Uma biblioteca para comunicação ciente de localização geográfica entre dispositivos moveis

Renato Avila dos Santos e Tonny Costa Cordeiro

Orientador: Prof. Dr. Daniel Macêdo Batista

Trabalho de Conclusão de Curso
Bacharelado em Ciência da Computação

Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Universidade de São Paulo

Durante o desenvolvimento deste trabalho os autores receberam auxílio financeiro da FAPESP por meio de projeto coordenado pelo Prof. Roberto Marcondes

São Paulo, Fevereiro de 2013

Resumo

O GPS e a bússola digital são tecnologias que, embarcadas em dispositivos móveis, estão em crescente utilização. Hoje já é possível encontrar diversas aplicações que utilizam esses recursos.

Apesar dessas tecnologias fornecerem os dados necessários para que aparelhos se comuniquem através de referências geográficas, são raras as aplicações que permitem que um dispositivo envie uma mensagem a outro pelo simples fato de estar apontando para ele. Esse trabalho de conclusão de curso (TCC) propõe a criação de uma biblioteca que atenda essa funcionalidade, baseando-se em um protocolo de aplicação e sem a necessidade de utilização de servidores externos.

A biblioteca terá sua eficácia avaliada através da implementação de um jogo, cujo código-fonte será disponibilizado sob uma licença de software livre.

Palavras-chave: acelerômetro, magnetômetro, giroscópio, gps, android, biblioteca, redes, ponto de acesso

Abstract

The GPS and the digital compass are technologies that when embedded in mobile devices have more and more users. Today there are many applications that use those types of resources.

Although those technologies provide the necessary data for device communication through geographic landmark, there are only a few applications that enable one device to send messages to another device simply by "pointing"at it. This work proposes a library to enable such functionality, based on an application protocol without the need of external servers.

In order to attest the library functionality, it will be used to develop a simple game whose source code will be available under an open software license.

Keywords: accelerometer, magnetometer, gyroscope, gps, android, library, networks, access point.

Sumário

Capítulo 1

Introdução

O GPS e a bússola digital são tecnologias que, embarcadas em dispositivos móveis, estão em crescente utilização. Hoje já é possível encontrar diversas aplicações que utilizam esses recursos.

Apesar dessas tecnologias fornecerem os dados necessários para que aparelhos se comuniquem através de referências geográficas, são raras as aplicações que permitem que um dispositivo envie uma mensagem a outro pelo simples fato de estar "apontando"para ele. O desenvolvimento de uma biblioteca para comunicação ciente de localização geográfica entre dispositivos móveis pode ser um agente motivador para criação de aplicativos com esse perfil por diversos programadores.

Neste trabalho de conclusão de curso é apresentada uma biblioteca para comunicação ciente de localização geográfica entre dispositivos. As próximas seções deste capítulo estão organizadas da seguinte forma: A Seção 1.1 apresenta a motivação do trabalho, a Seção 1.2 apresenta os objetivos e a Seção 1.3 resume os conteúdos dos demais capítulos da monografia.

É importante observar que, para o desenvolvimento da biblioteca, foi escolhido o sistema operacional Android pelo fato da utilização do mesmo estar crescendo de forma impressionante. De 2010 a 2012, a quantidade média de novos usuários desse sistema operacional, por dia, aumentou de 200.000 para 900.000 [3, 21].

1.1 Motivação

A motivação para a realização deste trabalho vem do fato dos autores não terem encontrado na literatura e em sites especializados nenhuma biblioteca que permita o desenvolvimento de aplicações para celulares rodando o sistema operacional Android com comunicação ciente de localização geográfica. Uma biblioteca como esta levaria ao surgimento de diversas novas aplicações, principalmente na área de jogos.

Pelas buscas realizadas, o mais próximo que foi encontrado foi um jogo de tiro chamado *Mobile War* [8], uma espécie de "*paintball* virtual". Através do GPS e dos sensores acelerômetro e giroscópio dos dispositivos móveis, esse aplicativo permite a criação de um ambiente interativo nos quais os celulares são simulados como armas. A biblioteca apresentada nesse trabalho de conclusão de curso se propõe, por exemplo, a facilitar a implementação de um jogo desse tipo, dando-lhe ainda uma possível vantagem de ficar independente de servidores externos, o que não existe na versão 1.71 do *Mobile War*, que foi a versão mais recente encontrada pelos autores em Novembro de 2012.

1.2 Objetivos

Esta monografia tem como objetivos:

- Permitir a comunicação entre dispositivos móveis baseada unicamente em referências geográficas, através de uma rede que dispense a utilização de um servidor externo e que identifique, de forma rápida, o aparelho receptor de mensagens.
- Criar a biblioteca intitulada como *GeoCommunication* para facilitar o desenvolvimento de aplicações que utilizem as funcionalidades descritas no item anterior.
- Criar um aplicativo Android que utilize a biblioteca do item anterior.
- Avaliar a precisão dos sensores e GPSs de alguns celulares disponíveis no mercado brasileiro.

1.3 Organização da monografia

A monografia está organizada da seguinte forma: Os Capítulos 2 e 3 descrevem os estudos que se fizeram necessários para o entendimento dos conceitos que serão abordados na criação da biblioteca e da aplicação. No Capítulo 4 são apresentados os resultados dos experimentos que foram realizados com o objetivo de identificar informações técnicas referentes à precisão das bússolas e dos GPSs de diversos celulares. No Capítulo 5 a biblioteca é descrita através do seu diagrama de classe, do diagrama de caso de uso e de alguns detalhes da sua implementação. No Capítulo 6 são mostrados os resultados obtidos por meio de experimentos para avaliar a eficácia da biblioteca. No Capítulo 7 são apresentadas as conclusões. Informações adicionais são apresentadas no Apêndice.

Capítulo 2

Conceitos sobre redes

Este capítulo apresenta alguns conceitos básicos sobre redes. A Seção 2.1 fornece uma pequena introdução sobre as camadas da Arquitetura TCP/IP. A Seção 2.2 apresenta algumas informações sobre a camada de aplicação e seus principais protocolos. A Seção 2.3 apresenta os principais protocolos da camada de transporte. Na Seção 2.4 são apresentados conceitos básicos sobre modos de configurações de rede. Na Seção 2.5 são discutidas as formas de envio de uma mensagem via rede. A Seção 2.6 apresenta, de forma resumida, algumas tecnologias de redes sem fio como Wi-fi, Bluetooth e *Near Field Communication* (NFC).

2.1 Arquitetura TCP/IP

Arquiteturas de comunicação especificadas através de camadas facilitam a compreensão e permitem uma modularização que facilita o trabalho do desenvolvedor. Por conta disso algumas das seções deste capítulo descrevem algumas informações sobre camadas da Arquitetura Internet, também conhecida como Arquitetura TCP/IP. A Tabela 2.1 apresenta a organização das camadas, sendo cada uma responsável por fornecer serviços de rede para a camada imediatamente superior.

Aplicação
Transporte
Rede
Enlace
Física

Tabela 2.1: *Arquitetura TCP/IP em camadas.*

2.2 Camada de aplicação

A camada de aplicação é responsável por prover serviços às aplicações abstraindo a comunicação em rede entre os nós comunicantes. Uma aplicação tem a "impressão"de que ela está se comunicando diretamente com a outra, pois não tem conhecimento do que acontece entre o envio e o recebimento das mensagens.

Essa camada é de extrema importância, pois se não fosse possível criar aplicações de rede úteis, tais como a navegação web, o correio eletrônico e a transferência de arquivos, não haveria necessidade de criar uma arquitetura como a arquitetura TCP/IP.

