecido neste campo, mais sensível e fácil será a detecção dos cliques efetuados com o movimento do acelerômetro.

No Quadro 5 é apresentado o método `doSalvar` da classe `Arquivo`, responsável por criar o arquivo de configuração no formato XML. Este método tem como primeiro parâmetro uma string, que representa o caminho físico no disco onde o arquivo será salvo. O segundo parâmetro é uma referência para a classe `MouseConfig`, responsável por fazer o encapsulamento dos parâmetros de configuração.

```
public static void doSalvar(String caminhoArquivo, MouseConfig configMouse){
   try
   {
        StringBuilder conteudoArquivo = new StringBuilder();
        conteudoArquivo.append("<MouseConfig>\r\n");
        conteudoArquivo.append("  <Parametro ID=\"SensibilidadeClique\" Valor=\""+
String.valueOf(configMouse.SensibilidadeClique) +"\"/>\r\n");

        conteudoArquivo.append("  <Parametro ID=\"VelocidadeMovimento\" Valor=\""+
String.valueOf(configMouse.VelocidadeMovimento) +"\"/>\r\n");

        conteudoArquivo.append("</MouseConfig>");

        FileWriter arquivoWr = new FileWriter(caminhoArquivo, false);
        arquivoWr.write(conteudoArquivo.toString());
        arquivoWr.close();
   }
   catch (Exception err) {
         JOptionPane.showMessageDialog(null, err.getMessage(), "Erro ao salvar
arquivo!", JOptionPane.ERROR_MESSAGE);
   }
}
```

Quadro 5 - Método `doSalvar` da classe `Arquivo`

Para que o arquivo de configuração seja carregado e reconhecido pelo protótipo de *mouse*, este deve ter um nome e extensão pré-definidos e estar localizado no diretório raiz de

um cartão micro SD. Para garantir a autenticidade do nome e extensão do arquivo, estes valores são informados automaticamente pela rotina no momento em que o usuário seleciona o diretório para salvar o mesmo.

O Quadro 6 mostra o método `doSelecionarDiretorio` da classe `TelaPrincipal` e o Quadro 7 apresenta a classe `MouseConfig` onde está localizada a variável `NomeArquivoConfiguracao` com o nome e extensão pré-definidos para o arquivo de configuração.

```
private String doSelecionarDiretorio(){
        try
        {
            String caminhoArquivo = "";
            jFileChooser1.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

            // Mostra janela de selecção de arquivos ao usuário
            if (jFileChooser1.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
            {
                // Busca o diretório do arquivo selecionado
                caminhoArquivo = jFileChooser1.getSelectedFile().getAbsolutePath();
                caminhoArquivo += "\\"+ MouseConfig.NomeArquivoConfiguracao;
            }

            return caminhoArquivo;
        }
        catch (Exception err)
        {
            JOptionPane.showMessageDialog(null, err.getMessage(), "Erro ao salvar
arquivo!", JOptionPane.ERROR_MESSAGE);
            return "";
        }
}
```

Quadro 6 - Método `doSelecionarDiretorio` da classe `TelaPrincipal`

```
public class MouseConfig {

    public static String NomeArquivoConfiguracao = "MouseAcelerometro.config";
    public int SensibilidadeClique = 7;
    public int VelocidadeMovimento = 5;
}
```

Quadro 7 - Classe `MouseConfig`

O método `doCarregar` (Quadro 8) da classe `Arquivo`, é responsável por carregar uma configuração já salva para o aplicativo de configuração. O método busca o arquivo no diretório especificado pelo parâmetro `caminhoArquivo` e retorna uma classe `MouseConfig` contendo os parâmetros do respectivo arquivo carregado.

```
public static MouseConfig doCarregar(String caminhoArquivo) {
 try{
      MouseConfig configMouse = new MouseConfig();

      File arquivoXML = new File(caminhoArquivo);
      DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

      org.w3c.dom.Document doc = dBuilder.parse(arquivoXML);
      doc.getDocumentElement().normalize();

      NodeList nList = doc.getElementsByTagName("Parametro");
      int atributosEncontrados = 0;
      int atributosNecessarios = 2;

      for (int temp = 0; temp < nList.getLength(); temp++){
         Node nNode = nList.item(temp);
         if (nNode.getNodeType() == Node.ELEMENT_NODE){
             Element eElement = (Element) nNode;
             String name = eElement.getNodeName();

             if (name.toLowerCase().equals("parametro")){
                // Validação dos atributos 'ID' e 'Valor'
                if (eElement.getAttributeNode("Valor") == null)
                   throw new Exception("Tag 'Valor' não encontrada no arquivo.");

                if (eElement.getAttributeNode("ID") == null)
                   throw new Exception("Tag 'ID' não encontrada no arquivo.");

                String identificador = eElement.getAttributeNode("ID").getValue();
                int valor = 0;

                try{
                   valor =
                   Integer.parseInt(eElement.getAttributeNode("Valor").getValue());
                }
                catch (Exception err2 {
                   throw new Exception("Tag 'Valor' do identificador '"+
      identificador +"' não está em um formato numérico. \r\nValor encontrado: '"+
      eElement.getAttributeNode("Valor").getValue() +"'.");
                      }

                if (identificador.toLowerCase().equals("sensibilidadeclique")){
                  configMouse.SensibilidadeClique = valor;
                  atributosEncontrados++;
                }
                if (identificador.toLowerCase().equals("velocidademovimento")){
                   configMouse.VelocidadeMovimento = valor;
                   atributosEncontrados++;
                }
             }
         }
      }
      if (atributosEncontrados != atributosNecessarios)
          throw new Exception("Formato de arquivo inválido!");

      return configMouse;
   }
   catch (Exception err){
       JOptionPane.showMessageDialog(null, err.getMessage(), "Erro ao ler
       arquivo!", JOptionPane.ERROR_MESSAGE);
       return null;
   }
}
```

Quadro 8 - Método `doCarregar` da classe `Arquivo`

3.3.1.3 Protótipo de *mouse*

No desenvolvimento das rotinas do protótipo de *mouse* foi utilizada a linguagem de programação C# no ambiente de desenvolvimento Visual Studio 2010. Foi utilizada a versão 4.0 do .NET em conjunto com a biblioteca .NETMF 4.1.

Por meio da classe `Mouse`, o dispositivo FEZ Domino pode, através do método `doIniciar` (Quadro 9), emular um dispositivo de *mouse*. Esta emulação é iniciada a partir do momento em que a placa FEZ Domino é conectada ao computador, através do cabo USB/miniUSB. Como pode-se observar, a emulação do *mouse* é realizada de forma abstrata pelo método `USBClientController.StandardDevices.StartMouse`, disponibilizado pela biblioteca `GHIElectronics.NETMF.USBClient`.

```
public void doIniciar()
{
    try
    {
        Debug.Print("Iniciando mouse...");
        p_Mouse = USBClientController.StandardDevices.StartMouse();

        Debug.Print("Iniciando conexão USB...");
        USBHostController.DeviceConnectedEvent +=
      new USBH_DeviceConnectionEventHandler(Evento_DispositivoUSBHostConectado);

        Thread.Sleep(Timeout.Infinite);
    }
    catch (Exception err)
    {
        Debug.Print("Erro ao iniciar mouse! Mensagem erro: "
         + err.Message);
    }
}
```

Quadro 9 - Método `doIniciar` da classe `Mouse`

Para comunicação e obtenção dos dados de aceleração do acelerômetro eZ430-Chronos, foi desenvolvida a classe `Acelerometro`. Essa classe foi implementada com base no código fonte disponibilizado pela TINYCLR (2011). O método `GetDeslocamento` (Quadro 10) da classe `Acelerometro`, é responsável por buscar os dados dos deslocamentos efetuados com o acelerômetro eZ430-Chronos. O retorno desse método é uma classe `Deslocamento`, contendo os deslocamentos dos três eixos do acelerômetro, dos quais citam-se:

a) x: deslocamentos realizados para frente e para trás;
b) y: deslocamentos realizados para esquerda e direita;
c) z: deslocamentos realizados para cima e para baixo.

```
public Deslocamento GetDeslocamento()
{
   byte[] send = new byte[4] { 0x00, 0x00, 0x00, 0x00 };
   Packet response = SendAndReceive(Packet.Create(APCommand.BM_GET_SIMPLICITIDATA,
send), send.Length, 1);

   Deslocamento d = new Objetos.Deslocamento();
   d.Eixo_Z = (sbyte)response.Data[Constants.PACKET_DATA_START + 3];
   d.Eixo_X = (sbyte)response.Data[Constants.PACKET_DATA_START + 2];
   d.Eixo_Y = (sbyte)response.Data[Constants.PACKET_DATA_START + 1];

   CheckResponse(response);
   return d;
}

private bool CheckResponse(Packet response)
{
   APStatus BM_errorstate = (APStatus)response.Data[Constants.PACKET_BYTE_CMD];
   if (BM_errorstate != APStatus.HW_NO_ERROR)
      return false;
   return true;
}


private Packet SendAndReceive(Packet packet, int receive, int delay)
{
   p_USB.Write(packet.Data, 0, packet.Data.Length);
   Thread.Sleep(delay);

   Packet result = GetResponse(receive + Constants.PACKET_OVERHEAD_BYTES);
   return result;
}

private Packet GetResponse(int count)
{
   byte[] b = new byte[count];
   p_USB.Read(b, 0, count);
   Packet packet = Packet.CreateResponse(b);
   return packet;
}
```

