ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PROJETO MOMID

MONITORAMENTO DE MOVIMENTAÇÃO DE IDOSOS

São Paulo

2012

DANIEL CAVALLARI GONÇALVES

HALPH MACEDO FRAULOB

MARCOS CÉSAR VOLTOLINI

PROJETO MOMID

MONITORAMENTO DE MOVIMENTAÇÃO DE IDOSOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro.

São Paulo

2012

DANIEL CAVALLARI GONÇALVES

HALPH MACEDO FRAULOB

MARCOS CÉSAR VOLTOLINI

PROJETO MOMID

MONITORAMENTO DE MOVIMENTAÇÃO DE IDOSOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção dos títulos de Engenheiros

Área de Concentração: Engenharia Elétrica com ênfase em Sistemas Eletrônicos

Orientadores: Prof. Livre-Docente Marco Isaías Alayo Chávez e Prof. Livre-Docente Marcelo Nelson Páez Carreño

São Paulo

2012

AGRADECIMENTOS

Aos professores Marco Isaias Alayo Chávez e Marcelo Nelson Páez Carreño pela orientação, apoio e constante estímulo durante a realização deste trabalho.

Aos amigos Gustavo Pamplona Rehder pelo apoio, acompanhamento e colaboração, Luiz Medaglia e o restante dos técnicos de laboratório do departamento de Elétrica da Escola Politécnica da USP pela disposição e grande ajuda na realização deste trabalho.

RESUMO

O objetivo deste projeto é de criar um sistema sem fio que permite a detecção de quedas de pessoas. A queda de idosos foi o foco do projeto, uma vez que pessoas nessa faixa-etária podem sofrer graves consequência em tais acidentes, especialmente aquelas que moram sozinhas (que constituem um grande número de pessoas no Brasil). Um relógio, da Texas Instruments, que contém um acelerômetro tri-axial, foi utilizado para implementar o hardware, e um modelo de detecção de quedas foi desenvolvido e programado no microcontrolador do relógio. Utilizando o modelo criado, foi possível obter uma acurárcia de 90% na detecção de quedas. Este documento detalha a criação do sistema e permite a continuidade do trabalho.

Palavras-chave: Engenharia, Engenharia Elétrica, Detecção de quedas, Detecção de quedas de idosos.

ABSTRACT

The goal of this project is to create a wireless system that allows the detection of people's falls. The fall of the elderly was the focus of the project, since people in that age can suffer serious consequences from such accidents, especially those who live alone (who presently constitute a large number of people in Brazil). A clock, by Texas Instruments, which contains a tri-axial accelerometer, was used to implement the hardware, and a fall detection model was developed and programmed into the clock's microcontroller. Using the model created, it was possible to obtain a fall detection accuracy of over 90%. This document details the creation of the system and allows the work to be continued.

Key-words: Engineering, Electrical Engineering, Fall detection, Elderly fall detecction.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Problema Motivador

A popularização de métodos contraceptivos nos anos 80 e o aumento da expectativa de vida vêm causando, no Brasil, uma tendência à inversão da pirâmide etária brasileira [1], ou seja, há cada vez mais idosos e cada vez menos jovens no país; essa tendência é ilustrada na figura 1 abaixo, retirada de [2], comparando as faixas etárias do país em três períodos diferentes: 1980, 1991 e 2000.

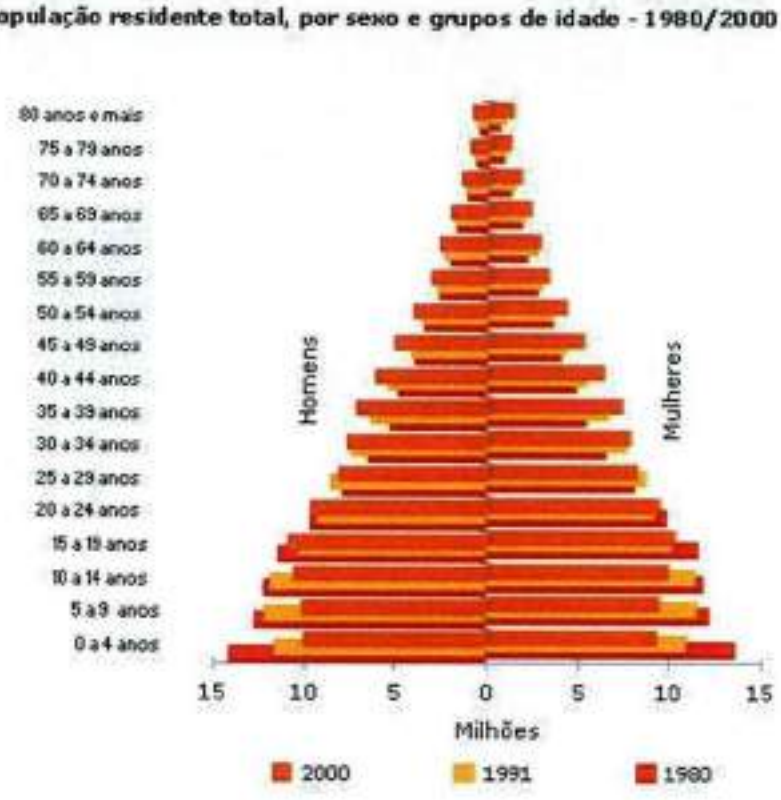

Figura 1 – Pirâmide etária brasileira em três períodos históricos

Analisando a figura, percebe-se um estreitamento na base da pirâmide, e um alargamento do seu topo, o que indica um crescimento da população idosa, ou seja, não economicamente ativa, em relação à faixa da população trabalhadora. Considerando que os idosos constituem a faixa de população mais suscetível a doenças e acidentes domésticos, principalmente quedas, os gastos com saúde

provenientes dessa faixa populacional é grande. Em 2009, o SUS gastou quase R$ 81 milhões com fraturas ósseas em pessoas idosas [3], a maioria dessas causadas por queda.

Além disso, o número de idosos com mais de 60 anos que moram sozinhos no Brasil representa 14% do total, ou quase 3 milhões, segundo o censo de 2011 realizado pelo IBGE [1]. Sendo assim, fica claro a necessidade da implementação de algum tipo de sistema remoto de socorro a idosos, principalmente os que moram sós, pois estes não possuem ninguém que possa socorrê-los caso fiquem impossibilitados de fazer ligações de socorro ou fiquem inconscientes.

1.2 Solução Proposta

A proposta do grupo para resolver o problema consiste em um sistema remoto de monitoramento de movimento de pessoas, com um foco especial em detecção de quedas, nomeado MOMID – Monitoramento de Movimento de Idosos. Esse sistema irá permitir que uma central entre em contato com a pessoa monitorada caso seja detectada uma queda e, caso o contato não seja possível, contate o resgate emergencial e informe que houve uma possível queda.

O objetivo do projeto é diminuir a taxa de mortalidade de quedas de idosos, principalmente os que moram sozinhos e que possam possivelmente ficar impossibilidades de pedir socorro ao caírem (devido a fraturas ou perda de consciência).

O projeto irá utilizar sensores acelerométricos (para monitorar movimentos e possíveis quedas) que enviarão dados e possíveis avisos de emergência para uma estação remota de processamento de dados. Esse sistema é confiável e pouco invasivo, permitindo ao idoso manter sua independência e sua privacidade.

2 REVISÃO DA LITERATURA E ESTADO DA ARTE

Considerando os dados apresentados na seção anterior, é fácil concluir que o assunto de detecção de quedas de idosos é de grande interesse econômico e social no mundo. Sendo assim, diversas pesquisas científicas com objetivos similares foram desenvolvidas e serviram de base para a construção do projeto MOMID.

Yu [4] descreve, em grandes detalhes, as características dos três diferentes tipos de quedas mais comuns aos idosos (queda da própria altura, queda de cadeiras e queda da cama; quedas de maior magnitude foram ignoradas em seu trabalho). Segundo ele, a maioria das quedas domésticas apresentam diversas características comuns, como duração (de um a três segundos), variação brusca na posição e aceleração do corpo, etc. Sendo assim, o autor sugeriu diversas soluções para monitorar pacientes, como câmeras, monitoramento do ambiente e, mais importante, dispositivos a serem usados no corpo do paciente. Esta última solução foi a adotada pelo projeto MOMID.

Wang et al [5] propõem um sistema muito similar ao proposto pelo projeto MOMID: um sistema de detecção de quedas de idosos utilizando acelerômetros fixados na cabeça dos pacientes, de modo a evitar ruído indesejado devido ao movimento natural do corpo durante atividades cotidianas, como andar, sentar, etc. Wang et al [6], por outro lado, propõem um sistema similar mas utilizando giroscópios além de acelerômetros, mas fixando os sensores no peito do paciente. Como há divergência no local de fixação dos sensores, fica clara a necessidade de um estudo extenso para determinar a melhor maneira e local de fixação do sensor para o projeto MOMID.

Além dos trabalhos de pesquisa citados acima, existem diversos produtos quase ou já implementados que procuram solucionar os problemas apresentados anteriormente. Um exemplo é o projeto SIMBAD [8], que utiliza sensores infravermelhos, similares a câmeras, montados nas paredes para gerar um perfil de temperatura que permite determinar a posição e velocidade de alvos dentro da área de visualização dos sensores. Segundo o autor, o sistema permite uma análise

confiável dos dados para detecção de quedas. Porém, com o sistema cria "imagens" térmicas, alguns usuários consideram o sistema muito invasivo.

Um projeto de análise do modo de andar (ou *gait*, em inglês) utilizando o acelerômetro presente no dispositivo iPod [9] permite adquirir dados sobre o modo de andar de uma pessoa com grande acurácia e consistência, utilizando um aparelho eletrônico muito comum atualmente. Esse projeto, combinado com um estudo relacionando o risco de queda e o modo de andar de idosos [10], permite a criação de um sistema confiável de prevenção de quedas, indicando que certos idosos precisam de monitoramento mais cuidadoso e contínuo.

Outro exemplo de projeto é o BIOTELEKINESY [11] propõe a criação de uma estação de monitoramento dentro da casa do idoso, utilizando câmeras para obtenção de dados e detecção de quedas e utilizando um computador com internet para transmitir dados para assistentes e familiares em caso de emergência.

Um produto implementado nos Estados Unidos é o Halo Monitoring [12] que apresenta um design muito parecido com o inicialmente desejado pelo projeto MOMID. Além de apresentar as características de outros PERS (Personal emergency response systems – Sistemas de resposta pessoal de emergência), como um botão de pânico, que o usuário aperta caso haja uma emergência, o Halo Monitoring apresenta sensores para detecção de quedas que automatizam o processo caso o usuário esteja incapacitado de apertar o botão. O sistema ainda conta com planos com GPS, o que aumenta sua utilidade.

3 DETALHAMENTO DO PROJETO

3.1 Proposta Inicial

Tomando como base as pesquisas e projetos mencionados na seção anterior, o projeto MOMID irá consistir, basicamente, na comunicação sem fio entre sensores de movimento fixados em pessoas e uma estação remota que processa, interpreta e traduz os dados coletados pelos sensores.

O grande desafio e objetivo primário do projeto consistem em estabelecer essa comunicação sem fio, tratar os dados e fazer com que os sensores e a estação remota interpretem corretamente quedas e ignorem movimentos normais que ocorrem quando a pessoa se senta, se deita, etc. O grupo não possui recursos para fazer a integração do projeto com um sistema médico ou de atendimento, mas pretende facilitar tal integração no futuro, em um outro trabalho ou possível comercialização.

Além disso, é objetivo secundário do trabalho criar um sistema de comunicação base remota-sensores que possa ser utilizado para outros tipos de sensores em outros trabalhos que envolvam o monitoramento de dados através de sensores. Esse esforço é motivado pela grande quantidade de projetos nessa linha que estão sendo realizados pelos laboratórios do departamento de Sistemas Eletrônicos da Escola Politécnica da USP.

3.2 Vantagens, Desvantagens e Justificativa

Uma vez que existem diversas soluções já implementadas e algumas pesquisas sobre o assunto, se fez necessário uma escolha objetiva de solução a ser adotada pelo projeto MOMID. Quatro critérios foram escolhidos para avaliar as possíveis soluções propostas pelo grupo ao analisar o problema. Os seguintes parâmetros foram julgados os mais importantes na avaliação, com seus respectivos pesos relativos entre parênteses:

- Respeito à privacidade do usuário (4)
- Disponibilidade de componentes (1)
- Menor incômodo ao usuário (2)
- Acurácia na detecção de quedas (4)

O grupo considera que um sistema que respeita ao máximo a privacidade do usuário com grande acurácia em detectar quedas é ideal, minimizando o incômodo ao usuário. Convém mencionar que o grupo não utilizou o critério custo na sua análise, uma vez que, sendo este um projeto de formatura, o grupo é seu próprio cliente.

Utilizando esses critérios, o grupo fez uma análise por matriz de decisão (AHP) comparando as soluções mais comuns e viáveis propostas pelo grupo, listadas abaixo:

- Sistema utilizando acelerômetros
- Sensores Infravermelhos
- Câmeras
- Pisos Piezoelétricos
- Sistema sonoro de detecção

A matriz de decisão, já normalizada, se encontra abaixo, com as notas relativas e o *score* final das soluções propostas.

