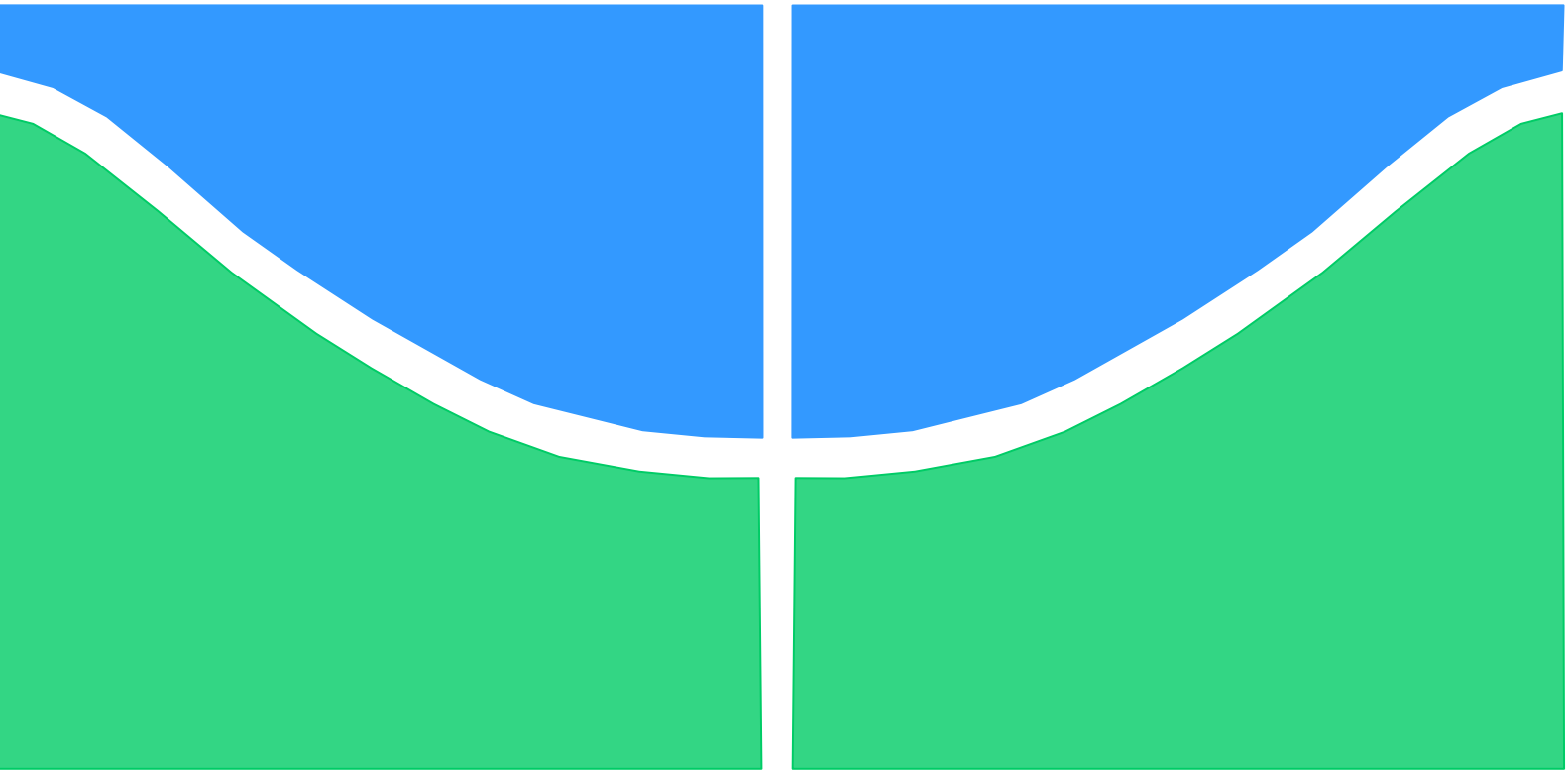

TRABALHO DE GRADUAÇÃO

Sistema de Detecção de Quedas e de Posicionamento Corporal com Monitoramento de Batimentos Cardíacos

Larinni Malheiros

Brasília, Dezembro de 2015

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

SISTEMA DE DETECÇÃO DE QUEDAS E DE POSICIONAMENTO CORPORAL COM MONITORAMENTO DE BATIMENTOS CARDÍACOS

Larinni Malheiros

Relatório submetido como requisito parcial para obtenção do grau de Engenheira de Redes de Comunicação.

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Dr., ENE/UnB
Orientador

Prof. Leandro Xavier Cardoso, Dr., FGA/UnB
Examinador Externo

Prof. Valério Aymoré Martins, Msc., ENE/UnB
Examinador Interno

Dedico este trabalho à minha família,
professores e colegas de curso.

Agradecimentos

A realização deste trabalho de graduação contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade e aos quais estarei eternamente grata.

Ao Professor, Georges Daniel Amvame Nze, pela sua orientação, apoio, disponibilidade, pelas opiniões, sugestões e críticas, com sua total colaboração no solucionar de dúvidas e problemas que foram surgindo ao longo da realização deste trabalho.

Aos docentes da Universidade de Brasília, que me ensinaram todo o conhecimento necessário para a realização deste trabalho, além de me auxiliarem ao longo de minha trajetória na graduação, com projetos de pesquisa e incentivo a atividades extracurriculares.

Aos meus colegas de curso, aos quais sempre me ajudaram nos momentos de dificuldade em projetos e trabalhos.

A Arina Bragança, pela paciência e auxílios prestados.

Por último, dirijo um agradecimento especial aos meus pais, pelo seu apoio incondicional, incentivo, amizade e paciência demonstrados e pela total ajuda na superação de obstáculos que surgiram ao longo desta caminhada para alcançar o ensino superior.

RESUMO

Em 2050 terá mais idosos no mundo do que crianças menores de 16 anos pela primeira vez na história[50]. Um dos maiores problemas enfrentados pela população acima de 60 anos são as quedas. Essa é uma preocupação de relevância na saúde pública, pois trazem problemas físicos e psicológicos. Alguns traumas podem direcionar à perda de autonomia, imobilidade, restrição de atividades, medo de ocorrer outras quedas , e também o risco de morte. Assim, no que se refere ao longo potencial de existência de idosos temos que preocupar em desenvolver técnicas para diminuir a incidência de quedas e melhorar sua qualidade de vida.

Este trabalho foi realizado com o objetivo de criar um equipamento capaz de auxiliar o acompanhamento à distância de pessoas com diagnóstico de Alzheimer, com crises convulsivas generalizadas, com quadros de desmaios frequentes, idoso e pessoas que precisam de monitoramento 24 horas. Esse dispositivo detecta quedas, posição corporal, valor dos batimentos cardíacos e a temperatura ambiente.

Para detectar quedas, será proposto um algoritmo baseado em threshold e em tempo de pausa, para detectar posição corporal será utilizado a orientação dos acelerômetros e giroscópios, para o valor dos batimentos cardíacos será utilizado um sensor de Pulse Senor e a coleta de temperatura será obtida através de um sensor de temperatura

Este trabalho irá apresentar todas as etapas necessárias para a implementação do dispositivo, como escolha dos sensores, diagrama de casos de caso, diagrama UML, algoritmo, linguagens de programação utilizadas e sua efetividade. Os resultados obtidos foram analisados em forma de porcentagem sendo cada medição realizada 10 vezes. A posição corporal foi corretamente detectada em 100% dos casos, já o movimento do corpo andando teve correta precisão em 90% dos casos, quedas foram detectadas em 60% dos casos.

Como trabalhos futuros, deseja-se aumentar a estatística de deteção de quedas, utilizando outro algoritmo para processar as informações obtidas dos sensores.

ABSTRACT

By 2050 the world will have more elderly people than children under 16 years old in the first time in history [50]. One of the biggest problems faced by the population over age 60 are falls. This is a concern of importance in public health, as they bring physical and psychological problems. Some trauma can direct the loss of autonomy, immobility, activities restriction, fear of suffering falls, and also the risk of death. It is therefore necessary to develop techniques to decrease the incidence of falls among the elderly.

This work was carried out in order to create a device capable of assisting people diagnosed with Alzheimer's, generalized seizures, frequent fainting frames, elderly and people who need 24-hours monitoring. This device detects falls, body position, heart rate and value of the ambient temperature.

To detect falls, an algorithm is proposed based on threshold and pause time, to detect body position will be used guidance from accelerometers and gyroscopes, to get the heartbeat value will be used a sensor pulse sensor and ambient temperature will be obtained through a temperature sensor

This paper will present all the necessary steps to implement the device, such as choice of sensors, User Case diagram, UML diagram, algorithm, programming languages used and its effectiveness. The results obtained were evaluated as percentage each measurement being carried out 10 times. The body position has been correctly detected in 100% of cases, walking body motion was detected in 90% of cases, and falls were detected in 60% of cases.

As future work, it’s necessary increase the statistical detection of falls by using another algorithm to process the information obtained from the sensors.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIAÇÕES

Amax	Aceleração máxima do corpo em uma janela de análise
Amin	Aceleração mínima do corpo em uma janela de análise
API	*Application Programming Interface*
AT	Transparent Mode
AUFT	Aceleração Upper Fall Threshold
AVC	Acidente Vascular Cerebral
B4A	*Basic4Android*
BPM	Batimentos Por Minuto
CSS	Cascading Style Sheets
DML	*Data Manipulation Language*
DMP	*Digital Motion Processor*
GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Services*
GPS	*Global Positioning System*
HDMI	*High –Definition Multimedia Interface*
HR	*Heart Rate*
HRV	*Heart Rate Variability*
HTML	HyperText Markup Language
I^2C	*Inter-Integrated Circuit*
IBGE	Instituto Brasileiro de Geografia e Estatística
IoT	*Internet of Things*
IDE	*Integrated Development Environment*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
JSON	*JavaScript Object Notation*
LED	*Light Emitter Diode*
MEMS	*Microelectromechanical Systems*
NOOBS	*New Out Of the Box Software*
P2P	*Peer-to-Peer*
PAN	*Personal Area Network*
PHP	*Hypertext Preprocessor*
RAM	Random Access Memory
RDBMS	*Relational Database Management System*
RFID	*Radio Frequency Identification*
RpTV	Repetição de Televisão
SDA	*Serial Data*

SCL	*Serial Clock*
SD	*Secure Digital*
SMS	*Short Message Service*
SQL	*Structured Query Language*
USB	*Universal Serial Bus*
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
Wmax	Velocidade Angular máxima do corpo em uma janela de análise
Wmin	Velocidade Angular mínima do corpo em uma janela de análise
WSN	*Wireless Sensor Network*
WUFT	Velocidade Angular Upper Fall Threshold)

1. Introdução

1.1. Motivação

O envelhecimento populacional apresenta-se como um fenômeno atual de grande relevância em todo o mundo, já sendo característico de países desenvolvidos e agora, se tornando de modo crescente, em países subdesenvolvidos [51]. Segundo a pesquisa realizada pelo *The Centre for Modern Family* [1], quase metade das pessoas acima de 60 anos vivem sozinhas, sendo que 48% destas dizem não ter família morando localmente e outros 25% dizem não terem mais família. No Brasil, de acordo com os dados do Censo [52], realizado pelo IBGE(Instituto Brasileiro de Geografia e Estatística) em 2011, são quase 3 milhões de pessoas acima de 60 anos que moram sozinhos. Com esses idosos morando sozinhos se faz necessário cuidados especiais, pois se algum acidente ocorrer, o alerta a centros de saúde ou familiares pode não acontecer no tempo hábil para evitar consequências indesejáveis.

Quando idosos moram com familiares também é necessário constante vigilância, pois, algumas vezes o familiar ou o cuidador não estará presente e nesses momentos é que podem ocorrer acidentes domésticos ou outros incidentes, como parada cardíaca. Para idosos com Alzheimer além da detecção de quedas é importante saber o estado da pessoa no momento, pois ela poderá sair de casa sem que os familiares a perceba. Novas tecnologias, como detectores de queda e monitores de posição são necessárias para dar suporte à independência do idoso e a sua segurança.

Não são apenas os idosos que precisam de monitoramento, pessoas com doenças cardíacas e pessoas com quadros de desmaios frequentes também necessitam de acompanhamento, pois o aumento dos batimentos e desmaios pode ocorrer a qualquer momento, com ou sem a presença de familiares e, portanto, seria adequado o uso de um equipamento que notificasse as pessoas interessadas sobre a situação do ente.

1.2. Objetivos

O objetivo geral desse trabalho de graduação é criar um dispositivo capaz de detectar quedas e posicionamento corporal, monitorar batimentos cardíacos e temperatura ambiente. Os objetivos específicos consistem em criar um sistema de alerta para informar os familiares se ocorreu uma queda ou elevação nos batimentos cardíacos. Esse projeto foi desenvolvido com o intuito de melhorar a qualidade de vida das pessoas que necessitam de monitoramento e de seus familiares.

1.3. Desenvolvimentos Recentes

Várias pesquisas recentes na área de desenvolvimento de algoritmos para detecção de quedas foram utilizadas como base para esse projeto. O artigo *Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information*[2] utiliza um sistema de detecção de quedas baseado em *thresholds*, com detecção de posição baseada no ângulo entre a coxa e o peito, e a partir disso, determina se o movimento foi intencional para inferir se a queda ocorreu . A forma mais comum e a metodologia mais simples de detecção de quedas é utilizar acelerômetros de três eixos com algortimos baseados em *thresholds* [2][12], mas eles são distintos em valores e método de processamento da informação. Lim, D., et al. [3] faz a combinação desse método com cadeia de Markov.

O artigo *Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement*[28] apresenta uma técnica baseada em tempo remanescente. Ele calcula o valor resultante da aceleração nos três eixos, depois a compara com *thresholds* preestabelecidos, se os valores estiverem acima de um limiar, uma possibilidade de queda é detectada e um contador é inicializado. Esse contador mede quantas vezes os valores obtidos foram maiores do que os *thresholds* dentro da janela de tempo de seis segundos. Se este contador tiver como resultado um valor entre um e quatorze, a queda é detectada. Caso contrário, considera-se que outra atividade estava sendo realizada.

Uma abordagem recentemente publicada [43] utiliza a rede de Petri para detectar quedas, contudo o idoso deve constantemente carregar em seu bolso um *smartphone*, o que na maioria das vezes, é inviável. Outro artigo que utiliza essa abordagem de detecção de quedas com *smartphones* está presente em [44].

Lustrek, M., Kaluza, B. [29] utiliza redes neurais para classificar o comportamento do usuário. Ele treina a rede para reconhecer qual atividade está sendo realizada. Foram utilizados oito algoritmos de aprendizado de máquinas, no qual o *Support Vector Machine* se mostrou o classificador mais eficiente com acurácia de 97,7%.

Dinh, C., Struck, M. [27] trabalha com o conceito de *“expert knowledge”*, no qual mantém a detecção independente das características físicas do paciente, como peso e altura. A lógica fuzzy é utilizada para aplicar esse conceito e o resultado é classificado por uma rede neural. Mundher, Zaid., & Zhong, J. [26] utilizam sensores *Kinect* embutidos em robôs móveis que fazem o reconhecimento de gestos e voz.

Torres, R., et al. [46] utiliza tags RFID (*Radio-Frequency Identification*) instaladas no tapete para obter informações dos pacientes e a encaminha para algoritmos de aprendizado de máquina, na qual faz decisão sobre a informação do paciente. Com base nos artigos mostrados acima verifica-se que há várias abordagens diferentes e todas elas tem como objetivo detectar a queda em 100% das ocorrências e não realizar falsos positivos.

Dois termos técnicos são utilizados em todos os artigos apresentados acima para definir se uma queda foi corretamente detectada. O primeiro termo é chamado de “falso positivo” e o segundo é chamado de “falso negativo”. Falso positivo é um conceito que indica quando a queda foi erroneamente detectada, pois o paciente estava realizando outra atividade, como andar, correr ou levantar rapidamente de uma cadeira. Falso negativo indica quando uma queda não foi dectada.

Este trabalho irá utilizar uma metodologia baseada em *thresholds*, utilizando acelerômetros de três eixos. Também será utilizada a análise de tempo remanescente, como em [28], contudo não será utilizado contadores, mas sim, um estudo da média dos valores obtidos no tempo remanescente após a queda. Além disso, esse trabalho fará complemento aos artigos acima citados, realizando o alerta-via *email, sms* e ligação telefônica-e mostrando as informações em tempo real utilizando um aplicativo android e uma página web.

1.4. Estrutura do Texto

No Capítulo 2 é realizada uma introdução aos conceitos que serão utilizados para o desenvolvimento do projeto. Em seguida, o Capítulo 3 descreve a metodologia de como o protótipo foi realizado, seus recursos de hardware e software. Resultados experimentais são discutidos no Capítulo 4, seguido das conclusões no Capítulo 5.

2. Fundamentação Teórica

2.1. ZigBee (802.15.4)

XBees são soluções embarcadas que fornecem conectividade wireless entre dispositivos, esses rádios utilizam o padrão IEEE 802.15.4 para comunicação ponto a ponto e ponto-multiponto. Módulos XBee são produtos produzidos pela Digi International e fornecem soluções práticas para prototipagem de projetos que consistem em enviar dados para a nuvem, ou seja, são excelentes soluções para projetos de IoT. Os rádios XBees podem conter várias firmwares para suportar ZigBee/ZigBee Pro e DigiMesh.

ZigBee é uma aliança e padrão, criada pelo IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) em conjunto com a ZigBee Alliance com o intuito de disponibilizar uma rede de baixo custo e baixo consumo de energia, estendendo a vida útil das baterias que compõe os dispositivos, podendo durar anos. Desta forma, esse padrão é utilizado em aplicações que não necessitam de taxas de transmissão de dados tão altas quanto à fornecida por dispositivos Bluetooth e Wi-Fi (*Wireless Fidelity)*. Assim, seu uso é ideal para WSN (*Wireless Sensor Networks*). A Tabela 2-1 apresenta as diferenças entre as tecnologias de rede sem fio presentes.

O padrão IEEE 802.15.4 é responsável pela criação e operação das duas camadas mais baixas da tecnologia ZigBee, enquanto que a ZigBee Alliance trabalha nas camadas superiores. A Figura 1 apresenta a pilha de protocolos ZigBee.

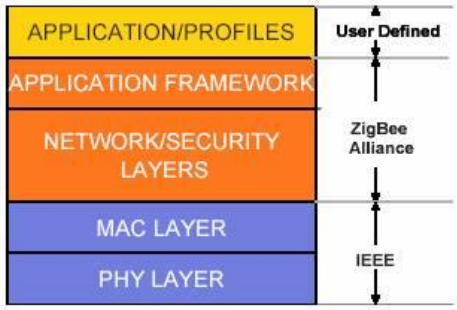

Figura 1 - Pilha de protocolos, retirado de [31]

Tabela 2-1 Tecnologias de Redes sem Fio

	ZigBee	Bluetooth	Wi-Fi
Padrão	IEEE 802.15.4	IEEE 802.15.1	IEEE 802.11
Aplicação	Monitoramento e Controle	Substituição de Cabos	Web, Video, Email
Recursos do Sistema	4 KB-32KB	250KB+	1MB+
Organizações Industriais	Zigbee Alliance	Bluetooth Sig	Wi-Fi Alliance
Topologia	Mesh, estrela, árvore	Estrela	Estrela
Rádio Frequência	868/900 MHz, 2.4GHz	2.4 GHz	2.4 GHz, 5.8 GHz
Taxa de Transmissão	250 kbits/s	723 kbits/s	11-105 Mbits/s
Alcance	10-300 m	10 m	10-100m
Consumo de energia	Muito baixo	Baixo	Alto
Tempo de vida útil da bateria (dias)	100-1000+	1-7	1-5
Número de nós	65000	8	32
Principais Atributos	Confiável, Baixa potência, Baixo custo	Conveniência	Velocidade, Flexibilidade

Há vários tipos de rádios XBees fornecidos pela Digi International, eles variam em alcance, consumo de energia, potência de transmissão, taxa de transmissão, frequência de operação e tipo de antena. Além disso, eles são categorizados por serem Regulares ou Pro. Os XBees Pros são um pouco maiores, utilizam maior potência e tem maior custo, por eles terem maior potência, consequentemente seu alcance é maior, 1,6 Km, em vez de aproximadamente 100 metros da versão regular. A maioria dos XBees operam em 2.4 GHz, contudo alguns operam em 900 MHz, como sabemos, 900 MHz pode ter um alcance muito maior para um mesmo ganho se comparado com 2.4 GHz, contudo 900 MHz não é permitido em muitos países. Também há módulos que operam na frequência de 868 MHz, contudo eles também sofrem restrições de uso em diferentes países. No Brasil, a Anatel regulamentou que a banda de 900 MHz é reservada para operações de GSM(*Global System for Mobile Communications)* e a banda 868 MHz para RpTV (Repetição de Televisão).

A Figura 2 apresenta o modelo XBee Series 2, com potência de transmissão de 2mW, alcance de 120 metros, consumo de energia de 40mA a uma tensão de 3.3V. Esse rádio não pode operar com tensões de entrada acima de +3.3V, pois pode-se comprometer o funcionamento do dispositivo.

Figura 2 - XBee Series2 [30].

O padrão IEEE 802.15.4 oferece suporte para as topologias de rede árvore, estrela, *mesh* e árvore de clusters (*cluster tree*), contudo como pode ser conferido na Tabela 2-1, a tecnologia ZigBee suporta apenas árvore, estrela e *mesh*. A ideia básica atrás da construção da topologia em redes de sensores é que ela utiliza uma estrutura de hierarquia; um dispositivo que está entrando na rede pode ser tanto roteador (router) ou um dispositivo final (*end-device*) e roteadores podem aceitar vários end-device. A Figura 3 mostra as diferentes topologias possíveis

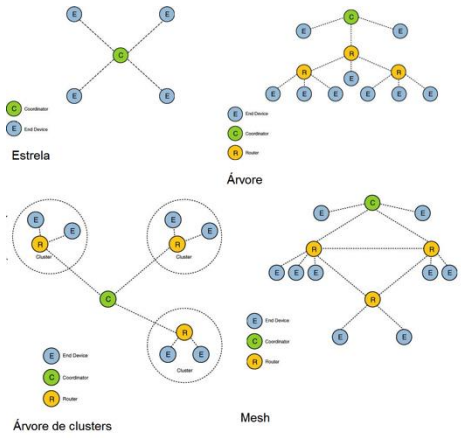

Figura 3 - Topologias, adaptado de [32]

WSN pode ser definida como uma rede de (pequeno porte e baixa complexidade) dispositivos, denominados nós, que são capazes de monitorar o ambiente e comunicar a informação obtida através de links sem fio; os dados são transmitidos, possivelmente através de múltiplos saltos, até um *sink*, que utiliza a informação localmente, ou está conectado a outras redes através de um gateway. Os nós podem estar estacionários ou se movimentando, eles podem saber de sua localização ou não e podem ser homogêneos ou não. A Figura 4 mostra a arquitetura de rede com um gateway fazendo a interconexão entre a Internet e duas WSN.

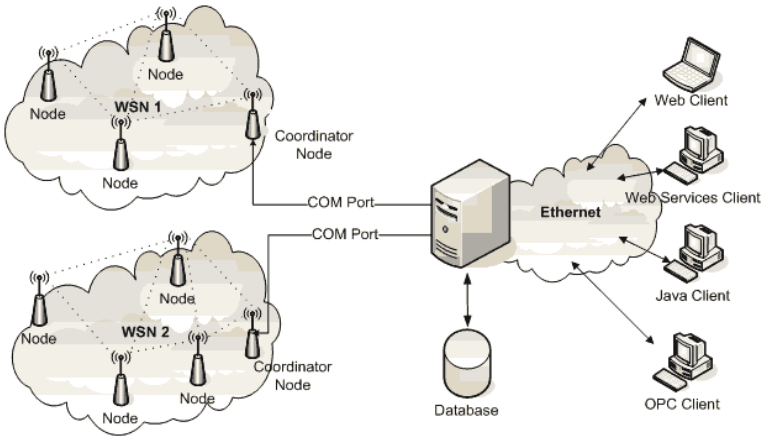

Figura 4 - Wireless Sensor Network [33]

Em redes de sensores os agentes que participam da comunicação são: coordenadores (*coordinators*), roteadores (*routers*) e dispositivos finais (*end-device*). Cada célula da WSN é caracterizada pelo seu PAN ID, o que significa Personal Area Network ID, por exemplo, na Figura 4 acima, a WSN1 e WSN2 apresentam diferentes PAN ID. O PAN ID define em qual cluster o dispositivo final se encontra.

O coordenador é o centro de um WSN e, encaminhas as informações dos sensores a uma central de monitoramento. A função do coordenador é dividida em duas partes: Servidor Web e fazer a interface para a WSN. O coordenador seleciona um PAN ID para iniciar a rede, permite nós entrarem na rede, pode armazenar pacotes enquanto seus filhos estão dormindo e faz a entrega dos mesmos assim que eles acordarem e não podem entrar no modo dormir. Os roteadores devem entrar na rede antes de transmitirem, receberem dados ou rotearem, e também não podem entrar no modo dormir. Os *end-devices* devem entrar na rede antes de transmitirem e receberem dados, não podem rotear pacotes e devem sempre enviar seus pacotes para seu pai e podem entrar no modo dormir para economizar bateria.