Há a necessidade de criar protocolos para que essas aplicações possam se comunicar com outras de um modo simples e único. Uma analogia a comunicação entre aplicação seria duas pessoas conversando, por exemplo, em português. Cada pessoa corresponderia a uma aplicação diferente e o idioma ao protocolo usado para comunicação.

Existem muitos protocolos de aplicação, mas, entre os principais e mais usados, podemos citar o HTTP (*HyperText Transfer Protocol* - Protocolo de Transferência de HiperTexto) usado na navegação web, o SMTP (*Simple Mail Transfer Protocol* - Protocolo de Transferência de Correio Eletrônico) usado no correio eletrônico e o FTP (*File Transfer Protocol* - Protocolo de Transferência de Arquivo) usado na transferência de arquivos. Exploraremos, na subseção a seguir, com mais detalhes o HTTP, para ter uma noção básica sobre como se deve criar um protocolo de aplicação.

2.2.1 HTTP

O HTTP é responsável pelo tratamento de mensagens de requisição (*request*) e resposta (*response*) entre um cliente e um servidor na *World Wide Web*. Esse protocolo especifica como deve ser a sintaxe (regras) que as mensagens devem seguir realizar, corretamente, uma comunicação HTTP.

Quando um cliente faz uma requisição de um recurso que está num determinado servidor, a mensagem deve conter, em sequência, uma linha inicial (*request-line*), linhas de cabeçalho (*request-header*), uma linha em branco obrigatória e um corpo de mensagens opcional. A linha inicial deve conter três partes: o método a ser usado, a identificação do recurso (URI) e a versão do HTTP.

O método da linha inicial indica o que deve ser feito com o recurso requerido. Por exemplo, o método GET indica que o cliente solicita o conteúdo do recurso, enquanto o método POST envia dados a serem processados pelo mesmo (os dados são enviados no corpo da mensagem). Segue um exemplo de uma requisição usando o método GET:

```
GET /index.html HTTP/1.1
Host: www.exemplo.com
```

Ao receber uma requisição, o servidor deve processá-la e responder ao cliente qual foi o resultado. Assim como na requisição, a mensagem de resposta tem um formato bem definido, sendo composta de uma linha inicial (*status-line*), linhas de cabeçalho (*response-header*), uma linha em branco obrigatória e um corpo de mensagem opcional. A linha inicial deve conter três campos: a versão do HTTP, um código de status e sua descrição.

O código de status indica se a requisição foi ocorrida com sucesso ou se teve algum erro e qual erro. Por exemplo, o código 200 indica que a requisição ocorreu com sucesso, enquanto o código 404 indica que o recurso requerido não foi encontrado no servidor. Segue um exemplo de parte de uma resposta bem sucedida a requisição:

```
HTTP/1.1 200 OK
Server: Apache/1.3.27
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html
```

2.3 Camada de transporte

A camada de transporte, definida na arquitetura TCP/IP, é responsável pelo envio das mensagens da origem até o destino, ou seja, entre o nó inicial e final, não sendo considerada nos nós intermediários da rede. Os protocolos mais comuns dessa camada são o UDP e o TCP.

2.3.1 Socket

Socket é o canal de comunicação entre dois processos. Esse conceito é muito importante para entender como as aplicações conseguem trocar mensagens entre elas. Segundo JAMES F KUROSE, "socket é a interface entre a camada de aplicação e a de transporte"[17].

Um socket pode ser identificado por uma porta, através da qual os processos enviam e recebem mensagens. Para entender melhor a ideia de socket, uma analogia com o cotidiano pode ser utilizada. Quando você deseja ir à casa do vizinho, você deverá passar pela porta da sua casa, atravessar a rua e entrar pela porta da casa do vizinho. A porta da sua casa é análoga ao socket do processo cliente e a porta da casa do vizinho é análoga ao socket do processo servidor.

Há vários tipos de sockets, entre eles o *Datagram Socket* e o *Stream Socket*. Esses sockets são usados, respectivamente, pelo protocolo UDP e o TCP.

2.3.2 UDP

O UDP (*User Datagram Protocol* - Protocolo de Datagrama de Usuário), definido na RFC 768 [5], é o protocolo que fornece um serviço não orientado a conexão, portanto não é necessária a apresentação (*handshaking*) dos processos envolvidos antes da comunicação. O UDP não provê um serviço confiável de entrega de mensagens, ou seja, não há nenhuma garantia de que o pacote que foi enviado será entregue ao destino. Além disso, os pacotes podem chegar fora de ordem e possíveis atrasos devido a um congestionamento na rede não são tratados. Se algum tipo de garantia de entrega das mensagens for necessária, então esta deverá ser implementada na camada de aplicação.

Logo, aplicações que usam o UDP devem ser tolerantes a falhas e perda de dados. Esse é o caso, por exemplo, das aplicações de transmissão de vídeo e áudio em tempo real.

Apesar dessas desvantagens, o UDP é um protocolo bem mais simples do que muitos outros, como o TCP, que será visto na Subseção 2.3.3. Por exemplo, é possível que durante uma conexão, para transmitir um determinado conjunto de dados, o UDP envie apenas dois pacotes, enquanto o TCP pode enviar mais de dez. O seu uso também permite enviar mensagens em modo Broadcast (Seção 2.5.2) e Multicast (Seção 2.5.3), dois modos que permitem o envio de uma mensagem para mais de um usuário. O TCP, por exemplo, não tem suporte a esses modos, já que é um protocolo utilizado entre, no máximo, duas entidades da rede.

2.3.3 TCP

O TCP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão), definido na RFC 793 [5], é um protocolo mais complexo que o UDP. É um protocolo orientado a conexão, ou seja, antes do envio de mensagens, os processos envolvidos deverão apresentar-se uns aos outros. Feito isso, a conexão está preparada para o tráfego de mensagens, com a possibilidade de transferência simultânea em ambas direções (*Full duplex*).

A principal vantagem desse protocolo, em relação ao UDP, é a confiabilidade da entrega de mensagem. Durante o tráfego de pacotes, é sua responsabilidade verificar:

- pacotes perdidos ou com bits modificados por erros de transmissão;
- chegada de pacotes fora de ordem ou duplicados.

Portanto, o TCP garante que os pacotes enviados chegarão e estarão na ordem correta. O protocolo inclui um mecanismo de controle de congestionamento, que opera limitando o número máximo de pacotes que podem ser enviados numa conexão antes do recebimento de um pacote de reconhecimento (ACK). Esse mecanismo não funciona com o intuito de melhorar o desempenho da aplicação, mas sim de melhorar o tráfego na rede. Assim como o UDP, o TCP não garante um limite de tempo para a entrega da mensagem.

Uma conexão TCP é ponto a ponto, portanto o único modo de envio é Unicast (Seção 2.5.1). Diferente do UDP que suporta os modos broadcast e multicast.

2.4 Redes infraestruturada e ad-hoc

A forma como é feita a comunicação entre nós de uma rede depende da arquitetura utilizada. Os dois principais modos de configuração de uma rede de dispositivos móveis são ad-hoc e infraestruturado.

2.4.1 Rede ad-hoc

Redes ad-hoc são redes onde não existe ponto de acesso (PA) centralizando as comunicações. Ou seja, os dispositivos são capazes de trocar informações diretamente entre si. Com isso, dois dispositivos que estão próximos um do outro podem trocar informações sem precisar que esta mensagem vá até alguma estação base ou ponto de acesso centralizado. Há grandes vantagens em utilizar redes ad-hoc para comunicação entre nós. Por não ser necessário o uso de uma estrutura física fixa para que os nó se comuniquem, esta é a rede ideal para lugares com infraestrutura de suporte precária, como estações móveis, desertos e florestas.