Quadro 10 - Método `GetDeslocamento` da classe `Acelerometro`

Os valores obtidos pelo método `GetDeslocamento` são processados pela classe `ControladorDeslocamento`, através do método `doProcessarDeslocamento` (Quadro 11). Esse método é responsável por verificar os tipos de movimentos que foram realizados com o acelerômetro, identificando os comandos de *mouse* que serão enviados ao computador. Entre os possíveis comandos, citam-se:

a) movimentos verticais e horizontais com o cursor;

b) simulação de clique com o botão esquerdo;

c) simulação de clique com o botão direito.

```
Public void doProcessarDeslocamento(Deslocamento d){
       // ======== Verifica se houve clique =========
       CliqueIdentificado = doVerificaClique(d, eTipoEixo.X);

       if (CliqueIdentificado == eTipoClique.Nenhum)
             CliqueIdentificado = doVerificaClique(d, eTipoEixo.Y);

      // ======================================================
      HouveClique = CliqueIdentificado != eTipoClique.Nenhum;

      // = Se houve clique, ou uma aceleração > q a
permitida, trava o deslocamento do mouse
      Bloqueado = HouveClique ||
       isDeslocamentoUltrapassaIndice(d, p_MouseConfig.AceleracaoMaximaPermitida)
      || !isDeslocamentoUltrapassaIndiceMouse(d,
         p_MouseConfig.SensibilidadeDeteccaoMovimento)
      || (p_UltimoClique != null && p_UltimoClique.HoraClique.AddMilliseconds(
         p_MouseConfig.IntervaloMinimoEntreClique/2) > DateTime.Now);

      doGuardarUltimoDeslocamento(d);
      doVerificarSensibilidadeMovimento(d);
}

private bool isHouveMovimento(int deslocamentoAtual, int deslocamentoAnterior,
int indiceDeslocamento){
    return (System.Math.Abs(deslocamentoAtual -
  deslocamentoAnterior) > indiceDeslocamento);
}

private bool isDeslocamentoUltrapassaIndice(Deslocamento d, int indiceDeslocamento){
   bool retorno = false;
   retorno |= isHouveMovimento(d.Eixo_X, p_UltimoDeslocamento.Eixo_X,
              indiceDeslocamento);
   retorno |= isHouveMovimento(d.Eixo_Y, p_UltimoDeslocamento.Eixo_Y,
              indiceDeslocamento);
   return retorno;
}

private bool isDeslocamentoUltrapassaIndice(int deslocamentoAtual,
int deslocamentoAnterior, int indiceDeslocamento){
  return isHouveMovimento(deslocamentoAtual,deslocamentoAnterior,indiceDeslocamento);
}

private bool isDeslocamentoUltrapassaIndiceMouse(Deslocamento d,
int indiceDeslocamento){
   bool retorno = false;
   retorno |= isHouveDeslocamentoMouse(d.Eixo_X, p_UltimoDeslocamento.Eixo_X,
              indiceDeslocamento);
   retorno |= isHouveDeslocamentoMouse(d.Eixo_Y, p_UltimoDeslocamento.Eixo_X,
              indiceDeslocamento);
   return retorno;
}

private bool isHouveDeslocamentoMouse(int deslocamentoAtual, int ultimoDeslocamento,
int precisao){
   int posMim = ultimoDeslocamento - precisao;
   int posMax = ultimoDeslocamento + precisao;
   return (posMim > deslocamentoAtual) || (posMax < deslocamentoAtual);
}

private void doGuardarUltimoDeslocamento(Deslocamento d){
   p_UltimoDeslocamento.Eixo_X = d.Eixo_X;
   p_UltimoDeslocamento.Eixo_Y = d.Eixo_Y;
   p_UltimoDeslocamento.Eixo_Z = d.Eixo_Z;
}
```

Quadro 11 - Método `doProcessarDeslocamento` da classe `ControladorDeslocamento`

A implementação dos cliques no protótipo de *mouse* foram desenvolvidos com base em uma análise realizada através do aplicativo Texas Instruments eZ430-Chronos Control Center 1.1[6] (Figura 18). Esse aplicativo permite visualizar três gráficos de movimento, um para cada eixo de deslocamento do acelerômetro. A taxa de atualização dos gráficos é feita com base no tempo de resposta do acelerômetro, que corresponde a aproximadamente 36,4 milissegundos.

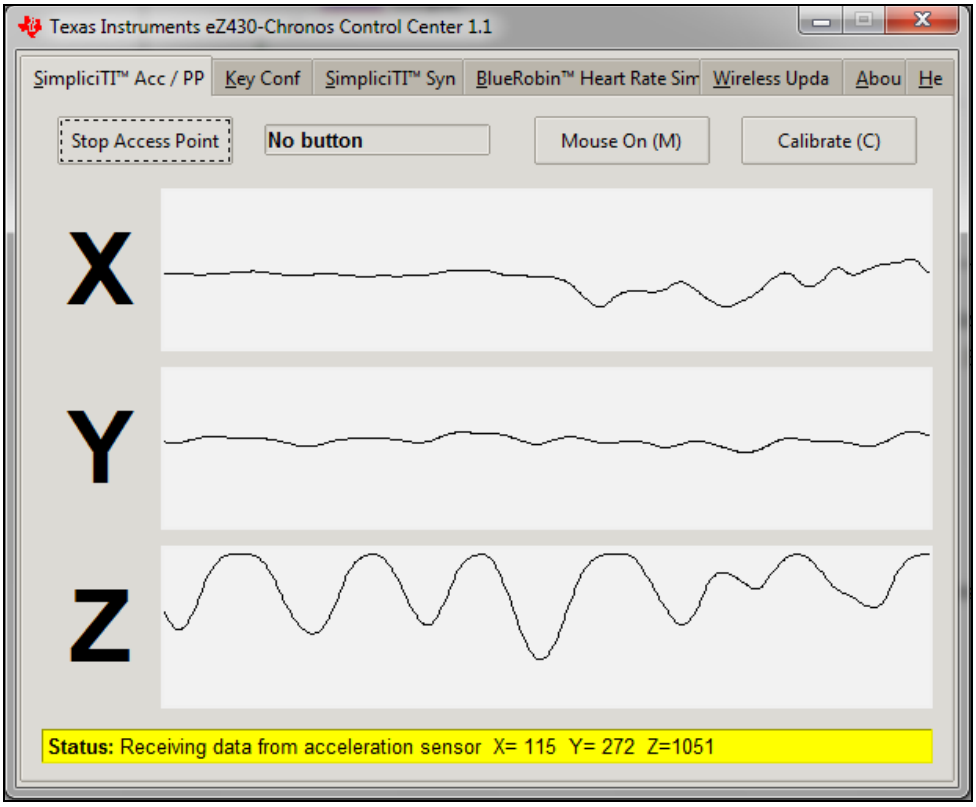


Figura 18 - Tela do aplicativo Texas Instruments eZ430-Chronos Control Center 1.1

Os gráficos dos eixos x, y e z apresentados na Figura 18, mostram várias oscilações para cada eixo. Quanto maior for o deslocamento realizado com o acelerômetro em determinada direção, maior será a curva de deslocamento no gráfico do respectivo eixo. Por exemplo, se for realizado um pequeno movimento girando lentamente o acelerômetro para frente e para trás, o gráfico do eixo x irá oscilar para cima e para baixo, conforme mostra Figura 19.

Figura 19 - Gráfico de oscilação do eixo x

[6] Aplicativo disponibilizado pela empresa Texas Instruments juntamente com o dispositivo eZ430-Chronos. Este aplicativo fornece vários recursos que demonstram a capacidade wireless do dispositivo eZ430-Chronos em PCs com Windows e Linux (TEXAS INSTRUMENTS, 2010, p. 26)

Já o gráfico da Figura 20 mostra um movimento realizado com o acelerômetro, girando-o rapidamente para frente e retornando à sua posição original. Pode-se observar nesta figura, que há uma mudança brusca na curva do gráfico. Esta mudança indica uma aceleração repentina.

Figura 20 - Gráfico de oscilação do eixo x com maior aceleração

É através das acelerações repentinas dos eixos x e y que os cliques são identificados. Cada eixo representa um tipo de clique, dos quais citam-se:

a) eixo x: representa um clique com o botão esquerdo do *mouse*;
b) eixo y: representa um clique com o botão direito do *mouse*.