2.2. Acelerômetro e Giroscópio (MPU6050)

O InvenSense MPU6050, mostrado na Figura 5 abaixo, integra três graus de liberdade para o giroscópio, três graus de liberdade para o acelerômetro e um DMP (*Digital Motion Processor*) no mesmo chip. O MPU6050 contém um hardware de conversão analógico-digital de 16 bits para cada canal, logo ele coleta os canais x,y e z ao mesmo tempo. O sensor é baseado na tecnologia MEMS (*Microelectromechanical System*) e utiliza o protocolo de comunicação I^2C. A detecção de movimentos rápidos ou lentos pode ser diferenciada com a programação de um intervalo de escala fornecido pelo equipamento. O giroscópio apresenta um intervalo de $\pm 250^o/s$, $\pm 500^o/s$, $\pm 1000^o/s$ e $\pm 2000^o/s$ e o acelerômetro apresenta um intervalo de ±2g,±4g,±8g e ±16g.

Figura 5 - MPU6050 [4]

O protocolo de comunicação I^{2}C (*Inter-Integrated Circuit*) descreve o funcionamento de um barramento de comunicação serial que utiliza apenas dois fios: SDA (*Serial Data*) e SCL (*Serial Clock*). O protocolo I^{2}C atua em dois dispositivos denominados por Mestre e Escravo, sendo que o Mestre é responsável por coordenar todos os periféricos e o Escravo é responsável pela transferência de dados, podendo atuar como transmissor ou receptor dependendo da natureza do dispositivo conectado, por exemplo teclado ou LCD. A linha SCL é responsável pelo clock do barramento e a linha SDA pela transmissão de dados, quando ambas as linhas estão em nível alto, o estado é neutro. Para iniciar a comunicação entre dispositivos conectados, o SDA é trazido para o nível alto no nó mestre. Para escrever os dados no barramento, um bit SDA é lido a cada pulso SCL. A Figura 6 mostra esse processo e é possível verificar que a primeira linha corresponde ao SDA e a linha abaixo é o SCL.

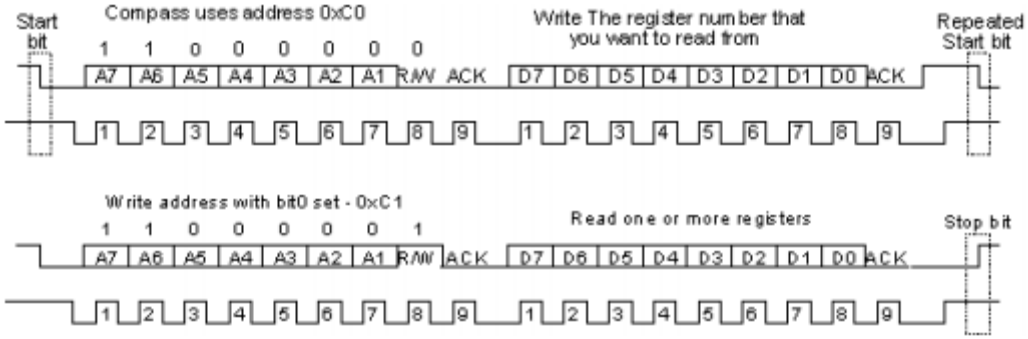

Figura 6 - Leitura e Escrita utilizando I2C, adaptado de [5]

2.3. Pulse Sensor

O *Pulse Sensor* é um sensor *open-source* de batimentos cardíacos *plug-and-play* que foi desenvolvido para ser anexado ao arduino. O sensor é utilizado por estudantes e desenvolvedores que desejam coletar batimentos cardíacos em tempo real e inseri-los em seus projetos.

Esse sensor realiza a fotopletismografia, que é a medição e registro das modificações de volume de uma parte do corpo, órgão ou membro decorrente de fenômenos circulatórios. Essa técnica é conhecida pela área biomédica por permitir dispositivos médicos não invasivos coletarem os batimentos cardíacos [38]. Outra aplicação da fotopletismografia é medir os níveis de oxigênio no sangue (SpO2), isso é realizado iluminando a pele e analisando mudanças na absorção da luz. O oxímetro convencional monitora a perfusão do sangue para a derme e tecido subcutâneo da pele.

O funcionamento do sensor se baseia na quantidade de luz refletida pela pele. Quanto maior a quantidade de luz refletida, maior a amplitude do sinal gerado pela fotoplestimografia; quanto menor, o contrário. Quando o coração bombeia o sangue para o corpo uma onda de pulso é gerada, a qual percorre as artérias até a extremidade do tecido capilar, onde o *Pulse Sensor* está anexado.

O valor dos batimentos cárdiacos é obtido fazendo a análise da Figura 7. Partindo do ponto ‘T’. Um pico no gráfico ocorre quando uma onda de pulso passa pelo sensor, depois o sinal volta a ficar em estado estacionário. Como a onda é periódica, um ponto de referência é escolhido, por exemplo, o pico, e a medida dos batimentos cardíacos é obtida realizando operações matemáticas nos tempos entre picos, chamado de IBI (*Inter Beat Interval*). O valor do Batimento Por Minuto é obtido a partir da média dos 10 últimos valores de IBI.

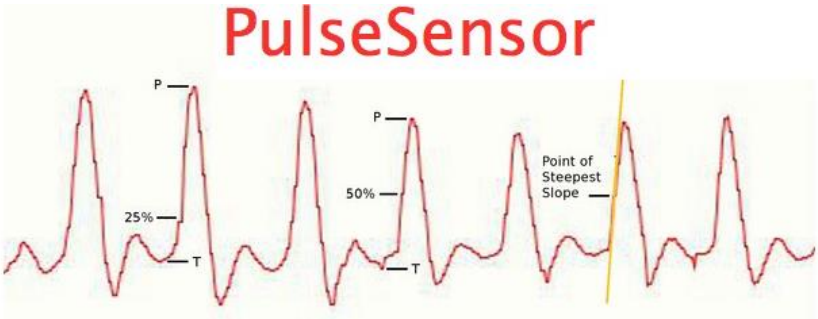

Figura 7 - Onda gerada pela Fotopletismografia [39]

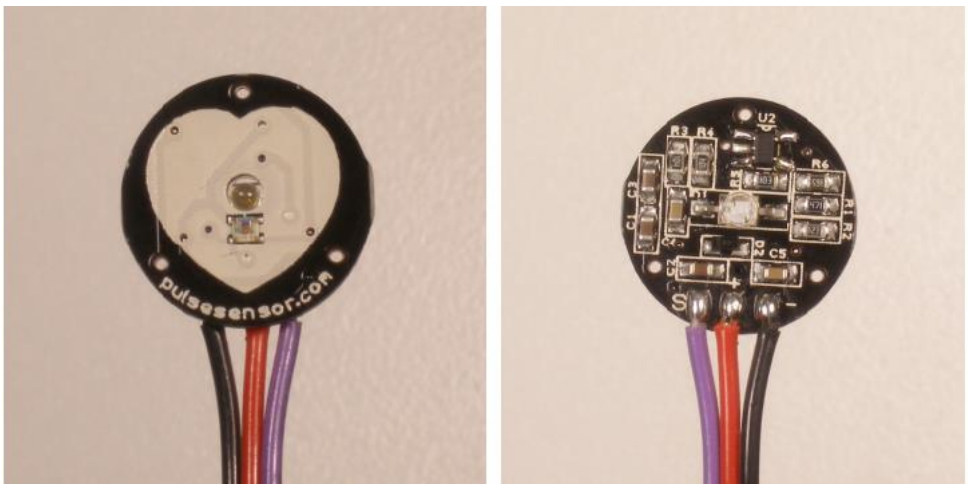

Figura 8 - Pulse Sensor [39]

A Figura 8 mostra uma foto do sensor de batimentos cardíacos. A frente do sensor contém uma logo com formato do coração. É esse lado que faz o contato com a pele, podemos ver que ele contém uma pequena lâmpada em seu centro, que é onde o LED (*Light Emitter Diode)* é irradiado, o pequeno quadrado logo abaixo, é um sensor de luz ambiente, exatamente o mesmo utilizado em celulares, *tablets* e notebooks que ajusta intensidade da tela em diferentes condições de luz presente no ambiente. O LED ilumina um tecido capilar (ponta dos dedos, lábios e orelha geralmente apresentam melhor precisão) e o sensor faz a leitura da intensidade da luz que retorna para o *Pulse Sensor.* A parte de trás é onde o circuito é montado.

O *Pulse Sensor* pode ser conectado ao arduino ou inserido em uma protoboard. Ele aceita tensões de +3V até +5V. Ao comprar o dispositivo, ele contém um código em Arduino, responsável pela leitura dos dados e seu envio para a porta serial, e um código em *Processing*, o qual apresenta graficamente os dados recebidos. A Figura 9 mostra o software de visualização.

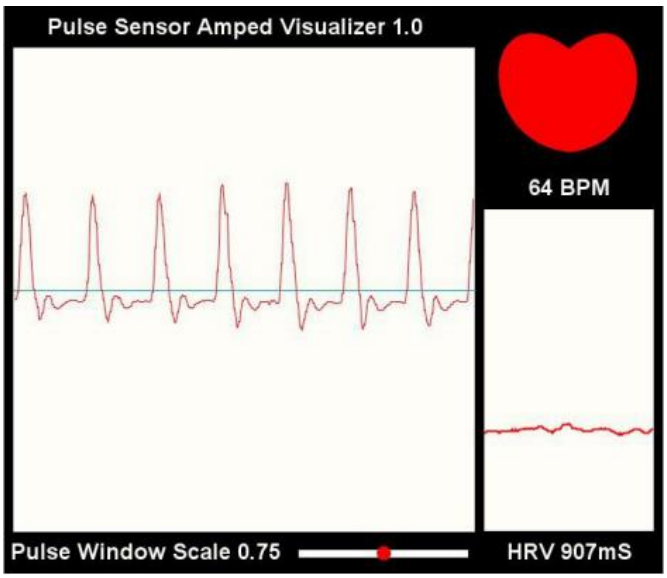

Figura 9 - Visualização do Pulse Sensor no software Processing [39]

A tela maior mostra o gráfico dos dados recebidos pelo sensor ao longo do tempo, a janela "*Pulse Window*", pode ser escalonada utilizando o *scrollbar*, aumentando ou diminuindo a escala dependendo se há um sinal com pequenas ou baixas amplitudes. A janela à direita mostra os batimentos cardíacos ao longo do tempo, ou seja, a variação dos batimentos HRV (*Heart Rate Variability*). Esse gráfico aumenta a cada pulso e o BPM (Batimentos Por Minuto) é atualizado com a média dos 10 últimos pulsos.

2.4. Raspberry Pi

O RaspberryPi é um mini-microcomputador, do tamanho de um cartão de crédito, que abriga processador, processador gráfico, slot para cartões de memória, interface USB (*Universal Serial Bus*), HDMI (*High –Definition Multimedia Interface*) e seus respectivos controladores. Além disso, ele também contém memória RAM (*Random Access Memory*), entrada de energia e barramentos de expansão. Ainda que pequeno, o Raspberry é um computador completo de baixo custo. O usuário pode conectá-lo a um monitor de computador ou a uma TV, juntamente com teclado e mouse e executar as atividades como navegar na Internet, escrever textos, ver vídeos, ouvir música, criar planilhas e realizar praticamente qualquer tarefa possível num computador convencional. A sigla "Pi" é uma abreviação para Python e foi intitulada pelos desenvolvedores por eles recomendarem o Python como uma linguagem de programação para aprendizado e desenvolvimento, contudo há várias outras linguagens que também podem ser utilizadas pelo RaspberryPi. O sistema operacional, o qual deverá ser instalado em um cartão de memória SD(Secure Digital), já que o mini-microcomputador não contém disco rígido próprio, é baseado no GNU/Linux, inclusive em suas várias distribuições: Raspian, Debian, Fedora Remix e Arch Linux.

O RaspberryPi apresenta-se em duas versões e em vários modelos. Os modelos existentes são A, A+, B, B+, Compute Module,Blue Pi e Red Pi. As versões se apresentam como versão 1 e versão 2. A versão 2, surgiu em Fevereiro de 2015 e conta com um processador *quad-core* de 900MHz da Broadcom, que promete trazer seis vezes o desempenho do chip anterior (700MHz). Graças ao novo processador, ele poderá rodar Ubuntu e também o Windows 10. A Microsoft já disponibilizou a versão gratuita do Windows 10 IoT destinada ao RaspberryPi 2. Os modelos podem ser fabricados na China e Reino Unido e não há diferenças de funcionalidades, esses modelos podem ser observados na Figura 10.

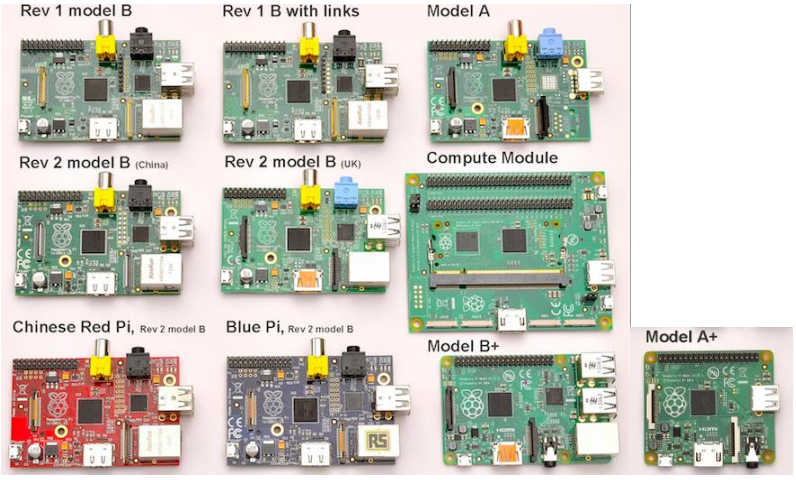

Figura 10 - Modelos de RaspberryPi, adaptado de [36]

2.5. Arduino

Arduino é uma plataforma *open-source* eletrônica para prototipagem que permite a criação de objetos eletrônicos interativos. Ele foi criado em 2005 com o objetivo de elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi utilizado o conceito de hardware livre, o que significa que qualquer pessoa pode montar, modificar, melhorar e personalizar o Arduino.

Foi criada uma placa composta por um microcontrolador Atmel e circuitos de entrada/saída, na qual pode ser facilmente conectada a um computador e programada via IDE (*Integrated Development Environment*) utilizando uma linguagem baseada em C/C++.

Depois de programado, o microcontrolador Arduino pode ser utilizado de forma independente, ou seja, pode-se colocá-lo para controlar um robô, cortinas, lâmpadas, ar-condicionado, além de utilizá-lo como um aparelho de medição, o qual realiza leituras de sensores.

Desde a criação do Arduino, diversas revisões foram realizadas e atualmente há mais de dez modelos disponíveis. A Figura 11 mostra seis destes modelos.

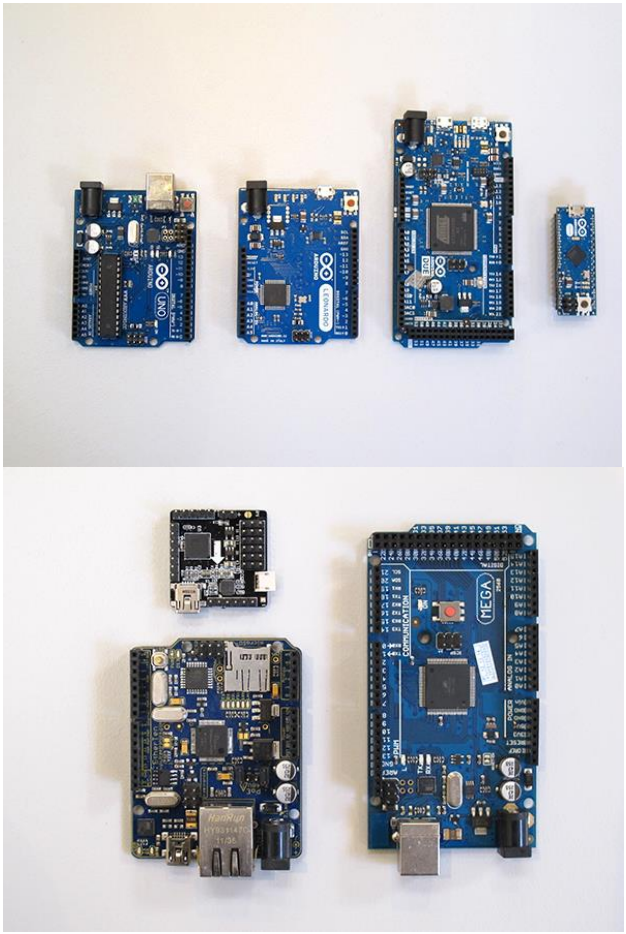

Figura 11 - Modelos de Arduino (à esqueda para direita: Uno, Leonardo, Due, Micro, Multiwii Nano, EtherTen e Mega 2650), adaptado de [37]

Todo programa em Arduino é chamado de *sketch* e contém uma estrutura básica composta por dois blocos de funções:

- **setup()**: É a primeira função a ser invocada quando o *sketch* inicia. Essa parte do programa é responsável por configurar as opções iniciais: iniciar valores de variáveis, definir portas, entre outros. Essa função é executada apenas quando o Arduino é inicializado ou resetado.
- **loop()**: Depois de criar a função setup (), que inicializa e define os valores iniciais, a função loop() faz a repetição do programa, permitindo-o realizar respostas e mudanças até que algum comando de "parar" seja enviado ao Arduino.

2.6 Basic4Android

O *Basic4Android* é uma ferramenta de desenvolvimento para dispositivos Android, iOS e PC desktop. A linguagem de programação utiliza é similar ao Visual Basic.

Java é a linguagem principal de desenvolvimento para Android, mas existem alternativas, tais como o *PhoneGap*, que pode tonar sites em aplicativos, e existem ferramentas como o Basic4Android, que oferecem outra abordagem.

O B4A (*Basic4Android*), da Anywhere Software, é um sistema de desenvolvimento completo para fazer aplicativos Android, também conhecido como IDE. Ele utiliza a linguagem de programação popular Basic.

Quando um aplicativo é desenvolvido no B4A, ele é automaticamente convertido em código Java, e então, o padrão Java é "empacotado" em um aplicativo Android, pronto para ser instalado em um dispositivo celular ou tablet. Logo, é necessário realizar a instalação das ferramentas de desenvolvimento Java.

A comunidade do B4A é extensa de tal forma que há várias dicas, sugestões e códigos já realizados. Ele se apresenta em três versões: gratuita, plano inicial e plano profissional. Nesse projeto foi utilizada a versão gratuita, que funciona por apenas 30 dias e há acesso limitado ao fórum e suporte.

Ao iniciar o programa, a tela mostrada na Figura 12 é apresentada ao usuário. As "funções" *Activity*, que são chamadas de *threads* pelo B4A, representam o ciclo de vida de um *software* Android.

Code:

```
Sub Process_Globals
    'These global variables will be declared once when the application starts.
    'These variables can be accessed from all modules.

End Sub

Sub Globals
    'These global variables will be redeclared each time the activity is created.
    'These variables can only be accessed from this module.

End Sub

Sub Activity_Create(FirstTime As Boolean)

End Sub

Sub Activity_Resume

End Sub

Sub Activity_Pause (UserClosed As Boolean)

End Sub
```

Figura 12 - Tela inicial do Basic4Android

Variáveis podem ser locais ou globais. Variáveis locais são declaradas dentro de uma Sub que não seja *Process_Globals* ou *Globals*. Essas variáveis existem apenas dentro de uma Sub e uma vez que esta termina, a variável, deixa de existir. Variáveis globais podem ser acessadas por todas as Subs. Há dois tipos de variáveis globais: *Process_variables* e *Activity_variables*.

As variáveis *Process_variables* existem enquanto o processo (programa) existir. Elas devem ser declaradas dentro de *Process_Globals*. Essa Sub é invocada apenas no momento em que o processo inicia. Contudo, nem todo tipo de objeto pode ser declarado como *Process_variables*, aqueles que não desejamos que seja destruído junto com a atividade, pois uma vez que a atividade é destruída, todos os objetos declarados como *Process_variables* também serão destruídos.

As variáveis *Activity_variables* estão contidas dentro de uma atividade. Elas devem ser declaradas dentro da Sub *Globals*. Essas variáveis são privadas e podem ser acessadas apenas pela atividade que está sendo executada. Qualquer tipo de objeto pode ser declarado como *Acitivity_variables*. Toda vez que uma atividade é criada, a Sub *Globals* é invocada.

Sub *Activity_Create* (*FirstTime As Boolean*) : é chamada quando uma atividade é criada. A atividade é criada quando o usuário lança a primeira aplicação, a configuração do dispositivo foi alterada (usuário rotacionou o dispositivo), atividade foi destruída, ou quando a atividade estava no fundo e o sistema operacional decidiu destruí-la, a fim de liberar memória. Esta Sub deve ser utilizada para carregar ou criar o *layout*. O parâmetro *FirstTime* indica se esta é a primeira vez que esta atividade é criada. *FirstTime* pode ser utilizado para executar todos os tipos de inicializações relacionadas com as variáveis do processo, por exemplo, se há um arquivo com uma lista de valores que precisa ser lido, pode-se lê-lo se *FirstTime* é verdadeiro e armazenar a lista como uma variável do processo. Para resumir, pode-se testar se *FirstTime* é verdadeira e, em seguida, inicializar variáveis de processo.

Sub *Activity_Pause* (*UserClosed As Boolean*): cada vez que a atividade se move a partir do primeiro plano para o segundo plano (*background*) a Sub *Activity_Pause* é invocada. Esta Sub também é invocada quando a atividade está em primeiro plano e uma alteração de configuração ocorre (o que leva a atividade a fazer uma pausa e, em seguida, destruí-la). Não é recomendado guardar informações importante no *Activity_Pause*.

Sub *Activity_Resume*: é invocada quando *Activity_Create* termina ou quando retorna-se de uma atividade em pausa (atividade mudou-se para o *background* e agora retorna para o primeiro plano).

Como discutido acima, *Activity_Pause* é chamada cada vez que a atividade se move a partir do primeiro plano para o segundo plano. Isso pode acontecer por que: uma atividade diferente foi iniciada, o botão *Home* foi pressionado, a configuração foi alterada, um evento foi gerado (orientação mudou, por exemplo) ou o botão voltar foi pressionado.

O B4A disponibiliza uma ferramenta chamada *Service* na qual pode-se colocar uma aplicação em background e um determinado evento poderá invocar a aplicação e colocá-la em primeiro plano.

2.7 Desenvolvimento Web

2.7.1 PHP

O PHP é um acrônimo para *Hypertext Preprocessor*. É uma linguagem de *script* devido ao fato do código PHP ser "interpretado" e não compilado. Essa linguagem foi criada para o desenvolvimento de conteúdos web, ou seja, a linguagem é utilizada para o desenvolvimento de páginas dinâmicas e automáticas.

2.7.2 HTML

O HTML significa *Hyper Text Markup Language*. Um arquivo HTML é um arquivo de texto contendo pequenas etiquetas de marcação, essas são utilizadas pelo navegador Web para mostrar como a página deve ser exibida.

2.7.3 CSS

O CSS (*Cascading Style Sheets*) é uma linguagem de programação que define como os elementos HTML serão visualizados. O CSS controla fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, posicionamento, entre outros. Ele é suportado por todos os navegadores atuais. HTML é utilizado para estruturar conteúdos, enquanto que o CSS formata os conteúdos estruturados.

2.7.4 MySQL

O MySQL é um gerenciador de banco de dados relacional de código aberto mais popular do mundo e possibilita a entrega econômica de aplicativos de banco de dados confiáveis, de alto desempenho e redimensionáveis.

Banco de dados é uma aplicação separada que armazena uma coleção de dados. Cada banco de dados tem uma ou mais APIs(*Application Programming Interface*) diferentes para criar, acessar, gerenciar, pesquisar e replicar os dados que ele contém. Outros tipos de armazenamento de dados também podem ser utilizados, tais como arquivos no sistema de arquivos ou grandes tabelas Hash na memória, mas a busca e escrita de dados não seria tão

rápido e fácil com esses tipos de sistemas. Então atualmente, utilizamos RDBMS(*Relational database management systems*) para armazenar e gerenciar grandes volumes de dados. Isso é chamado de banco de dados relacional, porque todos os dados são armazenados em diferentes tabelas e relações são estabelecidas usando chaves primárias ou chaves estrangeiras.

Um RDBMS permite a implementação de um banco de dados utilizando tabelas, colunas e index. Ele atualiza o index automaticamente, garante a integridade referencial entre linhas de várias tabelas, interpreta um query SQL e combina informações originadas de várias tabelas.

Alguns conceitos são essenciais para entender o MySQL:

- **Banco de dados:** um banco de dados é uma coleção de tabelas, com dados relacionados.
- **Tabela:** uma tabela é uma matriz com os dados. Uma tabela em um banco de dados parece uma planilha.
- **Coluna:** uma coluna contém os mesmos tipo de dados, por exemplo, o CEP.
- **Linha:** a linha (tupla, a entrada ou registro) é um grupo de dados relacionados, por exemplo, os dados de uma assinatura.
- **Redundância:** o armazenamento de dados duas vezes, de forma redundante para tornar o sistema mais rápido.
- **Chave primária:** a chave primária é única. Um valor de chave não pode ocorrer duas vezes em uma tabela. Com uma chave, você pode encontrar no máximo uma linha**.**
- **Chave estrangeira:** uma chave estrangeira é o que conecta duas tabelas.
- **Index:** um índice em um banco de dados se assemelha a uma página de livro.
- **Integridade Referencial:** integridade referencial garante que um valor de chave estrangeira sempre aponta para uma linha existente.