Esse tipo de rede serve também para computadores próximos um do outro, quando há necessidade de troca de informações entre eles e não é desejável o uso de um ponto de acesso.

Entretanto, há algumas desvantagens em sua utilização. Por ser auto gerenciável e depender dos nós para se manter conectada, há muita mudança na topologia da rede, logo seus protocolos devem ser auto configuráveis para se adaptarem aos novos caminhos das mensagens. Essa necessidade de busca constante por caminhos tem como consequência um maior atraso no envio das mensagens quando comparado com uma arquitetura que tenha os caminhos pré-definidos.

2.4.2 Rede infraestruturada

São redes em que os terminais estão conectados diretamente a uma Estação Base ou Ponto de Acesso. Toda comunicação entre nós, de sub-redes diferentes, deverá, necessariamente, passar pelo Ponto de Acesso, impossibilitando qualquer tipo de comunicação direta, independente da proximidade dos dispositivos.

No modo ad-hoc, quando um nó deseja trocar informação com outro, este precisa estar no raio de alcance do primeiro, mas em redes infraestruturadas cada nó precisa estar, apenas, no raio de alcance do Ponto de Acesso. Por ter um ponto de acesso centralizador, o processo

de envio de mensagem é de responsabilidade do mesmo, simplificando as tarefas realizadas por cada nó.

2.5 Modo de transmissão das mensagens

Ao criar uma aplicação deve-se escolher qual o protocolo de transporte a ser usado, TCP ou UDP. Para tomar essa decisão deve-se levar em conta as vantagens e desvantagens de cada protocolo e, principalmente, ter em mente o que esperar da aplicação.

Um importante fator na escolha do protocolo é o modo de transmissão da mensagem, que pode ser por unicast, multicast ou broadcast. Por exemplo, se uma aplicação precisar do multicast como modo de transmissão de mensagens, o seu protocolo de transporte será o UDP. Visto isso iremos apresentar uma breve explicação de cada um dos modos.

2.5.1 Unicast

Também conhecida como transmissão de ponto a ponto, é uma comunicação entre um único nó de origem e um único nó de destino. Um pacote é enviado de um nó ao outro carregando as informações de endereço IP de origem e endereço IP de destino.

Talvez o nome unicast seja menos conhecido que o broadcast e o multicast, porém esse modo de transmissão é o mais usado hoje em dia por aplicações. Uma prova disso é que os protocolos FTP (transferência de arquivos), SMTP (correio eletrônico) e HTTP (navegação web) usam transmissão unicast.

2.5.2 Broadcast

É uma comunicação entre um único nó de origem e todos nós da rede. Ou seja, quando for enviada uma mensagem em modo broadcast, todos os nós a receberão. Não é necessário que no envio sejam especificados, explicitamente, todos os endereços IPs de destino, bastando apenas indicar que a mensagem é do tipo broadcast no endereço IP de destino. Portanto, todo nó deve aceitar mensagens com dois possíveis endereços de destino, o próprio endereço IP e o endereço broadcast da rede.

Seus algoritmos de roteamento são mais complexos que o do unicast, que compreende apenas um destinatário, sendo um dos seus principais desafios enviar o menor número de pacotes pelos enlaces a fim de evitar a sobrecarga da rede.

2.5.3 Multicast

É uma comunicação entre um único nó de origem e múltiplos destinatários. Esse modo é muito parecido com o broadcast, com a diferença que a mensagem é enviada a um grupo, não necessariamente a todos os nós.

Assim como no broadcast, o multicast também necessita de um endereço específico. O endereço multicast é usado para formar o grupo de receptores. Portanto, cada aplicação que desejar se juntar ao mesmo, para receber mensagens a ele direcionadas, deverá saber qual é esse endereço. Enfim, para direcionar mensagens ao grupo, o endereço IP de destino deverá ser o de multicast.

Hoje os algoritmos de roteamento para mensagens em multicast são muito eficientes. Isso se deve ao fato do seu uso intensivo em transmissão de vídeo e áudio em tempo real para mais de um usuário ao mesmo tempo.

2.6 Tecnologias wireless

Através das tecnologias *Wireless* é possível a comunicação entre dispositivos sem que haja cabo de comunicação ligando uns aos outros. Existem diversas tecnologias wireless: Bluetooth, Wi-fi, infravermelho e GPS são alguns exemplos.

Essas tecnologias são caracterizadas pela potência de alcance e taxa de transmissão de dados. Essas características influenciam diretamente no consumo de energia do dispositivo que a usa. O Bluetooth, por exemplo, tem baixo consumo de energia comparado ao Wi-fi, porém seu alcance chega, no máximo, a 10 metros e a sua taxa de transmissão é, no máximo, de 3Mbit/s, ao passo que o Wi-fi tem alcance de até 300 metros (em locais abertos [14]) e sua taxa de transmissão é de até 100Mbit/s.

A tecnologia de interesse da biblioteca é o Wi-fi, que é suportada pela API do Android e utiliza o padrão IEEE 802.11g [6]. Além dessa, apresentaremos nessa seção também o Bluetooth e NFC.

2.6.1 Wi-fi

O Wi-fi permite que um dispositivo conecte-se a outro (ponto de acesso) usando a tecnologia Wireless. Isso facilita a criação de LANs (*Local Area Network* - Rede de Área Local), pois, onde é difícil ou inviável o uso de cabos, também é possível a criação de uma rede.

Apesar de ter um alcance de até 300 metros em locais abertos, o Wi-fi tem apenas 25 metros de alcance em locais fechados [14], o que limita uma aplicação que faça uso da biblioteca.

Outro fator que pode diminuir o alcance do sinal é a interferência de outras redes Wi-fi. Esse tipo de rede usa um canal de frequência para a comunicação, que pode sofrer interferência de outra rede Wi-fi que esteja usando o mesmo canal ou um canal que tenha intervalos de frequência em comum. Para minimizar esse problema é possível mudar o canal de frequência da rede para um que seja menos usado.

2.6.2 Bluetooth e NFC

As tecnologias Bluetooth e NFC foram desenvolvidas com objetivos diferentes do Wi-fi. As duas primeiras são usadas em comunicação pareada de curta distância, enquanto a última é uma tecnologia para comunicação em rede de média distância.

Vantagens

O Bluetooth e o NFC têm algumas vantagens em relação ao Wi-fi:

1. Comunicação simples, rápida e segura;
2. Menor gasto de energia.

Essas vantagens tornam muito atrativas essas tecnologias, pois todos esses aspectos são de grande valia nos dispositivos móveis.

Desvantagens

Visto as vantagens, agora é importante analisar as desvantagens e assim poder concluir qual a tecnologia melhor para ser usada na biblioteca:

1. Curto alcance. Cerca de 10m com o Bluetooth e 10cm com o NFC;

2. Necessidade de pareamento prévio.

A primeira desvantagem citada já torna essas tecnologias quase inúteis para a biblioteca. Porém é a segunda que torna impossível o uso delas. A necessidade de um pareamento para que haja comunicação inviabiliza que a mesma seja feita baseada em referências geográficas, que é o objetivo central dessa biblioteca.

Portanto, a melhor tecnologia para comunicação a ser usada nesse trabalho é o Wi-fi.

Capítulo 3

Conceitos sobre sensores e Android

Este capítulo apresenta alguns conceitos básicos sobre sensores e Android para compreensão dos demais capítulos da monografia. Na Seção 3.1 são explicados os principais conceitos do sistema operacional Android. A Seção 3.2 apresenta os sensores de um dispositivo com SO Android e como usá-los para obter o vetor de orientação do mesmo. A Seção 3.3 explica como obter uma posição geográfica através dos sistemas de posicionamento existentes. A Seção 3.4 mostra como filtrar os erros existentes na coleta das informações dos sensores.