Tabela 1 – Matriz de decisão referente às soluções mais comuns ao problema proposto

	PESO	ACELERÔMETRO	IV	CÂMERAS	PISOS	SOM
PRIVACIDADE	0.36	0.31	0.22	0.09	0.22	0.16
DISPONIBILIDADE	0.18	0.24	0.18	0.26	0.13	0.18
INCÔMODO	0.09	0.19	0.16	0.16	0.27	0.22
ACURÁCIA	0.36	0.20	0.20	0.24	0.20	0.17
SCORE		0.24	0.20	0.18	0.20	0.17

Como se pode observar, a solução utilizando acelerômetros foi considerada a melhor e por isso foi escolhida. Os sensores aceleremétricos permitem a criação de um sistema que respeite a privacidade do usuário (não há gravação de imagens ou sons) e com uma boa acurácia.

4 REQUISITOS DO SISTEMA

Os requisitos de engenharia e de marketing do sistema foram analisados e estão listados na tabela a seguir, além de uma descrição mais detalhada do que se espera do projeto levando em conta tais requisitos. Tal análise foi feita visando uma possível futura comercialização do produto, de modo que os requisitos analisados foram, conforme são referenciados na tabela:

1 – Fácil Configuração;

2 – Fácil Usabilidade;

3 – Leve;

4 – Pequeno;

5 – Discreto;

6 – Barato;

7 – Seguro;

8 - Grande autonomia;

9 – Alcance.

Tabela 2 – Requisitos do sistema

Requisito de marketing	Requisitos de engenharia	Justificativa
3, 4, 5,	O Token utilizado pelo usuário deverá ser ter máximo o tamanho e a massa de um *pendrive* comum.	Este Token será utilizado pelo usuário o dia inteiro. Possivelmente através cordão pendurado no pescoço ou ainda através de uma presilha
1, 2	O token e a base possuirão apenas um botão. E qualquer tipo de configuração é efetuada conversando com um atendente.	Como nossos usuários finais são idosos, acreditamos que o sistema deve ser o mais simples possível. Usuários com uma idade mais avançada certamente preferem conversas com uma pessoa ao invés de configurar um equipamento através de um menu.
2, 8	O sistema deve operar por no mínimo 24 horas ininterruptas. (24 horas é o suficiente).	Como é um sistema que deverá passar o máximo de tempo junto ao usuário, se faz necessária uma autonomia elevada.
6	O preço do produto não deverá ultrapassar R$300,00.	Preço baseado nos concorrentes.
9	O sistema deverá funcionar em toda a região interna da casa do usuário.	Como é um sistema de auxílio a quedas e emergências, ele deve funcionar a qualquer momento e em qualquer lugar.
7	O alarme de quedas não pode falhar. É preferível um alarme de queda quando não existiu uma queda, do que a falta deste alarme em caso positivo. Ou seja, falsos negativos (não queda) não serão permitidos.	A partir do momento em que garantimos o funcionamento de um sistema autônomo o usuário passa a confiar neste sistema. Logo assumi-se esta responsabilidade, pois em um sistema como este vidas estão envolvidas.
7, 8	Mesmo em caso de falta de energia a base deve possuir uma autonomia no mínimo igual a do Token	A base é a ponte de comunicação entre o Token e a central de atendimento. De nada vale o Token estar funcionando se a base não está.

5 DESIGN

5.1 Introdução

De forma geral o MOMID será um sistema capaz de monitorar quedas e situações adversas, auxiliar de maneira remota o usuário e efetuar pedidos de socorro em casos extremos. Esse capítulo visa descrever o design proposto inicialmente; durante o decorrer do projeto alterações foram feitas e o produto final não corresponde exatamente ao descrito aqui. O design descrito neste capítulo representa o ideal, que poderá ser implementado dessa forma futuramente, para comercialização ou maior desenvolvimento do projeto. A forma final do projeto é detalhada no capítulo seguinte.

O sistema é composto basicamente por 3 partes: um Token, usado pelo usuário 24 horas por dia, uma base, responsável por efetuar a comunicação externa a residência e uma central, responsável por atender o usuário, coletar dados de movimentação e efetuar chamadas de emergência.

Figura 2 - Diagrama do sistema

5.2 Token

O Token certamente é a parte mais crítica do sistema, é um componente que o usuário usará 24 horas por dia e por isso possui uma série de requisitos que devem ser atendidos.

5.2.1 Partes constituintes

Figura 3 - Token

5.2.2 Funcionalidades

O Token possui a princípio as seguintes funcionalidades:

- Monitoração de queda
- Monitoração de movimento
- Botão de socorro

5.2.2.1 Monitoração de queda

Através de um acelerômetro o movimento do usuário é constantemente avaliado verificando-se possíveis padrões que indiquem queda. Ao sinal de uma queda o Token envia um sinal à base e esta entra em contato com a central. Em seguida a base cria uma ponte entre o Token e a central, desta forma um operador se comunica com o usuário diretamente pelo Token, visto que este possui um speaker e um microfone. Caso o usuário informe que está tudo bem nada é feito. Mas se o usuário pedir ajuda ou mesmo não se comunicar com o atendente um chamado de socorro é efetuado.

5.2.2.2 Monitoração de movimento

O acelerômetro e o giroscópio presentes no Token estão constantemente sensoriando a movimentação do usuário. Desta forma é possível verificar se este anda fazendo, por exemplo, uma quantidade adequada de exercício diariamente, levando assim uma vida mais saudável. Ou até mesmo monitorar situações adversas como o usuário ter ido dormir mais depois de 16 horas ainda não ter se movimentado, algo provavelmente aconteceu.

Registrando estes dados ao longo do tempo é possível ainda verificar mudanças na forma como o usuário se movimenta, por exemplo, o usuário pode começar a mancar de um dia para o outro, a central pode entrar em contato com ele perguntando se está tudo bem.

Os dados de monitoração a princípio não são enviados a central visando principalmente economia de bateria. Estes dados são enviados apenas quando o Token está conectado a base recarregando sua bateria.

5.2.2.3 Botão de socorro

Ao apertar este botão o Token se comunica com a base e esta entra em contato com a central que por sua vez passa a se comunicar diretamente com o usuário através do speaker e do microfone presentes no Token. Verificando a situação do usuário o operador na central toma alguma ação, caso não obtenha resposta do usuário uma chamada de emergência é efetuada.

5.3 Base

A base é responsável por criar uma ponte entre o Token e a central. Nela também possível recarregar a bateria do Token e fazer a coleta dos dados de movimentação do usuário via USB. A alimentação da base é feita diretamente pela rede elétrica, possuindo também uma bateria com autonomia de 12 horas.

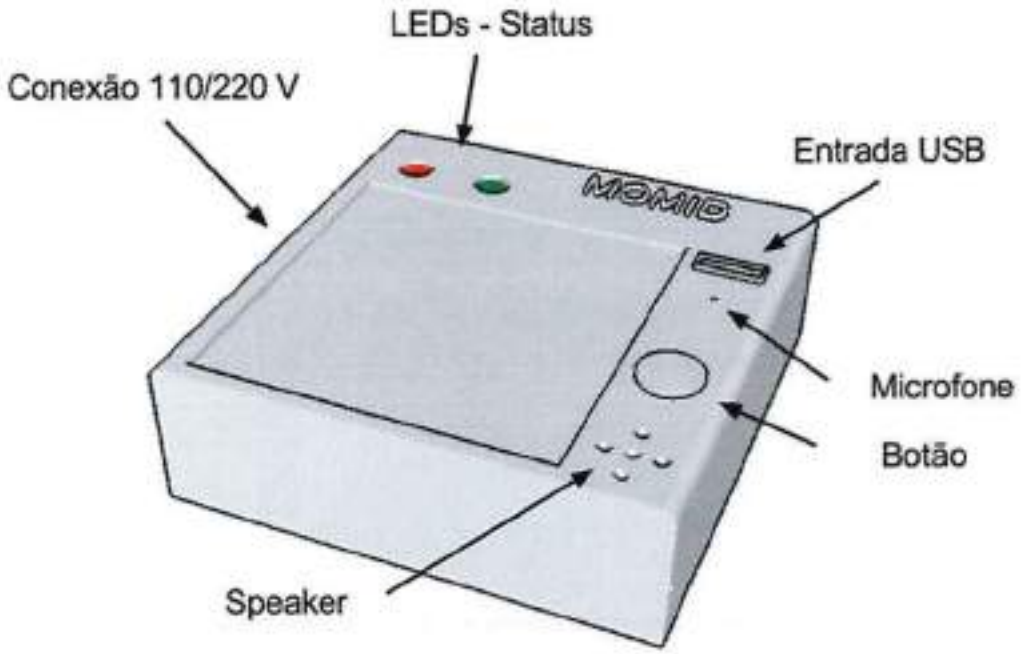

Figura 4 - Base

5.3.1 Funcionalidades

O Token possui a princípio as seguintes funcionalidades:

- Efetuar uma "ponte" entre a central e o Token
- Carregar a bateria do Token
- Coletar os dados do Token
- Botão de comunicação com a central

5.3.1.1 Ponte entre a central e o Token

São requisitos do Token, ter uma autonomia elevada e ser pequeno, logo este precisa consumir pouca energia e possuir o menor número de componentes possível. Justamente partes das tarefas que poderiam ser executadas pelo Token são feitas na base. Assim na base há, por exemplo, um módulo GSM que possui tamanho e consumo de energia razoáveis, este módulo se comunica com a central. Utilizando agora um módulo Bluetooth, por exemplo, a base se comunica com o Token. Internamente esta cria uma "ponte" entre o Token e a central, desta forma não foi necessário adicionar um módulo GSM ao Token.

5.3.1.2 Carregar a bateria e coletar dados do Token

O Token possui uma bateria que precisa ser carregada, além disso, possui uma memória com as informações de movimento do usuário que precisam ser enviada a central. Tudo isto é feito pela base através da entrada USB.

5.3.1.3 Botão de comunicação com a central

Através deste botão é possível se comunicar diretamente com a central, a base, assim como o Token, também possui um speaker e um microfone. Desta

forma caso ocorra algum problema com o Token o usuário pode entrar em contato com a central para pedir informações ou até mesmo pedir socorro.

5.4 Central MOMID

A central é composta basicamente por duas partes:

- Servidores de coleta e tratamento de dados
- Atendimento ao usuário

5.4.1 Servidores de coleta e tratamento de dados

Quando os dados são enviados do Token para a central eles são armazenados nos servidores de coleta, para então, em algum momento serem "interpretados" por algum algoritmo de análise de padrões procurando possíveis fatores de risco, por exemplo, de um dia para o outro um usuário passou a mancar.

Possuindo uma grande quantidade de dados de vários usuários diferentes será possível, por exemplo, analisar padrões que antes não imaginaríamos que poderiam existir. Por exemplo, o possível procurar alguma relação entre a modificação na forma de andar de um usuário com algum tipo de situação crítica futura, como um infarto.

5.4.2 Atendimento ao usuário

É nesta parte da central em que ficam os atendentes responsáveis pela comunicação com o usuário. Qualquer pedido de socorro feito pelo botão do Token ou um alarme de queda é redirecionado a esta área, que deve sempre possuir um atendente livre, que então procura conversar com o usuário e se necessário efetuar um pedido de socorro.

6 IMPLEMENTAÇÃO

6.1 Estratégia

Conforme comentado na proposta inicial, é de suma importância que o sistema diferencie um movimento de queda entre os demais movimentos comuns do usuário.

O desenvolvimento do algoritmo de reconhecimento de quedas baseia-se em uma localização pré-determinada no corpo do usuário, afim de que se consiga comparar padrões de movimento. Desta forma, localizar o sensor de movimento próximo ao centro de massa tem-se demonstrado popular nos estudos deste tipo [13, 14], uma vez que a força equivalente exercida sobre um corpo e, por conseguinte seu deslocamento, são equivalentes ao seu centro de massa [15].

Previamente, considerou-se a possibilidade do Token conter acelerômetro, como forma de detectar variações bruscas de movimento, além de giroscópio, que possibilita a percepção da inclinação do corpo do usuário.

Acelerômetros são dispositivos eletrônicos capazes de medir acelerações na qual está submetida segundo uma direção fixada, doravante chamada de eixo.

Os acelerômetros comerciais podem ter um, dois ou três eixos perpendiculares entre si, possibilitando a decomposição da aceleração total em componentes isolados. Por exemplo, deixado um acelerômetro de três eixos com um de seus eixos alinhado com a direção vertical obtém-se a aceleração de 1G (9,8m/s²) em módulo nesta direção e 0G (0m/s²) nos outros dois eixos. Deslocando-o horizontalmente da situação de repouso, detecta um acréscimo inicial na medida de aceleração conforme a decomposição do movimento entre os outros dois eixos.

Como meio de determinar a aceleração total submetida ao dispositivo, é feito um cálculo com o dado obtido por cada eixo. As direções perpendiculares serão denominadas por x, y, e, z e seus respectivos componentes de aceleração por a_x, a_y e a_z.