MySQL é um RDBMS utilizado por muitas pequenas e grandes empresas. Ele é desenvolvido, comercializado, e apoiada pela MySQL AB, que é uma empresa sueca. MySQL é tão popular por causa de muitas razões, entre elas: é liberado sob uma licença de código aberto, utiliza um formulário padrão da língua de dados do SQL, funciona em muitos sistemas operacionais e em muitas linguagens de programação, incluindo PHP, Python, Perl, C, C ++, Java, etc, funciona rapidamente mesmo se apresentar um grandes conjuntos de dados, é

amigável para PHP, suporta grandes bancos de dados, até 50 milhões de linhas ou mais em uma tabela, é customizável.

O MySQL contém um objeto chamado *trigger*. Ele é associado a uma tabela e permite a realização de processamentos em consequência de uma determinada ação como, por exemplo, a inserção de um registro. *Triggers* são executados ANTES ou DEPOIS das operações de DML (*Data Manipulation Language*): *INSERT*, *DELETE* e *UPDATE*. Um exemplo de sintaxe está descrito abaixo:

```
CREATE TRIGGER nome momento evento

ON tabela

FOR EACH ROW

BEGIN

/* corpo do código */

END
```

2.7.5 Google Charts

O *Google Charts* é uma API fornecida pela google na qual é possível criar gráficos dinâmicos e colocá-los em páginas web. Ela utiliza a linguagem JavaScript, objetos JSON (*JavaScript Object Notation*) e XML (*Extensible Markup Language*). Esses gráficos podem variar de um simples gráfico de linhas para um gráfico complexo com hierarquias.

Para construir os gráficos dinâmicos deve-se importar algumas bibliotecas *Google Charts*, fazer uma lista dos dados a serem inseridos no gráfico, selecionar as opções para customizá-lo, e finalmente criar um objeto "*chart*", com um id que deve ser escolhido. Esse id será utilizado na página web que foi criado com a .

Todos os tipos de *charts* contém dados que utilizam a classe *DataTable*, facilitando a troca entre diferentes tipos de gráficos, permitindo escolher o mais apropriado para a aplicação. A classe *DataTable* fornece funções para classificar, modificar e filtrar os dados e podem seus valores podem ser atualizados diretamente da página web ou de um banco de dados. Alguns exemplos de tipos de gráficos que o *Google Chart* suporta são: gráfico de barras, calendário, coluna, combo, Gantt, histogramas, área, bolhas, entre outros.

```html
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

      // Load the Visualization API and the piechart package.
      google.load('visualization', '1.0', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.setOnLoadCallback(drawChart);

      // Callback that creates and populates a data table,
      // instantiates the pie chart, passes in the data and
      // draws it.
      function drawChart() {

        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Topping');
        data.addColumn('number', 'Slices');
        data.addRows([
          ['Mushrooms', 3],
          ['Onions', 1],
          ['Olives', 1],
          ['Zucchini', 1],
          ['Pepperoni', 2]
        ]);

        // Set chart options
        var options = {'title':'How Much Pizza I Ate Last Night',
                       'width':400,
                       'height':300};

        // Instantiate and draw our chart, passing in some options.
        var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>

  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div"></div>
  </body>
</html>
```

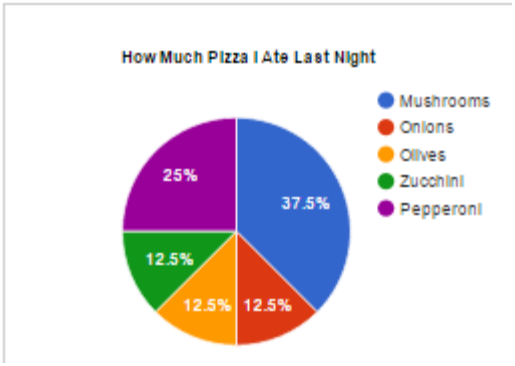

Figura 13 - Google Charts [20]

De acordo com a Figura 13, o primeiro passo para desenhar um *Google Chart* é carregar a biblioteca JavaScript do Google (JSAPI) e depois utilizá-la para carregar a biblioteca de visualização. Com a biblioteca carregada, o próximo passo é selecionar os dados a serem exibidos. Os dados devem ser inseridos na classe JavaScript *google.visualization.DataTable().* Essa classe é definida pela biblioteca de visualização carregada anteriormente.

DataTable é uma tabela bidimensional com linhas e colunas, onde cada coluna tem um tipo de dado(*type*), um ID opcional e uma descrição, também opcional. A *DataTable* que representa a Figura 13 está demonstrada na Figura 14.

type: string label: Topping	type: number label: Slices
Mushrooms	3
Onions	1
Olives	1
Zucchini	1
Pepperoni	2

Figura 14 - DataTable Google Charts [20]

Existe a opção de adicionar elementos na tabela fazendo *query* em um website que suporta o protocolo *Chart Tools Datasource* e utilizar um objeto *google.visualization.Query*, de tal forma a enviar um *query* para o website e adicionar a resposta recebida à tabela.

Agora que os dados já estão inseridos na tabela, customiza-se o gráfico, indicando o título, legendas, cor e tamanho. A variável *options* contém todas as customizações que desejamos para o gráfico. Para finalizar, deve-se indicar qual tipo de gráfico satisfaz a aplicação.

2.8 Cron Job

Cron Job é uma ferramenta fornecida pelo sistema operacional Linux, que permite a realização de agendamento de tarefas programadas em dias e horários determinados pelo programador.

Para executar as tarefas, o *cron* usa uma espécie de tabela conhecida como *crontab*. O arquivo *crontab* normalmente fica localizado no diretório /etc, mas também pode estar em um diretório que cria um *crontab* para cada usuário do sistema, tudo depende das configurações do sistema operacional.

Para configurar tarefas *cron*, primeiramente devemos abrir o *crontab*, utilizando o comando "*crontab* – e" e configurar de acordo com a sintaxe abaixo:

[minutos] [horas] [dias do mês] [mês] [dias da semana] [usuário] [comando]

Minutos: números de 0 a 59;

Horas: números de 0 a 23;

Dias do mês: números de 0 a 31;

Mês: números de 1 a 12;

Dias da semana: números de 0 a 7;

Usuário: informa o usuário que irá executar o comando (não é necessário especificá-lo caso o

arquivo do próprio usuário for utilizado);

Comando: a tarefa que deve ser executada

No lugar destes valores pode-se colocar “*” para especificar uma execução constante, por exemplo, se o campo “Dias do mês” conter “*”, o comando será executado todos os dias.

2.10. Centro de Gravidade

O Centro de Gravidade ou baricentro de um corpo é a posição onde pode ser considerada a aplicação da força de gravidade resultante equivalente de todo o corpo. É o ponto no qual o corpo se equilibra, levando-se em conta a aceleração da gravidade. De uma forma geral, quando se considera a não uniformidade de campos gravitacionais, as determinações da força de gravidade total e do seu ponto de aplicação ficam dependentes da posição e orientação do corpo. Portanto, o centro de gravidade não pode ser considerado uma característica específica de um corpo rígido.

Pessoas com tamanhos diferentes apresentam centros de gravidades diversos, como pode ser verificado na Figura 15.

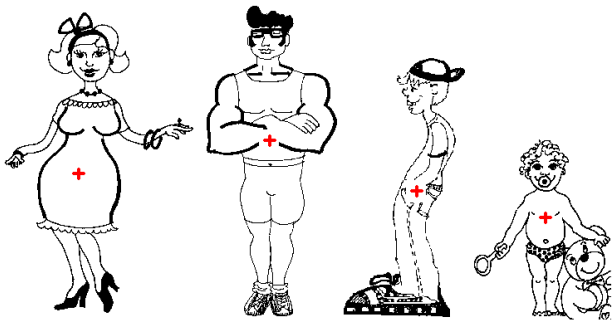

Figura 15 - Pessoas de tamanho diferentes com centro de gravidade localizado em partes diferente do corpo [47]

Posições diferentes também oferecem diferentes centros de gravidade, como pode ser visto na Figura 16.

Figura 16 - Pessoas em posições diferentes oferecem diferentes centros de gravidade [48]

Centro de massa é um ponto no qual se concentra toda a massa do corpo. Esse termo às vezes é confundido com centro de gravidade, ambos designam coisas diferentes, mas eles podem ser coincidentes se a aceleração da gravidade for a mesma para cada partícula que compõe o corpo.

2.11. Princípio Fundamental da Dinâmica (Segunda Lei de Newton)

Um corpo de massa **m** em queda livre na Terra está submetido a uma aceleração de módulo **g** (aceleração da gravidade, de valor aproximadamente 9,8 m/s^2). Se desprezarmos os efeitos do ar, a única força que age sobre o corpo é a força gravitacional ($\overrightarrow{F_G}$). Podemos relacionar essa força à aceleração correspondente através da segunda Lei de Newton, $\vec{F} = m.\vec{a}$. A força gravitacional atua sobre o corpo mesmo quando não está em queda livre, mas se encontra, por exemplo, em repouso, como em pé, sentado ou deitado.

O peso de um corpo é igual ao módulo da força gravitacional que age sobre o corpo e temos a seguinte relação:

$$P = mg$$

3. Metodologia

3.1. Delimitação do Tema

O presente trabalho tem como proposta desenvolver um equipamento que detecta quedas e posição corporal juntamente com o monitoramento dos batimentos cardíacos e da temperatura ambiente. Esse dispositivo permite ao usuário acompanhar o estado do paciente através de um aplicativo Android e de uma Plataforma Web.

Foi utilizada uma WSN para realizar o monitoramento do indivíduo. A topologia escolhida foi a estrela visto que no momento de desenvolvimento do projeto estavam disponíveis apenas três rádios Xbees e ela satisfaz a baixa complexidade do sistema. Dois rádios foram utilizados como roteadores, pois em aplicações de monitoramento os nós não podem dormir, e um rádio foi configurado como coordenador.O papel do coordenador é desempenhado pelo RaspberryPi e o papel dos roteadores é realizada pelos dispositivos de monitoramento. As etapas necessárias para fazer a instalação da rede estão apresentadas no Capítulo 3.

O protocolo de camada física escolhido para realizar o projeto foi o 802.15.4, pois é uma tecnologia de baixo custo, baixo consumo de energia e baixa velocidade se for comparado ao padrão 802.11, como mostra a Tabela 2-1.

O sistema é responsável não apenas por detectar o estado do paciente, mas também alertar o cuidador que uma situação está anormal, estas são caracterizadas por quedas ou elevação do batimento cardíaco acima de um *threshold*. O dispositivo realiza avisos através de *email*, *sms* e ligação telefônica. Quando os batimentos estão acima do estabelecido, um *sms* é enviado avisando a situação do indivíduo; se em vez disso, uma queda tiver ocorrido, o sistema envia um alerta para o *email* cadastrado e realiza uma ligação telefônica; se ninguém atender a chamada, o dispositivo faz uma nova ligação para outro número cadastrado.

O fluxograma da Figura 17 mostra as etapas realizadas para o desenvolvimento desse projeto.

Etapa 1 – Compra dos dispositivos de construção do Hardware: definição dos materiais necessários para a implantação do projeto, quais os valores e onde os dispositivos foram adquiridos.

Etapa 2 – Montagem do Hardware: apresenta a conexão dos dispositivos.

Etapa 3 – Configuração da Rede: aborda como foi criada a rede, sua arquitetura e seus participantes.

Etapa 4 – Desenvolvimento do algoritmo dos dispositivos de monitoramento: apresenta o processo necessário para o desenvolvimento do *software* que faz a leitura e envio dos dados coletados pelos sensores.

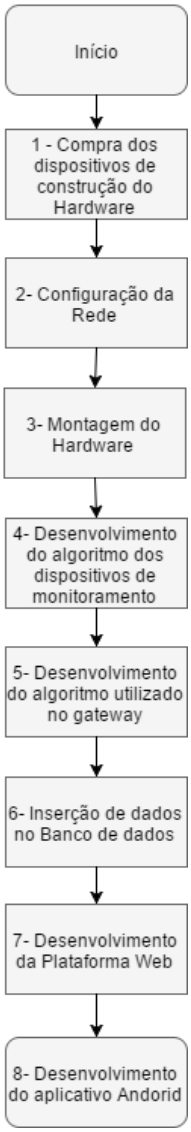

Figura 17 - Etapas para a construção do equipamento de detecção e monitoramento

Etapa 5 – Desenvolvimento do algoritmo utilizado no gateway: apresenta o processo necessário para o desenvolvimento do *software* utilizado no *gateway*, mostrando como o recebimento de informações é realizado via banco de dados.

Etapa 6 – Inserção de dados no Banco de dados: apresenta as tabelas que foram criadas para esse projeto, o modelo de banco de dados e o Diagrama de Classe.

Etapa 7 – Desenvolvimento da Plataforma Web: apresenta as linguagens e ferramentas utilizadas para o desenvolvimento da página web.

Etapa 8 – Desenvolvimento do aplicativo Android: apresenta quais ferramentas e quais funcionalidades foram implementadas para a construção do aplicativo.

3.1.1. Compra dos dispositivos de construção do Hardware

Após a definição do tema, da arquitetura e dos objetivos do projeto, foi realizada uma busca dos materiais necessários para sua implementação. A Tabela 3-1 mostra a quantidade, onde os dispositivos foram adquiridos e seus respectivos preços.

Tabela 3-1 Materiais necessários para construção do equipamento

Quantidade	Dispositivo	Preço unitário	Loja
2	Rádio Xbees	$22,95	Sparkfun
2	Breakout Board for Xbee Module	$2,95	Sparkfun
1	XBee Explorer Dongle	$24,95	Sparkfun
1	Pulse Sensor	$24,95	Sparkfun
2	Arduino Uno	$24,95	Sparkfun
1	Raspberry Pi modelo B	$41,95	Sparkfun
2	Sensores MPU6050	R$25,90	Huinfinito
2	Protoboard	$4,95	Sparkfun
1	Jumper Wire Kit	$6,95	Sparkfun
1	Ethernet Cable 3ft	$1,95	Sparkfun
1	SD Card 8GB	$11,95	Sparkfun
2	Fonte de Alimentação para Arduino	R$20,00	Labdegaragem
1	Fonte de Alimentação para RaspberryPi	$8,00	Amazon

3.1.2. Montagem do Hardware

Nessa etapa é realizada a montagem do *hardware* do sistema. Foram utilizados todos os dispositivos apresentados na Tabela 3-1. A arquitetura do projeto está apresentada na Figura 22. O dispositivo é composto por um *gateway* receptor e dois transmissores. Cada transmissor é composto por um sensor MPU6050, um rádio XBee e um Arduino. O *gateway* é composto pelo RaspberryPi, um rádio XBee e um cabo Ethernet.

Um dos transmissores é anexado ao peito e o outro à coxa do paciente. A Figura 18 apresenta os dispositivos anexados ao corpo e a Figura 19 apresenta os dispositivos peito e coxa, respectivamente. O sensor de batimentos cardíacos é instalado ao dispositivo peito e é mostrado na Figura 20.

Figura 18 - Dispositivo anexado ao corpo

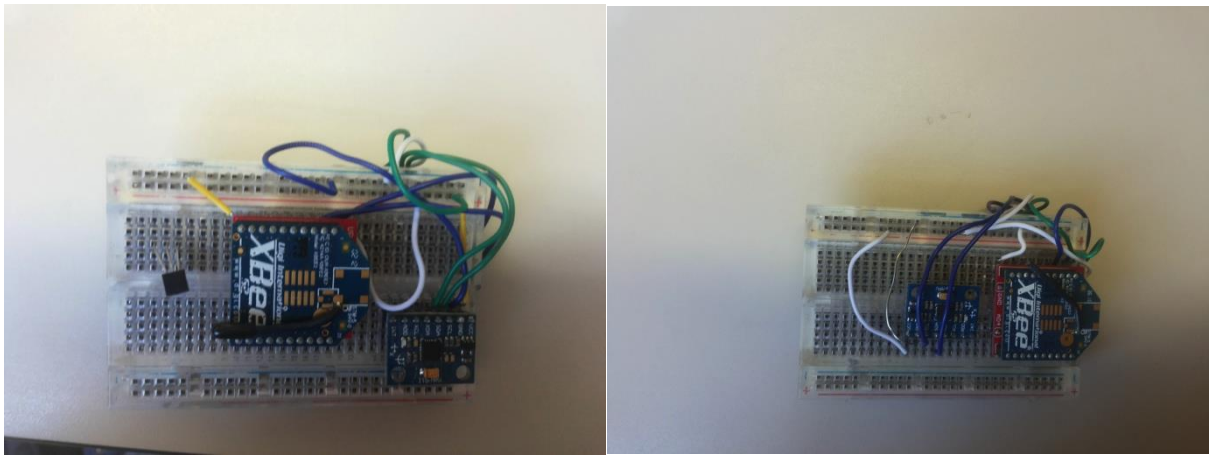

Figura 19 - Dispositivo peito (à esqueda) e dispositivo coxa (à direita)

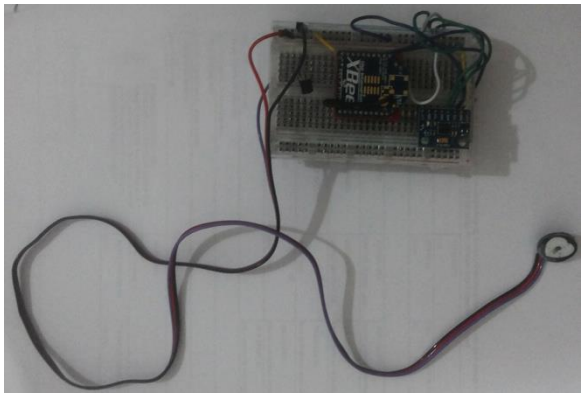

Figura 20 - Dispositivo anexado ao peito com sensor de batimentos cardíacos

O primeiro passo para realizar a montagem, foi soldar cada rádio XBee em uma *breakout board*. Essa placa é necessária porque o tamanho dos pinos dos rádios são diferentes do tamanho das entradas da *protoboard*, então um conversor de pinos é necessário. Para fazer a conexão entre o rádio e o Arduino foi adicionada uma tensão de +3,3V ao XBee e seu pino digital de saída foi conectado ao pino receptor (digital 0) do Arduino e, seu pino digital de entrada ao transmissor (digital 1) do Arduino. Com isso, foi montado o hardware dos dispositivos que farão o monitoramento do estado do paciente.

Após os XBees estarem fixados, foi anexado o MPU6050 à *protoboard*. Como dito na seção 2.2 o sensor necessita das linhas SCL e SDA, para controlar a pulsação do *clock* e a transmissão para o barramento, respectivamente. Para realizar a comunicação I^2C entre o MPU6050 e o Arduino, a linha SCL foi anexada à entrada analógica A5 e a linha SDA foi anexadaa à entrada analógica A4. O sensor foi alimentado com uma tensão de +3,3V, sendo que a mesma poderia também ser de +5V, contudo como o XBee utiliza uma tensão de +3,3V a mesma foi escolhida. O sensor de batimentos cardíacos foi adicionado à entrada análogica A0 do Arduino.

O próximo passo é fazer a instalação do hardware do Gateway, que pode ser visualizado na Figura 21.

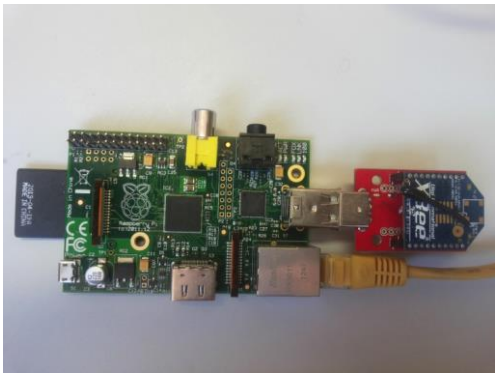

Figura 21 - Gateway

O Gateway é composto pelo RaspberryPi, cabo Ethernet e um módulo XBee. Primeiramente, deve-se instalar o NOOBS (*New Out Of the Box Software*), que é um gerenciador de instalação de um sistema operacional para o RaspberryPi, no cartão SD. Ao ligar o RaspberryPi pela primeira vez, uma lista de diferentes sistemas operacionais é apresentada, dentre as opções, foi escolhido o Raspbian.

Com o sistema operacional instalado, o rádio XBee foi conectado ao *dongle* e este conjunto foi anexado à entrada USB do RaspberryPi. O cabo Ethernet também é conectado, e assim, tem-se o hardware do gateway comunicando-se com a Internet. Apesar de sua instalação ser mais simples do que os dispositivos de monitoramento, o software do gateway é

bem mais complexo, como pode será visto na seção 3.1.5.

3.1.3. Configuração da Rede

A arquitetura e equipamentos desenvolvidos nesse projeto são mostrados nas Figuras 22 e 23, sendo esta última, com ênfase no hardware.

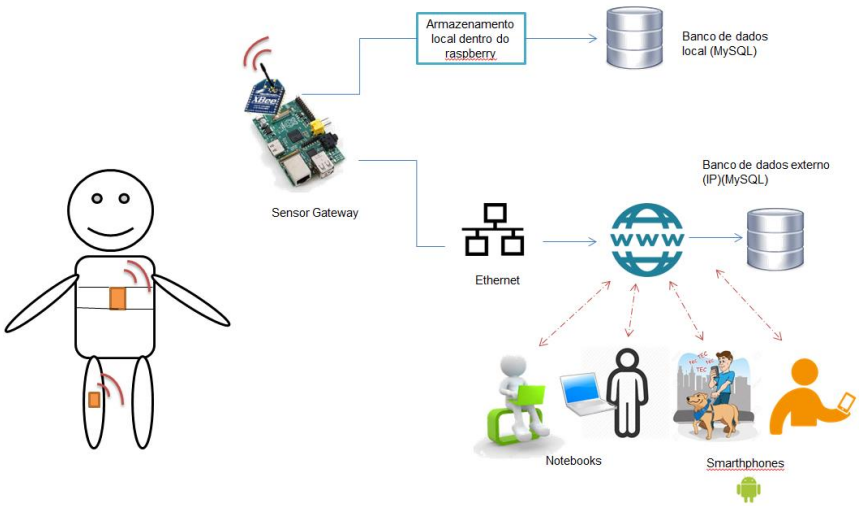

Figura 22 - Arquitetura do sistema

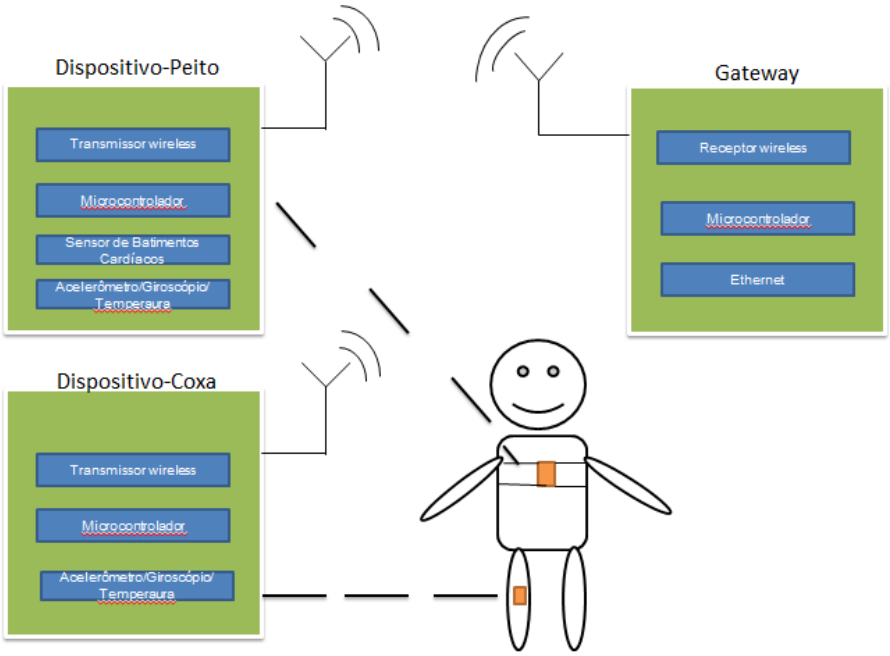

Figura 23 - Arquitetura do Projeto com ênfase no hardware

A Figura 23 mostra os componentes de hardware envolvidos na iteração entre o paciente, Arduinos (Dispositivos Peito e Coxa) e Raspeberry (*Gateway*). Cada um dos dispositivos de monitoramento, tem um microcontrolador (ATmega328P) utilizado pelo Arduino, sensor e um transmissor sem fio. O *gateway*, tem um receptor sem fio para fazer o recebimento dos dados, um processador Broadcom de 700 MHz, utilizado pelo RaspberryPi e um controlador Ethernet.

Como mostra a Figura 22, deve-se fazer a configuração da rede de tal modo que o equipamento utilizado pelo paciente se comunica via 802.15.4, utilizando rádios XBees, com o *gateway*, o mesmo, após receber as informações, utiliza a arquitetura Ethernet para enviar os dados para o banco de dados implementado no RaspberryPi.

O rádio XBee utilizado nesse projeto é o modelo apresentado na Figura 2, que permite um alcance de até 120 metros, logo o uso deste equipamento deve ser restrito até esse limite. Contudo, como 70% das quedas realizadas por idosos ocorrem dentro de casa [35] este equipamento satisfaz as necessidades do projeto. Como descrito em trabalhos futuros, a instalação de módulos 3G/GPRS (*General Packet Radio Services*) é desejável para fazer o monitoramento não apenas em ambientes domiciliares mas também coberturas de alcance superior à WLAN (*Wireless Local Network*), aumentando assim a utilização do equipamento em áreas densamente povoadas.