O nível de aceleração dos eixos é medido pela diferença entre o deslocamento atual e o deslocamento anterior do acelerômetro. Este nível de aceleração é comparado ao parâmetro de configuração `SensibilidadeClique` da classe `MouseConfig`, que determina o nível de aceleração mínimo para a detecção de cada clique. A aceleração somente será considerada um clique, se o nível de aceleração obtido for maior ou igual ao valor definido para o parâmetro `SensibilidadeClique`.

O valor do parâmetro `SensibilidadeClique` pode ser alterado pelo usuário através do aplicativo de configuração do protótipo, conforme mencionado anteriormente no tópico 3.3.1.2 Aplicativo de configuração.

Para identificar os cliques nos movimentos do acelerômetro, foi criado o método `doVerificaClique` (Quadro 12) na classe `ControladorDeslocamento`. Esse método recebe dois parâmetros. O primeiro é uma classe `Deslocamento`, contendo o deslocamento atual dos três eixos do acelerômetro, o segundo é um parâmetro do tipo `enum eTipoEixo`, que determina em qual eixo o clique está sendo verificado. O retorno do método é o tipo de clique identificado com as acelerações dos eixos. Sendo que os possíveis valores de retorno são:

a) nenhum;
b) esquerdo;
c) direito.

```
private eTipoClique doVerificaClique(Deslocamento d, eTipoEixo eixo){
   int valorEixoAtual = 0;
   int valorEixoAnterior = 0;

   // Somente verifica o clique se já houve algum deslocamento
   if (p_UltimoDeslocamento != null){
       #region Busca valor do eixo
       switch (eixo){
         case eTipoEixo.X:
              valorEixoAtual = d.Eixo_X;
                    valorEixoAnterior = p_UltimoDeslocamento.Eixo_X;
              break;
         case eTipoEixo.Y:
              valorEixoAtual = d.Eixo_Y;
              valorEixoAnterior = p_UltimoDeslocamento.Eixo_Y;
              break;
         case eTipoEixo.Z:
              valorEixoAtual = d.Eixo_Z;
              valorEixoAnterior = p_UltimoDeslocamento.Eixo_Z;
              break;
       }
       #endregion

        // Verifica se o mouse foi estabilisado depois do último clique efetuado
        if (p_EstabilizouAposClique || (p_UltimoClique != null &&
            p_UltimoClique.TipoClique == eTipoClique.Esquerdo)) {
           // Verifica se houve aceleração no eixo
          if (isDeslocamentoUltrapassaIndice(valorEixoAtual, valorEixoAnterior,
               p_MouseConfig. SensibilidadeClique)){
              eTipoClique clique = eTipoClique.Nenhum;
              if ((valorEixoAtual > const_NivelEstabilidadeEixoZ) ||
                    (valorEixoAtual < 0))
                 clique = eixo == p_MouseConfig.EixoCliqueEsquerdo ?
                     eTipoClique.Esquerdo : eTipoClique.Direito;

              if ((p_UltimoClique) != null){
                  // Verifica se houve um intervalo entre o clique atual
                  // e o último clique
                   int intervalo =
                         p_UltimoClique.TipoClique == eTipoClique.Esquerdo &&
                         clique == eTipoClique.Esquerdo ?
                         p_MouseConfig.IntervaloMinimoEntreCliqueEsquerdo :
                         p_MouseConfig.IntervaloMinimoEntreClique;

                   if (p_UltimoClique.HoraClique.AddMilliseconds(intervalo)
                         > DateTime.Now){
                     clique = eTipoClique.Nenhum;
                     Debug.Print("Não atingiu o intervalo de clique! ");
                  }
              }

              // Se houve clique, seta as variáveis de controle
              if (clique != eTipoClique.Nenhum){
                p_EstabilizouAposClique = false;
                p_UltimoClique = new Clique(clique);
              }
              return clique;
           }
        }
      }
      if (!p_EstabilizouAposClique && isDeslocamentoDentroIndice(valorEixoAtual,
          p_MouseConfig.SensibilidadeDeteccaoMovimento + 1)){
       p_EstabilizouAposClique = true;
        Debug.Print("Estabilizou!!");
      }
      return eTipoClique.Nenhum;
}
```

Quadro 12 - Método `doVerificaClique` da classe `ControladorDeslocamento`

A velocidade de deslocamento do ponteiro do *mouse* foi implementada com base no valor de inclinação dos eixos x e y do acelerômetro eZ430-Chronos. O valor de inclinação de cada eixo foi obtido através de uma amostra (Tabela 1) criada durante os testes do protótipo. Esta amostra exibe a inclinação do eixo x partindo de 0° e atingindo 90° ao final.

Tabela 1 - Amostra de inclinação do eixo x

Inclinação eixo X (Graus °)	Valor eixo X (eZ430-Chronos)	Hora resposta (hh:mm:ss:ms)
0,0	0	18:36:45:688
1,8	1	18:36:45:724
1,8	1	18:36:45:761
1,8	1	18:36:45:796
1,8	1	18:36:45:832
3,6	2	18:36:45:871
3,6	2	18:36:45:908
9,0	5	18:36:45:944
12,6	7	18:36:45:979
12,6	7	18:36:46:015
16,2	9	18:36:46:051
19,8	11	18:36:46:088
23,4	13	18:36:46:124
27,0	15	18:36:46:161
30,6	17	18:36:46:198
36,0	20	18:36:46:236
37,8	21	18:36:46:273
48,6	27	18:36:46:308
50,4	28	18:36:46:345
57,6	32	18:36:46:381
63,0	35	18:36:46:417
70,2	39	18:36:46:453
75,6	42	18:36:46:489
77,4	43	18:36:46:525
77,4	43	18:36:46:560
77,4	43	18:36:46:596
84,6	47	18:36:46:635
90,0	50	18:36:46:670

Observando a amostra apresentada na Tabela 1, foi possível calcular o intervalo de tempo entre cada resposta do acelerômetro, que foi de aproximadamente 36,4 milissegundos, o que equivale à aproximadamente 27,5 respostas/segundo. Constatou-se, através de testes, que cada unidade de valor dos eixos x e y equivale ao deslocamento de um *pixel* do cursor de *mouse* na tela do computador.

Considerando o deslocamento de um *pixel* para cada unidade de valor do eixo x e y, foi criada uma fórmula que permitiu aumentar e reduzir a velocidade de deslocamento do ponteiro do *mouse* de acordo com uma velocidade pré-definida. Esta fórmula é calculada

considerando os parâmetros Deslocamento do eixo (D), Velocidade pré-definida (V) e um valor Heurístico (H). Este cálculo é representado pela fórmula D*(V/H). O valor heurístico utilizado foi o número cinco, cujo valor foi obtido através de testes realizados durante a implementação do protótipo.

O Quadro 13 exibe o método `doAplicarVelocidadeMovimento` da classe `ControladorDeslocamento`, o qual aplica a fórmula D*(V/H) considerando a velocidade definida no parâmetro `VelocidadeMovimento` da classe `MouseConfig`. O método recebe como parâmetro o valor original do deslocamento e retorna o novo valor calculado pela fórmula.

```
private int doAplicarVelocidadeMovimento(int deslocamentoEixo)
{
    double velocidade = double.Parse(p_MouseConfig.VelocidadeMovimento.ToString());
    double deslocamento = double.Parse(deslocamentoEixo.ToString());
    double novaVelocidade = deslocamento * (velocidade / 5);

    return int.Parse(System.Math.Round(novaVelocidade).ToString()); ;
}
```

Quadro 13 - Método `doAplicarVelocidadeMovimento` da classe `ControladorDeslocamento`

O valor do parâmetro `VelocidadeMovimento` da classe `MouseConfig` pode ser alterado pelo usuário através do aplicativo de configuração do protótipo, conforme mencionado anteriormente no seção 3.3.1.2.

Na classe `MouseConfig` foram criados atributos que são utilizados como parâmetros para as rotinas do protótipo de *mouse*. O objetivo dessa classe é centralizar todos os atributos que possuam valores constantes. O método `doInicializar` (Quadro 14) exibe a inicialização dos atributos da classe `MouseConfig`.

```
private void doInicializar()
{
   // Seta os valores padrões
   SensibilidadeDeteccaoMovimento = 2;
   IntervaloMinimoEntreClique = 1000;
   IntervaloMinimoEntreCliqueEsquerdo = 150;
   AceleracaoMaximaPermitida = 4;
   VelocidadeMovimento = 5;
   SensibilidadeClique = 7;
   AceleracaoClique = 10;
   EixoCliqueEsquerdo = ControladorDeslocamento.eTipoEixo.X;

   // Cria uma instância de MicroSD para buscar as configurações do cartão microSD
   MicroSD p_MicroSD = new MicroSD(this);
   p_MicroSD.doIniciar();
}
```

Quadro 14 - Método `doInicializar` da classe `MouseConfig`

Para permitir a alteração dos valores de alguns parâmetros da classe `MouseConfig`, foi criada a classe `MicroSD`, cuja responsabilidade é buscar um arquivo de configuração em um cartão micro SD inserido na placa Fez Domino. Os parâmetros cujos valores podem ser alterados são:

a) `SensibilidadeClique`;

b) `VelocidadeMovimento`.