A magnitude da aceleração (MA) é definida por:

$$MA = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Para situações de equilíbrio estático, a informação dado por MA terá sempre o valor de 1G (9,8m/s²) para qualquer posição relativa dos eixos com a direção vertical e com o plano horizontal. Desta forma, um cálculo simples de MA indica a característica geral do movimento submetido ao acelerômetro.

A variação de MA ao longo do tempo indica o evento de queda a partir de um limiar determinado empiricamente por Giureski et al[14] e, assim, constituirá o método inicial de detecção de quedas desde sistema.

Como descrito por Giureski et al[14], este primeiro método de detecção de quedas tem sua eficiência limitada a quedas livre simples, quando o impacto do corpo do usuário é absolvido principalmente pelo solo.

A fim de que o projeto MOMID tenha resultados consistentes, com confiabilidade acima de 90% é necessário que a detecção se estenda para o maior número de quedas possíveis, respeitando sempre as especificações de custo e autonomia.

De maneira a simplificar o projeto sem perder funcionalidade, um acelerômetro de três eixos é capaz de substituir o giroscópio que inicialmente consta no conceito do projeto. A inclinação do dispositivo pode ser extraída pela componente de aceleração de um eixo qualquer em relação à resultante, cujo módulo é MA, da seguinte forma:

$$\theta_x = cos^{-1}\left(\frac{a_x}{MA}\right)$$

Onde θ_x é o ângulo do eixo x referente à direção da aceleração resultante. Portanto, caso seja respeitada a condição de fixação do acelerômetro no corpo do usuário, posicionando-se um dos eixos sempre na vertical por exemplo, é possível determinar a angulação do indivíduo continuamente. Logo, a utilização de acelerômetro no token foi descartada.

Os valores de MA e de θ que conceituam uma provável queda serão estudados posteriormente.

Pensando na utilização do token pelo usuário final, deve-se admitir que o dispositivo não será usado fixado corretamente em seu corpo sempre. Prevendo esse comportamento, em [14] faz-se uma média dos ângulos de cada eixo durante 15 segundos da colocação do dispositivo como forma de inicialização dos parâmetros para o acelerômetro. A diferença entre o valor do ângulo da média inicial (θ_x) e seu valor ideal (θ_{0x}) caracteriza um fator de normalização (θ_{nx}) para as medidas de inclinação do corpo do usuário. Assim,

$$\theta_{nx} = \theta_{0x} - \theta_x$$

$$\theta_{ny} = \theta_{0y} - \theta_y$$

$$\theta_{nz} = \theta_{0z} - \theta_z$$

devem ser somados continuamente em seus respectivos eixos a cada novo valor determinado de inclinação. Esses parâmetros serão configurados pelo projeto MOMID considerando o usuário em pé durante o tempo de 15 segundos.

6.2 Especificações de hardware necessárias

6.2.1 Acelerômetro

Segundo Bouten et al[16] acelerômetros com faixa de medida de até 6G são suficientes para medir movimentos localizados na região da cintura. Enquanto que Aminiam et al [17] afirma que não há potência de movimento significativa abaixo de 16 Hz, portanto, a frequência de amostragem do movimento humano deve ser superior a 32 Hz [18].

Desta forma, estabelecemos os primeiros critérios para a escolha do acelerômetro:

- Três eixos;
- Baixíssimo consumo de corrente (ordem de μA);
- Faixas de medida de aceleração acima de 6g;

- Frequência de amostragem do sinal acima de 32Hz.

6.2.2 Microcontrolador

Em um primeiro momento, idealizou-se a transmissão de dados do acelerômetro contido no Token de maneira contínua para a base, onde o processamento desses sinais seria realizado. No entanto, dado levantamento do consumo de energia feito por Karantonis et al[13] conclui-se que é mais vantajoso manter um microcontrolador de baixíssima potência junto ao usuário. No contexto daquele experimento, padrões pré-determinados de movimentos foram contidos na memória do microcontrolador, pelos quais dados oriundos do acelerômetro são comparados. Após a classificação dos movimentos do usuário, são transmitidos para um computador somente o suficiente capaz de nominar o movimento já classificado pelo microcontrolador. Nesta técnica baseou-se o projeto que prevê duração de três pilhas AAA (1100mAh cada) por até 17 dias.

Consoante à especificação do projeto MOMID de promover uma maior autonomia de funcionamento do sistema possível, utilizou-se a mesma técnica empregada em [13], ou seja, farão parte como componentes principais do Token uma bateria, um acelerômetro, um microcontrolador e um módulo de comunicação wireless.

Neste novo contexto, o consumo de corrente pelo microcontrolador torna-se o ponto chave do projeto a fim de promover uma maior autonomia do Token.

Outro aspecto relevante a ser considerado é a quantidade de memória do microcontrolador uma vez que esta precisa ser o suficiente para gravar os eventos de queda que serão comparados com os dados fornecidos pelo acelerômetro. Além disso, a quantidade de memória deve ser o suficiente para expansão da base de dados futuramente a fim de aperfeiçoar a detecção de queda.

6.3 Dispositivos Utilizados

6.3.1 Token

Devido a restrições de tempo, o grupo não pôde realizar a confecção de hardware próprio para o projeto, uma vez que a criação do algoritmo de detecção de quedas representou um desafio maior que o esperado, especialmente considerando que uma grande quantidade de testes foi realizada para determinar a eficácia do projeto. Sendo assim, foram utilizados dispositivos já estabelecidos (kits de desenvolvimento) para implementar o hardware do projeto.

Para o token, foi utilizado o kit de desenvolvimento EZ430-Chronos [19]. A escolha deste dispositivo para teste justifica-se por já possuir um acelerômetro [20] de acordo com 6.2.1 e ter um microcontrolador [21] compatível com a implementação descrita em 6.2.2. O microcontrolador integrado no EZ430-Chronos é da família CC430 da Texas Instruments, que possuem baixíssimo gasto de energia; ele possui 32KB de memória programável e 4KB de memória SRAM.

Uma caracteristica importante deste item é o fato de o EZ430-Chronos ser *open source* [22], o que significa que o software de desenvolvimento pode ser modificado e utilizado livremente.

O uso do EZ430-Chronos tornou a implementação da parte de hardware do "token" do projeto trivial; é simples fixar o relógio em várias partes do corpo (pulsos, cintura, pescoço) utilizando acessórios simples como cintos ou cordões. Além disso, o kit acompanha um software gratuito que permitiu reprogramar as funções do relógio para suprir as necessidades do projeto. Sendo assim, o "token" do protótipo do projeto se encontra pronto para uso, conseguindo se comunicar de modo sem fio com um computador qualquer que contenha seus drivers.

Figura 5 – EZ430-Chronos

6.3.2 Base

Para o projeto da base utilizamos o mesmo pensamento do *Token*, a escolha e especificação de todos os componentes e o projeto do circuito da base por si só já representam um grande desafio. Como estamos desenvolvendo não apenas uma placa eletrônica, mas sim um serviço, optamos por utilizar uma solução *open source* que já possua todos os recursos que necessitamos.

As principais características da base são a criação de uma ponte entre a central e o Token via comunicação sem fio simples (token-base) e via comunicação celular (token-central),e a coleta dos dados enviados pelo token. O EZ430-Chronos possui um *access point* USB para comunicação sem-fio em tempo real, este *access point* possui *drivers* para Windows, Mac e Linux [23]. Logo para facilitar o processo de desenvolvimento a escolha de um hardware já com comunicação USB seria ideal. Módulos que se enquadram nessas características são os chamados computadores de placa única, como o nome já informa são computadores que possuem em uma única placa todos os componentes de um computador pessoal comum. Geralmente estes módulos trabalham com processadores ARM e possuem como sistema operacional o Linux.

Um computador de placa única bastante conhecido atualmente e o Raspberry Pi [24] desenvolvido na universidade de Cambridge. O motivo deste *hardware* ser bastante conhecido está no fato ter seu custo bastante baixo, US$25,00, e ser

totalmente *open source*. Estudando as características eletrônicas deste computador percebemos que ele é ideal para o *hardware* da base, visto que roda Linux, possui duas portas USB e uma entrada para cartão SD.

Para efetuar a comunicação com a central via celular optamos pela escolha de um modulo GSM USB com *drivers* para Linux. Desta forma este modulo é conectado de forma bastante simples ao Raspberry Pi. Optamos pelo Huawei E220 devido ao fato de ser largamente comercializado no Brasil, visto a oferta de Internet 3G pela operadoras, e já possuir *drivers* para diversas distribuições Linux.

Utilizando os dispositivos descritos nesse item e no item 6.3.1, já temos o hardware completo do projeto, sobre o qual foi desenvolvido o algoritmo de detecção de quedas, ponto central do projeto. Um código em Python que roda na base e faz a comunicação entre base e central foi desenvolvido; o mesmo se encontra no anexo A.

6.3.3 Algoritmo de detecção de quedas

O maior desafio do projeto se encontrou no desenvolvimento do algoritmo de detecção de quedas em si, ponto central do projeto. Inicialmente, foi desenvolvido um algoritmo simples que simplesmente analisava o módulo do vetor aceleração e, caso o mesmo ultrapassasse um certo limiar, o sistema acusava queda. Porém, esse sistema era muito impreciso: um limiar muito alto falhava na detecção de algumas quedas, enquanto um limiar muito baixo acusava quedas em eventos cotidianos como um simples caminhar. Portanto, o desenvolvimento de um algoritmo mais refinado se tornou necessário.

O algoritmo desenvolvido a seguir levava em conta, também, o período de inatividade característico de uma queda grave, com perda de consciência ou perda de mobilidade. Caso haja um ou mais picos no módulo da aceleração (acima de 4G) do dispositivo *token* seguido de um período de baixa movimentação, então o sistema acusa um evento de queda. Com esse novo algoritmo, uma acurácia de 98% na detecção de quedas foi atingida nos testes realizados, descritos em detalhes no capítulo seguinte. Tal algoritmo foi desenvolvido no software MATLAB e o código

completo pode ser encontrado para consulta no anexo B (código principal e funções auxiliares). Mais detalhes sobre os resultados obtidos podem ser encontrados no capítulo seguinte.

6.3.4 Central

Uma parte muito importante de um sistema de detecção de quedas é o sistema responsável por receber os pedidos de ajuda e fazer o pedido de socorro. Em nosso projeto, quando uma queda é detectada, o token envia um sinal a base que então efetua um chamado a esta central, que nada mais é que um servidor web, esta é a central MOMID

A central MOMID poderia ser um projeto separado devido a seu tamanho e diversos requisitos. Por isso optamos por fazer uma versão simplificada da central contendo os principais requisitos:

- Receber pedidos de ajuda;
- Listar os últimos pedidos de ajuda;
- Fornecer informações sobre o usuário que está pedindo ajuda;
- Efetuar ligações telefônicas para números cadastrados.

O servidor foi desenvolvido em Python [25] utilizando o framework webapp2 [26] e a linguagem de template Jinja2 [27]. Optamos por estas bibliotecas porque hospedamos nosso servidor no Google App Engine [28] onde estes frameworks são oferecidos como padrão. A aplicação cliente, que roda em qualquer browser moderno, foi desenvolvida em HTML5 utilizando o framework jQuery Mobile [29]. Utilizamos um banco de dados não relacional oferecido pelo Google diretamente no App Engine, desta forma foi economizado um bom tempo de desenvolvimento e configurações.

Para efetuar as ligações telefônicas fizemos uso do serviço Twilio [30] que oferece através de uma API simples o acesso a diversos serviços telefônico, como, enviar sms, receber ligações e efetuar ligações. Todo o código do servidor desenvolvido em Python está no anexo C.

O servidor foi hospedado no Google App Engine, sendo possível acessá-lo através do endereço http://centralmomid.appspot.com. A listagem dos últimos pedidos de ajuda é acessada através da URL http://centralmomid.appspot.com/chamados, um exemplo do resultado desta página é apresentado na figura 6 abaixo, que mostra chamados testes realizados (Joao da Silva é um nome fictício criado para testes).

Ao clicar sobre o chamado são apresentados os dados do usuários e as opções de ajuda, como apresentado na figura 6.

Os pedidos de ajuda são enviados através da base que roda um código em Python presente no anexo C, este código é responsável por fazer o elo de comunicação entre o token e a central MOMID. Para efetuar um pedido de ajuda este código efetua uma requisição http POST na URL http://centralmomid.appspot.com/chamados, informando o ID do cliente previamente cadastrado no banco de dados do servidor. Este ID é gravado na base quando o usuário se cadastra no sistema.