A *Digi International* oferece um software de configuração de XBees chamado X-CTU. A Figura 24 mostra a tela de configuração dos rádios utilizados neste projeto no X-CTU.

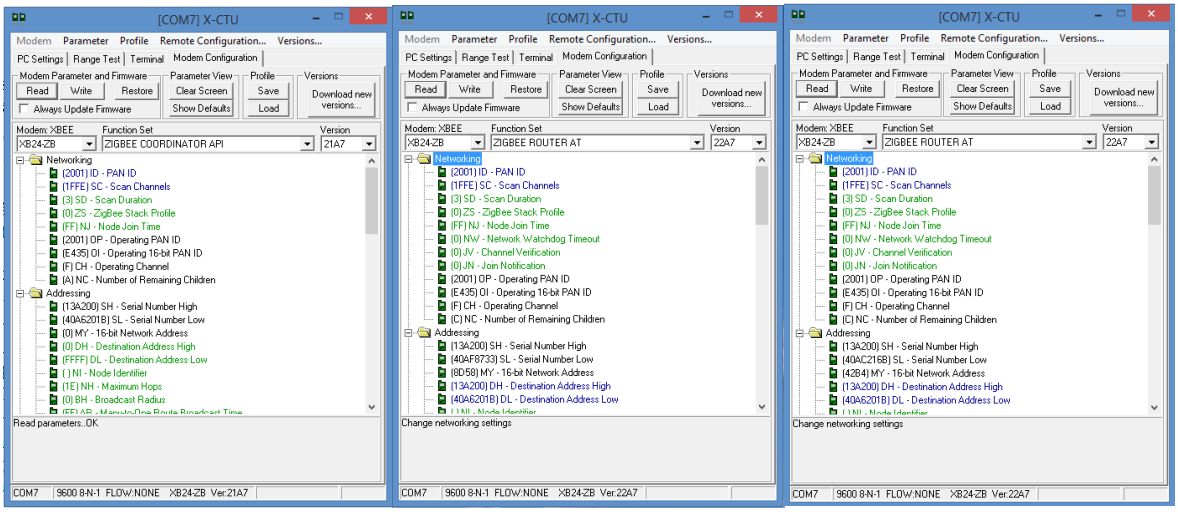

Figura 24 - Configuração dos XBees no X-CTU (coordenador à esqueda)

Na seção Networking da Figura 24 pode-se verificar que o PAN ID utilizado pelos três rádios é 2001. Esse valor pode ser aleatório, mas precisa ser hexadecimal e estar no range 0-0xFFFF. XBees apenas podem se comunicar se eles estiverem o mesmo PAN ID, ou seja, devem pertencer a mesma célula WSN. Os campos em verde permitem ter seus valores alterados, enquanto que os campos em azul e preto não são configuráveis.

Figura 25 - MY address XBees

Cada XBee na rede tem um endereço de 16 bits (entre 0-0xFFFF), o qual é referido como *MY address*, ou endereço da fonte, que está presente na parte de trás do mesmo, como pode ser visto na Figura 25 e comparar com a Figura 24, na seção "*Addressing*" do X-CTU. Após o PAN ID estar definido, outra configuração importante é o endereço de destino, que determina para qual endereço de fonte um rádio deve enviar dados. Para um XBee ser capaz de enviar dados para outro, ele deve ter como número de destino o mesmo endereço de fonte do outro, por exemplo, o coordenador acima tem como endereço de destino o *Broadcast*, ou seja, todo mundo da mesma WSN recebe os dados que ele envia, contudo, os *end-devices* tem como endereço de destino o endereço *MY address* do coordenador.

O *My address* do dispositivo anexado à coxa é 13A20040AF8733, enquanto que a do dispositivo anexado ao peito é: 13A20040AC216B.

Pode-se observar da Figura 24 que o coordenador está no modo API, enquanto que os roteadores estão no modo AT. Os módulos XBees podem ser configurados em dois modos: Modo Transparente (AT) e Modo API (API). O modo transparente limita a comunicação para ser P2P(*Peer-to-Peer*), enquanto que em modo API, pode-se comunicar não apenas com o coordenador, mas com vários outros XBees. A desvantagem do modo API é que ele adiciona várias informações ao pacote. Nesse projeto, escolheu-se o modo AT nos roteadores porque a comunicação sempre será P2P, entre o roteador e o coordenador e, evita-se o envio desnecessário de dados no canal de comunicação.

As informações adicionadas a um pacote API indicam como comandos, respostas de comandos e mensagens de estado do módulo devem ser enviadas e recebidas. As Figuras 26 e 27 indicam como é um frame API. Verifica-se que o modo API incorpora os comandos AT e adiciona outras informações para formar o quadro.

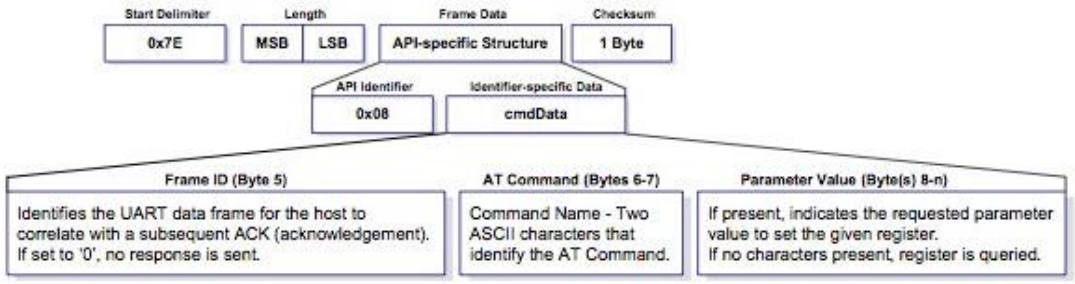

Figura 26 - Frame API [40]

```
7e 00 04 08 52 4D 59 FF

7E    : API Frame
00 02 : Length
08    : AT Command Frame id
52    : Frame id
4d 59 : MY (4d 59) (4d ⇒ M, 59 ⇒ Y) Get the 16-bit network address of the module.
FF    : checksum ff = ff - ((08+52+4d+59) & ff)
```

Figura 27 - Exemplo de dados API recebidos [40]

3.1.4. Desenvolvimento do algoritmo dos dispositivos de monitoramento

Todo algoritmo desenvolvido nesta seção foi implementado na plataforma Arduino. Ele apresenta um *script* que utiliza a biblioteca I^2C e a biblioteca MPU6050. Neste projeto foram utilizados três *scripts*: calibração, leitura dos dados dos sensores e o *script* de interrupção, o qual detecta se há sinais de batimento cardíaco. Os dados utilizam uma porta de 9600 bauds e são amostrados a 10Hz, ou seja, a cada 100ms.

A função do *software* nessa parte do equipamento consiste em fazer a calibração dos sensores, definir o *offset*, gerar a interrupção para leitura do sensor de batimentos, obter os valores lidos pelo giroscópio e acelerômetro nos eixos x, y e z, obter o valor da temperatura e enviar os dados pelo XBee. Esses dispositivos de monitoramento desempenham a função de roteadores de uma WSN, pois coletam as informações e as enviam para o nó coordenador sem entrar no modo dormir.

Primeiramente foi realizado o upload do código que faz a calibração dos sensores para o Arduino. A configuração padrão do MPU6050 apresenta uma sensibilidade de ±2g, que é o intervalo de maior sensibilidade, e este foi o valor utilizado nesse projeto. Para obter o valor correspondente de força gravitacional aplicada ao corpo, devemos utilizar um fator de sensibilidade, neste caso o valor adimensional de 16384. Logo, todos os valores obtidos pelo acelerômetro foram divididos por 16384 para chegar ao valor correspondente à força gravitacional, (como exemplo, tendo o sensor na posição horizontal e sem movimento, o valor obtido deverá ser aproximadamente zero para o eixo x, zero para o eixo y e um para o eixo z).

O *script* de calibração inicia assumindo valores iniciais de *offset* para os eixos e depois realiza um conjunto de médias de tal forma que o resultado final esperado para o sensor em posição horizontal sem movimento seja “0 0 16384 0 0 0”. Os valores de *offset* obtidos para os sensores são mostrados na Tabela 3-2.

Tabela 3-2 Valores de offset

	Coxa	**Peito**
x acelerômetro(g):	-1747	-4415
y acelerômetro(g):	-10	1795
z acelerômetro(g):	1299	1405
x giroscópio(°/s):	80	308
y giroscópio(°/s):	37	1
z giroscópio(°/s):	64	53

A biblioteca MPU6050 contém um conjunto de funções que ajustam o valor lido pelos sensores de acordo com o valor de *offset* que é passado como parâmetro, por exemplo: *setXAccelOffset(-4415)*, tendo uma função para cada eixo. Após esse procedimento, é iniciado um *loop* que utiliza a função *getMotion6()* da biblioteca MPU6050 para obter os valores lidos pelos sensores e por último, os dados são enviados pela porta serial. Na porta serial está conectado o XBee, e assim, os dados escritos na porta serial são automaticamente transmitidos.

O Arduino possui uma função chamada *delay()*, ela pausa o programa pela quantidade de tempo em milissegundos passada como parâmetro. Esse projeto utiliza um *delay* de 100 ms, de tal forma que o *gateway* recebe 60 dados do Arduino para fazer uma análise, sendo uma média de 30 dados para a coxa e os outros 30 para o peito, logo tem-se uma janela de tempo de aproximadamente 3 segundos.

Os dados emitidos pelo Arduino têm o seguinte formato:

```
13140,4668,9224,-14553,-12770,5567,30.98
7572,1552,868,-900,9972,-753,30.88
21432,-1608,-2488,-10081,265,5839,31.07
17452,72,1948,-6680,-6951,-1001,31.02
10148,-2720,336,-18013,-6654,-393,30.84
13000,-1832,2148,-9931,-3716,2563,31.07
16140,484,1840,-8004,-5765,-392,31.02
14100,-4336,320,-4523,6291,1506,30.98
14620,8196,728,-998,19696,20,30.93
23332,412,-1612,-25447,4788,-6816,31.02
4420,-8464,-20,3885,3598,-2636,30.98
10996,3312,4268,11882,-15204,-1968,30.98
19912,7968,8800,-7596,14012,-257,30.98
32767,-380,3188,1795,-10435,-7712,30.93
6736,-6964,-2760,4545,-10664,-3061,31.02
8344,2152,1576,-7690,-1671,6581,30.98
19064,-5156,1052,-7682,-3696,24,31.02
14496,848,4180,-3961,-3473,1854,31.02
27616,916,-888,-5889,14259,-3673,31.07
13220,10560,-2880,-18318,12482,-11371,31.12
13412,-5836,-2252,-24870,10400,-41,30.98
7364,-5788,1928,12228,-3088,-7577,31.12
14812,10552,9572,-2977,-8858,5104,31.07
9376,4780,3336,-3289,6129,-922,31.07
22752,-5280,-2112,-5857,879,12246,30.98
17204,3736,832,13878,-11261,-5568,30.93
4988,1260,1408,-6219,-5875,3773,30.93
13892,-7016,280,8563,-1069,2330,31.02
13912,6056,2696,11046,-6092,173,31.07
14064,-2916,1908,-1452,-2050,-1063,31.07
```

Figura 28 - Dados recebidos pela porta serial do dispositivo anexado à coxa

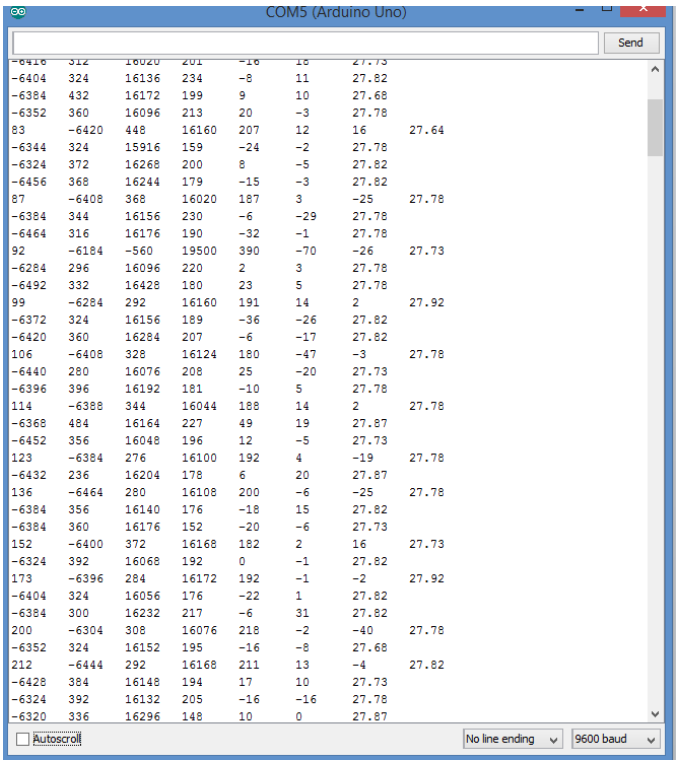

Figura 29 - Dados recebidos pela porta serial do dispositivo anexado ao peito

Há diferenças entre os dados transmitidos pelo dispositivo anexado ao peito e o anexado à coxa. O primeiro tem além do sensor MPU6050, o sensor de batimentos cardíacos, assim, mais dados são enviados em relação ao segundo. Como se pode ver na Figura 29, os valores dos batimentos não aparecem em todas as leituras, devido ao fato do Arduino enviar os dados antes da interrupção ter sido gerada. A ordem dos dados é: valor dos batimentos cardíacos, valores referentes à leitura do acelerômetro e giroscópio, nos três graus de liberdade, e o último termo é referente ao valor da temperatura.

O sensor anexado à coxa não contém o sensor de batimentos cardíacos, logo a sequência enviada tem a seguinte ordem: valores referentes à leitura do acelerômetro e giroscópio e o último termo é referente ao valor da temperatura.

Os valores recebidos pela porta serial devem ser divididos pelo valor adimensional 16384, para os dados provenientes do acelerômetro e pelo valor adimensional 131, para os dados provenientes do giroscópio. A primeira linha da Figura 28, por exemplo, apresenta os seguintes valores após fazer as devidas divisões dos valores respectivos ao acelerômetro por 16384 e dos valores respectivos ao giroscópio por 131:

```
0,8ax; 0,28ay;0.56az;-111wx;97,48wy;42,5wz;30,98°C
```

As palavras “ax,ay,az,wx,wy,wz” referem aos graus de liberdade do giroscópio e do acelerômetro. A seção 2.3 mostra que o *Pulse Sensor* contém um código em Arduino e outro código em *Processing*. Esse projeto não utiliza a IDE *Processing*.

3.1.5. Desenvolvimento do algoritmo utilizado no gateway

O RaspberryPi foi a plataforma escolhida para compor o *gateway*, que tem a função de coordenador na WSN, por ele possuir uma entrada Ethernet e um poder de processamento maior do que o Arduino, além de permitir a construção de um código na linguagem Python, que é mais simplificada para o desenvolvimento, em relação à linguagem do Arduino, que é inspirada em C e C++. Ele faz a comunicação entre um serviço que utiliza Ethernet e ZigBee.

Antes de iniciar o desenvolvimento do algoritmo que será utilizado no *gateway*, foi coletado alguns dados a fim de aferir comportamentos semelhantes entre as posições corporais. Os dados foram obtidos utilizando o monitor serial do *software* Coolterm e depois foi realizada a respectiva plotagem no *software* Matlab. A plotagem desses dados serviu de embasamento teórico para o desenvolvimento deste algoritmo e estão mostrados no capítulo 4 deste trabalho.

Primeiramente o algoritmo utilizado no gateway faz uma análise para verificar se o corpo está em movimento ou estático. O *flowchart* da Figura 30 mostra como é feita essa verificação. Se for constatado que o corpo está estático, é feita uma verificação da posição em que o corpo se encontra, representado pelo *flowchart* da Figura 34. Caso o corpo esteja em movimento, deve-se diferenciar o movimento entre queda e outros tipos de movimentos, para isso, é utilizada a análise dos *flowcharts* representados pelas Figuras 35 e 36.

A janela de tempo de análise é de 3 segundos. Esse valor foi escolhido porque ele é o mínimo necessário para que se possa fazer alguma inferência sobre o movimento. Os dados são enviados pelo Arduino a cada 100 ms e o algoritmo faz um *loop* 60 vezes, o que equivale a 6 segundos, contudo, como os dados são recebidos da coxa e do peito a janela de tempo é dividida por 2, resultando em 3 segundos. Wmax e Wmin referem-se ao valor máximo e mínimo da velocidade angular do corpo em uma janela de análise, respectivamente, Amax e Amin referem-se ao valor máximo e mínimo da aceleração do corpo em uma janela de análise, respectivamente, Δw e Δa indicam a variação média entre Wmax e Wmin e Amax e Amin, respectivamente. Esses valores foram determinados a partir da análise gráfica do movimento, como mostra o capítulo 4, nas Figuras 52, 53, 54 e 55.

A análise representada pela Figura 30 inicia recebendo os dados dos sensores, e logo após, verifica se há dados resultantes da última análise, isso é necessário porque como pode ser visto nos gráficos da seção 4.2.2, para inferir uma queda é necessário que haja um momento de pausa logo após o valor máximo da aceleração, se este ocorrer no final da janela não se pode afirmar que uma queda ocorreu, pois o momento de pausa será registrado somente pela próxima amostra dos dados, então para solucionar este problema verifica se na última análise o valor máximo do acelerômetro ocorreu após 75% da janela, ou seja, se há dados anteriores. Caso afirmativo, esses dados são adicionados aos novos vetores formados pelos dados atuais e então, com esse vetor originado dessa junção, realiza-se a análise dos primeiros 25% valores. Isso ocorre porque não se pode fazer uma análise de detecção de movimento considerando todo o comprimento da janela, pois os dados recebidos recentemente iriam causar imprecisão na decisão do movimento anterior. Caso não haja dados anteriores, a análise é realizada ao logo do vetor gerado pelos dados recebidos recentemente.

É realizado o cálculo do vetor resultante das componentes do acelerômetro e giroscópio, seleciona-se o maior e o menor valor e os compara com Δw e Δa, que foram escolhidos baseados nos gráficos da seção 4.2.1.

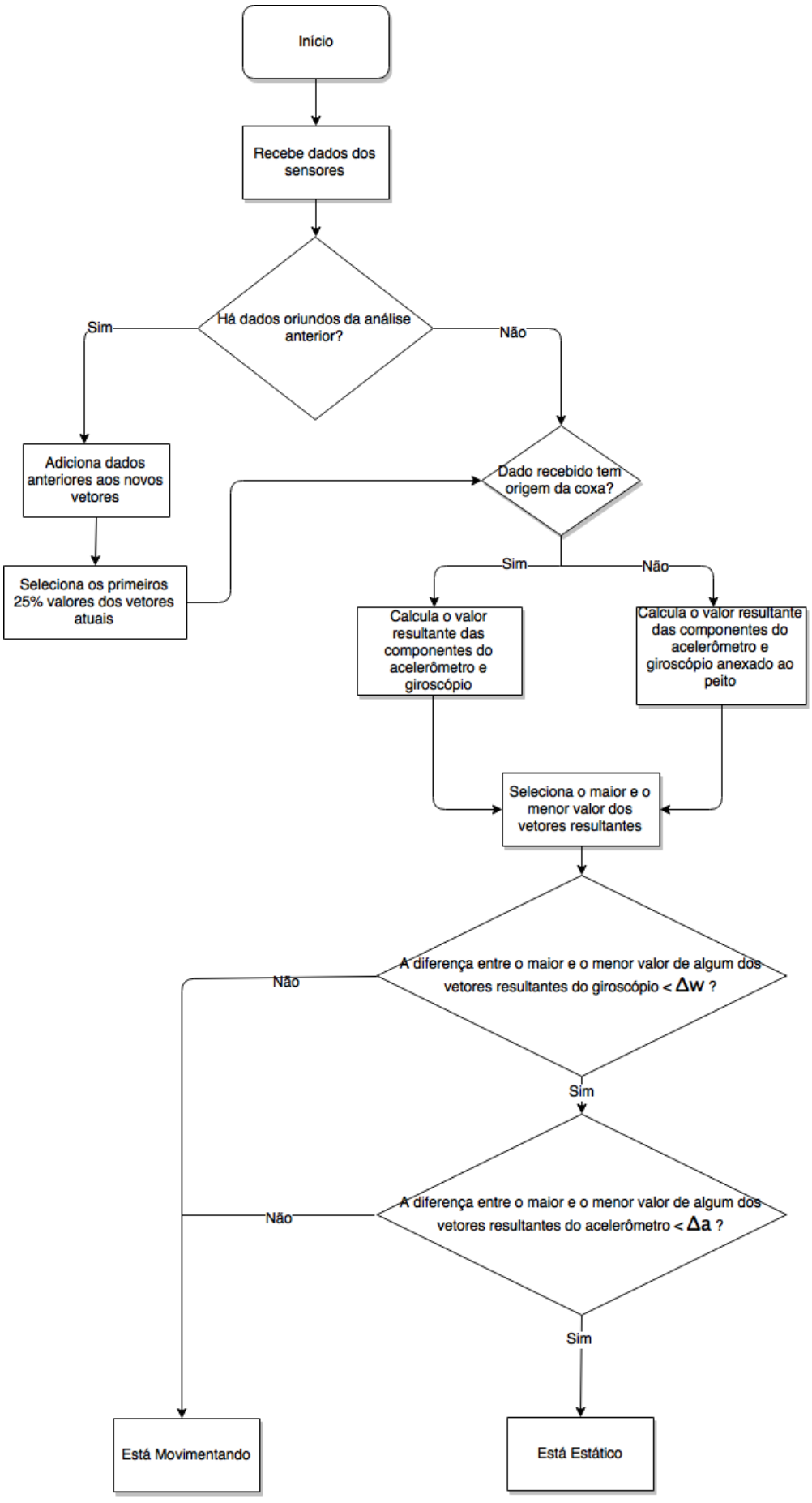

Figura 30 - Flowchart do algoritmo detecção de movimento

Se for constatado que o corpo está estático, deve-se determinar em qual posição ele se encontra. Para fazer essa inferência é feita a análise dos planos de secção do corpo humano. Como pode ser visto na Figura 5, o MPU6050 indica a orientação dos eixos coordenados utilizado pelo acelerômetro e giroscópio, e a partir disso, é realizada uma escolha dos planos de secção que podem ser vistos nas Figuras 31,32 e 33.

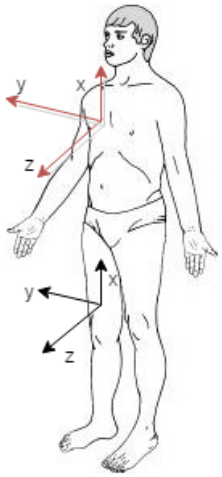

Figura 31 - Planos que atravessam o corpo quando está em pé

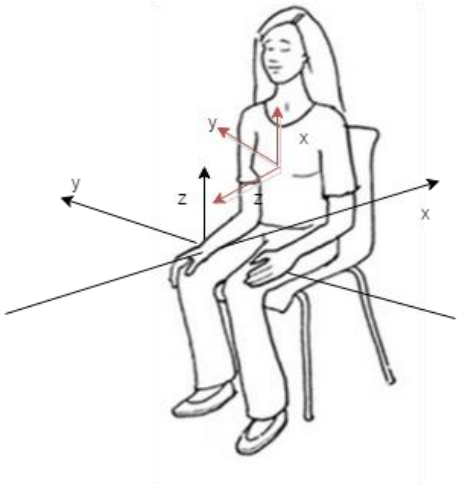

Figura 32 - Planos que atravessam o corpo quando está sentado

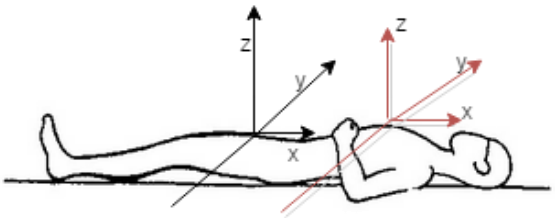

Figura 33 - Planos que atravessam o corpo quando está deitado

Para diferenciar a posição corporal determina-se o eixo que tem uma aceleração própria (difere da aceleração no sentido convencional de taxa de variação da velocidade, pois está atrelada a sensação de peso medida em um dado referencial) de aproximadamente 1g

(9,81m/s^2), na qual a unidade g indica referência em relação à aceleração da gravidade. A aceleração igual a 1g indica qual eixo está sofrendo ação da força peso (linha reta para cima). Quando o corpo está em pé, os sensores da coxa e do peito apresentam a aceleração no eixo x de aproximadamente 9,81m/s^2, enquanto que quando o corpo está sentado o eixo x, no peito, e o eixo z, na coxa, apresentam aceleração de aproximadamente 9,81m/s^2, já quando o corpo está deitado o eixo z da coxa e do peito apresenta aceleração de aproximadamente 9,81m/s^2.

Foi realizado o cálculo das médias dos valores recebidos do acelerômetro nos três eixos. Os valores registrados pelo giroscópio não foram utilizados nessa parte do algoritmo porque não há velocidade angular, pois o corpo encontra-se estático. O *flowchart* da Figura 34 mostra como é feita a detecção da posição.