A detecção do cartão micro SD na placa FEZ Domino é feita pelo método `doDetectarCartao` da classe `MicroSD`, exibido no Quadro 15, o qual retorna `True` quando detectar um cartão micro SD inserido e `False` quando não detectar.

```
private bool doDetectarCartao()
{
   try
   {
       if (PersistentStorage.DetectSDCard() && p_CartaoMicroSD == null)
       {
           p_CartaoMicroSD = new PersistentStorage("SD");
           Debug.Print("MicroSD inserido!!");
           return true;
       }
       else if (!PersistentStorage.DetectSDCard() && p_CartaoMicroSD != null)
       {
           Debug.Print("MicroSD removido!!");
           p_CartaoMicroSD.Dispose();
           p_CartaoMicroSD = null;
       }
       return false;
   }
   catch (Exception err)
   {
      Debug.Print("Erro ao detectar MicroSD! Mensagem erro: " + err.Message);
      return false;
   }
}
```

Quadro 15 - Método `doDetectarCartao` da classe `MicroSD`

No Quadro 16 é apresentado o método `doLerMicroSD` da classe `MicroSD`, responsável por efetuar a leitura do arquivo de configuração no cartão micro SD, cujo arquivo pode ser gerado pelo aplicativo de configuração do protótipo de *mouse*, conforme detalhado na seção 3.3.1.2. A busca pelo arquivo de configuração é feita somente no diretório raiz do cartão e somente pelos arquivos que tenham o nome igual à `MouseAcelerometro.config`.

```
private void doLerMicroSD(){
   try{
        // Monta o sistema de arquivos para o dispositivo de armazenamento
        p_CartaoMicroSD.MountFileSystem();

        // Aguarda algum tempo para finalizar a montagem de arquivos
        Thread.Sleep(500);

        // Verifica se há valumes no cartão micro SD
        if (VolumeInfo.GetVolumes().Length > 0){
            // Verifica se o cartão microSD está formatado com FAT32/FAT16
            if (VolumeInfo.GetVolumes()[0].IsFormatted){
                string rootDirectory = VolumeInfo.GetVolumes()[0].RootDirectory;
                string arquivoConfiguracao = rootDirectory + @"\" +
                            this.p_MouseConfig.NomeArquivoConfiguracao;

                // Verifica se o arquivo de configuração existe dentro
                //do cartão microSD
                if (isExisteArquivo(Directory.GetFiles(rootDirectory),
                arquivoConfiguracao)){
                    FileStream fileStream = new FileStream(arquivoConfiguracao,
                                FileMode.Open, FileAccess.Read);
                    doCarregarConfiguracoes(fileStream);
                    fileStream.Close();
                }
                else
                    Debug.Print("Arquivo não encontrado!");
            }
            else
                Debug.Print("O cartão MicroSD não está formatado!");
        }

        p_CartaoMicroSD.UnmountFileSystem();
    }
    catch (Exception err) {
        Debug.Print("Erro ao ler cartão MicroSD! Mensagem erro: " + err.Message);
    }
}

private bool isExisteArquivo(string[] listaArquivos, string arquivoProcurado) {
   try{
        foreach (string arquivo in listaArquivos) {
            if (arquivo == arquivoProcurado)
                return true;
        }
        return false;
   }
   catch (Exception err) {
       Debug.Print("Erro ao verificar existência de arquivo! Mensagem erro: " +
       err.Message);
       return false;
   }
}
```

Quadro 16 - Método `doLerMicroSD` da classe `MicroSD`

No Quadro 17 é exibido o método `doCarregarConfiguracoes` da classe `MicroSD`, responsável por carregar as configurações do arquivo de configuração criado pelo usuário.

```
private void doCarregarConfiguracoes(Stream streamFile){
   try{
        XmlReader xmlConfig = XmlReader.Create(streamFile);

        while (xmlConfig.Read()){
            if (xmlConfig.NodeType == XmlNodeType.Element
                &&  xmlConfig.Name.ToUpper() == "PARAMETRO"){

                string identificador = "";
                if (xmlConfig.MoveToAttribute("ID", ""))
                    identificador = xmlConfig.Value;

                int valor = int.MinValue;
                if (xmlConfig.MoveToAttribute("Valor", ""))
                    valor = int.Parse(xmlConfig.Value);

                if ((identificador != "") && (valor != int.MinValue)) {
                    switch (identificador.ToUpper()){
                           case "SENSIBILIDADECLIQUE":
                                  this.p_MouseConfig.SensibilidadeClique = valor;
                                  break;
                           case "VELOCIDADEMOVIMENTO":
                                  this.p_MouseConfig.VelocidadeMovimento = valor;
                                  break;
                    }
                }
            }
        }
    }
    catch (Exception err){
          Debug.Print("Erro ao carregar configurações! Mensagem erro: " +
           err.Message);
    }
}
```

Quadro 17 - Método `doCarregarConfiguracoes` da classe `MicroSD`

3.3.2 Operacionalidade do protótipo

Nesta seção é apresentada a operacionalidade do protótipo. Na seção 3.3.2.1 será apresentado a criação e alteração de um arquivo de configuração utilizando o aplicativo de configuração. Na seção 3.3.2.2 será apresentada a utilização do protótipo de *mouse* por uma pessoa que possui deficiência motora nos membros superiores.

3.3.2.1 Criação do arquivo de configuração

Nesta seção é apresentada a criação de um arquivo de configuração utilizando-se do aplicativo de configuração. Na Figura 21 é apresentado o aplicativo de configuração.

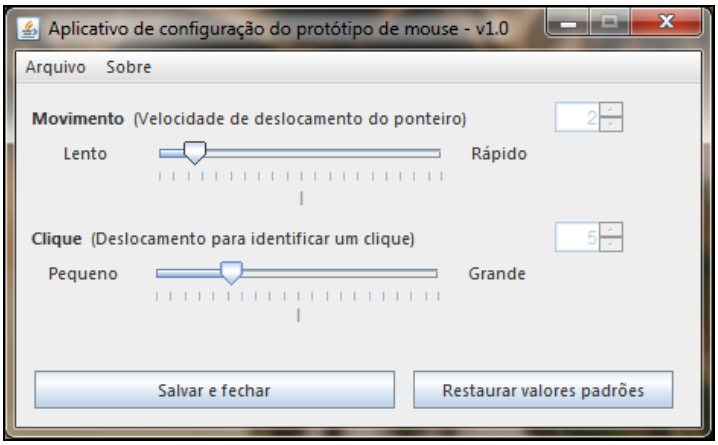


Figura 21 - Aplicativo de configuração do protótipo de *mouse*

O usuário seleciona os valores desejados para os campos movimento e clique, e após isso clica no botão `Salvar e fechar`. O sistema irá abrir uma janela solicitando o diretório onde o arquivo deve ser salvo, conforme Figura 22.

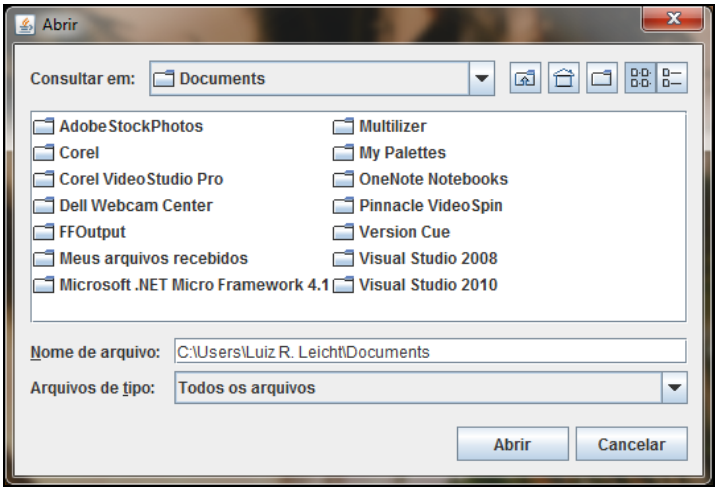


Figura 22 - Tela para seleção de diretório

Após selecionar o diretório onde o arquivo deve ser salvo, o usuário clica no botão Abrir. O arquivo de configuração será gerado e criado no respectivo diretório selecionado.

Para alteração de um arquivo de configuração, o usuário deve selecionar a opção `Carregar` no menu `Arquivo`. A Figura 23 mostra a opção Carregar do aplicativo de configuração.