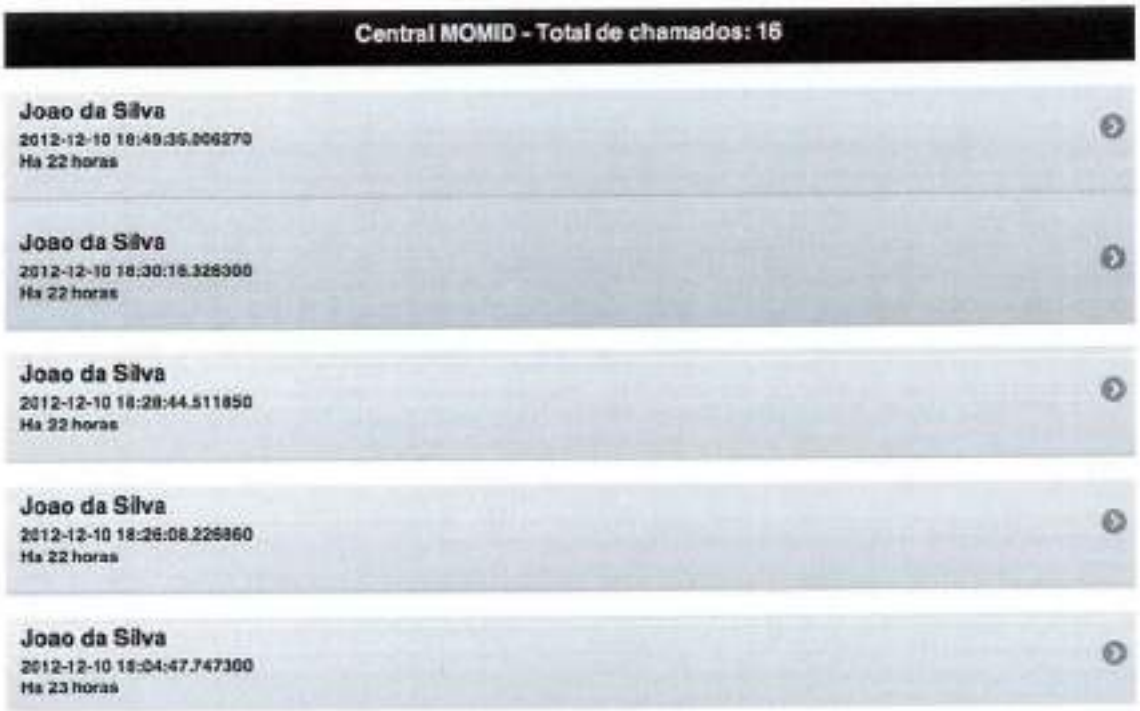

Figura 6 - Central MOMID

7 TESTES E RESULTADOS

7.1 Testes de Hardware

7.1.1 Posição do sensor

O primeiro desafio do trabalho foi determinar a posição ótima para fixação do sensor no corpo do usuário. Os locais que foram julgados potencialmente úteis para o projeto foram o pulso, pelo comodidade, o pescoço, pela facilidade de remoção e recolocação, e a cintura, pelo baixo nível de ruído detectado pelo sistema devido a atividade cotidianas.

Sendo assim, um indivíduo utilizando três relógios EZ-Chronos 430 (um em cada local do corpo supracitado) realizou diversas simulações de eventos como quedas, sentar em uma cadeira, deitar em uma cama, caminhar, correr, subir e descer escadas, etc., enquanto os dados dos acelerômetros eram monitorados em um computador. De dez a quinze testes de cada evento foram realizados, exceto nos eventos de queda onde foram realizados vinte testes. Fotos ilustrativas de cada um dos testes realizados, para ilustração do contexto de testes, se encontram no anexo C – Ilustração dos testes do acelerômetro.

Observando os resultados obtidos com tais testes, percebeu-se que o nível de ruído detectado pelos sensores fixados no pulso e no pescoço é muito alto, de modo que o algoritmo desenvolvido iria acusar falsas quedas muito frequentemente.

Porém, analisando os dados obtidos do sensor fixo na cintura, percebeu-se que o nível de ruído era bem inferior, de modo que a posição do sensor no tronco do usuário foi julgada a mais conveniente.

Abaixo se encontra um gráfico referente a um dos testes de quedas realizados, de modo a ilustrar os resultados obtidos acima. Os três eixos de cada sensor podem ser vistos. Convém notar que os três sensores não estão sincronizados no tempo. Resultados similares foram obtidos analisando os gráficos dos outros eventos; como os gráficos de um evento de queda ilustram muito mais

claramente o ruído presente nos acelerômetros fixos no pulso e no pescoço (os picos de sinal são muito mais discerníveis comparados com o ruído nesse caso), os gráficos referentes aos outros eventos foram omitidos.

Para a realização destes testes, foram utilizados três scripts criados na linguagem de programação Python, que permitem a aquisição de dados (uma versão para o computador que recebe os dados e uma versão para o próprio relógio para enviar os dados para o computador), e a gravação de um vídeo do evento de teste sendo realizado para futura referência e consulta. Estes códigos se encontram no anexo E.

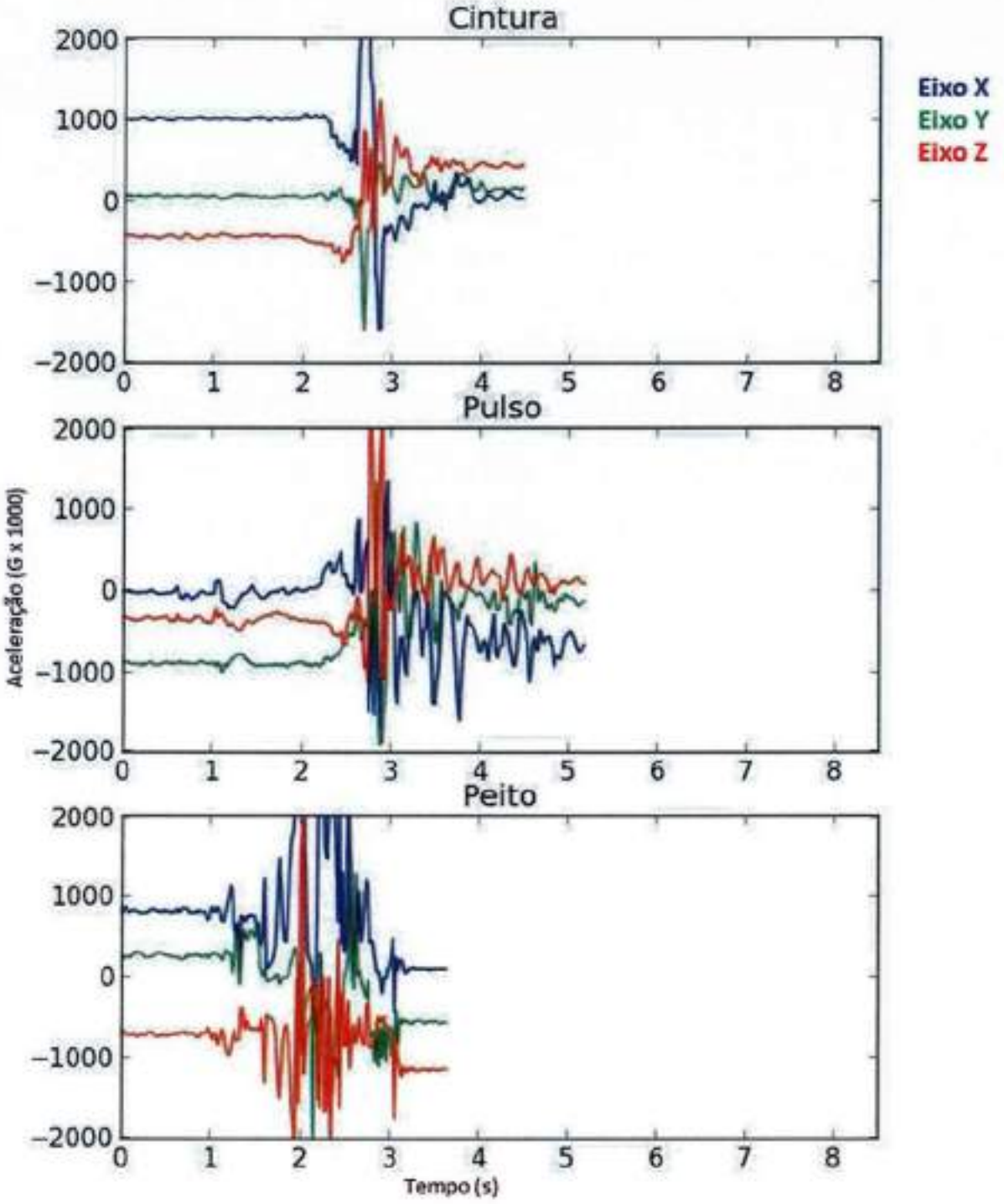

Figura 7 – Comparação dos dados do sensor de detecção em três locais de fixação para um evento de queda.

7.1.2 Testes de unidade

7.1.2.1 EZ430 – Chronos

O EZ430 – Chronos é o principal componente do sistema, uma vez que a segurança do usuário depende crucialmente do funcionamento correto do sensor. Sendo assim, é necessário ter controle das ferramentas de aquisição e processamento de dados. Então, foram testados a capacidade de aquisição de dados do acelerômetro e seu acionamento a partir do evento de toque de um botão.

Para isso, o dispositivo foi seguro em mãos de forma a ser colocado em diversas posição e balançado vigorosamente. Aproveitou-se os *drivers* de configuração dos botões e do acelerômetro embutidos no relógio para a análise.

Como os três eixos do acelerômetro estão disposto ortogonalmente, é possivel captar a aceleração espacial total do sensor. Assim, verifica-se que quando cada eixo do acelerômetro é colocado paralelamente à direção vertical, o mostrador do relógio acusa uma aceleração de 1G (aproximadamente 9.8m/s²) neste eixo e 0G (0m/s²) para os demais. Isso significa que a calibração do acelerômetro, bem como a conversão dos sinais brutos pelo microcontrolador, são feitos corretamente.

A transmissão dos dados do acelerômetro para um computador é realizada com êxito pelo relógio e o Access Point a uma distância de pelo menos um metro entre eles, fato observado pela consistência dos gráficos de aceleração apresentados no computador, alternando-se as escalas de aquisição de dados do acelerômetro entre 2G e 8G, no próprio relógio.

7.1.2.2 Raspberry Pi

Conforme consta nos requisitos do sistema, a base deve permanecer funcionando corretamente por um período de pelo menos 24 horas sem suprimento de energia. Portanto, um teste de consumo de energia foi realizado.

O Raspberry Pi foi alimentado por uma fonte regulada de tensão de 5V. Buscou-se exigir o máximo de processamento da unidade para corretamente dimensionar a bateria, rodando diversos processos em software. As medidas foram feitas por multímetros conectados a periféricos necessários ao projeto conectados ao dispositivo: modem GSM e Access Point.

Observou-se que o consumo de corrente médio é aproximadamente de 770mA na condição de alto processamento, portanto a potência consumida pela base nas configurações citadas é de 3,5W.

Logo, em 24 horas de uso intenso, o Raspberry Pi consome cerca de 18480mAh. Tomando como exemplo baterias recarregáveis de 2450mAh, comumente encontradas no mercado, oito baterias deste tipo são suficientes para alimentar a base por 24 horas.

7.1.3 Testes de integração

Para certificar que o sistema funciona corretamente, é necessário testar se as partes individuais funcionam quando integradas. Para minimizar a chance de erro nos testes, foi testada a integração dos componentes dois a dois e por fim, do sistema completo.

7.1.3.1 Integração EZ430 – Chronos e Access Point

A confiabilidade da comunicação entre o sensor e o Access Point é um ponto crucial para a eficiência do monitoramento. A especificação do projeto indica que o sistema deve ser capaz de funcionar em uma casa de 60 metros quadrados. Considera-se para fins de testes que a dimensão da casa seja de 6 metros por 10 metros e que pode haver um máximo de três barreiras (parede de alvenaria) entre token e base.

Para o teste, utilizou-se um EZ430 – Chronos, um Access Point e um computador para monitorar a comunicação. Foram empregados dois tipos de comunicação: direta, sem protocolo de comunicação, e uma outra utilizando o protocolo Simpliciti, previamente configurado no relógio.

Uma contagem sequencial simples foi feita no relógio e transmitida para o computador, de modo a verificar se há erro no sincronismo de transmissão de dados.

A comunicação direta, sem protocolo de comunicação, apresenta cerca de seis erros a cada minuto a uma distância de um metro entre relógio e Access Point, aumentando a medida que se incrementa a distância entre eles.

Já a comunicação feita com o protocolo Simpliciti não apresenta erros finais de transmissão da contagem feita pelo relógio a distâncias inferiores a trinta metros sem barreiras, confirmando a especificação do fabricante que prevê confiabilidade para até duzentos metros. Com três barreiras, a transmissão utilizando o protocolo Simpliciti apresenta erros de transmissão a partir de 20 metros de distância.

7.1.3.2 Integração Raspberry Pi e Access Point

Para garantir o funcionamento das tarefas de monitoramento e processamento de sinais da base, é necessário que o Raspberry Pi e o Access Point funcionem perfeitamente quando integrados. Apesar da base ser considerada um pequeno computador, apresenta limitações consideráveis de memória e poder de processamento. Em razão disso, foi utilizado uma distribuição Linux bastante enxuta, a Wheezy-Raspbian, originada de uma das mais estáveis distribuições Linuz, o Debian. Como o Access Point é conectado ao Raspberry Pi por uma porta USB, a comunicação serial é utilizada. Devido à eficiente biblioteca de comunicação serial do Raspberry Pi, a Pyserial, a linguagem de altíssimo nível Python 2.6 foi utilizada para gerenciar a integração dos componentes da base.