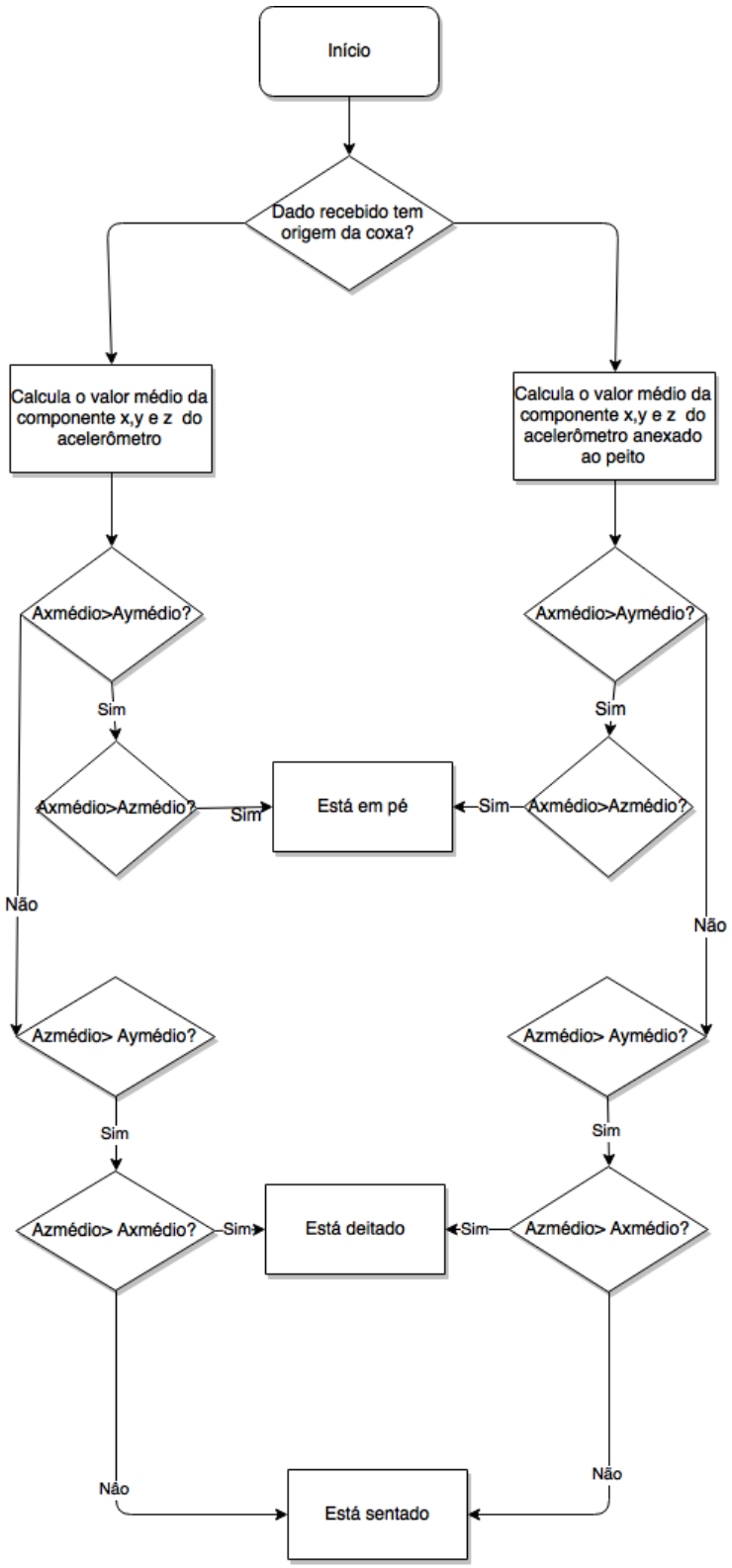

Figura 34 - Flowchart do algoritmo de detecção de posição

Se for constatado que o corpo está em movimento, é necessário diferenciar esse movimento entre quedas e atividades como andar ou correr. Para isso, foi realizado um algoritmo baseado em *thresholds* e em tempos de pausa, no qual o *flowchart* que representa valores acima do *threshold* é mostrado na Figura 35 e o que representa valores menores do que o *threshold* é mostrado na Figura 36. Para explicar esses *flowcharts* é necessário verificar

o comportamento dos dados obtidos no acelerômetro e giroscópio durante quedas. A seção 4.2.2 mostra os gráficos que apresentam o comportamento dos dados quando ocorrem quedas bruscas e quedas suaves, pode-se verificar que quando ocorrem quedas bruscas obtém-se um valor de aceleração maior do que 2,2g, que foi escolhido como AUFT (Aceleração *Upper Fall Threshold*), e valores de velocidade angular maiores do que 220°/s, que foi escolhido como WUFT(Velocidade Angular *Upper Fall Threshold*), contudo quando ocorrem quedas suaves o valor máximo da aceleração é aproximadamente 1,6g, por isso o tratamento desses valores deve ser diferente.

Para ter maior precisão na hora de decidir se ocorreu queda deve-se levar em conta outros parâmetros além do valor da aceleração e da velocidade angular. Quando uma queda ocorre geralmente o paciente permanece em posição estática por pelo menos 1 segundo, pois idosos apresentam dificuldades em levantar e pessoas que sofrem de doenças que ocasionam desmaios, por exemplo, permanecem deitadas por um período de pelo menos 1 segundo, por isso, esse aspecto também foi considerado para a realização do algoritmo.

Após determinar que o corpo está em movimento, a análise da Figura 35 é realizada. Esse *flowchart* também faz uma diferenciação se os dados são originados de uma análise anterior, caso sejam, a aceleração e velocidade angular máxima registrada anteriormente serão utilizadas para fazer a comparação entre os *thresholds*. Se os dados anteriores são maiores do que os *thresholds*, uma média da diferença entre cada um dos valores dos vetores dentro dos primeiros 25% da janela é realizada. O objetivo é verificar se ocorreu momento de pausa após a aceleração máxima na análise anterior. Caso não haja dados anteriores, a aceleração e velocidade angular máxima dos dados recebidos atualmente serão utilizadas para verificar a condição dos *threshold* e verifica-se o valor máximo da aceleração ocorreu antes dos 75% da janela, caso não tiver ocorrido, os 25% valores remanescentes são armazenados para análise posterior, se tiver ocorrido antes, é realizada uma média da diferença entre os valores dos vetores desde o *index* de onde foi determinada a aceleração máxima até o final do vetor.

Como mostra a Figura 35, o valor escolhido para fazer comparação da média no algoritmo com valores acima do *threshold* foi de 0,3g para aceleração e para o giroscópio foi de 40°/s. Geralmente a análise do *flowchart* da Figura 35 remete ao caso de quedas bruscas (valores acima do *threshold*) e o *flowchart* da Figura 36 remete ao caso de quedas suaves (valores abaixo do *threshold*).

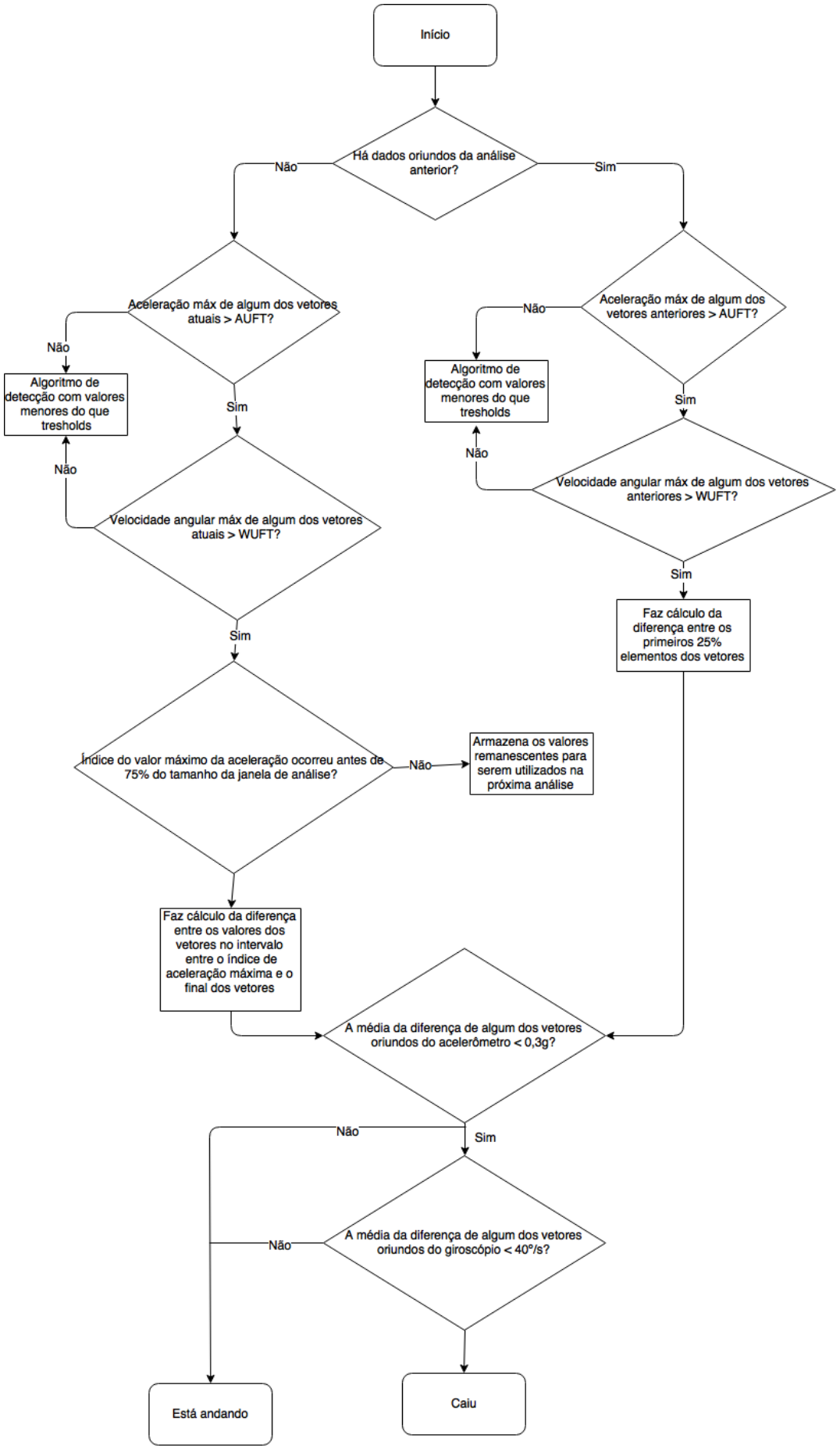

Figura 35 - Flowchart do algoritmo de detecção de queda com valores maiores do que os *thresholds*

Se os *thresholds* do *flowchart* da Figura 35 não forem satisfeitos, o programa executa a análise representada pela Figura 36. A primeira condição nessa etapa consiste em verificar se a aceleração máxima está entre 1 e AUFT. Esse valor foi escolhido porque quando o corpo cai, a aceleração máxima geralmente não é menor do que 1, criando assim o primeiro critério para detecção de queda. Caso as condições dos *thresholds* não forem satisfeitas o algoritmo determina que o corpo está andando, caso sejam satisfeitas uma análise seguinte é realizada. O WUFT para quedas suaves tem valor de 95°/s, pois após uma série de testes verificou-se que quedas geralmente não apresentam velocidades angulares menores do que 95°/s. Depois é verificado se após o pico máximo ocorreu algum momento no qual o corpo ficou estático, para fazer essa inferência observa-se se o pico ocorreu antes de 75% do tamanho da janela de tempo, ou seja, se a queda ocorreu entre 0 e 2,25 segundos, e se sim, o cálculo da média entre as diferenças dos valores em cada um dos vetores é realizado à partir do ponto em que foi detectado o pico máximo até o final da janela de análise, compara-se esse valor com um *threshold* determinado por inferência. O valor de *threshold* da média para aceleração própria diminuiu de 0,3g para 0,2g porque os valores de quedas bruscas devem ser ponderados. O mesmo ocorre com a velocidade angular, que diminuiu de 40°/s para 20°/s. Após fazer a análise com os valores dos *thresholds* da média conclui-se se ocorreu um momento de pausa.

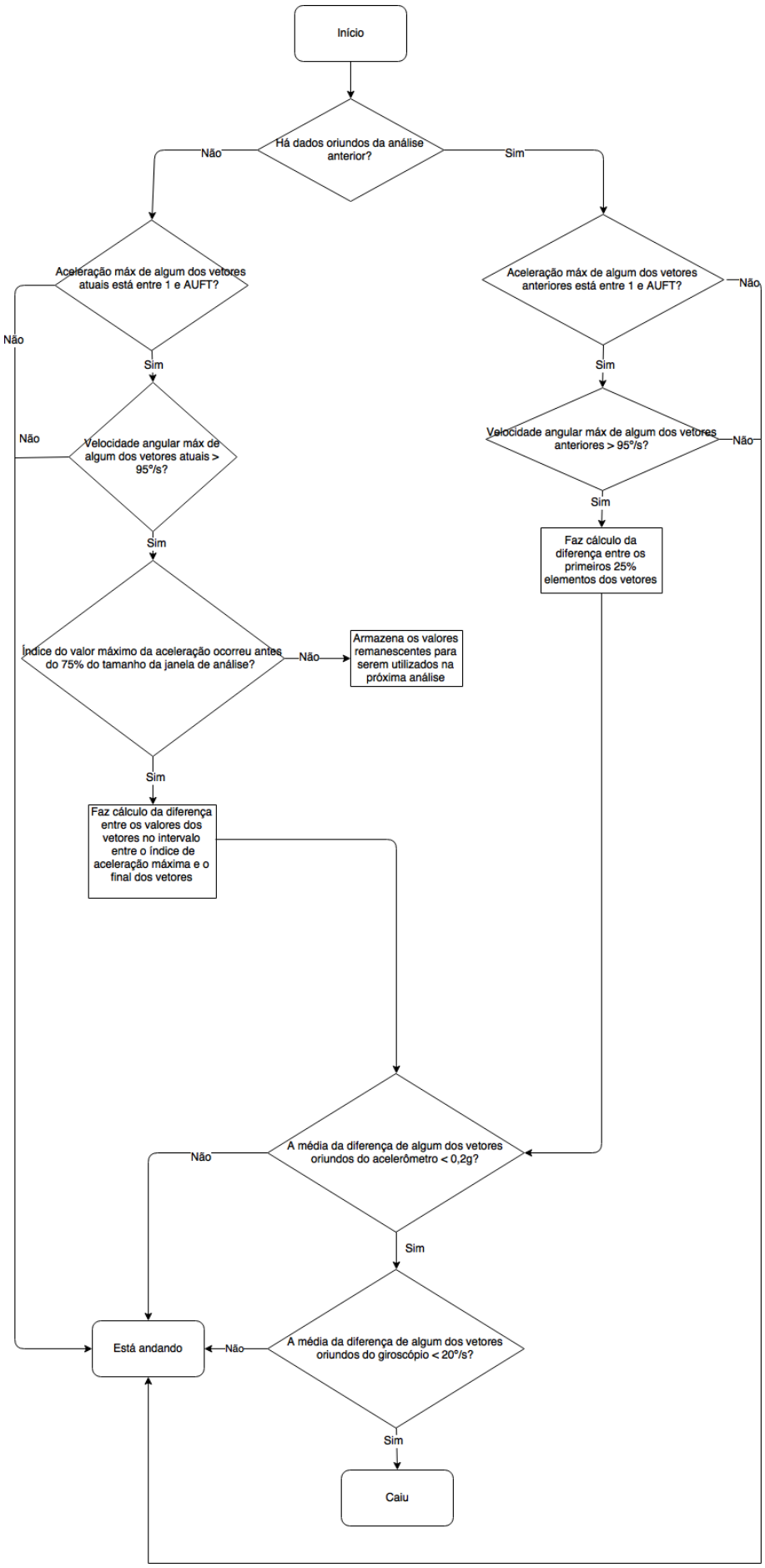

Figura 36 - Flowchart do algoritmo de detecção de queda com valores menores do que os *thresholds*

3.1.6. Inserindo dados no Banco de dados

Nesse projeto foi utilizado o banco de dados MySQL e o servidor de hospedagem Hostgator. No banco de dados, foram adicionadas quatro tabelas: quedaApp, data_th, queda e notification_queue. A tabela data_th é responsável por conter os campos posição, temperatura, hora,data, id e batimentos cardíacos, seus valores são atualizados toda vez que o programa instalado no *gateway* faz a leitura de uma posição. A tabela quedaApp contém os campos id, hora, data e posição. Quando o *gateway* detecta uma queda, ele atualiza o campo posição da tabela quedaApp para "queda" e o *software* android que está sendo executado em *background* verifica se esse campo está configurado como "queda", se sim, ele faz uma ligação telefônica e configura esse campo como "nada". O *flowchart* que representa o algoritmo do aplicativo é mostrado na seção 3.1.8. A tabela queda contém os campos id, hora e data, seus dados são inseridos toda as vezes que uma queda é detectada, isso faz com que o MySQL dispare um *trigger* de tal forma que um novo elemento na tabela notification_queue é inserido. O *Cron*, que está sendo executado a cada 2 minutos, verifica se uma linha foi inserida na tabela notification_queue, se sim, ele envia um *email* aletando a queda, mais detalhes do funcionamento desse processo é mostrada na seção 3.1.7.

O diagrama de classe da Figura 37 foi realizado utilizando o *software* Astah e mostra como ocorrem os relacionamentos entre as classes.

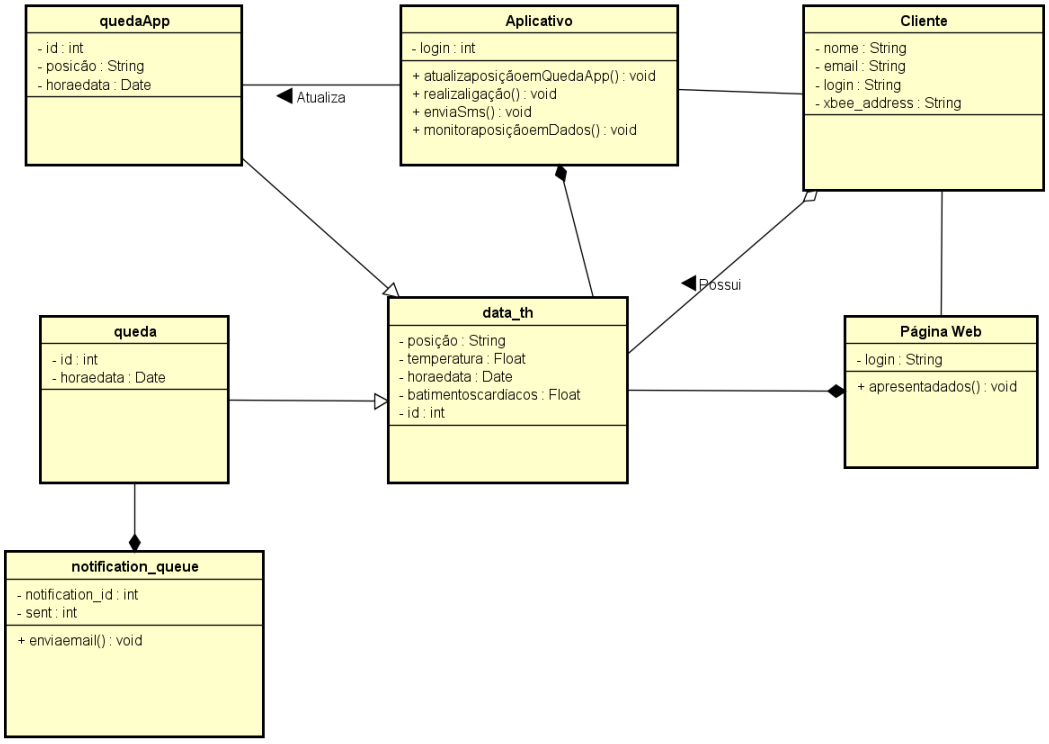

Figura 37 - Diagrama de Classe

A classe data_th tem forte relação com as demais. A relação entre quedaApp e data_th e entre as classes queda e data_th é de especialização, ou seja, existem apenas para o caso do campo posição em data_th ser “queda”. O relacionamento entre as classes notification _queue e queda é de composição, pois se a última não for atualizada, o trigger não será disparado e a primeira não irá existir. O mesmo acontece com os relacionamentos entre data_th e Aplicativo e entre data_th e Página Web. A classe Cliente faz uma associação com as classes Aplicativo e Página Web, mostrando a interação do mesmo com estas.

A Figura 38 mostra as tabelas utilizadas nesse programa. O *phpMyAdmin* foi utilizado para fazer a administração do MySQL por ser uma ferramenta fornecida pelo servidor de hospedagem.

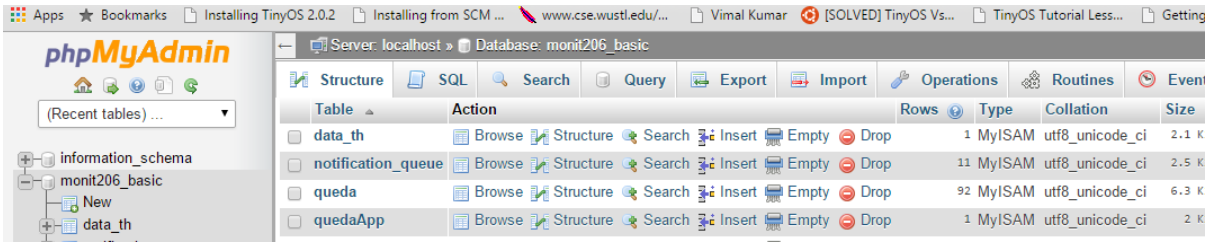

Figura 38 - Exemplo de tabelas do Gerenciador de Banco de dados

3.1.7. Desenvolvimento da Plataforma Web

A plataforma Web foi construída com o objetivo de fornecer monitoramento remoto para o usuário. As linguagens de programação utilizadas para seu desenvolvimento foram: PHP, HTML e CSS.

A página web contém um sistema de login e seção de tal forma que apenas as informações relativas ao usuário logado serão exibidas. Outra atividade realizada pelo programa é enviar automaticamente um *email* de alerta quando ocorre uma queda. Este *email* está devidamente cadastrado no banco de dados, junto com outras informações do usuário. O envio desse alerta é realizado através do serviço *Cron Job*, que são tarefas executadas em determinado período de tempo. O *Cron* executa a cada 2 minutos um *script* PHP, chamado Cron.php, o qual verifica se o campo *sent* da tabela Notification_queue é igual a ‘0’, se for, isso indica que uma linha da tabela queda foi inserida, logo ele envia um *email* notificando que uma queda ocorreu e configura o campo *sent* para ‘1’ e assim. Um dado é inserido na tabela queda quando o *gateway* detecta queda e isso faz com que um *trigger* MySQL seja disparado, adicionando uma linha na tabela notification_queue, com valor de *sent* igual a ‘0’.

Um servidor remoto de hospedagem foi utilizado e o link para ter acesso ao website, ao longo do desenvolvimento deste trabalho foi: http://108.179.252.198/~monit206/. Escolheu-se um servidor remoto em vez de um servidor local porque o acesso ao banco de dados realizado pelo dispositivo android a um servidor remoto evita erros que foram encontrados no momento da compilação do programa, como será comentado na seção 3.1.8.

3.1.8. Desenvolvimento do aplicativo Android

Um aplicativo android foi desenvolvido para que o usuário tenha mais acesso e conhecimento sobre o estado do paciente. Este foi desenvolvido utilizando a ferramenta *Basic4Adroid.*

O aplicativo funciona em *background*, utilizando a ferramenta *Service* do B4A, ou seja, fica constantemente em funcionamento e se for detectado que o campo posição da tabela quedaApp está como “queda”, automaticamente uma ligação telefônica é realizada para o número registrado e se a mesma for perdida, o aplicativo liga para um segundo número. Depois que o alerta foi feito, ele configura o campo posição da tabela como “nada”, evitando assim que a ligação volte a acontecer desnecessariamente. Dessa forma, diminui-se o risco do cuidador não ser informado sobre a queda. Outro sistema de alerta que o aplicativo possui é detectar a elevação do batimento cardíaco, se o mesmo ultrapassar um *threshold* (150 BPM), ele envia um *sms* avisando que está alto. O aplicativo faz a ligação e o envio do *sms* através da biblioteca *Phone* do B4A, utilizando as funções *PhoneSms() e Call()*.

Os batimentos cardíacos são comparados com valores retirados da Tabela 3-3, que apresenta a frequência cardíaca média em repouso para mulheres acima de 65 anos, o valor e estado do batimento por minuto é mostrado no aplicativo, indicando se está excelente, bom, na média ou alto. Os valores dos batimentos variam de acordo com a idade, sexo, frequência de atividade física entre outros fatores, contudo, para o escopo desse projeto, foi utilizado apenas o critério idade e sexo. Se a frequência está maior do que 150 BPM o aplicativo envia um alerta.

Tabela 3-3 Frequência cardíaca média de uma mulher acima de 65 anos, adaptado de [6]

Estado dos batimentos	Batimentos por Minuto (BPM)
Atleta	54-59
Excelente	60-64
Bom	65-68
Acima da média	69-72
Média	73-76
Abaixo da média	77-84
Pobre	84+

Além do valor e estado dos batimentos cardíacos, o aplicativo também fornece outras informações como temperatura ambiente, o endereço do dispositivo utilizado, um botão de emergência e apresenta a posição do paciente de uma forma interativa. O *flowchart* que representa o algoritmo do aplicativo é mostrado na Figura 39.