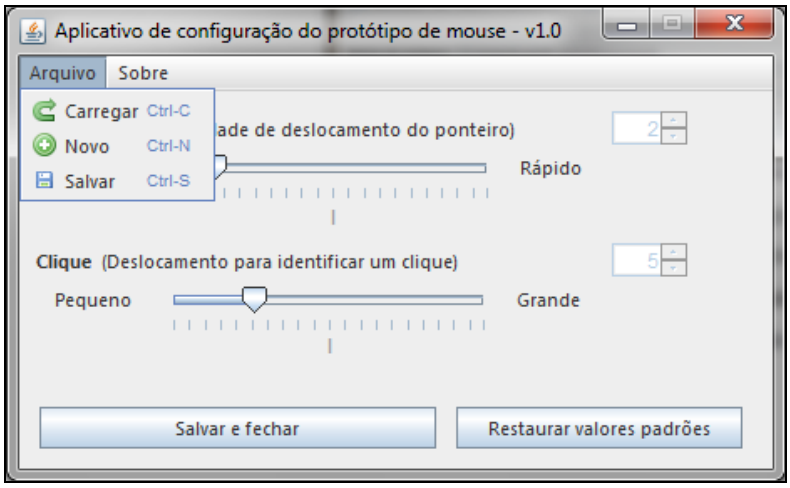


Figura 23 - Opção `Carregar` no aplicativo de configuração

Após clicar na opção Carregar, será exibida uma tela para selecionar o arquivo de configuração que será alterado. A Figura 24 exibe a tela para seleção do arquivo.

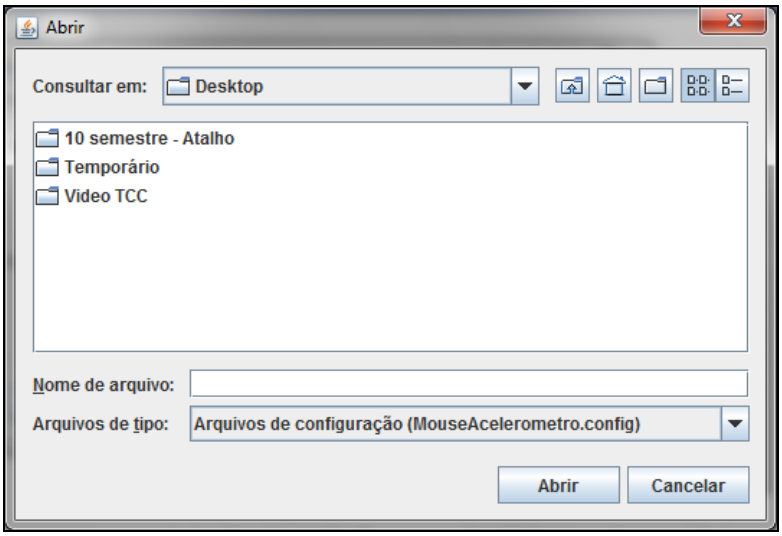


Figura 24 - Tela para seleção do arquivo de configuração

Após selecionar o arquivo de configuração, o usuário clica no botão `Abrir`. Após isso a aplicação preenche os campos `Movimento` e `Clique` com os respectivos valores do arquivo selecionado e permite que o usuário altere os mesmos. Após a alteração o usuário clica no botão `Salvar e fechar`. O arquivo selecionado será atualizado com os novos valores informados pelo usuário

3.3.2.2 Utilização do protótipo de *mouse*

Nesta seção será apresentada a utilização do protótipo de *mouse* por uma pessoa que possui deficiência motora nos membros superiores e que com ajuda de um usuário auxiliar

está efetuando a preparação do protótipo de *mouse*.

O usuário auxiliar conecta a placa FEZ Domino ao computador utilizando o cabo USB/miniUSB. A Figura 25 ilustra o processo de conexão da placa FEZ Domino ao computador.

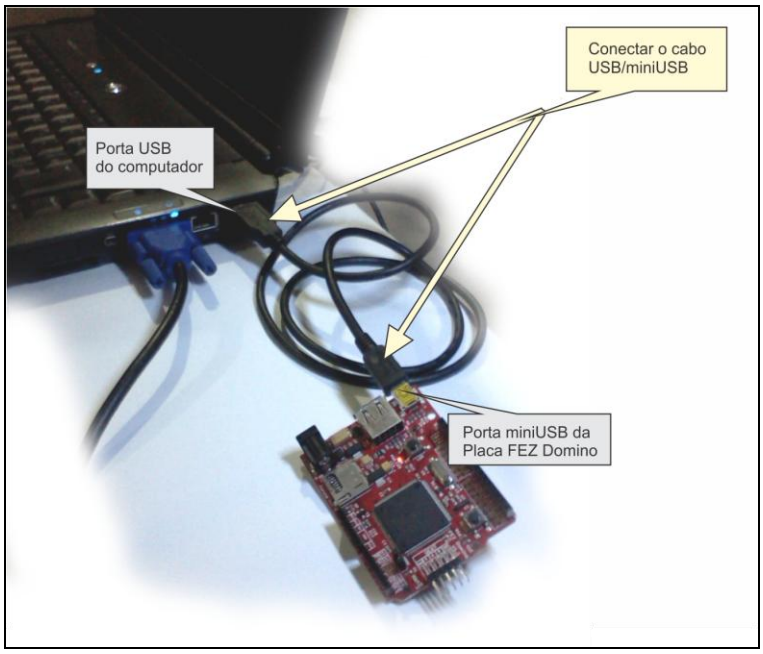


Figura 25 - Conexão da placa FEZ Domino ao computador

Após a conexão do cabo USB/miniUSB, o usuário auxiliar conecta à placa FEZ Domino o *access point* CC1111 USB RF. Na Figura 26 é demonstrada a conexão do *access point*.

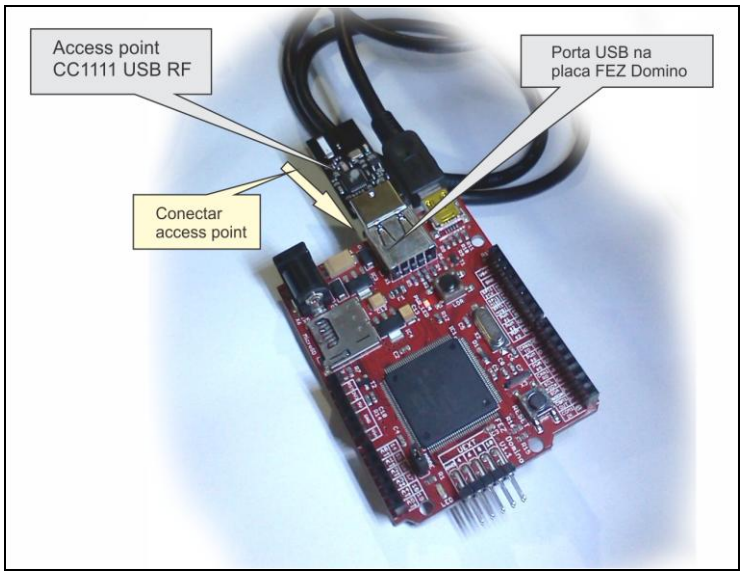


Figura 26 - Conexão do *access point* CC1111 USB RF na placa FEZ Domino

Para finalizar a preparação do protótipo, o usuário auxiliar deve ativar o acelerômetro no dispositivo eZ430-Chronos. A Figura 27 ilustra o acelerômetro ativado.

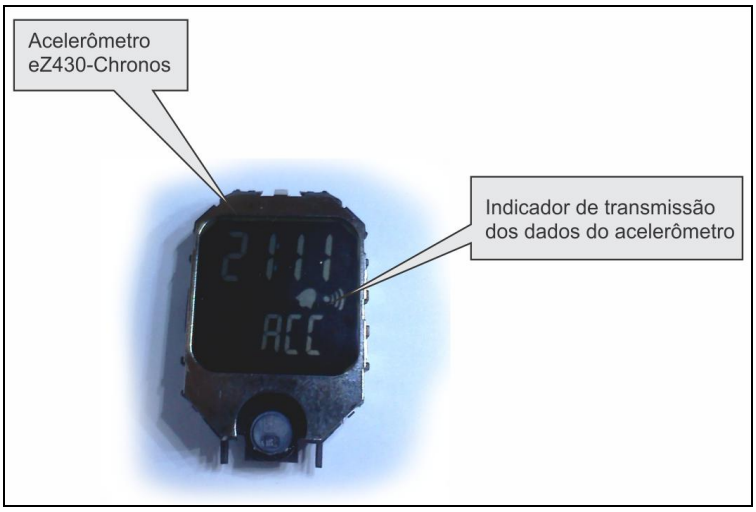


Figura 27 - Acelerômetro eZ430-Chronos ativado

Após o preparo do protótipo, o usuário auxiliar coloca o acelerômetro no adaptador, e após isso, ajusta o adaptador na cabeça do usuário que possui deficiência motora. Caso tenha-se criado um arquivo de configuração com os valores customizados para a velocidade e o clique do *mouse*, o usuário auxiliar coloca o cartão micro SD, o qual contém o arquivo de configuração, no leitor de cartão micro SD da placa FEZ Domino. Na Figura 28 é apresentada a conexão do cartão micro SD na placa FEZ Domino.