A uma distância de cerca de um metro entre o relógio e a base, o desempenho gráfico dos dados do acelerômetro mostrados no monitor através do software de monitoramento do relógio são consistentes com o esperado.

7.1.3.3 Integração Access Point, EZ430-Chronos, Raspberry Pi

Para testar a integração do sistema completo, foram realizados os mesmos testes do item 7.1.3.1, mas utilizando o Raspberry Pi como *stub* ao invés de um computador comum. Os resultados obtidos foram exatamente os mesmo, após testes de comunicação com e sem o protocolo Simpliciti, chegando a um alcance de 20 metros de distância com três barreiras de alvenaria.

7.2 Testes do algoritmo de detecção

7.2.1 Teste da acurácia do algoritmo

Para validação do algoritmo de detecção de quedas, o mesmo foi rodado nos dados obtidos no teste descrito na seção 7.1.1 (que foram previamente gravados e armazenados num computador) para determinar a porcentagem de acerto do modelo. Os resultados foram dividido em:

- Acertos, onde o modelo detectou queda onde realmente houve queda e não detectou nada onde não houve queda;
- Falsos positivos, onde o modelo detectou queda onde não houve queda;
- Falsos negativos, onde o modelo não detectou queda, mas houve evento de queda.

Utilizando o primeiro algoritmo descrito em 6.3.3 que levava em conta apenas o módulo da aceleração total, cerca de 10% dos eventos de queda não foram detectados. Porém, utilizando o segundo algoritmo, que também leva em conta o

período de inatividade característico de uma queda, não houve nenhum falso negativo. Foram observados alguns falsos positivos, de modo que a acurácia total do sistema com o segundo algoritmo ficou alta: 98% dos eventos foram detectados corretamente. Uma matriz de confusão referente á acurácia do segundo algoritmo, rodado nos testes realizados em laboratório, se encontra abaixo. A matriz de confusão cruza as informações dos eventos de testes: as linhas representam o resultado obtido e as colunas os resultados esperados. Sendo assim, os acertos do modelo estão localizados na diagonal principal da matriz (células verdes) enquanto os erros estão localizados nas células onde saída real e saída esperada do sistema não são consistentes (células vermelhas). Como os testes foram realizados em laboratório, a saída esperada era conhecida e a matriz pôde ser construída. Quarenta e nove eventos julgados importantes foram utilizados para este teste. Eventos banais, onde não havia variação perceptível nos dados do acelerômetro, como um simples andar, foram desconsiderados e foram utilizados apenas eventos críticos, como quedas, movimentos em escadas e sentar em cadeiras.

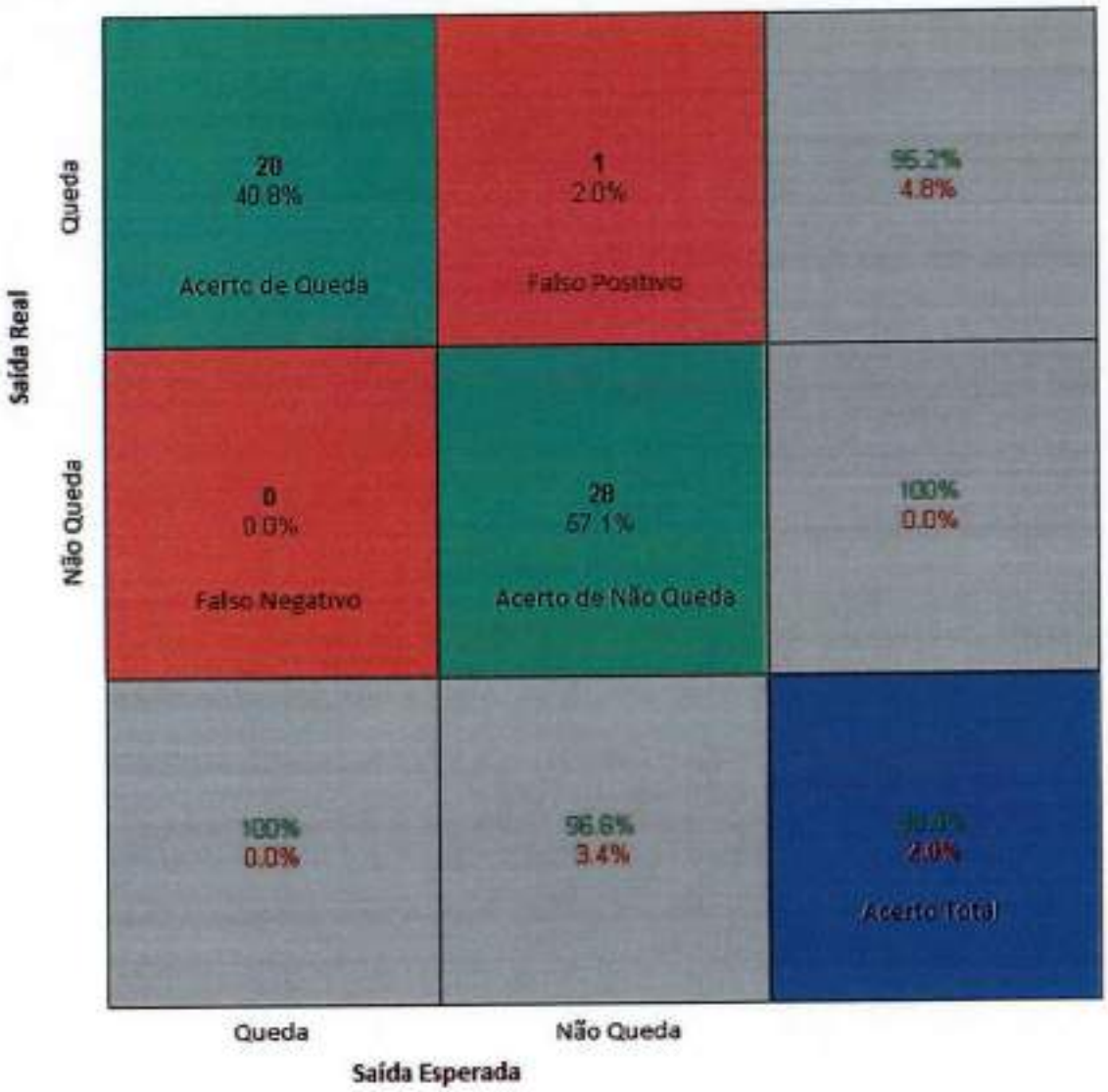

Figura 8 – Matriz de confusão referente aos testes feitos com o algoritmo de detecção.

7.2.2 Teste de campo

De modo a verificar quais eventos cotidianos causavam falsos positivos e verificar a quantidade de falsos negativos eram detectados pelo modelo implementado, o algoritmo utilizado foi carregado no microcontrolador do EZ-

Chronos 430 e o mesmo foi programado para apitar toda vez que uma queda fosse acusada.

Um indivíduo prendeu o relógio à cintura e realizou suas atividades cotidianas normalmente durante sete dias para a realização deste teste. Foram observados alguns eventos que causavam falsos positivos:

- Parar em uma escada durante a descida da mesma em velocidade elevada;
- Pular com força e parar em seguida;
- Sentar com força ("se jogar") em um sofá ou cadeira;
- Deitar com força ("se jogar") em uma cama;
- Correr a toda velocidade e parar em seguida.

Apesar dos falsos positivos, não foi detectado nenhum falso negativo nas simulações de queda realizadas pelo usuário, como se jogar em camas, em sofás e no chão.

8 DISCUSSÃO DOS RESULTADOS

8.1 Referentes aos testes de Hardware

Os resultados obtidos com os testes de hardware levaram à conclusão que o melhor local para fixar o token é, de fato, na cintura ou próximo dela no tronco do usuário. Esse fato foi constatado experimentalmente, mas deve-se ao fato do tronco ser a parte do corpo que menos se movimenta bruscamente em ações cotidianas. Sendo assim, o local de fixação do EZ430-Chronos será na cintura através de um cinto comum.

Os testes de integração levaram à conclusão de que o alcance máximo do sistema é de 20 metros, considerando que numa casa (local onde o sistema será implementado) a presença de três barreiras físicas entre token e base é algo comum. Mas, para casa de dimensões maiores, há a possibilidade de instalação de roteadores de sinal; o protocolo Simpliciti permite facilmente a instalação de tais roteadores sem perda de sinal, aumentando consideravelmente o alcance. Não foram realizados testes com tais roteadores, mas é uma sugestão interessante para a implementação final do projeto.

8.2 Referentes aos testes do Algoritmo

Analisando os resultados dos testes realizados com o algoritmo, vemos que o algoritmo atual detecta com precisão todos os eventos de queda, apesar de acusar alguns falsos positivos. Considerando que nenhum sistema é perfeito e que o sistema lida com acidentes possivelmente fatais, julgou-se satisfatório o sistema atual, que detecta quedas com eficiência, mesmo que alguns eventos cotidianos sejam detectados como quedas.

Além disso, é possível que o algoritmo seja melhorado de alguma forma para minimizar os falsos positivos. Uma característica de quedas que não foi abordada neste trabalho por falta de tempo foi a inversão de eixos do acelerômetro: em um

evento de queda, a angulação do acelerômetro muda, em geral, bruscamente; isso poderia ser utilizado em um trabalho futuro que dê continuação ao projeto.

9 CONCLUSÃO

O sistema de detecção de quedas obtido após o término do projeto MOMID atingiu uma porcentagem de acurácia de 98%. Esse valor é adequado à proposta do trabalho, uma vez que o sistema lida com acidentes envolvendo vidas, especialmente levando em conta que os 2% de erros do sistema correspondem a falsos positivos, ou seja, eventos que não são quedas mas que o sistema detecta como quedas.

Considerando que o mercado para um produto deste tipo no Brasil é elevado (cerca de 3 milhões de brasileiro acima de 60 anos moram sozinhos), o apelo do produto final para comercialização é elevado. Seria necessário desenvolver hardware próprio para tal fim, mas de posse do algoritmo de detecção desenvolvido e dos kits de desenvolvimento utilizados como base (EZ430-Chronos e Raspberry Pi), a produção de hardware próprio é simples.

A premissa secundária do projeto, que era a criação de um sistema de comunicação sem fio genérica, que utiliza sensores de algum tipo, de interesse de vários projetos do departamento de sistemas eletrônicos da Escola Politécnica da Universidade de São Paulo, também foi atendida. A metodologia deste trabalho pode ser adaptada para qualquer outro projeto que utilize um sistema de sensores similares.

10 REFERÊNCIAS

[1] http://www.ibge.gov.br/home/ Site oficial do IBGE – Instituto Brasileiro de Geografia e Estatística, acessado em março de 2012.

[2] http://www.ibge.gov.br/ibgeteen/pesquisas/demograficas.html Site IBGE teen, acessado em março de 2012.

[3] http://portal.saude.gov.br/portal/saude/visualizar_texto.cfm?idtxt=33674&janela=1 Site oficial do Ministério da Saúde, acessado em março de 2012.

[4] Yu, X. **Approaches and Principles of Fall Detection for Elderly and Patient**, E-health Networking Applcation and Services, Health Com 2008.

[5] Wang, C.; Chiang, C.; Lin, P.; Chou Y.; Kuo, I.; Huang, C.; Chan, C. **Development of a Fall Detecting System for the Elderly Residents**, IEEE 2008.

[6] Hwang, J.Y.; Kang, J.M.; Jang, Y.W.; Kim, H.C. **Development of Novel Algorithm and Real-time Monitoring Ambulatory System Using Bluetooth Module for Fall Detection in the Elderly**, 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, September 1-5, 2004.

[7] http://ieeexplore.ieee.org Site IEEE Xplore, acessado em abril de 2012.

[8] Sixsmith, A.; Johnson, N. **A Smart Sensor to Detect the Falls of the Elderly, Pervasive Computing**. p. 42-44.

[9] LeMoyne, R.; Mastroianni, T.; Grundfest, W. **Wireless accelerometer iPod application for quantifying gait characteristics**. 33rd Annual International Conference of the IEEE EMBS, Boston, Massachusetts USA, August 30 - September 3, 2011.

[10] Jiang, S.; Zhang, B.; Wei, D. **The Elderly Fall Risk Assessment and Prediction Based on Gait Analysis.** 2011 11th IEEE International Conference on Computer and Information Technology.

[11] Barelle, C.; Vellidou, E.; Koutsouris, D. **The BIOTELEKINESY home care service: a holistic concept to prevent fall among the elderly based on ICT and computer vision**, IEEE 2010.

[12] http://www.halomonitoring.com/ Site do Halo Monitoring, acessado em novembro de 2012.

[13] Karantonis, D. M.; Narayanan, M. R.; Mathie, M.; Lovell, N. H.; Celler, B. G. **Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring,** IEEE 2006.