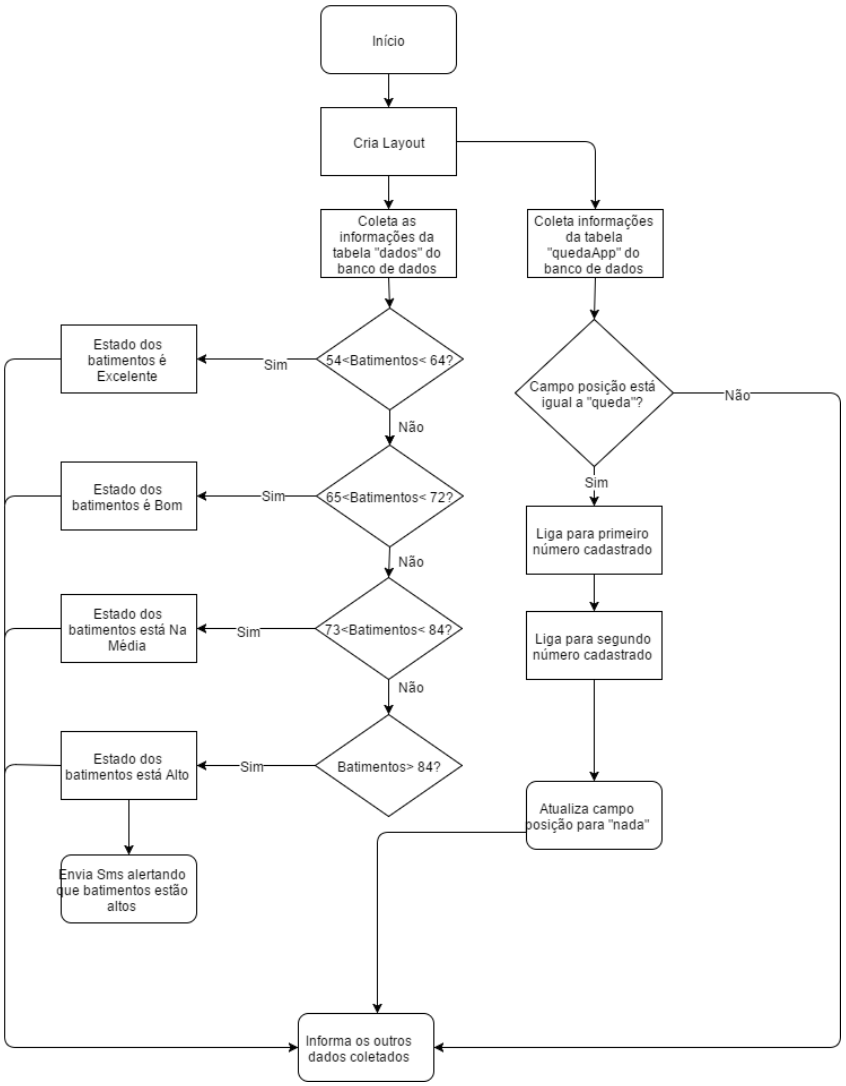

Figura 39 - Flowchart do Algoritmo Android

O botão de emergência foi adicionado para o caso de pacientes que estejam precisando de ajuda, e não conseguem fazer uma ligação telefônica ou falar. Esse botão faz imediatamente uma ligação para o número do cuidador.

A justificativa sobre o fato de ter-se optado por utilizar um servidor remoto é que na primeira fase de desenvolvimento do aplicativo Android tentou-se utilizar um servidor local, contudo o programa apresentou erros, informando que não conseguia se conectar ao servidor. Para solucionar esse problema, foi utilizado um servido remoto privado com IP fixo chamado Hostgator.

3.2 Estudo de Caso

O Estudo de Caso procura uma forma de investigar o uso do dispositivo em um contexto real. Os testes foram realizados por uma pessoa do sexo feminino, 24 anos, 1,63m de altura e 55 quilos. Foi realizada uma medição de 10 vezes em cada posição. A arquitetura utilizada para realizar os testes é mostrada na Figura 40.

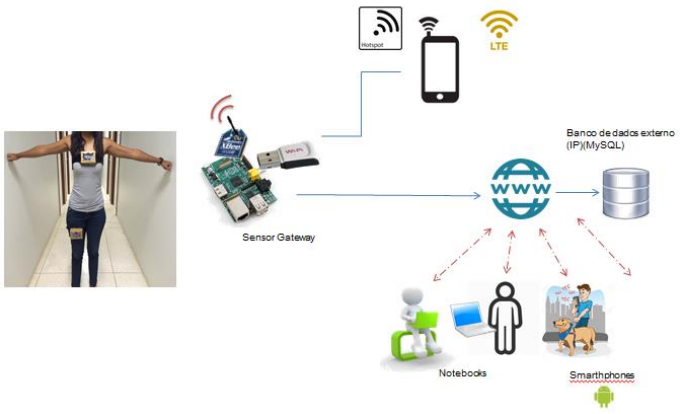

Figura 40 - Arquitetura utilizada no estudo de caso

A arquitetura da Figura 40 foi utilizada em vez da arquitetura normal do programa, mostrada na Figura 22, porque no ambiente de coleta dos dados não havia saída Ethernet, então foi necessário criar uma rede *Wi-Fi* utilizando o celular como *Hotspot*. A diferença entre posições estáticas e dinâmicas foram detectadas em 100% dos casos, contudo quando uma posição dinâmica é detectada, erros quanto à determinação se o corpo está em queda ou se está andando ocorrem.

Um dos erros apresentados pelo dispositivo é quando ocorrem quedas e o sistema afirma que o corpo está andando, nesse caso o sistema apresenta falsos negativos. O outro tipo de erro é quando o corpo está andando e ele afirma que ocorreu queda, nesse caso o sistema apresenta falsos positivos. A Figura 41 apresenta os histogramas dos dados obtidos quando foi

realizado testes com o equipamento. O sistema detectou corretamente 60% das quedas e 90% dos casos quando o corpo está andando. Este último apresenta maior ocorrência porque como pode ser verificada no flowchart da Figura 36, uma maior quantidade de fatores indicam que o corpo está andando.

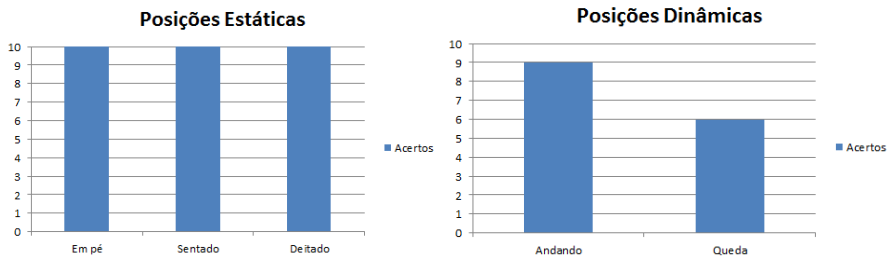

Figura 41 - Acurácia do sistema

Outra análise realizada foi diferenciar os resultados quando os dados são recebidos via serial e via sem fio. Após alguns testes, verifica-se que ocorrem alguns erros devido a comunicação entre XBees e pela leitura errônea dos sensores. Os possíveis erros são: rajada de dados, dados faltando ou a não leitura dos sensores. O algoritmo de detecção de quedas contém algumas táticas para tratar esses eventuais erros, que serão explicadas ao longo desta seção.

A ordem dos dados recebidos do dispositivo coxa é: valores obtidos da leitura do acelerômetro e giroscópio e valor da temperatura, logo o tamanho da *string* recebida deve ser igual a 7 (3 dados referente à leitura do acelerômetro, 3 referentes ao giroscópio e 1 de temperatura). Esse dispositivo não contém o sensor de batimentos cardíacos. O endereço do XBee anexado à coxa é: 0013A20040AF8733.

Caso os dados sejam recebidos pelo sensor anexado à coxa, o código utiliza um sistema para tratamento de erros, o qual consiste em ignorar a leitura quando a *string* tem tamanho diferente 7 ou 14. Quando a *string* tem tamanho igual a 14, dois dados são enviados ao mesmo tempo, então para resolver esse problema, apenas os valores referentes ao primeiro dado é armazenado. Em uma comunicação sem erros, é recebida uma *string* de tamanho 7, como mostra a Figura 42. A Figura 43 mostra o caso em que dois dados são enviados ao mesmo tempo (*string* de tamanho 14).

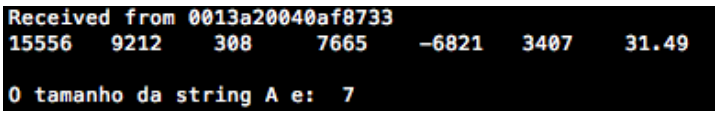

Figura 42 - Dados recebidos do dispositivo anexado à coxa (*string* de tamanho 7)

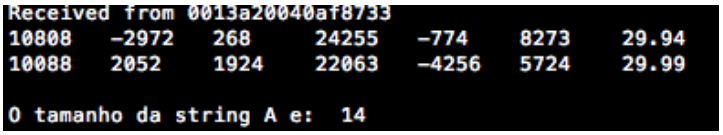

Figura 43 - Dados recebidos do dispositivo anexado à coxa (*string* de tamanho 14)

O endereço do XBee anexado ao peito é 0013A20040AC216B. As Figuras 44 e 45 mostram um *screenshoot* dos dados recebidos com erro pelo *gateway*, originados do sensor anexado ao peito, como se pode ver, a rajada de dados recebidos pode conter componentes faltando ou componentes em excesso. Analisando a Figura 44, verifica-se que há dados faltando depois do sinal "-", já no caso da Figura 45 a string tem tamanho igual a 17 e pode-se ver que há alguns dados que deveriam ser enviados na comunicação e há outros dados em excesso (3 dados foram enviados aos mesmo tempo, mas apenas 1 foi enviado sem erro). O ideal é receber uma *string* de tamanho 8 do dispositivo peito, que apresenta o primeiro termo como o valor do batimento cardíaco. A ordem de dados recebidos do dispositivo peito é a mesma de quando eles foram enviados pela porta serial, como é mostrada na Figura 29. A Figura 46 mostra os valores recebidos do sensor anexado ao peito sem erro na comunicação.

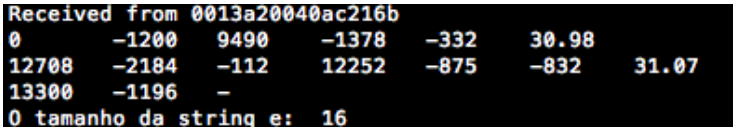

Figura 44 - Dados recebidos do dispositivo anexado ao peito (string de tamanho 16)

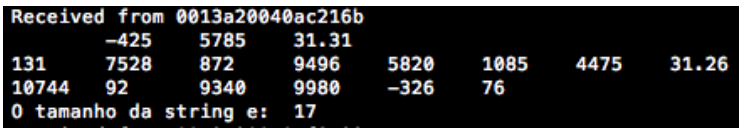

Figura 45 - Dados recebidos do dispositivo anexado ao peito (string de tamanho 17)

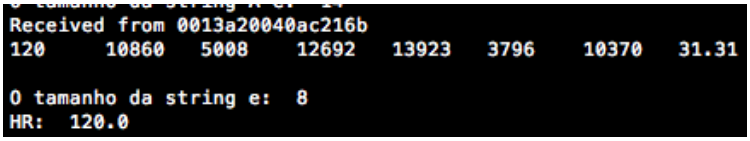

Figura 46 - Dados recebidos do dispositivo anexado ao peito (string de tamanho 8)

O símbolo HR da Figura 46 indica *Heart Rate*, que é o valor do batimento cardíaco. Nem todos os dados recebidos pelo dispositivo anexado ao peito contém o valor do batimento cardíaco, pois às vezes quando o dado é enviado, o Arduino ainda não gerou a interrupção necessária para fazer a leitura do sensor de batimentos, então o algoritmo realiza um processo de tratamento de erros, na qual consiste em aguardar o próximo dado recebido que apresenta o valor dos batimentos e realiza os devidos cálculos. Essa situação é mostrada na Figura 47, onde a *string* tem tamanho igual a 7.

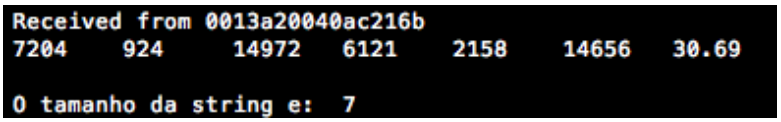

Figura 47 - Dados recebidos pelo dispositivo anexado ao peito sem o valor do batimento cardíaco

O MPU6050 é bastante sensível à variação de tensão, por isso, às vezes, quando uma queda ocorre ele faz uma leitura apenas da temperatura; velocidade angular e aceleração não são computadas. Um exemplo desse erro é mostrado na Figura 48. A única solução encontrada foi reiniciar o equipamento e aguardar a tensão estabilizar.

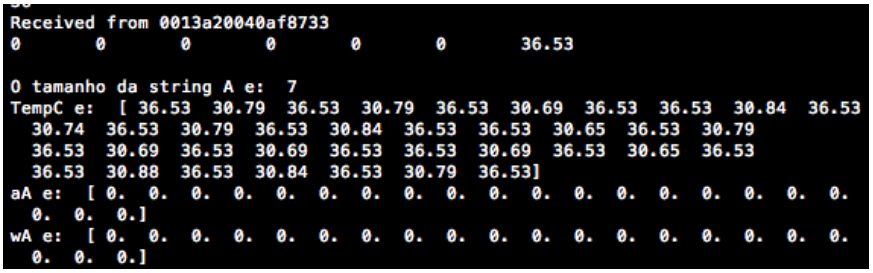

Figura 48 - Dados recebidos com erro devido à variação de tensão

De acordo com a Figura 48, a leitura referente aos valores da temperatura é computada, mas os valores do acelerômetro e giroscópio aparecem como uma sequência de zeros.

A detecção da posição com o corpo estático ocorre 100% do tempo, pois a orientação do corpo em posição estática não gera incertezas e o fenômeno deixa de ser estatístico e passa a ser determinístico.

Movimentos dinâmicos, como andar e quedas, são aleatórios. Esse projeto trabalha com padrões, se algo ocorrer fora do padrão a queda não será detectada ou será detectada erroneamente. Isso faz com que o algoritmo apresente falhas. Falsos positivos são fenômenos conhecidos por algoritmos que detectam queda, pois ocorre um dilema, se eleva o *threshold*, mesmo em casas decimais, irá aumentar o número de falsos alarmes, se diminui-se o *threshold*, quedas podem não ser detectadas, então encontrar os valores que irão fornecer exatidão em 100% dos casos ainda não é possível, pois há várias atividades realizadas por uma pessoa que assemelham a uma queda, por exemplo, levantar rápido de uma cadeira ou

andar em maior velocidade. O projetista deve adicionar outros parâmetros para auxiliar a decisão.

Alguns estudos têm sido feitos para determinar diferentes parâmetros, como em [2], que determina se o movimento foi intencional ou não intencional, já o estudo em [7] utiliza vários valores de *threshold* para ser comparado. Bourke, A., O'Brien, J., & Lyons G.[8] utiliza a análise da queda no momento de início, sua velocidade, o impacto e o monitoramento da postura. Ele utiliza com parâmetro o comportamento padrão de uma queda, mostrada na Figura 49.

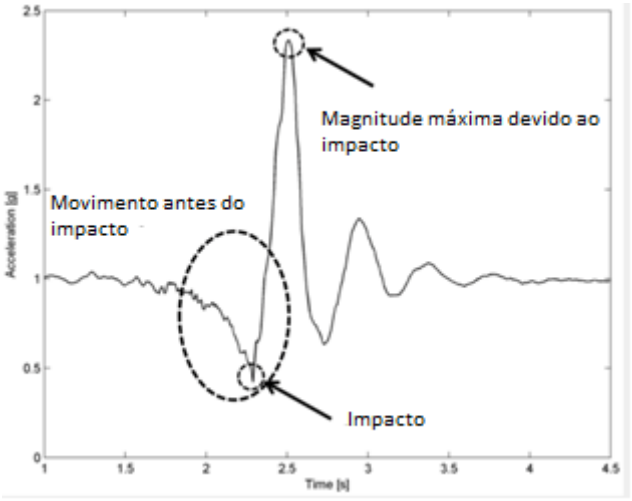

Figura 49 - Aceleração de uma queda típica, adaptado de [8]

De acordo com o estudo em [8] e a Figura 49 pode-se ver que a queda segue alguns padrões de comportamento, como ter um momento de pequeno valor na aceleração e logo depois um aumento abrupto na magnitude. Quando a magnitude da aceleração varia entre um valor muito baixo para outro muito alto em menos de 0.5 segundos pode-se dizer que é o impacto do corpo caindo no chão, entretanto há casos em que o menor valor da aceleração não ocorre exatamente antes do maior. Esse parâmetro foi considerado para a construção do algoritmo desse projeto, após vários testes essa solução se mostrou pouco eficiente, pois apresentou grande número de falsos negativos.

A Figura 50 mostra o *screenshot* do aplicativo quando o dispositivo realizou um falso negativo, afirmando que o corpo estava andando, enquanto uma queda havia ocorrido.

Figura 50 - Falso Negativo

4. Resultados

4.1. Diagrama de Caso de Uso

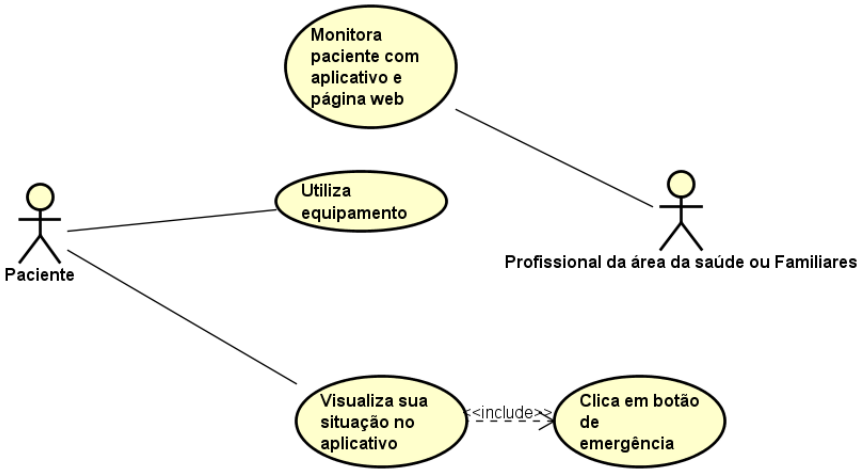

Figura 51 - Diagrama de caso de Uso

A Figura 51 apresenta o diagrama de caso de uso desenvolvido utilizando o *software* Astah. Esse diagrama descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. Os atores desse sistema são: paciente e o profissional da área da saúde ou familiares.

O paciente pode utilizar o equipamento (peito e coxa) e conferir sua atual situação no aplicativo Android, se ele estiver em uma situação de risco e não conseguir se comunicar então ele poderá clicar no botão de emergência. O caso de uso “Clica em botão de emergência” está incluso no caso de uso “Visualiza sua situação no aplicativo”. O paciente não poderá inserir nenhuma informação ao sistema. O equipamento automaticamente irá inserir as informações à medida que as leituras dos sensores forem feitas.

O profissional da área de saúde ou familiares estarão frequentemente monitorando a situação do paciente na página web ou no aplicativo. Ele não poderá inserir nenhuma informação no sistema, além de logar com seus dados, fazendo com que as informações relacionadas ao seu paciente sejam recuperadas do banco de dados.

4.2. Gráficos utilizados como embasamento para construção do algoritmo utilizado pelo Gateway

Os dados foram obtidos utilizando o monitor serial do *software Coolterm* e depois foi realizada a respectiva plotagem no *software Matlab*.

As quedas geralmente são simuladas utilizando um colchão, nesse projeto, a queda foi realizada no chão para os dados terem maior precisão. O indivíduo que realizou as quedas é do sexo feminino, 24 anos, 1,63 metros de altura e 55 quilos, esses dados são importantes porque a velocidade que a pessoa se desloca varia com o peso e com a idade e a aceleração própria na hora da queda também varia com esses dados. Cada experimento foi realizado três vezes.

4.2.1 Gráficos EstáticoxMovimento

As Figuras 52, 53,54 e 555 mostram os valores obtidos do acelerômetro e giroscópio para as posições estáticas e para o corpo em movimento. Quando o corpo está estático o Δa (variação entre o maior e o menor valor da aceleração) e Δw (variação entre o maior e o menor valor da velocidade angular) sofrem pouca variação se comparado com o corpo em movimento, neste caso, andando. Essa variação foi utilizada como base para o algoritmo da Figura 30. O *threshold* utilizado pelo algoritmo foi Δa igual a 0.3g e Δw igual a 30°/s.

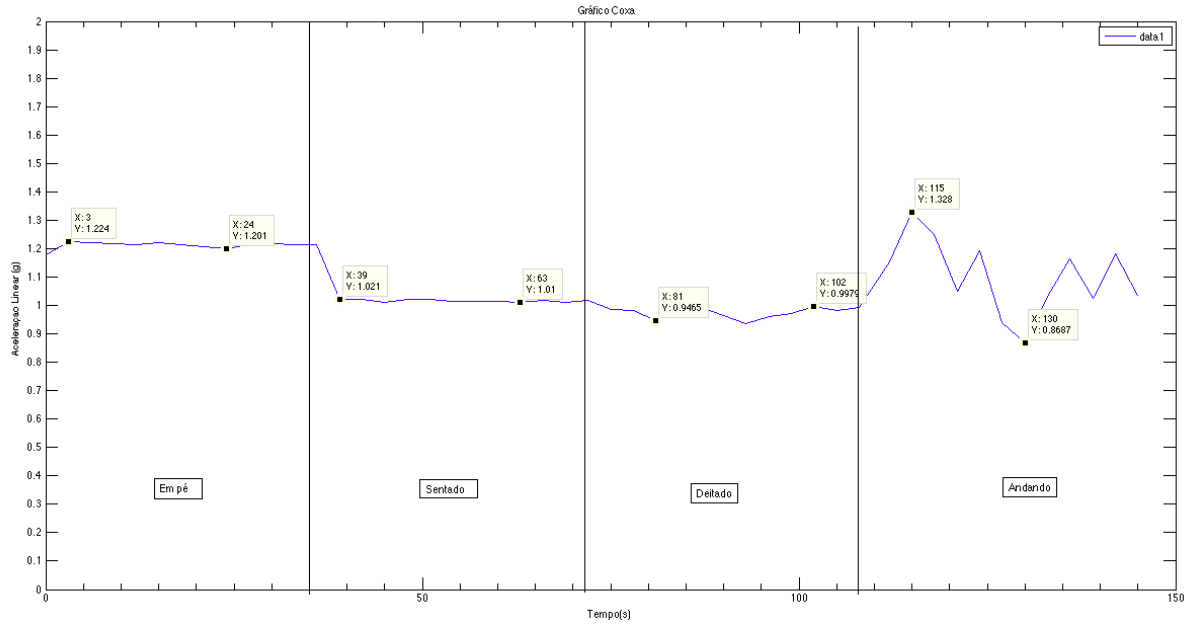

Figura 52 - Diferença de aceleração com corpo em posição estática e em movimento com sensor instalado na coxa

O gráfico da Figura 52 mostra o valor registrado pelo acelerômetro anexado à coxa para diferentes posições. Pode-se verificar que o maior valor registrado para aceleração é de 1,328g e o menor valor é aproximadamente 0,8687g, logo Δa neste caso é aproximadamente 0,459g. Também é possível verificar que quando a posição está estática nenhum valor de Δa é maior do que 0,3g.

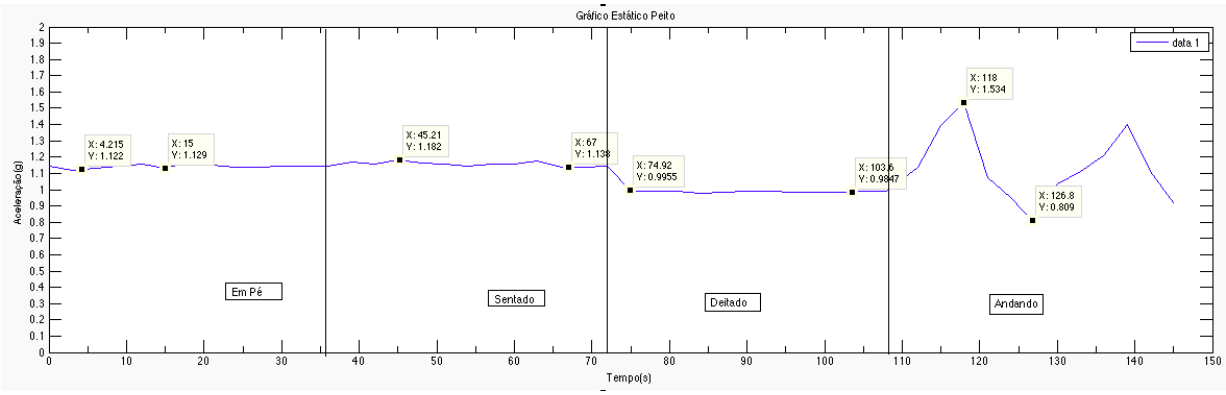

Figura 53 - Diferença de aceleração com corpo em posição estática e em movimento com sensor instalado no peito

O gráfico da Figura 53 mostra o mesmo cenário da Figura 52, contudo para o sensor instalado no peito, nesse caso o maior valor registrado para aceleração é de 1,534g e o menor valor é 0,809g, logo Δa é aproximadamente 0,725g. Como no caso anterior, para posições estáticas nenhum valor de Δa é maior do que 0,3g.