3.1 Sistema operacional Android

Android é um sistema operacional que surgiu da Android, Inc., uma pequena empresa na Califórnia, EUA, que tinha como objetivo criar uma plataforma flexível para telefonia móvel. A Google, interessada nessa área, comprou a empresa em 2005. O primeiro *smartphone* com esse sistema operacional foi vendido em 2008, desde então a popularidade da plataforma cresceu e continua crescendo de forma impressionante. Segundo a própria Google, eram vendidos mais de 200.000 aparelhos por dia em 2010 [3], e, recentemente, em junho/2012, segundo o Vice Presidente Sênior da Google, Andy Rubin, vende-se mais de 900.000 aparelhos por dia [21].

3.1.1 Algumas características

O sistema operacional Android foi desenvolvido para aparelhos móveis como celulares e tablets, mas já está a caminho dos computadores. Trata-se de um sistema baseado no kernel do Linux [23], com código aberto desde Outubro de 2008 e licença Apache.

Cada aplicação roda em uma própria instância de uma máquina virtual Dalvik, que é baseada na JVM e otimizada para dispositivos móveis. Várias instâncias dessa máquina virtual podem rodar concorrentemente em um dispositivo. Diferentemente da JVM, onde apenas uma instância é criada e vários processos a usam.

3.1.2 Programação para Android

Aplicações desenvolvidas para Android devem ser escritas em Java. Uma ótima opção para programar para Android é utilizando o IDE Eclipse, com o SDK do Android e o plugin ADT (*Android Development Tools*) [6].

O SDK do Android provê diversas APIs modernas, com avanços significativos de uma versão para outra. Implementações novas de software e/ou hardware são adicionadas a cada

nova versão. O Android garante que uma aplicação criada para uma dada API funcione numa API mais nova, porém o inverso não é válido.

A documentação oficial de desenvolvimento para Android [6] fornece detalhes sobre o fluxo de funcionamento e exemplos de uso.

3.1.3 Rede ad-hoc no Android

O Sistema Operacional Android não provê na sua interface para o usuário e nem para o desenvolvedor a criação de uma rede ad-hoc, mas isso não significa que não é possível criar uma rede desse modo. Para criá-la é preciso programar no Kernel do Linux e, portanto, deve-se programar em C/C++.

Para manipular o Kernel do Linux no Android, é necessário ter acesso *root* ao seu aparelho. Como esse tipo de acesso depende de cada dispositivo e versão do SO [14], e devido aos inúmeros dispositivos existentes e à rotatividade muito alta de novos modelos no mercado, a criação de um suporte para redes ad-hoc na primeira versão da biblioteca tornou-se inviável. Apesar de existirem vários aplicativos disponíveis no mercado que dêem acesso *root* ao celular, isso pode se tornar uma barreira para aplicações que desejem usar a biblioteca.

3.1.4 Celular como ponto de acesso

Dada a dificuldade de criar uma rede ad-hoc, optou-se por criar uma rede infraestruturada. Para isso, é necessário um ponto de acesso que seja um provedor de sinal Wi-Fi. Muitos dos dispositivos com o sistema operacional Android provêm essa funcionalidade.

Uma limitação importante imposta pelo Android é que apenas oito celulares podem se conectar ao mesmo tempo nesse ponto de acesso [6], o que restringe a biblioteca a algumas aplicações.

A principal vantagem de usar o ponto de acesso do próprio Android é que torna o uso da biblioteca totalmente independente de infraestrutura externa, tais como a Internet e roteadores externos.

Um ponto fraco do seu uso é o alto consumo de energia dessa função. Como apenas um dispositivo será o ponto de acesso, então este gastará mais energia que os demais, o que não é interessante, pois dispositivos iguais rodando a mesma aplicação podem ter desempenhos diferentes. É importante ressaltar que o gasto de energia é um grande problema atual e, por isso, é tema de muitos estudos, como, por exemplo, em [13].

3.2 Sensores para Android

Atualmente, micro sistemas eletromecânicos de baixo custo, correspondentes a múltiplos sensores atrelados a várias condições ambientais, são integrados em quase todos os dispositivos com o S.O. Android, cuja API provê a disponibilização e interpretação dos sinais de saída dos mesmos.

Conforme a documentação oficial online [6] de desenvolvimento para Android, não é requerido que os fabricantes criem e implantem sensores nos dispositivos, acarretando na existência de uma vasta variedade de configurações. Por esse motivo, sua API permite a criação de aplicações específicas para fabricantes ou versões diferentes de sensores.

Os sensores possuem uma série de peculiaridades que permitem o seu manuseio, desde a simples detecção de novos valores, como também eventos mais avançados, tais como obtenção de acurácia (baixa, média, alta ou indeterminada) e potência física, definição de da frequência de amostragem, entre outros.

A falta de acurácia de um sensor revela o quão necessário é sua calibração com o ambiente. Existem programas e maneiras específicas para calibração de acordo com o tipo de sensor.

O framework para os sensores acelerômetro, giroscópio e magnetômetro, utilizados na biblioteca, usam três eixos de coordenadas para expressar suas medidas. O sistema de coordenadas dos mesmos é definido em relação à tela do dispositivo em sua orientação natural, conforme Figura 3.1. Ou seja, nos tablets a disposição tradicional seria paisagem, enquanto nos celulares o tradicional seria retrato. Mesmo quando a disposição da tela muda com as ações do usuário, os eixos não são trocados automaticamente.

O restante dessa seção comentará mais a respeito dos sensores acelerômetro, giroscópio e magnetômetro, que serão a base da criação do vetor de orientação pretendido pelo sistema de comunicação da biblioteca.

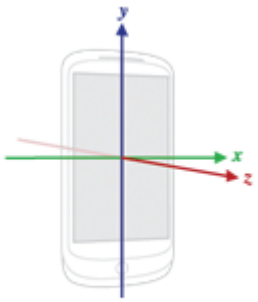

Figura 3.1: *Sistema de coordenadas do dispositivo conforme API de Sensores do Android. Figura extraída de [6].*

3.2.1 Taxa de amostragem

A taxa de amostragem define o intervalo de tempo em que os eventos de sensores são acionados na aplicação. Conforme Tabela 3.1, o tempo de atraso a ser definido pelo programador está atrelado ao tipo de atividade na API do Android: normal, jogos, interface de usuário ou "o mais rápido", podendo ser definido manualmente a partir da versão 3.0 do Android (nível de API 11). Conforme documentação [6], o valor padrão, SENSOR_DELAY_NORMAL, é apropriado para monitorar mudanças típicas de orientação da tela.

Quanto maior for o intervalo de tempo de atualização dos sensores, mais imprecisão passa-se a ter quando se busca fazer integrações sobre o histórico de resultados, para chegar a algum resultado baseado em estados anteriores. Por outro lado, uma opção como SENSOR_DELAY_GAME ou SENSOR_DELAY_FASTEST implicaria em um alto tempo de processamento, com impacto na duração da bateria.

Devido à particularidade dessas escolhas estarem atreladas à natureza da aplicação, cabe ao programador que usará a biblioteca *GeoCommunication* configurar a taxa de amostragem de acordo com seus propósitos.

Embora o programador possa definir a frequência de amostragem, cada tipo de sensor possui suas próprias restrições relativas à frequência de amostragens. A Tabela 3.1 exemplifica esse fato, trazendo dados relacionados a um experimento feito em [22].