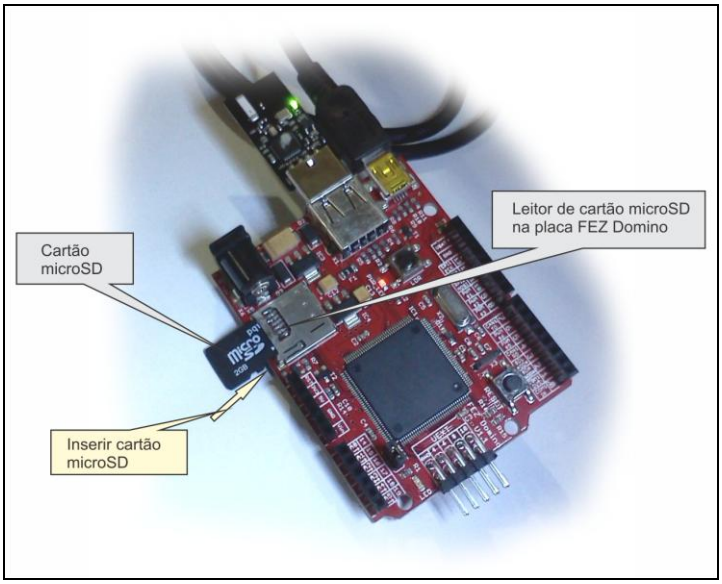


Figura 28 - Conexão do cartão micro SD na placa FEZ Domino

Para realizar o movimento do cursor para baixo, o usuário com deficiência física motora deve inclinar a cabeça levemente para frente. Quanto maior for a inclinação, mais rápido será o deslocamento do ponteiro de *mouse*. Para movimentar o cursor para cima, deve-se inclinar a cabeça levemente para trás. Para deslocar o cursor para o lado esquerdo, deve-se inclinar a cabeça levemente para a esquerda. Para o deslocamento do cursor para o lado direito, deve-se inclinar a cabeça levemente para o lado direito.

Para simular um clique com o botão esquerdo do *mouse*, o usuário com deficiência física motora efetua um rápido movimento com a cabeça inclinando-a para frente e retornando a posição anterior. O protótipo irá identificar esse movimento como um clique com o botão esquerdo e irá enviar esse comando ao computador. Para simular um clique com o botão direito do *mouse*, deve-se efetuar um rápido movimento inclinando a cabeça para o lado esquerdo ou direito e retornar à posição anterior. O protótipo identifica o movimento como um comando de clique com o botão direito e envia esse comando ao computador. Para simular um duplo clique com o botão esquerdo do *mouse*, deve-se realizar dois rápidos movimentos com a cabeça, inclinando-a para frente ou para trás, e retornar a sua posição anterior. O protótipo identificará cada movimento como sendo um clique simples esquerdo, e enviará estes ao computador, que realizará o comando de clique duplo.

Caso seja necessário alterar a precisão de identificação do clique ou a velocidade de deslocamento do cursor, o usuário auxiliar remove o cartão micro SD da placa FEZ Domino e altera os parâmetros do arquivo de configuração utilizando o aplicativo de configuração do protótipo de *mouse*. Depois de alterado o arquivo, o usuário auxiliar insere novamente o cartão micro SD na placa FEZ Domino. As novas configurações são carregadas pelo protótipo e o usuário com deficiência física motora realiza novos movimentos com a cabeça.

3.4 RESULTADOS E DISCUSSÃO

O paciente não teve nenhum treinamento ou preparo para utilização do protótipo, apenas foram-lhe dadas algumas instruções básicas de como proceder para efetuar os cliques e movimentos com o protótipo. O computador utilizado foi o do próprio paciente, no qual, nenhum aplicativo foi instalado e nenhuma configuração foi alterada. Durante os testes, o paciente, que possui tetraplegia causada por um acidente de moto, foi submetido a um circuito de testes, dos quais citam-se: movimentos para cima, para baixo e para os lados com o cursor do *mouse*; movimentos para efetuar clique simples com o botão esquerdo e direito do *mouse* e movimentos para efetuar o clique duplo do *mouse*. Após um curto período de tempo para adaptação e alguns ajustes na calibração da sensibilidade de clique e na velocidade de deslocamento do *mouse*, o paciente demonstrou ter adquirido domínio sobre os movimentos efetuados com o acelerômetro, conseguindo completar todos os testes ao qual foi submetido.

O paciente somente encontrou dificuldades para efetuar o clique duplo, o qual é obtido

efetuando-se dois movimentos rápidos com a cabeça para frente. Para resolver este problema, foi necessário efetuar uma alteração na rotina de detecção de cliques. Após as alterações, o paciente foi submetido a novos testes, cujo resultado foi bastante satisfatório, pois o paciente conseguiu efetuar o clique duplo e os demais movimentos sem dificuldades. O que mostrou que o trabalho desenvolvido atingiu satisfatoriamente os objetivos iniciais.

Durante os testes, o paciente, utilizando-se do protótipo de *mouse* e de um teclado virtual, conseguiu abrir um *browser* e efetuar diversas pesquisas no Google. Acessou também diversas pastas do seu computador e alterou o plano de fundo da sua área de trabalho. Isso mostrou o quanto o protótipo está apto para ser utilizado por pessoas que possuem algum tipo de deficiência motora nos membros superiores.

O paciente quando questionado sobre a usabilidade do protótipo em comparação aos demais *mouses* convencionais disponíveis no mercado, afirmou que o protótipo, para o seu caso em particular, é muito melhor, pois lhe permite efetuar movimentos e cliques com mais destreza.

Na fase de validação, foi encontrada uma limitação do protótipo para carregar o arquivo de configuração para o cartão micro SD. Pois somente era possível carregar o arquivo de configuração se o computador utilizado disponibilizasse uma porta de comunicação para cartões micro SD. Para resolver esse problema, o protótipo foi alterado para emular também um disco de armazenamento removível. A emulação do disco de armazenamento é alternada com o *mouse* sempre que o botão `LDR` da placa FEZ Domino é pressionado. Esta implementação permitiu ao protótipo ser reconhecido pelo computador como um *mouse* e também como um disco de armazenamento removível, o que possibilitou carregar o arquivo de configuração para dentro do cartão micro SD sem a necessidade de ter um leitor de cartões micro SD no computador.

Para comprovar a usabilidade do protótipo em múltiplas plataformas, o protótipo foi submetido a testes nos sistemas operacionais Windows, Linux e Mac OS X. Os testes incluíram movimentos com o cursor do *mouse* em todas as direções, clique simples com o botão esquerdo e direito e clique duplo. Em todos os testes efetuados, o protótipo apresentou cem por cento de compatibilidade com os sistemas operacionais.

Com relação aos trabalhos correlatos e ao trabalho de Jennrich (2010), o protótipo desenvolvido contempla as principais características de um *mouse* convencional, tais como movimento com o cursor, clique simples com o botão direito e esquerdo e clique duplo, além de ser multiplataforma e não necessitar da instalação de nenhuma aplicação auxiliar para detecção ou funcionamento do *mouse*. O Quadro 18 apresenta algumas funcionalidades e

características entre o protótipo desenvolvido neste trabalho, os trabalhos correlatos e o trabalho desenvolvido por Jennrich (2010).

Comparação					
Funcionalidade	Tongue Drive	Mouse visual Bradesco	Capacete Emotiv EPOC	Protótipo Jennrich (2010)	Protótipo desenvolvido
Funciona sem instalação de aplicativos	--	--	--	--	X
Multiplataforma	--	--	X	--	X
Movimenta cursor	X	X	X	X	X
Clique direito	--	--	--	X	X
Clique esquerdo	X	X	--	X	X
Clique seleção	--	--	--	X	--
Clique duplo	X	--	--	X	X
Comunicação sem fio	X	X	X	--	X

Quadro 18 - Comparação entre trabalhos

Após analisar o Quadro 18, é possível evidenciar que o protótipo desenvolvido possui grandes vantagens quando comparado aos trabalhos correlatos, tais como, comunicação sem fio, funciona sem a necessidade de instalação de aplicativos auxiliares, é multiplataforma e realiza quase todos os tipos de clique, com exceção apenas do clique seleção.

Segundo Texas Instruments (2010, p. 64), o dispositivo eZ430-Chronos possui um tempo de bateria reduzido quando utiliza o acelerômetro ativado por muito tempo. O Quadro 19 apresenta um demonstrativo da duração da bateria para alguns dos recursos do dispositivo eZ430-Chronos.

Recurso	**Tempo duração**
Data/hora	27,7 meses
Medição de temperatura	25 meses
Medição de altitude (contínuo/direto)	13,8 meses
Modo acelerômetro (contínuo/direto)	2 dias
Modo acelerômetro (1h/dia)	1,4 meses

Quadro 19 - Demonstrativo da duração de bateria do dispositovo eZ430-Chronos

O Quadro 20 apresenta os valores gastos para a construção do protótipo de *mouse* e aquisição dos produtos que se fizeram necessários para o desenvolvimento deste trabalho.