De acordo com os gráficos das Figuras 52 e 53, para posições estáticas a disposição das acelerações está em ordem decrescente, ou seja, o corpo quando está em pé tem aceleração maior do que quando está sentado, e este por sua vez, é maior do que o corpo deitado. Para explicar esse acontecimento, deve-se retornar aos conceitos de centro de gravidade e à segunda Lei de Newton (Princípio Fundamental da Dinâmica).

Centro de gravidade é o ponto no qual se pode considerar concentrado todo o peso do corpo. Isso implica que posturas diferentes da mesma pessoa e pessoas diferentes têm como centros de gravidade lugares diferentes. Ele está no ponto de interseção dos três planos cardinais do corpo: sagital, frontal e transverso. Nele está localizada a força de gravidade, que é a soma de todas as forças aplicadas aos constituintes do corpo. Quando o corpo está em pé geralmente o centro de gravidade é o ponto no qual, a partir dos pés, o corpo atinge metade de seu peso, geralmente está localizado no umbigo. Se o corpo está deitado, o centro de gravidade se localiza a partir do ponto sobre a superfície horizontal até o ponto em que é a metade do peso. Quando o corpo sentado, o centro de gravidade não está mais próximo ao umbigo porque as pernas não são contabilizadas no cálculo, ele se localiza na proximidade do peito. Para exemplificar, considere uma pessoa de 1,8 metros de altura, quando ele está em pé, o centro de gravidade deve estar aproximadamente a 1,0m; se está deitado, o centro está aproximadamente a 0,15 m; se está sentado, aproximadamente 0,8m. Todos esses valores são em relação ao chão.

A segunda Lei de Newton diz que a força resultante aplicada em um determinado corpo é igual ao produto da massa deste corpo pela sua aceleração adquirida. Assim, é possível relacionar a força resultante exercida em determinado corpo com a aceleração sofrida por este. Aplicando o princípio fundamental da dinâmica e considerando que a aceleração de um corpo em repouso é igual à aceleração da gravidade temos:

$$g = \frac{P}{m}$$

A força peso e o valor da massa do corpo são grandezas diretamente proporcionais, e assim quanto maior o centro de massa do corpo humano, em relação ao chão, maior é a intensidade da força peso sobre o corpo, e consequentemente, maior é a aceleração. Por isso, quando o corpo está em pé há uma aceleração maior, do que quando está sentado, e este por sua vez, é maior do que quando está deitado. Essa diferença pode ser verificada nas Figuras 52 e 53. As Figuras 54 e 55 representam os valores obtidos do giroscópio para o sensor anexado à coxa e ao peito, respectivamente.

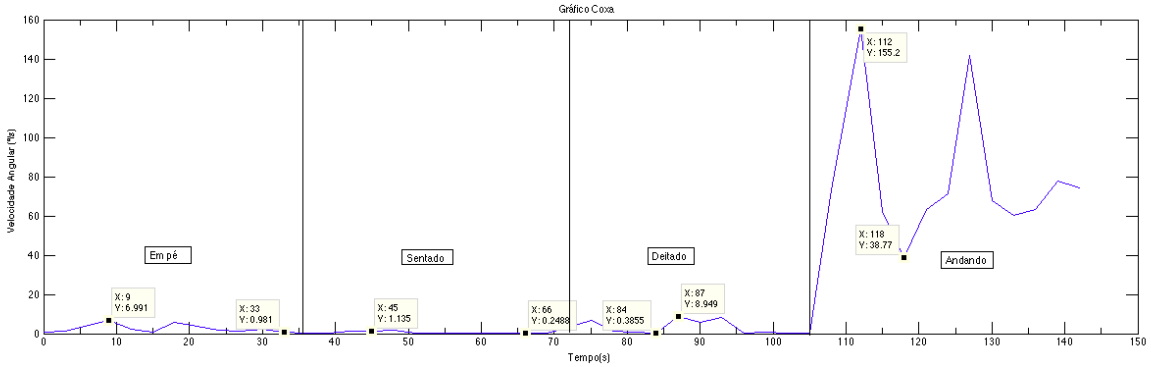

Figura 54 - Diferença de velocidade angular com corpo em posição estática e em movimento (Coxa)

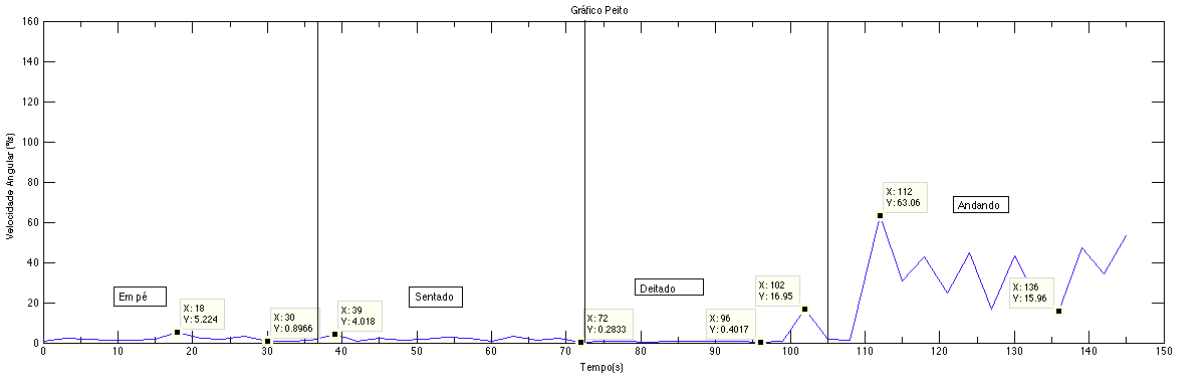

Figura 55 - Diferença de velocidade angular com corpo em posição estática e em movimento (Peito)

De acordo com as Figuras 54 e 55, a diferença máxima entre o maior e o menor valor das velocidades angulares (Δw) nos dois gráficos para o corpo estático foi de 16,55 °/s e para o corpo em movimento foi de 76,43°/s. O menor Δw encontrado para o corpo estático foi de 0,89°/s e para o corpo em movimento foi de 47,11°/s. A escolha mais adequada para o valor de *threshold*, baseada no gráfico, para detectar se o corpo está em movimento ou estático é de 30°/s, pois com esse valor, ocorreria detecção correta das posições e ainda haveria um intervalo de incerteza caso a diferença entre as posições estáticas superasse 16,55°/s.

Quando o corpo está estático não há mudança na orientação do eixo corporal, logo a taxa de variação tende a ser nula, por isso, os valores da velocidade angular permanecem constantes próximo de zero, quando o corpo está em movimento há uma constante mudança da orientação, aumentando a taxa de variação e consequentemente a velocidade angular.

4.2.2 Gráficos AndandoxQueda

Após ser constatado que o corpo está em movimento é necessário diferenciar se este está em queda ou se está realizando outro tipo de movimento. Para fazer a coleta dos dados, foram realizados experimentos com três quedas, no qual o indivíduo caminhava e caia. A queda também pode se diferenciar entre brusca e suave, os gráficos das Figuras 56,57,58 e 59 representam quedas bruscas, enquanto que as Figuras 60 e 61 representam quedas suaves.

Como pode ser verificado, quedas bruscas são mais fáceis de serem identificadas do que quedas suaves, pois em quedas bruscas ocorre um pico na aceleração, em torno de 2g e em seguida, ocorre um momento de desaceleração e o valor remanescente permanece estático e aproximadamente 1g. Os valores da velocidade angular, quando ocorrem quedas bruscas geralmente são maiores do que em quedas suaves.

Quando o corpo esta andando a aceleração geralmente não tem valores maiores do que 1,5g, enquanto que, em movimento de queda brusca os valores normalmente ultrapassam 2g, facilitando a identificação de queda. Além disso, quando o corpo está em movimento não há um momento em que a aceleração fica aproximadamente constante. Todas essas informações podem ser verificas nas Figuras 56 e 57. Os fenômenos trabalhados neste projeto são aleatórios, então se pode ter diferentes valores de aceleração e velocidade angular para a mesma atividade executada, por isso, os valores da queda brusca 1 e 2 são diferentes.

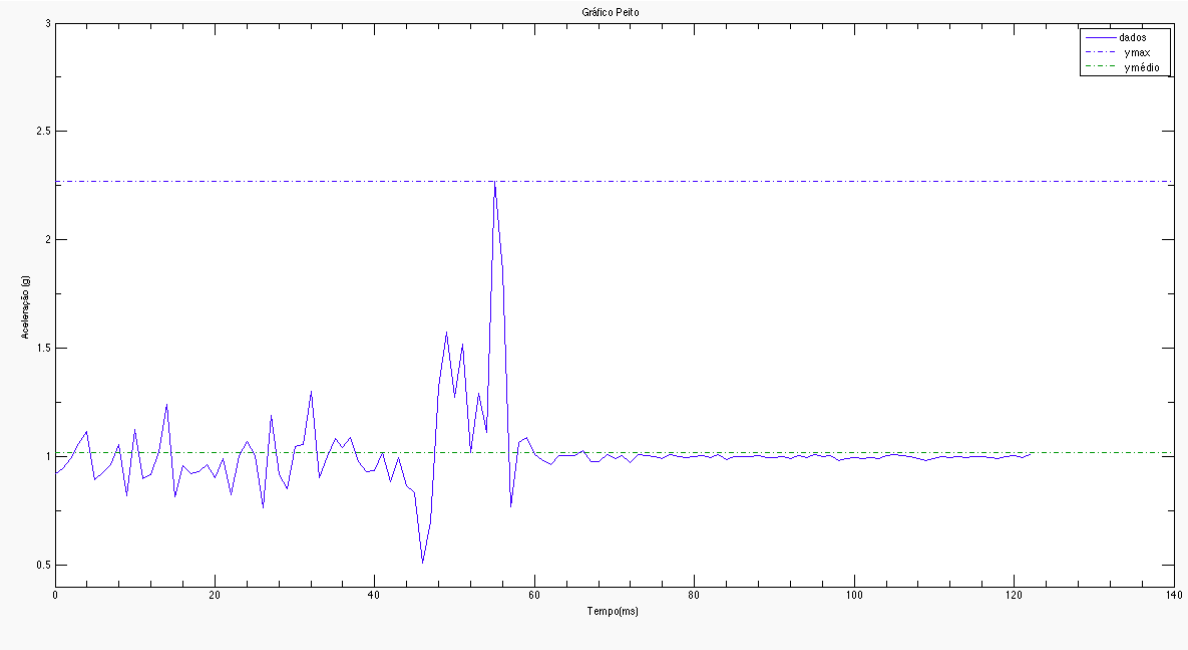

Figura 56 - Aceleração do corpo andando e em queda brusca 1 (Peito)

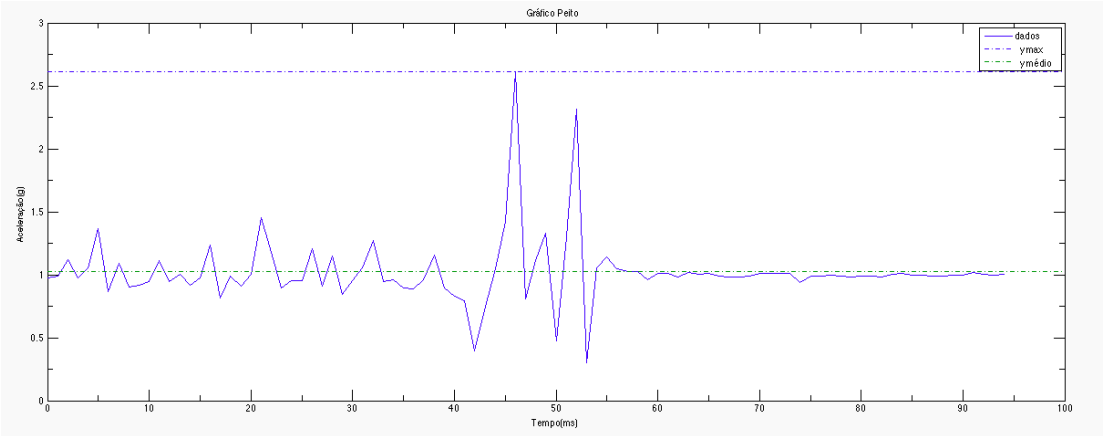

Figura 57 - Aceleração do corpo andando e em queda brusca 2 (Peito)

Também é possível identificar a queda ao analisar a velocidade angular. Quando o corpo em movimento, a velocidade angular fica variando devido à mudança da orientação corporal. Quando uma queda ocorre, há uma rápida mudança na orientação e isso eleva a taxa de variação da orientação, o que consequentemente eleva a velocidade angular, por isso existem os picos observados nas Figuras 58 e 59.

Como esse projeto trabalha com *thresholds*, é necessário escolher um valor de velocidade angular que fornece a menor quantidade de erro possível para diferenciar os tipos de movimentos. Essa escolha é bastante difícil, pois como se pode observar na Figura 59, a velocidade angular quando o corpo está em movimento pode ser bastante similar àquela da queda, com diferença de apenas 100°/s, o que é muito pouco se for considerar um fenômeno aleatório. Nesse trabalho utiliza-se o *threshold* de velocidade angular com o valor de 220°/s para o movimento ser identificado como queda, pois como ser visto nas Figuras 58 e 59, este acontece geralmente apenas quando a velocidade angular é acima de 200°/s, com objetivo de aumentar o intervalo de detecção esse valor foi aumentado de 20°/s.

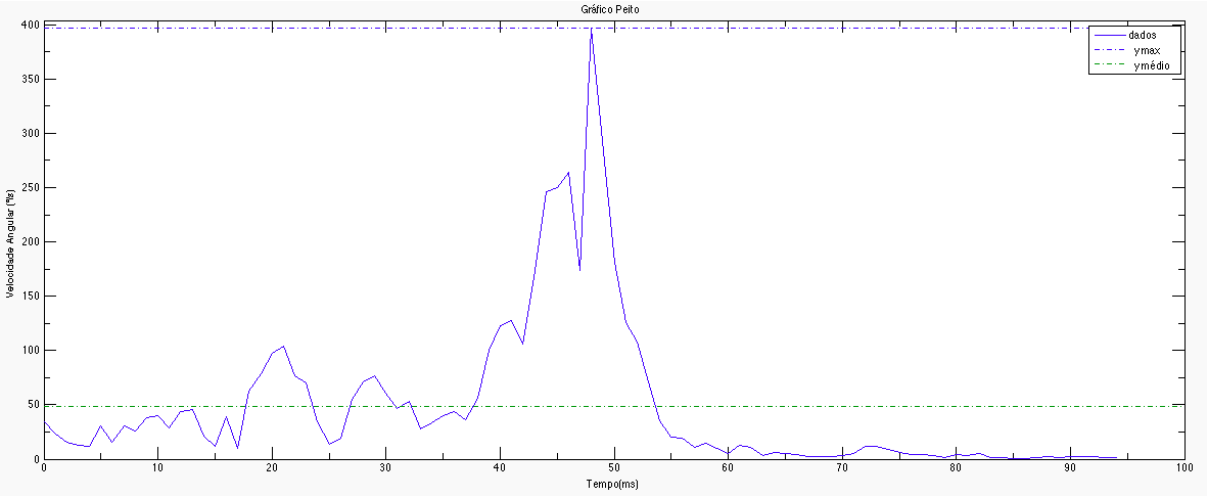

Figura 58 - Velocidade angular do corpo andando e em queda brusca 1 (Peito)

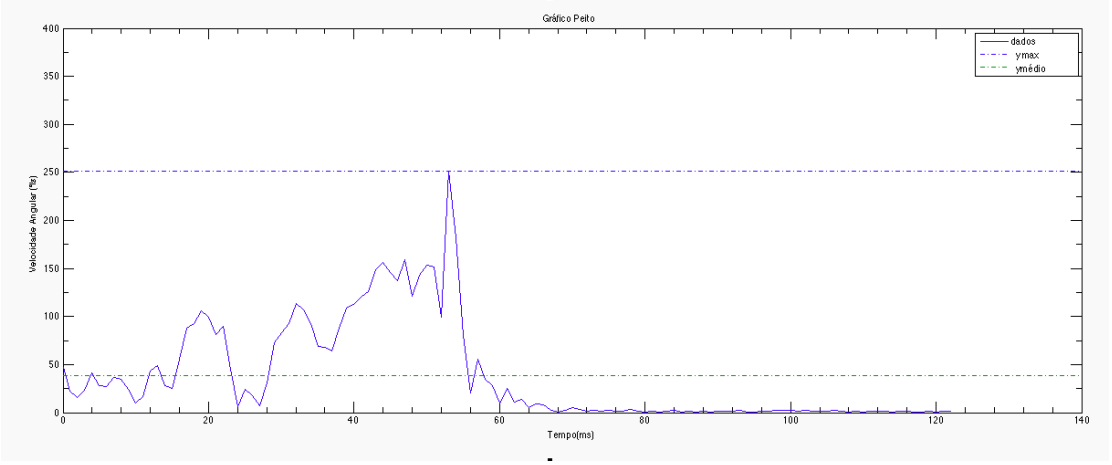

Figura 59 - Velocidade angular do corpo andando e em queda brusca 2 (Peito)

Nem todas as quedas seguem padrões, como no caso de queda suave, representada nas Figuras 60 e 61. Nesses cenários, a utilização de detecção apenas por *threshold* limita o sistema, por isso o momento de pausa foi utilizado. Pode-se ver da Figura 60 que quando ocorre queda o pico tem valor de aproximadamente 1,5g, mas esse valor também é o mesmo de quando o corpo está andando, por isso deve-se verificar a velocidade angular, que é maior que 220°/s, identificando a queda.

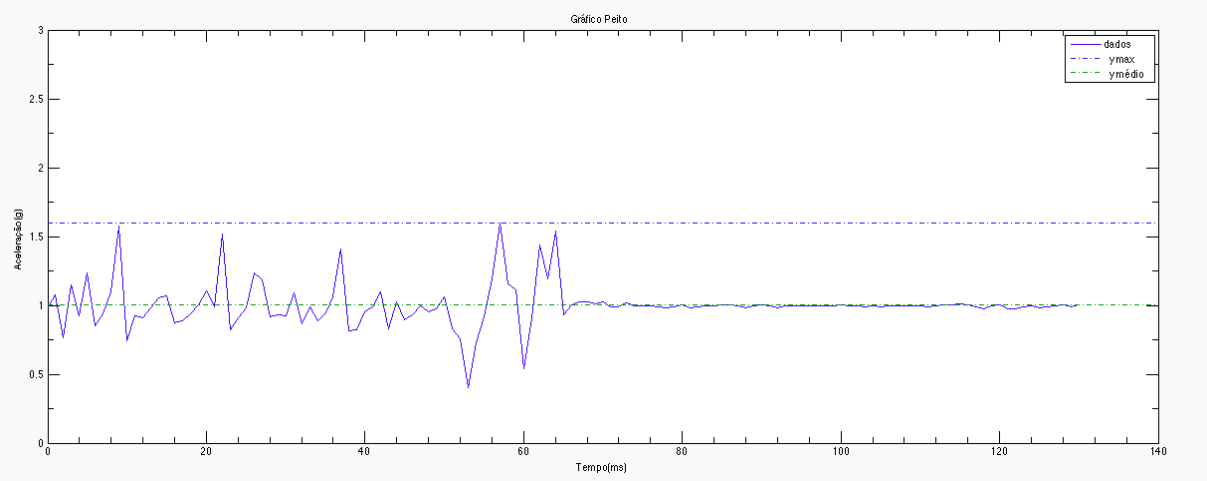

Figura 60 - Aceleração do corpo andando e em queda suave (Peito)

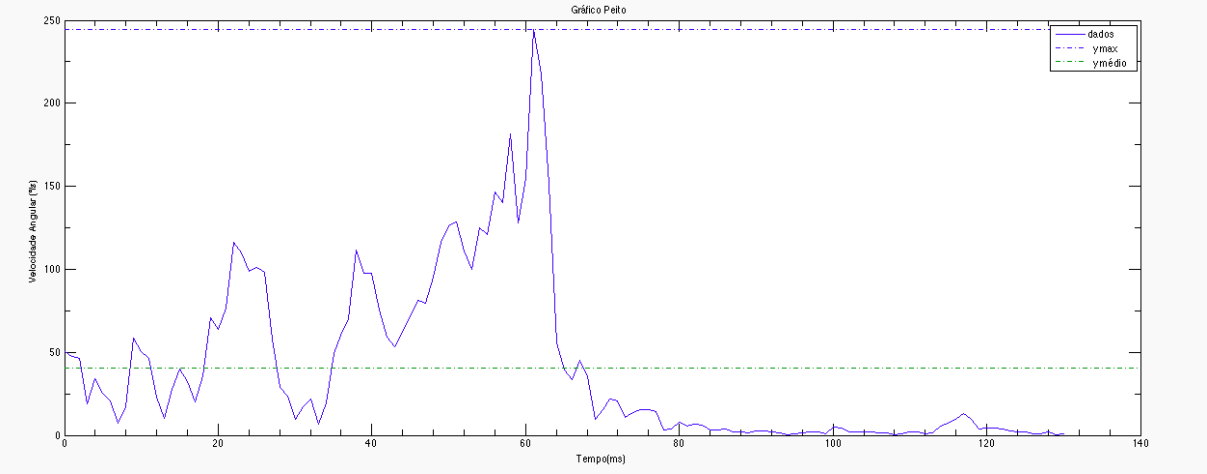

Figura 61 - Velocidade angular do corpo andando e em queda suave (Peito)

Os gráficos acima são referentes ao sensor anexado ao peito, contudo, os gráficos do sensor anexado à coxa apresentam os mesmos resultados, como pode ser visto nas Figuras 62 e 63. É possível verificar nestes gráficos que os valores máximos da aceleração e da velocidade angular ocorrem quando há queda e logo em seguida, um momento de pausa.

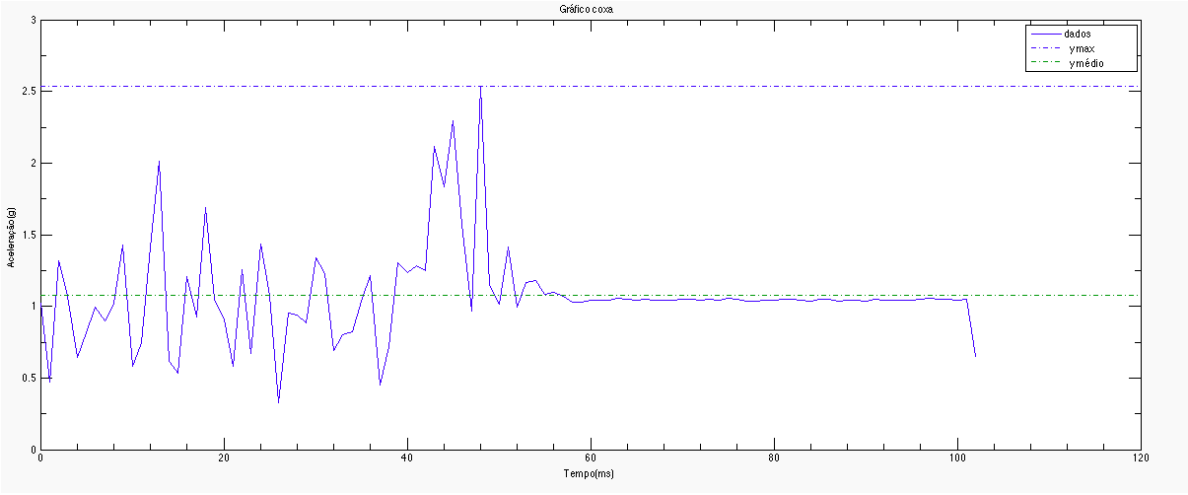

Figura 62 - Aceleração do corpo andando e em queda brusca (Coxa)

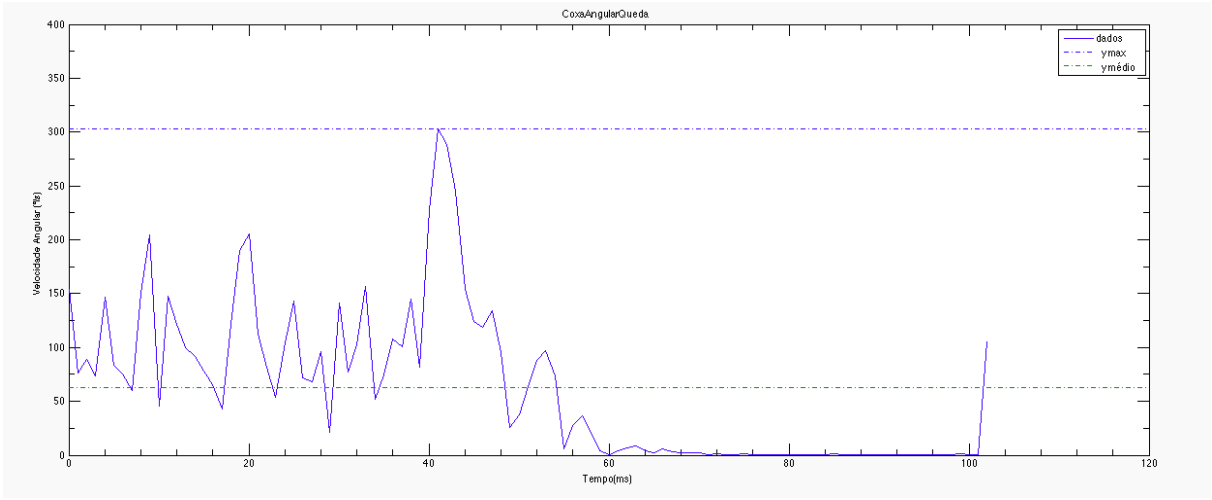

Figura 63 - Velocidade angular do corpo andando e em queda brusca (Coxa)

Para completar a análise, dados foram coletados quando o corpo está somente andando. As Figuras 64 e 65 mostram essa situação. Pode-se verificar que um momento de pausa não ocorre e a aceleração e a velocidade angular estão sempre variando.