Sensor	Taxa de amostragem (média)	Max	Min	Desvio padrão
Giroscópio	1.154713	3.512	0.589	0.2714452
Magnetômetro	16.81256	19.300	14.380	0.4401434
Acelerômetro	20.57853	23.703	18.256	0.4601378

Tabela 3.1: *Frequência de amostragem, em ms, obtida em experimento com dispositivos ora em movimentos aleatórios ora parados. Tabela extraída de [22]*

3.2.2 Magnetômetro

O magnetômetro corresponde a um sensor para medida do campo geomagnético do ambiente em μT nos 3 eixos físicos do dispositivo, servindo para monitoramento de mudanças no campo magnético da Terra e criação de bússolas digitais a partir do *azimuth* magnético, o componente horizontal do campo que deve, em condições ideais, apontar para o norte magnético.

Suas principais fontes de erros são as interferências magnéticas no ambiente e no dispositivo, que podem ser geradas por objetos metálicos, fios elétricos, motores, entre outras variadas situações comuns aos ambientes urbanos. Dessa forma, o sinal de saída pode estar sujeito a mais de uma interpretação, devendo ser feito a distinção, por exemplo, entre um distúrbio magnético e um movimento abrupto.

Há também uma falha intrínseca ao próprio sensor, cujo dispositivo portador deve ser mantido na horizontal durante as leituras para gerar um *azimuth* magnético mais próximo do valor real.

Pelo fato de estarem vinculados, a uma dada localização global, uma força e uma inclinação do campo magnético (ângulo do vetor gerado em relação à superfície), foi investigada em [16] a possibilidade de identificação de interferências magnéticas a partir da variação desses valores esperados. Porém, a admissão de um valor constante para a inclinação implicou em erros muito grandes na prática.

3.2.3 Acelerômetro

O Acelerômetro é um sensor baseado em hardware, com resolução de 10 bits, utilizado para detecção de movimento. Ele retorna, em m/s^2, a medida da aceleração somada à gravidade que é aplicada nos três eixos físicos que compõem o sistema de coordenadas do dispositivo.

Conforme Gabor Paller em [19], sua primeira utilização ocorreu em 2005 no celular Nokia 5500 "*sport device*", no sistema operacional Symbian. Trata-se de um sensor disponível desde a versão 1.0 do Android e presente, amplamente, nos diversos modelos com este sistema operacional

Devido à dificuldade do isolamento da gravidade dos resultados obtidos pelo sensor quando o seu dispositivo portador está em movimento, o mesmo costuma fornecer boa indicação de orientação apenas em condições estáticas (aceleração escalar nula) e costuma gerar informações restritas em algumas soluções para sistemas de posicionamento inercial. Por exemplo, pode ser utilizado para medição da gravidade quando há ausência de movimento, ou seja, quando não há nenhuma aceleração linear presente; ou para detecção de movimento quando houver mudança da norma euclidiana do vetor de aceleração.

Esse sensor costuma apresentar ainda distúrbios em movimento de inclinação e respostas lentas quando comparadas às do giroscópio.

Como concluído a partir de experimentos abordados em [18], a utilização de uma tradi-

cional forma para uso dos dados do acelerômetro conhecida como *Dead-reckoning* [1], quando testada em celulares, gerou erros de diversas ordens de magnitude. Isso ocorre porque as leituras desse sensor sofrem regularmente por largas interrupções, aumentando o ruído sucessivamente e em casos mais extremos, até mesmo omitindo a leitura.

3.2.4 Giroscópio

O giroscópio é um sensor com resolução de 16 bits que mede a velocidade angular em rad/s em volta dos eixos x, y e z do dispositivo. Normalmente, a variação angular é extraída da velocidade, através da sua multiplicação pelo intervalo de tempo entre a atual e a última saída, gerando um incremento de rotação que, integrado ao longo do tempo, permite a formação do vetor de orientação.

Esse sensor teve sua primeira utilização em 2009, no acessório de vídeo-game *Wii Motion Plus*, sendo a sua primeira ocorrência em celulares no final de 2010, no modelo Nexus S.

Quando comparado ao acelerômetro, sua produção é relativamente mais barata e possui respostas muito rápidas a mudanças de ângulos, porém requer significantemente mais potência para operar (cerca de 8 vezes mais, segundo Viktor Moel [18]).

Dois erros muito comuns em giroscópios são o bias (polarização) e o drift (deriva). O primeiro corresponde à ocorrência de sinais de saída do sensor quando não há movimento. O segundo diz respeito à variação do bias ao longo do tempo, gerada por aquecimento dos componentes físicos do sensor e/ou por erros de integração das taxas de giro. Além dessas falhas, há também problemas relativos à calibração, que são correspondentes a erros em fatores de escala, alinhamento e linearidades dos giros.

3.2.5 Vetor de orientação

A orientação de um dispositivo é baseada em ângulos de rotação relativos a algum quadro de referência estruturado sobre um sistema de coordenadas fixo. Para construir um vetor de orientação no quadro de referência da Terra (*Earth Reference Frame* - ERF), também conhecido como quadro de navegação ou quadro global, deve-se utilizar os dados do quadro de referência do dispositivo (*Device Reference frame* - DRF).

A API de programação do Android disponibiliza o método `getRotationMatrix` para gerar uma matriz de rotação, passando-se como parâmetros as medidas do campo geomagnético do ambiente e da gravidade. Dessa matriz, é possível extrair o vetor de orientação do quadro de referência da Terra, tal como ilustrado na Figura 3.2 e Tabela 3.2, através do método `getOrientation` ou multiplicando-a pelo vetor de orientação obtido através do giroscópio (quadro de referência do dispositivo).

	Tipo de Rotação	
Eixo de rotação	Nomenclatura em Português	Nomenclatura em Inglês
X	Arfagem ou Tangagem	Pitch
Y	Rolamento, Inclinação lateral ou bancagem	Roll
Z	Guinada ou Azimute	Yaw/Azimuth

Tabela 3.2: *Nomenclatura de rotações*

[1] cálculo da posição atual pelas anteriores

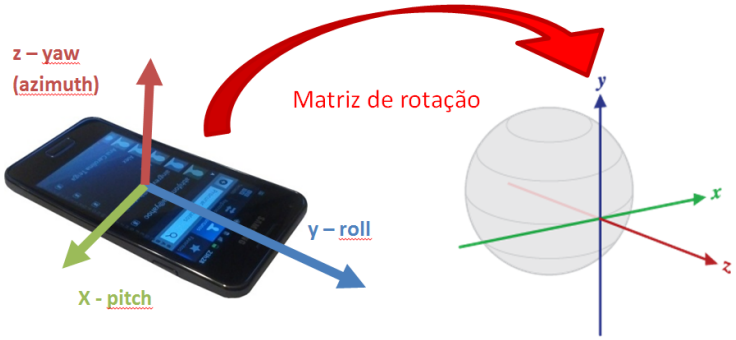

Figura 3.2: *Transposição do vetor de orientação do quadro de referência do dispositivo para o da Terra. Figura extraída e modificada de [6]*

3.3 Sistemas de posicionamento

Em virtude das limitações de vários métodos e tecnologias, existem vários sistemas e abordagens sobre localização geográfica.

Dados sobre as localizações de dispositivos podem ser importantes, por exemplo, em algoritmos de roteamento de dados em redes ad-hoc. Conforme [9], protocolos que usam localizações de nós como endereços são classificados como protocolos de "consciência geográfica"(*geographic awareness*). Nesse tipo de abordagem, existem dificuldades para serem feitos roteamentos em torno de espaços vazios e obstáculos como, por exemplo, funcionar em ambientes esparsos com densidade de rede muito baixa. Mas, por outro lado, há a vantagem dos nós nas redes necessitarem manter apenas o estado dos seus vizinhos e suportar um padrão de comunicação sem rotas explícitas estabelecidas.