[14] Gjoreski, H.; Luštrek, M.; Gams, M. **Accelerometer Placement for Posture Recognition and Fall Detection**, 7th Conference of Inteligent Environments, 25-28 July 2011.

[15] Newton, I. **Philosophiae Naturalis Principia Mathematica,** 1687.

[16] Bouten, C.V.; Koekkoek, K.T.; Verduin, M.; Kodde, R.; Janssen, J.D. **A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity**, IEEE Transactions on Biomedical Engineering , 44(3): 136— 147, 1997.

[17] Aminian, K.; Robert, P.; Jéquier, E.; Schutz. Y. **Incline, speed, and distance assessment during unconstrained walking. Medicine and Science in Sports and Exercise**, 27(2):226—234, 1995.

[18] Nyquist, H. **Certain topics in telegraph transmission theory**, abril de 1928.

[19] http://processors.wiki.ti.com/index.php/EZ430-Chronos?DCMP=Chronos&HQS=Other+OT+chronoswiki Consultado em agosto de 2012.

[20] http://www.muratamems.fi/en/products/accelerometers/cma3000-accelerometers Consultado em agosto de 2012.

[21] http://www.ti.com/product/cc430f6137#feature Consultado em agosto de 2012.

[22] http://sourceforge.net/projects/ez430chronosup/files/ Consultado em agosto de 2012.

[23] http://processors.wiki.ti.com/index.php/EZ430-Chronos Consultado em agosto de 2012.

[24] http://www.raspberrypi.org/ Consultado em agosto de 2012.

[25] http://www.python.org/ Site oficial da linguagem de programação Python, acessado em dezembro de 2012.

[26] http://webapp-improved.appspot.com/ Site do framework webapp2, acessado em dezembro de 2012.

[27] http://jinja.pocoo.org/docs/ Site da linguagem de template Jinja2, acessado em dezembro de 2012.

[28] https://appengine.google.com/ Site da Google App Engine, acessado em dezembro de 2012.

[29] http://jquerymobile.com/ Site do framework jQuery Mobile, acessado em dezembro de 2012.

[30] https://www.twilio.com Site do serviço telefônico Twilio, acessado em dezembro de 2012.

ANEXO A – CÓDIGO EM PYTHON DA BASE

A seguir se encontra o código na linguagem de programação Python programado dentro do Raspberry Pi, ou seja, a unidade de "base" do projeto MOMID. A função de tal código é monitorar o sinal enviado pelo *token* e, caso seja detectada um queda, enviar um sinal via modem 3G conectado na base à central.

```
from TIChronos import TIChronos
import time
import urllib
import urllib2
from threading import Thread

client_id = 2001

versao = "v0.9"

def enviar_chamado():
    global client_id
    print "enviando chamado"
    #url = 'http://localhost:8080/chamados'
    url = 'http://centralmomid.appspot.com/chamados'
    values = {'client_id' : client_id}
    data = urllib.urlencode(values)
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)
    resposta = response.read()
    print "Chamado",resposta
    return resposta=="OK"
```

```
def enviar_acknowledge():
    print "enviando acknowledge"
    global client_id
    url = 'http://centralmomid.appspot.com/acknowledge'
    values = {'client_id' : client_id}
    data = urllib.urlencode(values)
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)
    resposta = response.read()
    print "acknowledge",resposta
    return resposta=="OK"

def enviar_inicializacao():
    print "enviando inicializacao"
    global client_id
    url = 'http://centralmomid.appspot.com/inicializacao'
    values = {'client_id' : client_id}
    data = urllib.urlencode(values)
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)
    resposta = response.read()
    print "inicializacao",resposta
    return resposta=="OK"


porta = 10


print "Momid Base",versao

chronos = TIChronos(porta)
chronos.AP_start()
```

```
chronos.sync_start()



Thread(target=enviar_inicializacao).start()

i = 0
while True:
    resposta = chronos.sync_read_watch()
    print ".",i
    if resposta:
        print "Chamado"
        Thread(target=enviar_chamado).start()

        for i in range(10):
            chronos.sync_ajuda()
    i+=1
    if i > 20:
        i = 0
        Thread(target=enviar_acknowledge).start()



chronos.AP_stop()
chronos.COM_close()
```

ANEXO B – MODELO EM MATLAB

Este anexo contém os códigos em MATLAB referentes ao modelo de detecção de quedas desenvolvido. Cada função deve estar em um arquivo de script separado e na mesma pasta para que a função principal funcione.

Função principal

```
figura_i = 1;
esperado = zeros(1,1);
resultado = zeros(1,1);

i_saida = 1;

for i = 1:length(pastas)
  tipo = pastas(i).name;
  if (strcmp(tipo, '.') || strcmp(tipo, '..'))
     continue
  end

  pastas_tipo = dir(strcat(path,tipo));

  disp([tipo]);
  eh_queda = strcmp(tipo(length(tipo)-2:length(tipo)),'sim');
  for j = 1:length(pastas_tipo)
     teste_numero = pastas_tipo(j).name;
     if (strcmp(teste_numero, '.') || strcmp(teste_numero, '..'))
        continue
     end

     filename = sprintf('%s/%s/%s/68.csv',path,tipo,teste_numero);
     [dados, x,y,z] = pegar_dados_arquivo(filename);
```

```
      saida = false;

      for ii = 1:length(x)
        if modelo_limiar(x(ii),y(ii),z(ii))
          if modelo_parado(ii,x,y,z)
            saida = true;
            break;
          end

          saida =false;
          break;

          %disp([filename,x(ii),y(ii),z(ii)]);
        end
      end
      if saida==true && eh_queda==false
        gerar_grafico_modulo(dados,filename,figura_i,1,1);
        figura_i=figura_i+1;
      end
      esperado(1,i_saida) = eh_queda;
      resultado(1,i_saida) = saida;

      i_saida = i_saida+1;
    end
end

function [ queda ] = modelo_limiar( x,y,z )
    modulo = sqrt(double(x^2 + y^2 + z^2));
    queda = false;
    if modulo > 4000
```

```
        queda = true;
    end
end
```

Funções auxiliares

```
function [ queda ] = modelo_parado( inicio_queda, x,y,z )
    t = 0:1/50:(length(x)/50 - 1/50);
    modulo = sqrt(double(x.^2 + y.^2 + z.^2));
    soma = 0;
    n = 0;
    inicio_queda = inicio_queda+150; %3 segundo depois
    fim_queda = inicio_queda+100;%1 segundo depois
    fim_queda = min(fim_queda,length(x));
    for i = inicio_queda:fim_queda
        soma = soma + abs(modulo(i) - modulo(i-1));
        n = n + 1;
    end

    media = soma/n

    queda = false;
    if media < 60
        queda = true;
    end

end


function [dados, x, y, z ] = pegar_dados_arquivo( filename )
        fid = fopen(filename);
```

```
    c = textscan(fid,'%f%d%d%d','delimiter',',');
    fclose(fid);
    dados = [c{:}];

    x = dados(:,2);
    y = dados(:,3);
    z = dados(:,4);

end
```

ANEXO C – CÓDIGO EM PYTHON DA CENTRAL

Este anexo contém o código em Python criado para simular a central de operações do projeto MOMID. Esse código está hospedado no servidor de aplicativos do Google e permite receber os dados enviados da base de modo que um operador pode escolher entre diversas opções como ligar para o usuário, ligar para um familiar ou ligar direto para o resgate, sendo que a central possuirá os dados de cada um dos usuário (como endereço, telefone, etc.). Essas funções não foram ainda implementadas e a central do jeito que se encontra é apenas um protótipo ilustrativo das funções da central do projeto MOMID.

```
import webapp2
from google.appengine.ext import db
import jinja2
import os
import logging
import datetime
from twilio.util import TwilioCapability
import twilio.twiml
import re



jinja_environment =
jinja2.Environment(loader=jinja2.FileSystemLoader(os.path.dirname(__file__)))

# Add a Twilio phone number or number verified with Twilio as the caller ID
caller_id = "+18482604027"

# put your default Twilio Client name here, for when a phone number isn't given
default_client = "jenny"
```

```
def gerarTokenTwilio():
                    account_sid = "segredo"
                    auth_token = "segredo"

                    # This is a special Quickstart application sid - or configure your
own
                    # at twilio.com/user/account/apps
                    application_sid = "AP0a2b5ae316d36668a841090139b23ea1"

                    capability = TwilioCapability(account_sid, auth_token)

                    capability.allow_client_outgoing(application_sid)
                    capability.allow_client_incoming("jenny")
                    token = capability.generate()
                    return token

class Chamado(db.Model):
    date = db.DateTimeProperty(auto_now_add=True)

class Cliente(db.Model):
    nome = db.StringProperty()
    endereco = db.StringProperty(multiline=True)
    telefone = db.IntegerProperty()
    telefone_familiar = db.IntegerProperty()
    ultima_inicializacao = db.DateTimeProperty()
    ultimo_acknowledge = db.DateTimeProperty()


class MainHandler(webapp2.RequestHandler):
    def get(self):
        template_values = {
```

```
            'token':gerarTokenTwilio(),
        }

        template = jinja_environment.get_template('index.html')
        self.response.out.write(template.render(template_values))

class CriarStub(webapp2.RequestHandler):
    def get(self):
        cliente = Cliente()
        cliente.nome = "Joao da Silva"
        cliente.endereco = "Av. Paulista, 1230, Apt. 52A Sao Paulo, SP"
        cliente.telefone = 5511984455680
        cliente.telefone_familiar = 5511984455680
        cliente.put()
        self.response.write("OK")

class Clientes(webapp2.RequestHandler):
    def get(self):



        client_id = self.request.get('id')
        cliente = Cliente.get_by_id(int(client_id))
        template_values = {
            'cliente' : cliente,
            'token':gerarTokenTwilio(),
        }

        template = jinja_environment.get_template('cliente.html')
        self.response.out.write(template.render(template_values))

class Voice(webapp2.RequestHandler):
```

```
    def get(self):
        from_number = self.request.get('PhoneNumber')

        resp = twilio.twiml.Response()

        with resp.dial(callerId=caller_id) as r:
            # If we have a number, and it looks like a phone number:
            if from_number and re.search('^[\d\(\)\- ]+$', from_number):
                r.number(from_number)
            else:
                r.client(default_client)

        self.response.out.write(str(resp))

class Inicializacao(webapp2.RequestHandler):
    def post(self):
        try:
            client_id = self.request.get('client_id')
            cliente = Cliente.get_by_id(int(client_id))
            if not cliente:
                raise
            cliente.ultima_inicializacao = datetime.datetime.now()
            cliente.put()
            self.response.write("OK")
        except:
            self.response.write("ERROR")

class Acknowledge(webapp2.RequestHandler):
    def post(self):
        try:
            client_id = self.request.get('client_id')
            cliente = Cliente.get_by_id(int(client_id))
```

```
            if not cliente:
                raise
            cliente.ultimo_acknowledge = datetime.datetime.now()
            cliente.put()
            self.response.write("OK")
        except:
            self.response.write("ERROR")


class Chamados(webapp2.RequestHandler):
    def get(self, funcao):
        chamados_query = Chamado.all()
        chamados_query.order('-date')
        chamados = chamados_query.fetch(100)

        for chamado in chamados:
            chamado.cliente = chamado.parent()
            chamado.cliente.id = chamado.cliente.key().id()
            diferenca = (datetime.datetime.now()-chamado.date)

            chamado.diferenca = "Ha %d segundos"%(diferenca.seconds)
            if diferenca.seconds > 60:
                chamado.diferenca = "Ha %d minutos"%(diferenca.seconds/60)
                if diferenca.seconds > 60*60:
                                                chamado.diferenca = "Ha %d
horas"%(diferenca.seconds/(60*60))



        if funcao:
            if funcao=="total":
```

```
            self.response.write(len(chamados))
            return

        template_values = {
            'chamados': chamados,
            'total': len(chamados),
            'token':gerarTokenTwilio(),
        }

        template = jinja_environment.get_template('chamados.html')
        self.response.out.write(template.render(template_values))

    def post(self, funcao):
        try:
            client_id = self.request.get('client_id')
            cliente = Cliente.get_by_id(int(client_id))
            if not cliente:
                raise
            chamado = Chamado(parent = cliente)
            chamado.put()
            self.response.write("OK")
        except:
            self.response.write("ERROR")

app = webapp2.WSGIApplication([('/', MainHandler),
                               ('/chamados/*([^/]+)?', Chamados),
                               ('/criarstub', CriarStub),
                               ('/clientes', Clientes),
                               ('/inicializacao', Inicializacao),
                               ('/acknowledge', Acknowledge),
                               ('/voice', Voice)], debug=True)
```

ANEXO D – CÓDIGO DE AQUISIÇÃO DE DADOS

Este anexo contém os códigos auxiliares utilizados para realizar os testes referentes ao modelo de detecção e a posição ótima do sensor. Todos foram feitos na linguagem de programação Python. Um código se encontra no EZ430-Chronos utilizado como sensor, que envia os dados convenientes à um computador; um código se encontra no computador que recebe os dados através do Access Point do relógio, conectado por USB ao computador e um código adicional roda no computador para que a webcam do mesmo grave um vídeo do evento sendo realizado, em paralelo à aquisição de dados.