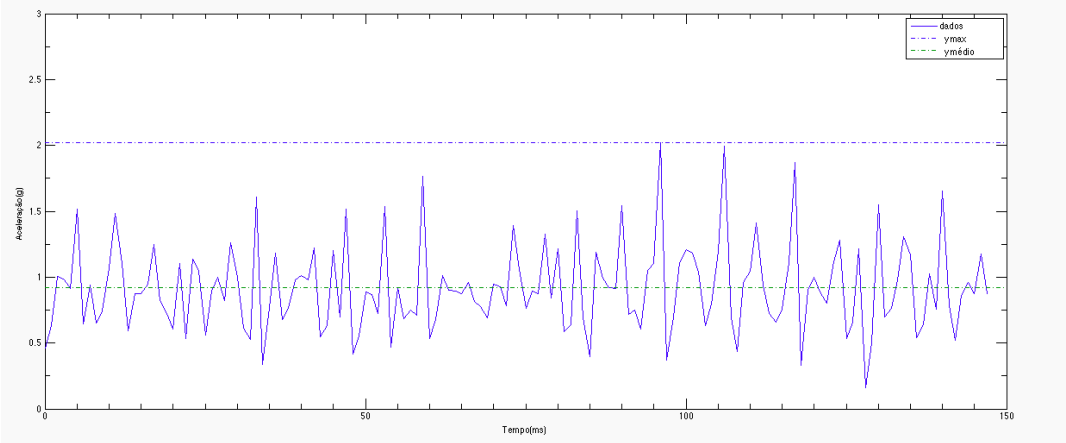

Figura 64 - Aceleração do corpo andando

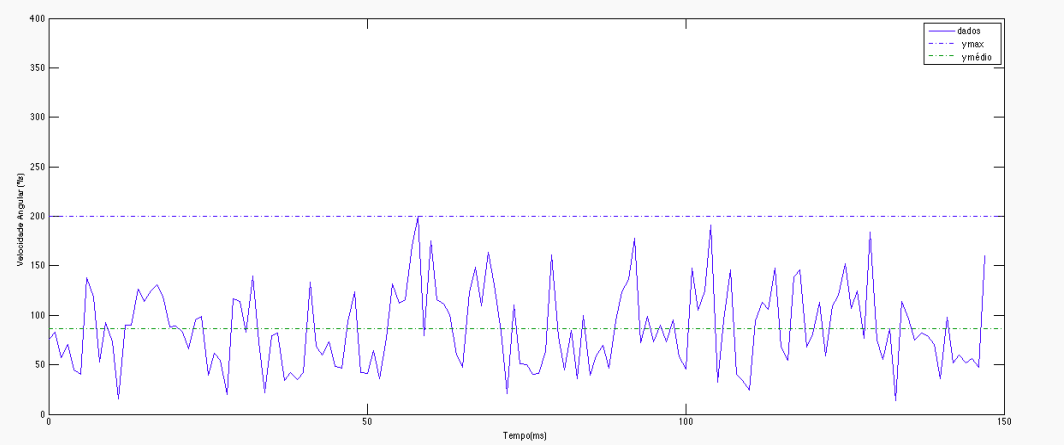

Figura 65 - Velocidade angular do corpo andando

A partir dos gráficos apresentados nessa seção, pode-se perceber que quando ocorre queda há um pico tanto na aceleração quanto na velocidade angular e logo após um período de pausa, enquanto que quando o corpo está apenas andando não há um período de pausa e a aceleração e velocidade angular estão sempre variando.

4.3. Site Web

A página inicial contém uma tela na qual o usuário deve informar seu *login* e senha, como pode ser visto na Figura 66, e com isso, é possível recuperar do banco de dados as informações referente àquele usuário, como o endereço do dispositivo que ele está utilizando, a temperatura no ambiente, posição e batimentos cardíacos do paciente, junto com a data e hora da última atualização. Esses dados são atualizados sempre que ocorrer alguma mudança em uma dessas variáveis. A Figura 67 mostra a tela de informações do usuário.

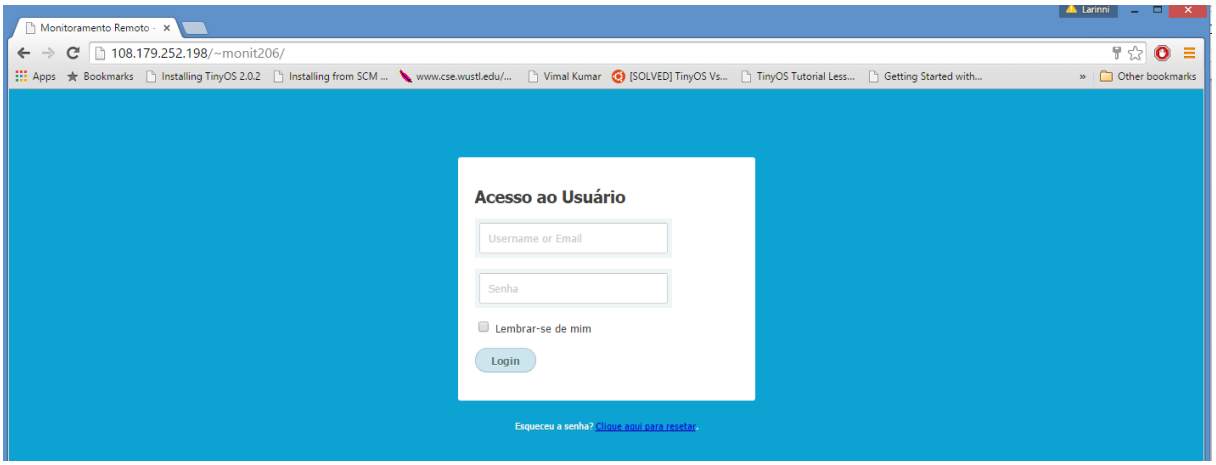

Figura 66 - Página inicial da Plataforma Web

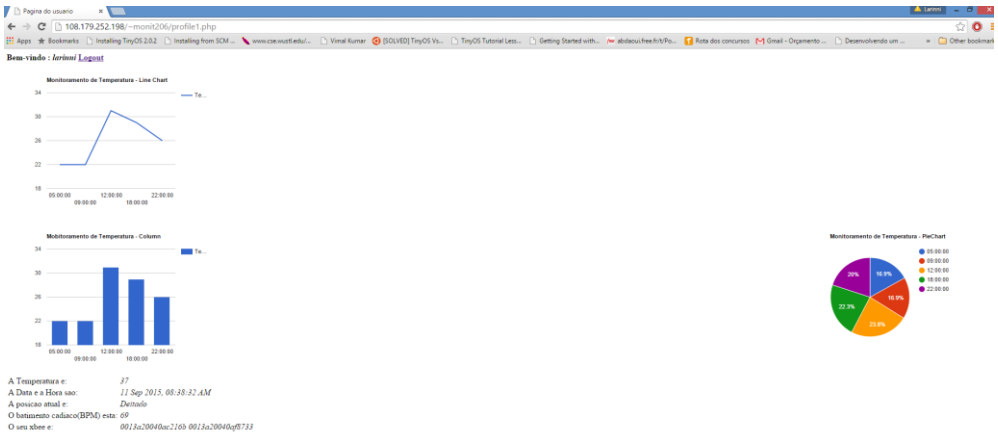

Figura 67 - Página de informações do usuário

Com o objetivo de complementar o equipamento e o monitoramento, na página que contém as informações do usuário é possível visualizar a variação da temperatura ao longo do dia utilizando gráfico de linhas, gráfico de barras e gráfico de pizza. Essa plotagem foi realizada utilizando a ferramenta *Google Charts*.

Como dito na seção 3.1.7, quando uma queda ocorre um script PHP envia um *email* de alerta, que pode ser verificado na Figura 68. Os campos que o compõem foram configurados na função *mail()* do PHP, que é responsável pelo seu envio.

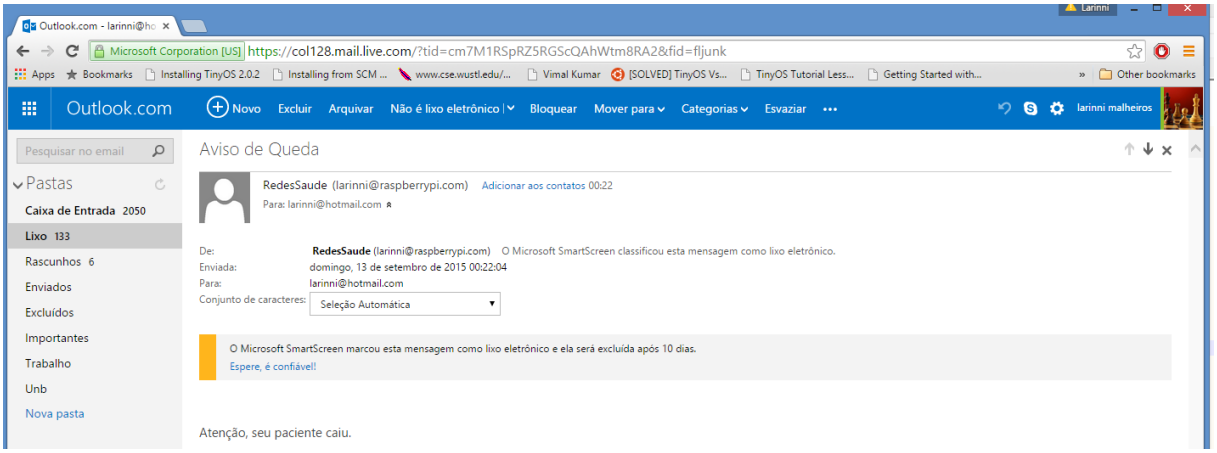

Figura 68 - Alerta de Email

4.4. Aplicativo Android

Ao iniciar o aplicativo, uma das telas da Figura 69 é apresentada ao usuário. Elas contêm um botão de emergência para pacientes que necessitam de ajuda e não conseguem se comunicar fazerem uma ligação para o número registrado no banco de dados, além disso, é apresentado o valor da temperatura, hora e data da última atualização, endereço do XBee,

valor e estado dos batimentos cardíacos. O aplicativo tem o nome fictício de MonitoraSaúde.

A posição em que o paciente se encontra é apresentada de forma iterativa com diferentes animações. Quando os batimentos cardíacos estão acima de 150 BPM, o aplicativo envia um alerta via *sms* para o número registrado, como pode ser visto na Figura 70.

Nesse aplicativo não é possível o paciente e o profissional da saúde fazerem algum tipo de alteração ou inserção de dados, pois todas as informações disponibilizadas são automaticamente atualizadas pelo sistema. A aplicação tem um tamanho total de 824KB.

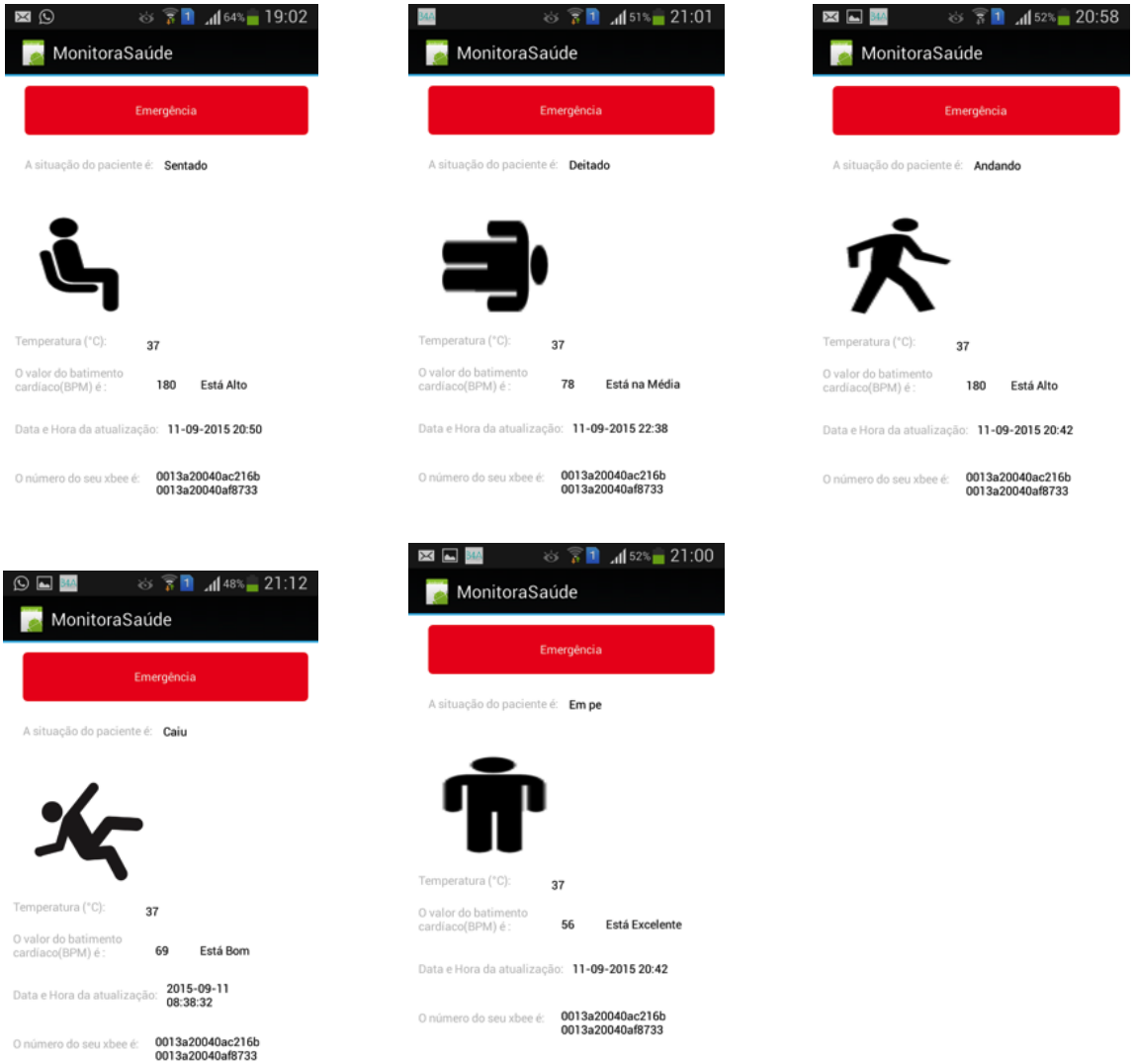

Figura 69 - Aplicativo Android

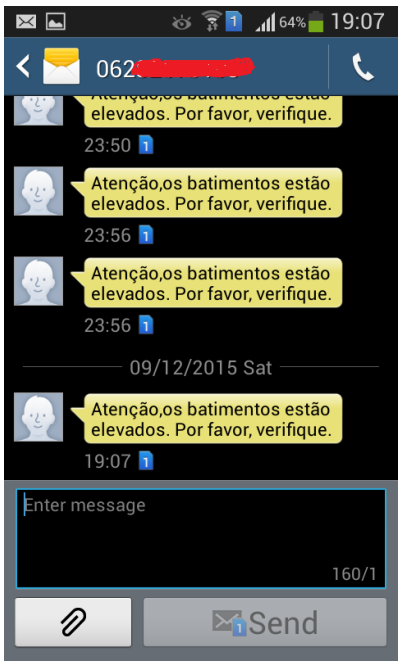

Figura 70 - Alerta de batimentos cardíacos

5. Conclusões e Trabalhos Futuros

O tema desenvolvido neste trabalho de graduação teve como objetivo principal construir um dispositivo capaz de detectar quedas, posicionamento corporal, monitorar batimentos cardíacos e temperatura ambiente, além de desenvolver um sistema de alerta para casos de quedas e elevação dos batimentos cardíacos. Podendo assim, contribuir para melhorar a qualidade de vida de idosos, pessoas que necessitam de monitoramento e de sesus familiares.

O projeto contemplou a construção do hardware e do software do equipamento. O equipamento é composto por 2 dispositivos de monitoramento e 1 dispositivo gateway. Os dispositivos de monitoramento são compostos por arduino, rádio XBee, sensor de batimentos cardíacos e por um sensor MPU6050, que é composto por acelerômetro, giroscópio e sensor de temperatura. O gateway é composto por RaspberyPi, rádio XBee e um cabo Ethernet.

A topologia de rede utilizada foi a estrela, com dois dispositivos roteadores e um coordenador. O protocolo de camada Física escolhido foi o 802.15.4 com o padrão de comunicação ZigBee. Utilizando essa tecnologia permitiu-se o equipamento apresentar baixo custo, baixo consumo de energia e baixa taxa de transmissão.

Foi desenvolvido um algoritmo para os dispositivos de monitoramento e outro para o gateway. O algoritmo dos dispositivos de monitoramento faz a coleta e transmissão de dados, enquanto que o algoritmo do gateway é responsável por receber os dados originados dos dispositivos de monitoramento, fazer o processamento da informação e informar ao usuário sobre o atual estado do paciente.

O equipamento apresentou uma precisão de 100% na detecção de posição, 60% de detecção de quedas e identificou corretamente 90% dos casos quando o corpo está andando. O objetivo geral deste projeto foi parcialmente realizado, pois não se conseguiu detectar a queda em todos os casos, contudo, o objetivo de detectar posição corporal, monitorar batimentos cardíacos e temperatura ambiente foi satisfatoriamente atingido. O sistema de alerta também foi corretamente desenvolvido utilizando *sms*, ligação telefônica e *email*. Além disso, foi desenvolvido um aplicativo Android para facilitar o monitoramento dos familiares.

Como trabalho futuro, deseja-se estudar e utilizar outros parâmetros para a determinação da queda, aumentando assim, a confiabilidade do sistema. Além disso, deseja-se adicionar um módulo GPS e um módulo 3G/GPRS de tal forma que a coleta e identificação de dados possa ser efetuadas fora de um ambiente domiciliar. Os estudos de como esses novos módulos afetam o gasto de energia também deve ser considerado.

Referências Bibliográficas

[1] Beckford, Martin. (2011). Almost half of elderly people live alone. Disponível online em: http://www.telegraph.co.uk/news/health/elder/8970490/Almost-half-of-elderly-people-live-alone.html . Acesso em: 04/06/2015.

[2] Li, Q., Stankovic,et al. (2009). Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information.

[3] Lim, D., et al. (2014).Fall-Detection Algorithm Using 3-Axis Acceleration: Combination with Simple Threshold and Hidden Markov Model.

[4] Interfacing Raspberry Pi and MPU-6050. (2013). Disponível online em: http://blog.bitify.co.uk/2013/11/interfacing-raspberry-pi-and-mpu-6050.html. Acesso em: 20/07/2015.

[5] Protocolo I2C. (2013). Disponível em: http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf. Acesso em: 21/07/2015.

[6] Frequência cardíaca média de repouso. Disponível em: http://saude-info.info/frequencia-cardiaca-media-de-repouso.html#sthash.UeKO7niW.K0bt3qel.dpbs. Acesso em: 15/08/2015.

[7] Bourke, A., & Lyons G. (2008). A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor.

[8] Bourke, A., O'Brien, J., & Lyons G. (2007). Evaluation of Accelerometer-Based Fall Detection Algorithms on Real-World Falls.

[10] Seifert, K., & Camacho O. (2007). Implementing Positioning Algorithms Using Accelerometers.

[11] Universidade Federal Fluminense. Departamento de Engenharia de Telecomunicações. Roteiro para configuração dos módulos Xbee.

[12] Tomkun, J., & Nguyen, B. (2010). Design of a Fall Detection and Prevention System for the Elderly.

[13] Ugulino, W., et al. (2012). Virtual Caregiver: um Sistema para Apoiar a Colaboração no Acompanhamento de Idosos.

[14] Oliveira, M., et al. (2013). Sistema de monitoramento de quedas para pessoas com deficiência motora.

[15] Miranda, A., Marik, P., Sistema Detector de Quedas(SDQ).

[16] Lukaszewski, A.(2010). MySQL for Python (pp. 247-312). Vol.1.

[17] Basic4Android Beginner's Guide(v3.0). Disponível em: http://www.b4x.com/android/documentation.html. Acesso em: 20/08/2015.

[18] Beazley, D., & Jones, B. (2013). Python Cookbook.

[19] Gilmore, J. (2010). Beginning PHP and MySQL.

[20] Charts-Google Developers. Disponível em: https://developers.google.com/chart/?hl=pt-BR. Acesso em: 17/05/2015.

[21] MPU-6000 and MPU-6050 Product Specification. Disponível em: http://www.waveshare.com/wiki/File:MPU-6000-and-MPU-6050-Product-Specification.pdf. Acesso em: 25/05/2015.

[22] Raspberry Pi Community. Disponível em: https://www.raspberrypi.org/forums/. Acesso em: 20/04/2015.

[23] Sorvala, A., et al. (2012). A two-threshold fall detection algorithm for reducing false alarms.

[24] Igual, R., Medrano, C., & Plaza, I. (2013). Challenges, issues and trends in fall detection systems.

[25] Ojetola, O., Gaura, E., & Brusey, J. (2011). Fall detection with wearable sensors-safe (smart fall detection).

[26] Mundher, Zaid., & Zhong, J. (2014). A Real-Time Fall Dection System in Elderly Care Using Mobile Robot and Kinect Sensor.

[27] Dinh, C., Struck, M. (2009). A new real-time fall detection approach using fuzzy logic and a neural network.

[28] Kostipoulos, P., et al., (2015). Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement.

[29] Lustrek, M., Kaluza, B. (2008). Fall Detection and Activity Recognition with Machine Learning.

[30] Xbee 802.15.4. Disponível em: http://www.digi.com/products/xbee-rf-solutions/modules/xbee-series1-module#specifications. Acesso: 04/04/2015

[31] Wireless Industrial Communication Networks. Disponível em: http://anp.tu-sofia.bg/djiev/Networks_Wireless.htm. Acesso em 08/09/2015

[32] Introduction to the ZigBee Wireless Sensor and Control Network. Disponível em: http://www.informit.com/articles/article.aspx?p=1409785&seqNum=4. Acesso em: 15/08/2015

[33] Bridging Wireless Sensor Networks and Ethernet. Disponível em : http://www.iebmedia.com/index.php?id=5429&parentid=63&themeid=255&showdetail=true. Aceso em 17/09/2015

[34] XBee and ZigBee basic concepts. Disponível em: http://ftp1.digi.com/support/documentation/html/90001399/90001399_A/Files/XBee-concepts.html#_Toc384719506. Acesso em: 02/05/2015

[35] Manual de prevenção de quedas da pessoa idosa. Disponível em: http://www.exerciciodorespeito.com.br/files/ManualQuedasPessoaIdosa.pdf?p=9. Acesso em 19/08/2015

[37] Arduino: An Introduction. Disponível em: http://computers.tutsplus.com/tutorials/arduino-an-introduction--mac-53232. Acesso em: 17/09/2015

[38] Martins, R. (2010). Desenvolvimento de um sensor de fotopletismografia para monitorização cardíaca para aplicação no pulso. Tese de Mestrado. Faculdade de Ciências e Tecnologia – Universidade de Coimbra

[39] Pulse Sensor Amped. Disponível em: http://pulsesensor.com/. Acesso em: 18/06/2015

[40] Example of XBee API Frames. Disponível em : http://rubenlaguna.com/wp/2009/03/12/example-of-xbee-api-frames/. Acesso em: 21/09/2015

[41] UML – Diagrama de Classes e Objetos. Disponível em: http://www.macoratti.net/net_uml1.htm. Acesso em : 20/09/2015

[42]Guedes, G. (2008). UML Uma abordagem Prática. Editora Novatec

[43] Shen, Victor., Lai, H., Lai A., (2015).The implementation of a smatphone-based fall detection system using a high level fuzzy Petri net

[44] Colon, L.N.V, DeLaHoz, Y., & Labrador, M. (2014). Human fall detection with smartphones.

[45] Zhuang, W., Sun, X., & Dai, D. (2015). Fall detection for elder people using single inertial sensor.

[46] Torres, R., et al. (2015). What if your floor could tell someone you fell? A device free fall detection method.

[47] Biomechanika. Disponível em: http://biomech.ftvs.cuni.cz/pbpk/kompendium/biomechanika/images/teziste.gif. Acesso em 23/09/2015
[48]http://www.upwithgravity.net/up-with-gravitysm-lesson-3-using-your-center-of-gravity-when-sitting/

[49] Hirata, M.(2005).Centro de massa. Universidade de Campinas.

[50] Department of Economic and Social Affairs,(2014).Population ageing and sustainable development. Disponível em: http://www.un.org/en/development/desa/population/publications/pdf/popfacts/PopFacts_2014-4.pdf. Acesso em: 20 de novembro de 2015

[51] Baldoni, A., Pereira, L.(2011). O impacto do envelhecimento populacional brasileiro para o sistema de saúde sob a óptica da farmacoepidemiologia: uma revisão narrativa.

[52] Oliveira, F. (2012). Cresce o número de idosos morando sozinhos no Brasil. Disponível em: http://jornalggn.com.br/blog/luisnassif/cresce-numero-de-idosos-morando-sozinhos-no-brasil. Acesso em: 14 de dezembro de 2015.