Um protocolo de roteamento desse tipo pode ser viabilizado através da atribuição de coordenadas virtuais para cada nó da rede. Tais coordenadas podem ser construídas a partir do conhecimento da localização dos vizinhos, que pode ser obtida através de mensagens de aviso e mecanismos de fluxo de rede.

Para o processo de determinação de uma localização relativa, em [9] discute-se sobre um algoritmo que constrói um sistema de coordenadas local para cada nó, computando a posição dos vizinhos em relação à sua, que é definida como a origem. Em seguida, todos os sistemas locais são convertidos em um único sistema global, a partir da eleição de uma origem universal que servirá como referência para rotações de direções e transferência das coordenadas dos sistemas locais para o global.

Após esse panorama de como essa questão é abordada na camada de rede, em [18] é apresentado outro modelo independente de GPS, que busca identificar as posições de dispositivos em uma rede de área local sem fio (WLAN) a partir dos sinais veiculados na mesma. É possível medir o índice da força do sinal recebido (*received signal strength index* - RSSI) de cada ponto de acesso em uma WLAN. Com isso, as posições de um dispositivo móvel podem ser estimadas através da comparação dos valores atuais de RSSI de todos os pontos de acesso com os valores RSSI das posições armazenadas, fazendo a interpolação com as posições atuais. Como em qualquer outro sistema de posicionamento, esse também está sujeito a falhas, pois as medidas dos sinais de uma WLAN contém ruído e os ruídos aleatórios do RSSI fazem a posição estimada variar.

Ao utilizar o Wi-fi, a acurácia de posicionamento do dispositivo será similar ao alcance de acesso de um típico roteador Wi-fi (cerca de 300m em locais abertos, conforme [14]). Além da alta margem de erro em relação ao GPS (discutido a seguir), a cobertura de Wi-fi não é

completa e varia de acordo com a localização.

As seções 3.3.1 e 3.3.2 descrevem os sistemas de posicionamentos utilizados na biblioteca.

3.3.1 GPS

GPS (*Global Positioning System*) é um sistema de posicionamento global baseado na radiodifusão de sinais por satélite com o intuito de gerar os dados necessários para o cálculo do posicionamento geodésico (latitude x longitude) do dispositivo solicitante, que deve estar localizado na superfície da Terra.

Todos os satélites possuem relógios sincronizados com exatidão e conhecem suas posições no espaço a partir dos dados que são enviados a eles por controladores de sistema, permitindo-lhes formular mensagens com informações sobre o tempo de transmissão e seus posicionamentos. Uma vez formuladas e enviadas a um dispositivo receptor, cada uma delas será usada para determinar o seu tempo de transição e para computar a distância do referido dispositivo a cada satélite através da velocidade da luz. Com isso, têm-se, virtualmente, formações de esferas centradas nas posições espaciais do satélite, com a posição do receptor pertencente às suas superfícies. Computa-se então a localização geodésica do receptor com auxílio do método de trilateração (algoritmo que usa a intersecção de esferas) e equações de navegação.

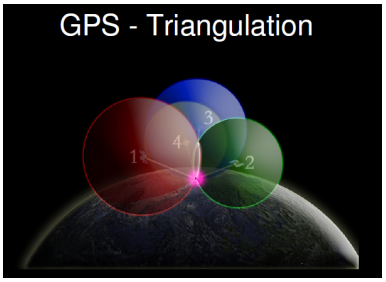

Figura 3.3: *Ilustração do algoritmo de obtenção do GPS. Imagem extraída de [1]*

Conforme [1], haviam 31 satélites em órbita em 2007 e existem 6 satélites visíveis em qualquer ponto da Terra. Normalmente, 4 ou mais satélites devem ser visíveis para obter um resultado acurado de posicionamento global. Porém, se uma variável (tal como a altitude ou valores anteriores de posicionamento) já é conhecida, um receptor pode determinar sua posição usando apenas 3 satélites.

Infelizmente, o uso do GPS está sujeito a uma série de problemas. Em uma rede móvel, pode haver situações onde os nós não possuem capacidade de GPS; ou em que sinais de satélites são bloqueados por obstáculos, como em ambientes internos; ou, ainda, em que o receptor de GPS é afetado por ruídos. As condições do tempo influenciam bastante na qualidade dos sinais (conforme abordado na Seção 4.1.2), sendo que, em 2012, existem aparelhos de GPS civis que, sob o céu limpo, possuem uma média de acurácia de mais ou menos 5m. passando para 15 ou mais metros em tempos nublados.

3.3.2 AGPS

A ativação do GPS está sujeita a um período de inicialização do sistema chamado *Time To First Fix* - TTFF, que ocorre devido à busca por dados orbitais. O AGPS (*Assisted* GPS) corresponde a um sistema para obtenção do posicionamento global de maneira assistida, permitindo a redução do referido atraso.

Enquanto as operações autônomas de GPS usam apenas sinais de rádio gerados por satélites, o AGPS usa, adicionalmente, recursos de rede (tais como GSM, CDMA, WCDMA, LTE ou Wi-fi) para a localização deles. Mesmo em condições ambientais impróprias que prejudiquem a intensidade dos sinais gerados por satélites, como, por exemplo, na presença de obstáculos físicos ou condições atmosféricas indesejáveis, o uso do AGPS não é prejudicado.

A assistência é feita amparada por uma infraestrutura que integra, além do dispositivo móvel e dos satélites, receptores de GPS, servidores e torres de celulares, tal como ilustrado na Figura 3.4. Conforme [4], muito frequentemente as torres de rede de celulares tem receptores de GPS ou uma estação de base próxima.

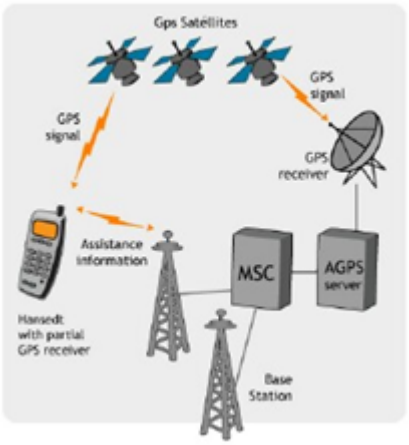

Figura 3.4: *Ilustração acerca do AGPS e sua infraestrutura de funcionamento. Imagem extraída de [16]*

O funcionamento de um sistema com AGPS age segundo dois modos de operação. Em ambos os casos, o dispositivo recebe assistência de dados do servidor de AGPS e, com o auxílio dos mesmos, recebe sinal dos satélites visíveis e envia as medidas para o servidor. Se o modo de operação for do tipo *Mobile Station Assisted* (MSA), o servidor calcula a posição e envia ao dispositivo. Caso contrário, se for do tipo *Mobile Station Based* (MBS), então o próprio AGPS responsabiliza-se pelo cálculo de posição a partir dos dados enviados pelo servidor.

Os receptores de GPS travam sua localização junto aos satélites constantemente, por serem atualizados com os dados orbitais e almanaques providos pelos mesmos. O acesso "imediato"a essa informação através da rede, permite tempos de resposta mais rápidos e precisos do que quando obtidos diretamente por sinais de satélite, tal como ocorre nos serviços autônomos de GPS.

Além de otimizar o tempo de obtenção e cálculo do posicionamento, o sistema AGPS garante, aos dispositivos, menores quantidades de processamento e menor consumo de bateria, pois o servidor de assistência possui um bom sinal de satélite e excelente poder computacional.