Produto	**Valor produto (R$)**	**Valor impostos (R$)**	**Total (R$)**
Placa FEZ Domino	271,76	133,92	405,68
Dispositivo eZ430-Chronos	164,32	217,79	382,11
Cabo serial com conversor USB	40,00	0,00	40,00
Adaptador para acelerômetro	10,00	0,00	10,00
Total	**486,08**	**351,71**	**837,79**

Quadro 20 - Custos para construção do protótipo de *mouse*

Para a preparação do hardware do protótipo, não foi necessário desenvolver um projeto no ambiente de simulação de circuitos Proteus. Isto porque o acelerômetro eZ430-Chronos escolhido já possui uma interface de comunicação com a placa FEZ Domino.

4 CONCLUSÕES

O presente trabalho teve como objetivo desenvolver um protótipo de *mouse* baseado nos requisitos funcionais do trabalho de Jennrich (2010), melhorando o desempenho, compatibilidade com outros SO, identificação dos cliques e usabilidade para o usuário final.

O protótipo de *mouse* é direcionado para pessoas portadoras de deficiência física motora nos membros superiores. Os resultados obtidos com o protótipo mostraram uma ótima usabilidade e facilidade de sua utilização até mesmo por pessoas que tenham um grau de deficiência física motora mais grave, como no caso de pessoas com tetraplegia. O protótipo permite ao usuário, através de movimentos e inclinações efetuadas com a cabeça, efetuar diversas tarefas em um computador, das quais citam-se: abrir pastas e arquivos; acessar *sites*; enviar *e-mail*; digitar textos utilizando-se de um teclado virtual e acessar redes sociais.

Inicialmente o acelerômetro do protótipo seria colocado no ombro do usuário, permitindo que fossem realizados movimentos para deslocar o cursor nas direções desejadas. Porém, nos primeiros testes com o protótipo, foi identificada certa dificuldade em se realizar movimentos com o ombro. Foi criado então um suporte para adaptar o acelerômetro à cabeça de uma pessoa. O resultado foi uma melhora significativa na precisão dos movimentos e maior facilidade para efetuar os cliques.

Durante o desenvolvimento do protótipo encontraram-se algumas dificuldades como, por exemplo, deficiência na exibição de mensagens de erros quando a versão de `dll` das bibliotecas era diferente da versão do firmware da placa micro-controladora, pois o erro não ocorria na compilação do código, somente ocorria na execução do projeto e ainda assim, as mensagens não eram claras com relação ao erro. Outra dificuldade encontrada foi para implementar o clique do *mouse*, pois quando o clique é efetuado há também um deslocamento que representa o movimento do cursor.

No desenvolvimento do protótipo, encontraram-se dificuldades também para efetuar a leitura do cartão micro SD, pois havia solda fria em algumas trilhas da porta micro SD da placa FEZ Domino que impossibilitavam a leitura do cartão. Outra dificuldade encontrada foi para efetuar a emulação de *mouse* e do disco removível alternadamente, pois a emulação alternada causava diversas exceções causadas por acesso à objetos nulos, causando interrupção no funcionamento do protótipo.

Os seguintes itens podem ser destacados como objetivos alcançados no desenvolvimento do protótipo: utilizada a placa FEZ Domino e o acelerômetro eZ430-

Chronos como parte de hardware do protótipo; desenvolvida aplicação para realizar a calibração da velocidade de deslocamento e sensibilidade de clique do *mouse*; utilizada comunicação via conexão USB para conectar a placa FEZ Domino ao computador e aperfeiçoada a forma como os cliques são efetuados e identificados.

Como limitação o protótipo possui apenas a ausência do clique seleção, o qual, entre outras funcionalidades, permite selecionar vários arquivos e trechos de textos. A ausência desse tipo de clique se decorreu por dificuldades encontradas para efetuar a calibração e identificação de um movimento que correspondesse ao respectivo clique e que houvesse extrema precisão em sua detecção.

4.1 EXTENSÕES

Como sugestão para continuação deste trabalho e melhoria do protótipo, pode-se sugerir:

a) implementar o clique seleção;
b) desenvolver um hardware que possa baratear o custo do protótipo, visando a comercialização do produto;
c) testar o protótipo em um órgão especializado no tratamento de pessoas com deficiência física motora, buscando aperfeiçoar ainda mais os movimentos e cliques realizados.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE MEDICINA FÍSICA E REABILITAÇÃO. **Espasticidade:** avaliação clínica. [Minas Gerais], 2006. Disponível em: <http://www.projetodiretrizes.org.br/5_volume/18-Espasticid.pdf>. Acesso em: 26 maio 2011.

BANCO BRADESCO S.A. **Bradesco lança mouse visual para clientes com deficiência física acessarem a conta-corrente pela Internet**. [S.l.], 2010. Disponível em: <http://www.bradesco.com.br/html/content/noticias/noticias_3.shtm>. Acesso em: 18 mar. 2011.

BBC BRASIL. **Dispositivo na língua pode ajudar comando de tetraplégicos**. [S.l.], 2008. Disponível em: <http://www.bbc.co.uk/portuguese/reporterbbc/story/2008/06/080630_linguadeficientes_np.shtml>. Acesso em: 21 mar. 2008.

BRASIL. Decreto n. 5.296, de 2 de dezembro de 2004. Regulamenta as Leis n. 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências. **Presidência da República**: Casa Civil - Subchefia para Assuntos Jurídicos, Brasília, 2004. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2004/Decreto/D5296.htm>. Acesso em: 27 mar. 2011.

DEFICIÊNCIA física. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2010. Disponível em: <http://pt.wikipedia.org/wiki/Defici%C3%AAncia_f%C3%ADsica>. Acesso em: 26 mar. 2011.

EMOTIV. **EPOC neuroheadset**. [S.l.], [2011?]. Disponível em: <http://www.emotiv.com/apps/epoc/299/>. Acesso em: 20 mar. 2011.

ESPASTICIDADE. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2011. Disponível em: <http://pt.wikipedia.org/wiki/Espasticidade>. Acesso em: 26 maio 2011.

FERREIRA, João C. **Periféricos de computadores**. [Porto], 2007. Disponível em: <http://paginas.fe.up.pt/~jcf/ensino/disciplinas/mieic/arq/2006-07/07-perifericos.notes-2.pdf>. Acesso em: 05 abr. 2011.

FURTADO JÚNIOR, Miguel B. **XML**. [Rio de Janeiro], [2010?]. Disponível em: <http://www.gta.ufrj.br/grad/00_1/miguel/index.html>. Acesso em: 09 ago. 2011.

GHI ELETRONICS. **FEZ Domino**, [S.l.], 2011. Disponível em: <http://www.ghielectronics.com/catalog/product/133>. Acesso em: 27 mar. 2011.

GHI ELETRONICS. **USBizi user manual**. [S.l.], 2011. Disponível em: <http://www.ghielectronics.com/downloads/USBizi/USBizi_User_Manual.pdf>. Acesso em: 08 ago. 2011.

HEITLINGER, Paulo. **O guia prático da XML**. [Porto], [2001]. Disponível em: <http://www.centroatl.pt/titulos/tecnologias/imagens/oguiapraticoda-xml-excerto.pdf>. Acesso em: 15 jul. 2011.

HUMAN interface device. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2011. Disponível em: <http://en.wikipedia.org/wiki/Human_interface_device>. Acesso em: 06 abr. 2011.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Censos demográficos**. [Brasília], 2010. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/08052002tabulacao.shtm>. Acesso em: 21 mar. 2011.

ISSA, Gus. **Guia para iniciantes em C# e .NET Micro Framework**. Tradução Miguel Alexandre Wisintainer. Blumenau, 2010.

JENNRICH, Gabriele. **Protótipo de um mouse utilizando acelerômetros**. 2010. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LEITE, Marcelo. **Uma palavra sobre Open Source**. [S.l.], 2005. Disponível em: <http://www.anysoft.com.br/content/view/13/52/>. Acesso em: 27 mar. 2011.

MACIEL, Maria C. B. T. Deficiência física. In: ARANTES, Elzira (Ed.). **Deficiência mental. Deficiência física**. Brasília : Ministério da Educação e do Desporto, 1998. p. 50-96. Disponível em: <http://www.dominiopublico.gov.br/download/texto/me000351.pdf>. Acesso em: 28 mar. 2011.

MACORATTI, José C. **C # - Apresentando o .NET Micro Framework (NETMF)**. [S.l.], 2010. Disponível em: <http://imasters.com.br/artigo/19313/dotnet/c___apresentando_o_net_micro_framework_netmf/>. Acesso em: 05 abr. 2011.

TEXAS INSTRUMENTS. **CC430 general overview**. [S.l.], 2010. Disponível em: <http://processors.wiki.ti.com/index.php/CC430_General_Overview>. Acesso em: 22 jun. 2011.

_____. **eZ430-Chronos development tool**. [S.l.], 2010. Disponível em: <http://robotics.ee.uwa.edu.au/courses/high-perf-emb-sys/TI-Chronos.pdf>. Acesso em: 11 ago. 2011.

TINYCLR. **Driver:** eZ430 chronos. [S.l.], 2011. Disponível em: <http://code.tinyclr.com/project/210/ez430-chronos/>. Acesso em: 22 jun. 2011.