Código de aquisição de dados no computador

```
from chronos import Chronos
import datetime
import sys
import os
import subprocess
import time

pasta = "c:/TF/curvas/"+sys.argv[1]

if not os.path.exists(pasta):
    os.makedirs(pasta)

for i in range(1000):
    dir = pasta+"/"+str(i)
    if not os.path.exists(dir):
        os.makedirs(dir)
        os.makedirs(dir+"/images")
        os.makedirs(dir+"/video")
```

```
            pasta = dir
            break

total = 0
t0 = datetime.datetime.now()


def chronos_process(camera):
    try:
        while True:
            Chronos.verificar()

    except:
        print "fechando porta"
        Chronos.closeAll()
        camera.kill()
        raise


camera = subprocess.Popen("python camera.py %s" % pasta, shell = True)
time.sleep(2.0)

chronos1 = Chronos(66,pasta)
chronos2 = Chronos(67,pasta)
chronos3 = Chronos(68,pasta)

chronos_process(camera)
```

Código de envio de dados pelo EZ430-Chronos

```
import serial
import array
import datetime
import time

def startAccessPoint():
    return array.array('B', [0xFF, 0x07, 0x03]).tostring()

def accDataRequest():
    return array.array('B', [0xFF, 0x08, 0x07, 0x00, 0x00, 0x00, 0x00]).tostring()



dados_conv_acelerometro = [ 71, 143, 286, 571, 1142, 2286, 4571 ];


def twos_comp(val, bits):
    """compute the 2's compliment of int value val"""
    if( (val&(1<<(bits-1))) != 0 ):
        val = val - (1<<bits)
    return val

def converter(valor):
        resultado = twos_comp(valor, 8)
        sinal = 1
        if resultado< 0:
                sinal = -1
                resultado = resultado*-1
        #print resultado
        saida = 0.0;
```

```
        for i in range(7):
                saida += ((resultado & (2**i))>>i)* dados_conv_acelerometro[i]
        saida = saida*sinal
        return saida


class Chronos:
        chronos_list = list()
        ser = serial.Serial(10,115200,timeout=1)
        ser.write(startAccessPoint())

        ser.write(accDataRequest())
        ser.read(7)
        ser.write(accDataRequest())
        ser.read(7)
        ser.write(accDataRequest())
        ser.read(7)


        def __init__(self, id,pasta):
                self.id = id
                self.__class__.chronos_list.append(self)
                self.file = open(pasta+"/"+str(id)+".csv", 'w')
                self.lines = list()
                self.first = None
                self.sample_i = 0
                self.t0 = datetime.datetime.now()
                self.t1 = datetime.datetime.now()


        def close(self):
                self.file.writelines(self.lines)
```

```
            self.file.close()

    @classmethod
    def requisitarDados(cls):
            cls.ser.write(accDataRequest())
            accel = cls.ser.read(7)
            saida = dict()
            saida['id'] = ord(accel[6])
            saida['x'] = converter(ord(accel[0]))
            saida['y'] = converter(ord(accel[1]))
            saida['z'] = converter(ord(accel[2]))
            return saida

    @classmethod
    def verificar(cls):
            dados = cls.requisitarDados()
            for chronos in cls.chronos_list:
                    if chronos.id == dados['id']:
                            dados = cls.requisitarDados()
                            if not chronos.first:
                                    chronos.first = time.time()
                                    print chronos.id,"pronto"
                            chronos.lines.append("%f,%d,%d,%d\n"%
(chronos.first+(chronos.sample_i*(1.0/50.0)),dados['x'], dados['y'], dados['z']))
                            chronos.sample_i += 1
                            print chronos.sample_i,"-",

                            """
                            if chronos.sample_i % 50 == 0:
                                    chronos.t1 = datetime.datetime.now()
                                    delta_t = chronos.t1-chronos.t0
                                    print delta_t
```

```
                chronos.t0 = chronos.t1
            """

    @classmethod
    def closeAll(cls):
        for chronos in cls.chronos_list:
            chronos.close()
        cls.ser.close()
```

Código para a gravação de vídeo

```
import cv2.cv as cv
import time
import datetime
import math
import time
import sys

def capturar(pasta):
    capture = cv.CaptureFromCAM(1)
    cv.SetCaptureProperty(capture,cv.CV_CAP_PROP_FRAME_WIDTH ,640)
    cv.SetCaptureProperty(capture,cv.CV_CAP_PROP_FRAME_HEIGHT ,480)
    i = 0
    t0 = datetime.datetime.now()
    while True:
        img = cv.QueryFrame(capture)


        if i % 30 == 0:
```

```
            t1 = datetime.datetime.now()
            delta_t = t1-t0
            print delta_t
            t0 = t1
        """

        cv.SaveImage("%s/images/%f.jpg"%(pasta,time.time()),img)


        i+=1
        if cv.WaitKey(10) == 27:
            break

capturar(sys.argv[1])
```

ANEXO E – ILUSTRAÇÃO DOS TESTES DO ACELERÔMETRO

O objetivo deste anexo é melhor ilustrar o processo de testes realizados com o EZ430-Chronos para a determinação da melhor posição de fixação do *token* e para os testes do modelo de detecção de quedas. Para isso, são apresentadas aqui tiras de fotos sequenciais ilustrando os eventos de um indivíduo realizando os eventos de teste, gráficos ilustrativos de um evento de cada e uma breve descrição de cada teste. Todos os testes descritos aqui foram realizados com um *token* fixo em cada um dos seguintes três pontos: pescoço, amarrado por um barbante; cintura, preso no cinto; pulso, preso como relógio. Como o local escolhido para fixar o *token* foi na cintura, os gráficos apresentados aqui são referentes ao *token* fixo na cintura do usuário.

Evento: Caminhada

O usuário simplesmente andou em linha reta por um curto período de tempo. Poucos testes desse tipo foram realizados pois, analisando os gráficos obtidos, não há variação significativa na aceleração do *token* quando fixo na cintura.

Número de testes realizados: 5.

Figura 9 – Usuário realizando evento de caminhada

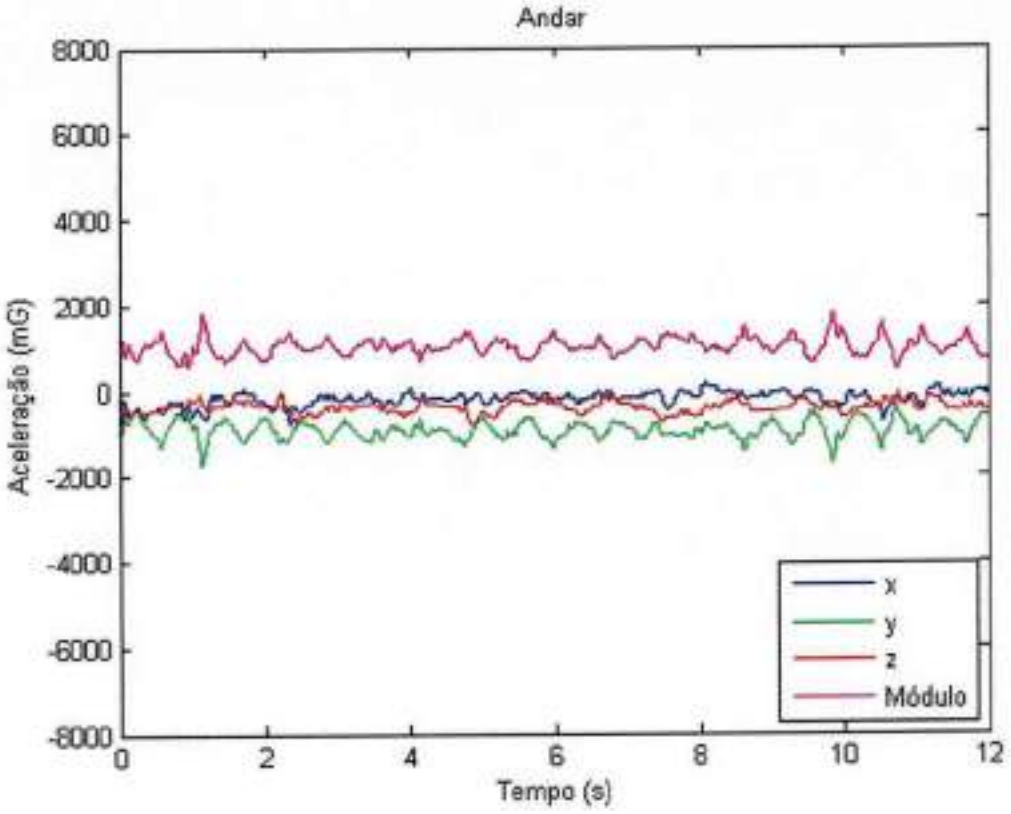

Figura 10 – Gráfico da aceleração referente ao evento de caminhada, com o sensor na cintura

Evento: Queda

O usuário caiu de costas em um colchão de ar durante estes testes e ficou imóvel por 5 segundos antes de repetir o teste, para simular perda de consciência.

Número de testes realizados: 30.

Figura 11 – Usuário realizando evento de queda

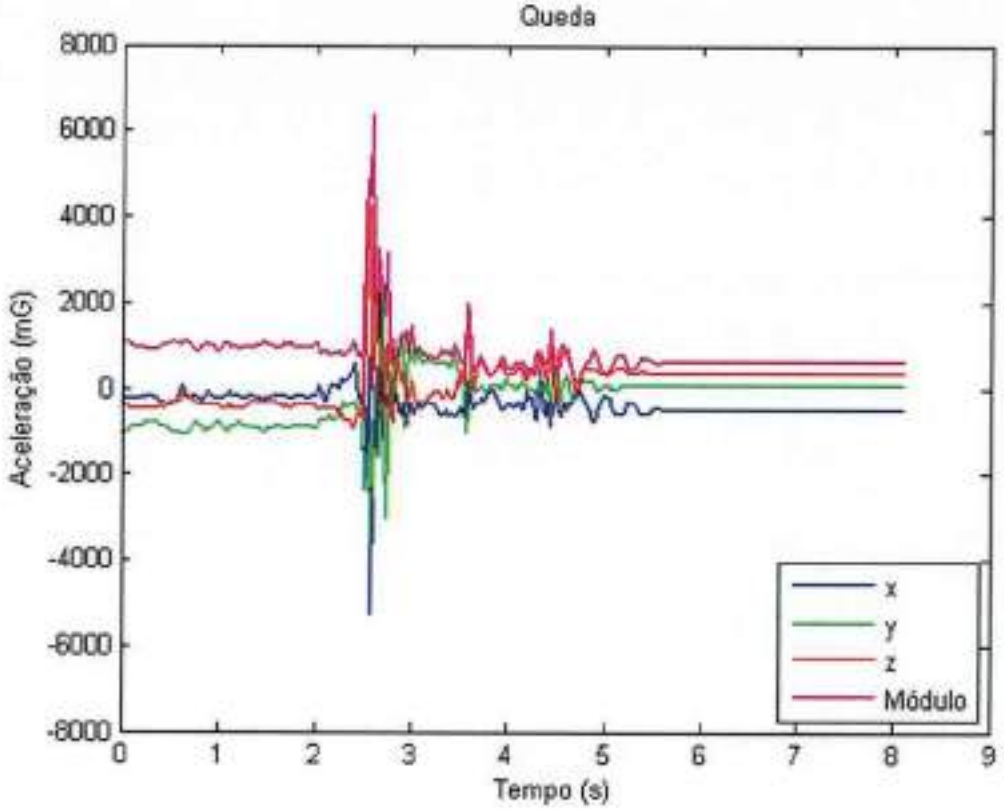

Figura 12 – Gráfico da aceleração referente ao evento de queda, com o sensor na cintura

Evento: Cair de uma cadeira

O usuário caiu de frente, partindo da posição sentada em uma cadeira e, novamente, ficou imóvel por 5 segundos para simular perda de consciência.

Número de testes realizados: 15.