Esse sistema é usado extensivamente por celulares. Sendo que alguns dispositivos possuem tanto o GPS quanto o AGPS, ou são integrados com outros serviços de localização, como posicionamento por Wi-fi e triangulação de celulares.

3.3.3 Cálculos geodésicos

Geodésia é a ciência que se ocupa da medição e representação da superfície da Terra. Nosso planeta possui um formato elipsoidal e o quadro de coordenas padrões que o descreve

tal como uma elipse é o WGS84 [11], correspondente ao sistema geodésico mundial criado em 1984, sujeito a revisões em anos posteriores, e usado por GPS.

Baseado na Figura 3.5, são apresentados os conceitos comuns relacionados à ciência em questão. Uma coordenada geodésica corresponde a um ponto formado por latitude(φ) e longitude(λ) situado na superfície de um elipsoide. As geodésicas correspondem às curvas de menor distância entre duas coordenadas Geodésicas (ponto A ao ponto B). O azimute(α) é o ângulo formado entre a geodésica e o meridiano, medido no sentido horário a partir do norte (zero graus). Ou seja, o azimute fornece uma direção de uma coordenada geodésica à outra.

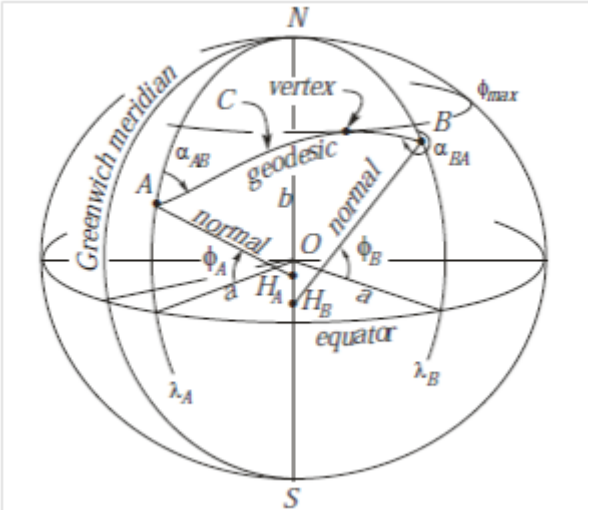

Figura 3.5: *Curva geodésica sobre um elipsoide. Figura extraída de [12]*

Os principais problemas dos cálculos geodésicos são:

- Problema direto: através de uma coordenada geodésica na superfície de um elipsoide, de um azimute inicial e de uma distância geodésica, obter a coordenada geodésica final.
- Problema inverso: através de duas coordenadas geodésicas na superfície do elipsoide, obter a distância e o azimute entre eles.

Para cada problema referenciado acima é possível também obter o seu azimute reverso (direção do ponto final ao inicial), levando-se em consideração que o módulo dos azimutes não serão os mesmos, pois os meridianos da Terra não são paralelos.

Ao longo de anos, uma considerável quantidade de trabalho foi realizada sobre cálculos geodésicos. Havendo contribuições, por exemplo, em 1805 com Puissant, em 1955 com Rainsford, em 1975 com Vicenty, em 1980 com Bomford e com Jank e Kivioja, em 2004 com Leick, entre muitas outras.

O método antigo de Puissant (século XIX), que se baseia em computações de posições intermediárias para os cálculos geodésicos, é completamente acurado para distâncias acima de 96,5 km.

Já o método de Jank e Kivioja (1980), muito mais moderno, permite, segundo a intervenção do usuário, resultados bastante acurados para qualquer tipo de distância, porém sua complexidade algorítmica é maior. O método em questão utiliza relações de geometria diferencial sobre triângulos elipsoidais que são pequenos suficientes para serem tratados como triângulos planos. Através desses recursos, a distância geodésica é dividida em muitos incrementos, sendo a quantidade diretamente proporcional à acurácia que se deseja obter.

Conforme T.Vicenty [24], que apresentou uma solução bastante utilizada em computação, os erros na obtenção de azimutes são grandes para pequenas distâncias, devido ao arredondamento das coordenadas dos pontos, e decrescem, progressivamente, até 10.000km, passando a aumentar após esse comprimento. Mais ou menos nesse limiar, o erro de posicionamento do ponto final, em função do azimute, alcança um valor máximo de 0,3mm por unidade da 5ª casa decimal de segundos das medidas de latitude e longitude.

3.4 Filtro de erros dos sensores

Em virtude dos inúmeros problemas inerentes à manipulação dos sensores abordados na Seção 3.2, existem variadas formas, propostas e estudos sobre a filtragem desses erros. Há dois métodos principais para essa finalidade e que costumam ser base para inúmeros outros: o fitro Kalman e o filtro Complementar, sendo o primeiro fundamentado no domínio do tempo e o segundo, no domínio da frequência.

O popular filtro Kalman, também conhecido como estimação quadrática linear, corresponde a um modelo matemático recursivo que se utiliza de dados com ruídos e observados anteriormente, para a obtenção de uma estimativa estatisticamente ótima dos novos valores do sistema.

Enquanto o anterior é solução para diversos tipos de sistemas, o filtro complementar é específico para o problema dos sensores. Seu funcionamento é divido pela atuação de 2 filtros, o filtro passa-baixa e o passa-alta. O primeiro objetiva filtrar (atenuar) as amplitudes de frequência maiores que a de corte, permitindo a passagem de baixas frequências sem dificuldades. Enquanto o segundo, obviamente, é responsável pelo processo inverso.

Esses filtros de frequência são muito comuns em diversas situações. Por exemplo, no sistema de navegação para pedestre proposto em [19], optou-se pelo filtro passa-baixa para compensar os erros individuais ocasionados por caminhadas, permitindo a eliminação das oscilações típicas das mesmas ao aplicá-lo sobre acelerações adicionais.

Em relação ao Kalman, o filtro Complementar possui resultados menos acurados, mas muito bons na prática, e exige uma capacidade de processamento muito menor, podendo ser submetido a altos valores de frequência de operação. Como concluído em [16], a maior limitação do filtro Kalman estendido implementado foi a intensidade do mesmo, respondendo por mais que 50% do consumo da CPU. Porém, utilizá-lo em baixa frequência para contornar essa situação, implicaria em imprecisões ou divergências decorrentes do processo de integração dos dados.

Essas características direcionaram a escolha do filtro Complementar para ser implementado na biblioteca, merecendo, portanto, uma abordagem mais detalhada na Seção 3.4.1.

3.4.1 Filtro complementar

A ideia principal do filtro Complementar é fazer uma fusão dos valores obtidos pelos sensores acelerômetro e giroscópio, com a finalidade de gerar um ângulo de rotação que tenha os erros peculiares de cada sensor filtrados durante sua composição. A Figura 3.6 apresenta a fórmula de composição desse ângulo, cujos procedimentos são enumerados abaixo:

1. Realiza-se a integração da posição anterior com a nova variação angular (taxa de giro em rad/s multiplicado pelo intervalo de tempo de amostragem em segundos) obtida pelo giroscópio.

2. Define-se uma constante de tempo relativa à duração do sinal digital, estabelecendo-se assim uma frequência de corte. Conforme [15], o filtro complementar é uma ótima

aproximação quando o intervalo de tempo definido é muito maior que o intervalo de amostragem, o que tornaria o ângulo final formado muito mais dependente do giroscópio que do acelerômetro. A escolha dessa constante deve estar sujeita a um *tradeoff* gerado através de experimentos para cada tipo de situação.

3. Aplica-se um filtro passa-alta sobre o resultado do item (1), para evitar erros de bias e drift do giroscópio, que são componentes de ruído de baixa frequência. O retângulo rosa da Figura 3.7 marca a parte do gráfico correspondente à correção dos erros de drift.