VELOSO, Thassius. **Bradesco lança mouse visual para deficientes físicos**. [S.l.], 2010. Disponível em: <http://tecnoblog.net/21499/bradesco-lanca-mouse-visual-para-deficientes-fisicos/>. Acesso em: 05 abr. 2011.

XML. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2011. Disponível em: <http://pt.wikipedia.org/wiki/XML#Caracter.C3.ADsticas_do_XML>. Acesso em: 10 ago. 2011.

W3SCHOOLS. **XML tutorial**. [São Paulo], [2011?]. Disponível em: <http://www.centroatl.pt/titulos/tecnologias/imagens/oguiapraticoda-xml-excerto.pdf>. Acesso em: 11 ago. 2011.

APÊNDICE A – Detalhamento dos casos de uso configuração

No Quadro 21 é apresentado o cenário para o caso de uso configurar protótipo de mouse.

UC001 – Configurar protótipo de mouse

1. **Descrição**: O usuário auxiliar configura o mouse de acordo com suas preferências.
   - 1.1. **Ator**: Usuário auxiliar.
   - 1.2. **Pré-condições**: Nenhum.
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Configurar protótipo de mouse.
       a. O usuário auxiliar inicia o aplicativo MouseConfig.jar.
       b. O usuário auxiliar preenche os campos movimento e clique.
       c. O usuário auxiliar clica em salvar e fechar.
       d. O sistema exibe uma janela com os diretórios do computador.
       e. O usuário auxiliar seleciona o diretório onde o arquivo deve ser salvo.
       f. O usuário auxiliar clica em abrir.
       g. O sistema cria no diretório selecionado um arquivo no formato XML com o nome MouseAcelerometro.config contendo as configurações do mouse.
     - 1.3.2. Fluxo alternativo.
       a. Carregar configuração.
       I. O usuário auxiliar inicia o aplicativo MouseConfig.jar.
       II. O usuário auxiliar clica em carregar.
       III. O sistema solicita ao usuário o arquivo que será alterado.
       IV. O usuário auxiliar seleciona o arquivo de configuração.
       V. O sistema carrega os dados do arquivo de configuração.
       VI. O usuário auxiliar altera os dados dos campos movimento e clique.
       VII. O usuário auxiliar clica em salvar e fechar.
       VIII. O sistema atualiza o arquivo de configuração.
   - 1.4. **Pós-Condição**: O arquivo de configuração do mouse está criado.

Quadro 21 - Cenário de caso de uso configurar *mouse*

No Quadro 22 é apresentado o cenário para o caso de uso salvar arquivo de configuração no cartão micro SD.

UC002 – Salvar arquivo de configuração no cartão micro SD

1. **Descrição**: O usuário auxiliar salva o arquivo de configuração no cartão micro SD.
   - 1.1. **Ator**: Usuário auxiliar.
   - 1.2. **Pré-condições**: O computador deve possuir leitor de cartão micro SD.
     Arquivo de configuração já deve estar criado;
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Salvar arquivo de configuração no cartão micro SD.
       a. O usuário auxiliar insere o cartão micro SD no leitor de micro SD do computador.
       b. O sistema operacional identifica o cartão micro SD.
       c. O usuário auxiliar seleciona o arquivo de configuração salvo no computador.
       d. O usuário auxiliar copia o arquivo de configuração para o cartão micro SD no diretório raiz.
       e. O usuário auxiliar remove o cartão micro SD do computador.
   - 1.4. **Pós-Condição:** O arquivo de configuração está salvo no cartão micro SD.

Quadro 22 - Cenário do caso de uso salvar arquivo de configuração no cartão micro SD

No Quadro 23 é exibido o cenário para o caso de uso carregar configuração.

UC003 – Carregar configuração

1. **Descrição**: O usuário auxiliar carrega as configurações salvas no cartão micro SD para a placa FEZ Domino.
   - 1.1. **Ator**: Usuário auxiliar.
   - 1.2. **Pré-condições**: Arquivo de configuração deve estar salvo no diretório raiz do cartão micro SD.
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Carregar configuração.
       a. O usuário auxiliar insere o cartão micro SD no leitor de micro SD da placa FEZ Domino.
       b. A placa FEZ Domino carrega o arquivo de configuração salvo no cartão micro SD.
   - 1.4. **Pós-Condição**: O cartão micro SD está inserido na placa FEZ Domino.

Quadro 23 - Cenário do caso de uso carregar configuração

APÊNDICE B – Detalhamento dos casos de uso protótipo de *mouse*

No Quadro 24 é exibido o cenário para o caso de uso conectar dispositivo FEZ Domino.

UC004 – Conectar dispositivo FEZ Domino

1. **Descrição**: O usuário conecta o dispositivo FEZ Domino ao computador.
   - 1.1. **Ator**: Usuário.
   - 1.2. **Pré-condições**: Nenhum.
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Conectar dispositivo FEZ Domino.
       - a. O usuário conecta à placa FEZ Domino na porta mini-USB um cabo com conector mini-USB.
       - b. O usuário conecta em um Computador a outra extremidade do cabo USB.
       - c. O sistema operacional identifica a placa FEZ Domino como um dispositivo de interface humana.
   - 1.4. **Pós-Condição**: A placa FEZ Domino está conectada ao computador pelo cabo USB.

Quadro 24 - Cenário do caso de uso conectar dispositivo FEZ Domino

No Quadro 25 é exibido o cenário para o caso de uso conectar *access point*.

UC005 – Conectar access point

1. **Descrição**: O usuário conecta o dispositivo access point CC1111 USB RF na placa FEZ Domino.
   - 1.1. **Ator**: Usuário.
   - 1.2. **Pré-condições**: Nenhum.
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Conectar access point.
       - a. O usuário conecta o dispositivo de access point CC1111 USB RF na placa FEZ Domino utilizando a porta USB.
       - b. A placa FEZ Domino identifica o access point conectado.
   - 1.4. **Pós-Condição**: O dispositivo access point CC1111 USB RF está conectado na placa FEZ Domino.

Quadro 25 - Cenário do caso de uso conectar *access point*

No Quadro 26 é exibido o cenário para o caso de uso ativar acelerômetro.

UC006 – Ativar acelerômetro

1. **Descrição**: O usuário ativa o acelerômetro eZ430-Chronos.
   - 1.1. **Ator**: Usuário.
   - 1.2. **Pré-condições**: Nenhum.
   - 1.3. **Fluxo de eventos**:
     - 1.3.1. Fluxo básico – Ativar acelerômetro.
       - a. O usuário seleciona a opção acelerômetro no dispositivo eZ430-Chronos.
       - b. O usuário ativa a transmissão wireless no dispositivo eZ430-Chronos.
   - 1.4. **Pós-Condição**: O acelerômetro está ativado no dispositivo eZ430-Chronos.

Quadro 26 - Cenário do caso de uso ativar acelerômetro

No Quadro 27 é exibido o cenário para o caso de uso colocar adaptador para *mouse*.

UC007 – Colocar adaptador para mouse

1. **Descrição**: O usuário auxiliar veste no usuário com deficiência física motora o adaptador para mouse.
   1.1. **Ator**: Usuário auxiliar.
   1.2. **Pré-condições**: Conectar dispositivo FEZ Domino.
   Conectar access point.
   Ativar acelerômetro.
   1.3. **Fluxo de eventos**:
   1.3.1. Fluxo básico – Vestir adaptador para mouse.
   a. O usuário auxiliar coloca o adaptador para mouse na cabeça do usuário com deficiência física motora.
   b. O usuário auxiliar prende o adaptador para mouse na cabeça do usuário com deficiência física motora.
   1.4. **Pós-Condição**: A adaptador para mouse está colocado no usuário com deficiência física motora.

Quadro 27 - Cenário do caso de uso colocar adaptador para *mouse*

No Quadro 28 é exibido o cenário para o caso de uso utilizar protótipo de *mouse* com acelerômetro.

UC008 – Utilizar protótipo de mouse com acelerômetro

1. **Descrição**: O usuário com deficiência física motora utiliza o protótipo de mouse com acelerômetro para movimentar o ponteiro do mouse no computador.
   1.1. **Ator**: Usuário com deficiência física motora.
   1.2. **Pré-condições**: O usuário com deficiência física motora deve estar com o adaptador para mouse colocado em sua cabeça.
   1.3. **Fluxo de eventos**:
   1.3.1. Fluxo básico –Utilizar protótipo de mouse com acelerômetro.
   a. O usuário com deficiência física motora realiza movimentos com a cabeça.
   b. O dispositivo do acelerômetro transmite os dados de aceleração e inclinação.
   c. A placa FEZ Domino processa os dados recebidos pelo acelerômetro através do access point.
   d. A placa FEZ Domino envia comandos para o computador movimentar o mouse.
   e. O sistema operacional do computador realiza os movimentos no ponteiro do mouse.
   1.4. **Pós-Condição**: O usuário com deficiência física motora movimentou o ponteiro do mouse no computador utilizando o protótipo de mouse com acelerômetro.

Quadro 28 - Cenário do caso de uso utilizar protótipo de mouse com acelerômetro