Figura 13 – Usuário realizando evento de queda de cadeira

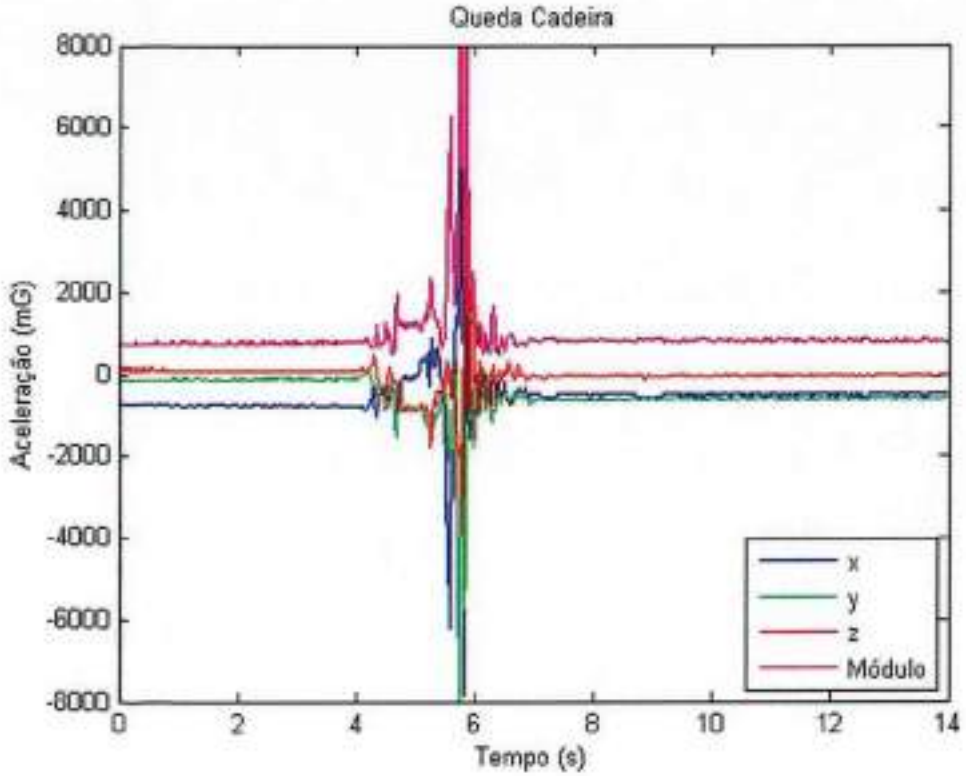

Figura 14 – Gráfico da aceleração referente ao evento de queda de cadeira, com o sensor na cintura

Evento: Deitar e Levantar de um colchão no chão

O usuário, partindo de uma posição em pé, deitou, esperou alguns segundos e em seguida levantou de um colchão de ar no chão. O objetivo era tentar simular os movimentos de deitar e levantar de uma cama.

Número de testes realizados: 10.

Figura 15 – Usuário realizando evento de deitar e levantar

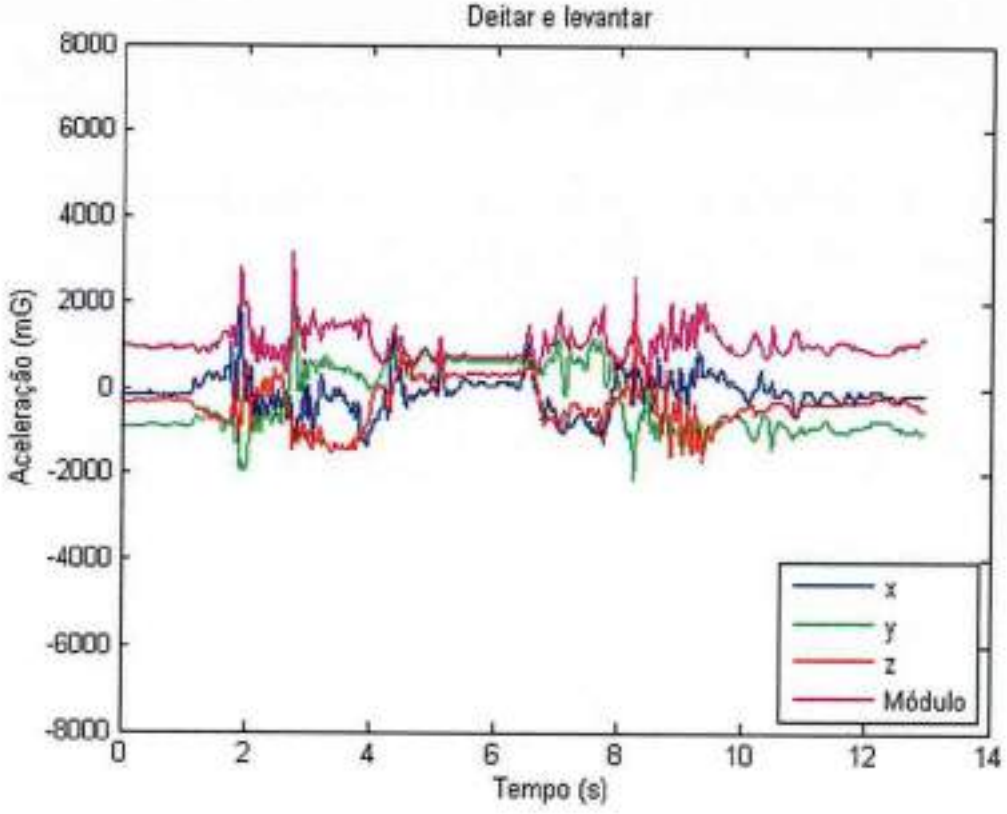

Figura 16 – Gráfico da aceleração referente ao evento de deitar e levantar, com o sensor na cintura

Evento: Subir e descer escadas

O usuário desceu e subiu escadas, parando por 5 segundos após a subida e após a descida. Em velocidades lentas (subidas e descidas normais) o sistema acusou apenas alguns falsos positivos, mas em velocidades altas (uma corrida) o sistema acusou vários falsos positivos, especialmente nas descidas. Porém, considerando que o foco do trabalho é em pessoas idosas que não são muito ativas, a chance de um usuário correr escada abaixo é baixa.

Número de testes realizados: 20.

Figura 17 – Usuário realizando evento de descer escada

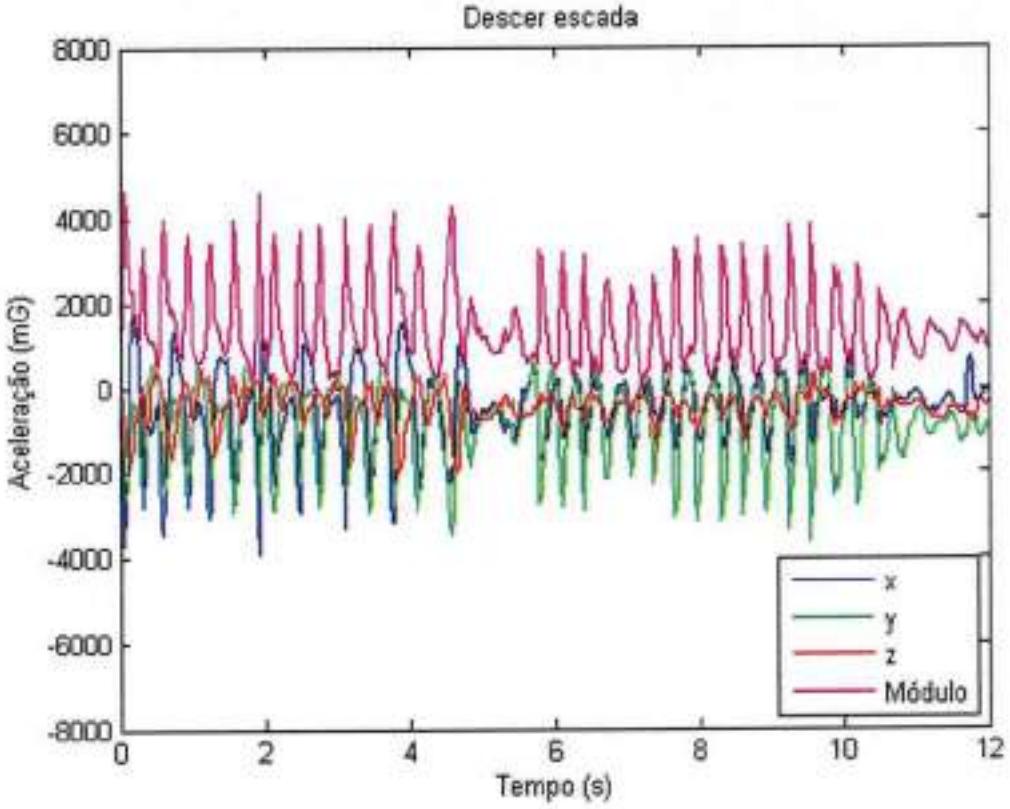

Figura 18 – Gráfico da aceleração referente ao evento de subir e descer escadas, com o sensor na cintura

Evento: Pulos

O usuário pulou algumas vezes por teste, variando a intensidade dos pulos. O pulo mais alto (e consequentemente com maior variação de aceleração) pode ser visto como o pico mais alto no gráfico de aceleração apresentado.

Número de testes realizados: 10.

Figura 19 – Usuário realizando evento de pular

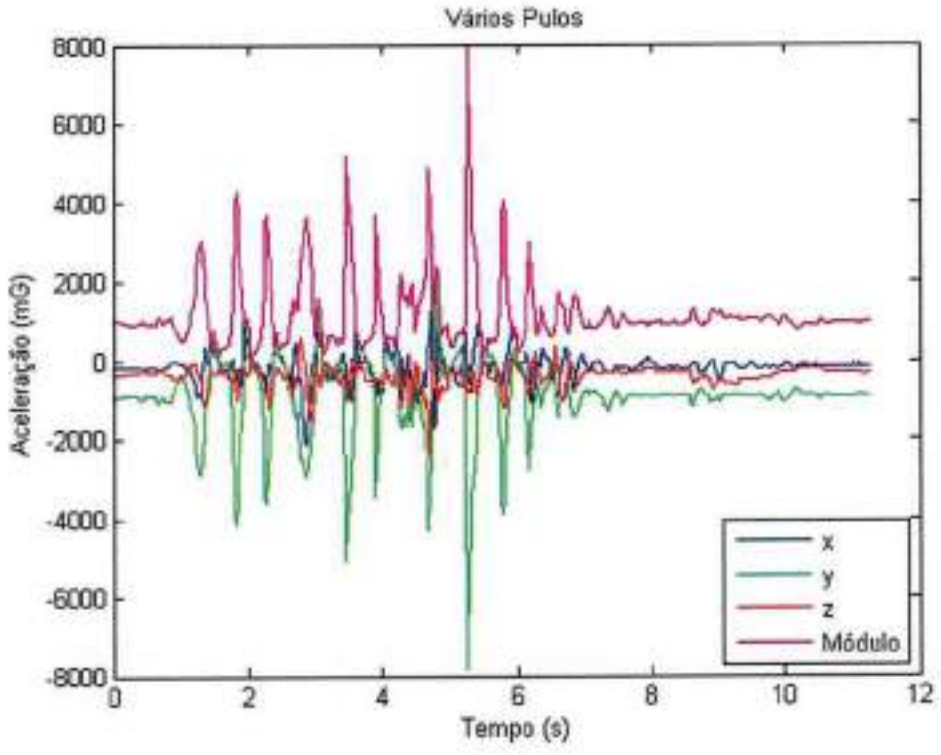

Figura 20 – Gráfico da aceleração referente ao evento de pular, com o sensor na cintura

Evento: Sentar e levantar de uma cadeira

O usuário sentou e levantou de uma cadeira estacionária uma vez por teste, ficando imóvel por alguns segundos após sentar e após levantar, para testar os limites do modelo.

Número de testes realizados: 20.

Figura 21 – Usuário realizando evento de sentar e levantar

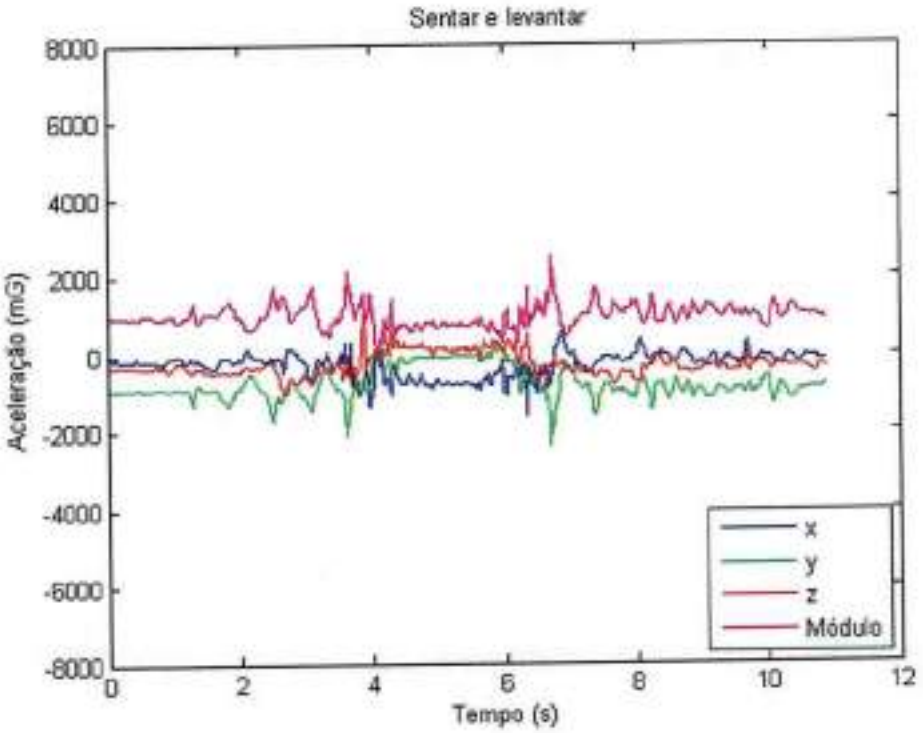

Figura 22 – Gráfico da aceleração referente ao evento de sentar e levantar, com o sensor na cintura