4. Aplica-se um filtro passa-baixa sobre os sinais dos acelerômetros, para tornar o ângulo gerado menos dependente dos ruídos ligados à ocorrência de aceleração horizontal quando não há rotação, que são componentes de alta frequência. O retângulo azul da Figura 3.7 marca a parte do gráfico correspondente à correção dos distúrbios da aceleração horizontal.

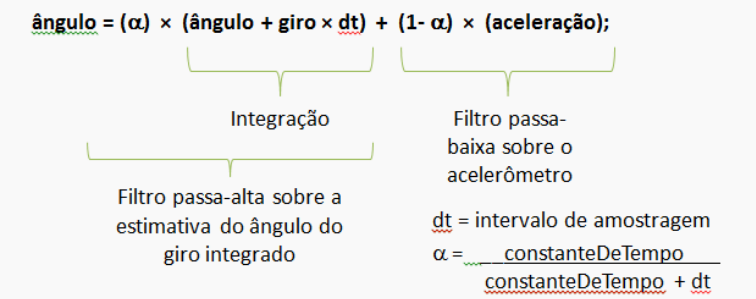

Figura 3.6: *Fórmula de obtenção do ângulo de orientação através do filtro Complementar*

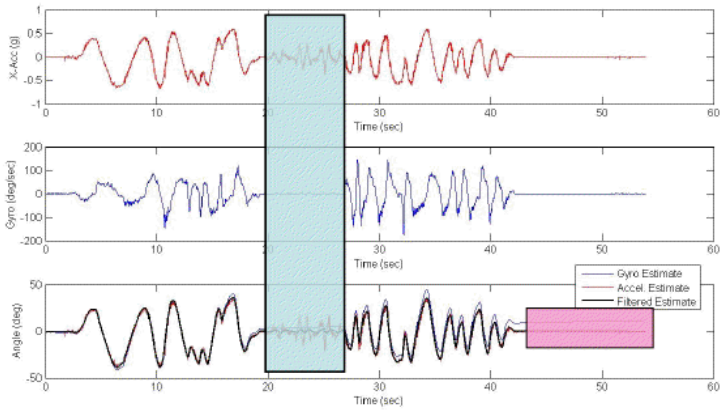

Figura 3.7: *Gerenciamento dos erros pelo filtro Complementar. Figura extraída de [10].*

3.4.2 Outras soluções

Um sistema de navegação inercial utiliza as medidas de sensores de movimentação (giroscópios e acelerômetros) para computar a posição, orientação e velocidade de objetos móveis via *dead reckoning*.

Existem inúmeras propostas, limitações e estudos sobre a criação desses sistemas. Seguem algumas citações sobre trabalhos desenvolvidos nessa área:

- D.E. Gebre-Egziabher, J.D. Powell e R.C.Hayward escreveram o artigo *"Design of multi-sensor attitude determination systems"*, no qual se discute o uso de quatérnios (sistema numérico que estende os números complexos) e algoritmos baseados em Euler.

- Um método baseado em DCM (*Direction Cosine Matrix*) para estimativa de orientação e comportamento foi proposto por Y.-S. S. Nguyen Ho Quoc Phuong, Hee-Jun Kang e Y.-S. Ro.

- Um filtro Kalman estendido baseado na fusão de algoritmos é discutido por P. Setoodeh, A. Khayatian, e E. Farjah. Uma sistemática aproximação baseada na decomposição em ondas pequenas é usada para estimar covariações de ruídos usadas na formulação do filtro Kalman.

- Uma outra proposta de filtro Kalman estendido foi feita por Nisarg Kothari em [16]. Esse filtro corresponde a um método Bayesiano para estimativas de estado não lineares, que controla os desvios da calibragem do sensor giroscópio e modifica os parâmetros de ruído dos sensores acelerômetro e magnetômetro caso haja desvios de propriedades dos sinais dos valores esperados. O artigo discute ainda sobre o controle da orientação do celular através do uso de quatérnios, para evitar o problema associado com a singularidade dos ângulos de Euler. A Tabela 3.3 atesta a qualidade do uso do filtro ao compará-lo com a utilização dos sensores sem tratamento de erros.

- Uma forma de integração da bússola digital e do giroscópio para navegação de pedestres proposta por Q. Ladetto e B. Merminod em [20], demonstrou a possibilidade de detectar e compensar distúrbios magnéticos com o giroscópio e de compensar o bias utilizando a bússola digital e o filtro Kalman.

	Erro médio aproximado		
Teste	**Acelerômetro + Magnetômetro**	**Giroscópio**	**Filtro Kalman**
rotação 360º	8º	2º	2º
rotações aleatórias (10s)	16º	10º	5º

Tabela 3.3: *Comparação dos erros relativos a métodos de obtenção de ângulo de rotação através de sensores. Tabela extraída e traduzida de [16].*

Capítulo 4

Experimentos

Antes de iniciar a implementação da biblioteca, foi necessário avaliar a precisão dos sensores dos celulares, já que a localização e a orientação dos dispositivos serão cruciais para o seu correto funcionamento. Este capítulo resume as medições que foram realizadas.

Os seguintes recursos são comuns a todos os experimentos:

- Um programa auxiliar, escrito em linguagem Java e com recursos da API do Android, que registra as informações lidas da bússola e do GPS;
- Utilização de cinco dispositivos móveis, sendo quatro celulares dos modelos LG-P350f (duas unidades), Samsung Galaxy SII e Samsung GT-S5570B; e um tablet do modelo Ideos S7.

Com respeito aos experimentos através dos quais foram extraídos dados dos sensores de celulares (GPS, magnetômetro e acelerômetro), é importante salientar que, além das imperfeições de hardware/software na obtenção dos dados, o que será demonstrado mais adiante, podem ter havido também margens de erro durante o processo de manuseio dos dispositivos e/ou na categorização/agrupamento posterior dos dados coletados. Por exemplo, podem ter havido imprecisões ao reconhecer se um dispositivo estava parado ou em movimento durante os testes em campo, pois, em ambos os casos, os sensores dos dispositivos podem estar identificando novos valores, como uma nova direção ou uma nova latitude.

As Seções 4.1 e 4.2 apresentam os dados e conclusões provenientes das medições realizadas, respectivamente, em experimentos com GPS e bússola digital. A Seção 4.3 aborda sobre tráfego de pacotes na rede, com o auxílio do programa Wireshark.

4.1 GPS no Android

A fim de realizar a persistência dos dados relativos ao posicionamento geográfico (latitude e longitude) de diferentes celulares, foi implementada uma aplicação que armazena, no sistema de arquivos do celular, o instante de tempo, a posição atual do dispositivo e sua acurácia. Abaixo são apresentados trechos do código referentes à aquisição e atualização periódica de coordenadas geográficas. O código completo está disponível em http://www.linux.ime.usp.br/~avila/mac499/codigo.html.

```
1     /*Gerenciamento das atualizações do GPS*/
2 public void configureLocationManager() {
3   /*intervalo de tempo mínimo para disparo do evento, em milisegundos*/
4   long minTime = 500;
5   /*distância mínima para disparo do evento, em metros*/
6   float minDistance = 0;
7
8   /*obtém atualizações periódicas da localização geográfica do dispositivo
        */
9   LocationManager lm = (LocationManager)
10          getSystemService(Context.LOCATION_SERVICE);
11  lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
12          minTime, minDistance, this);
13 }
14
15 /*disparo do evento a cada mudança de localização sujeita ao provedor de
      GPS (LocationManager.GPS_PROVIDER) e a uma distância e duração mínimas
      */
16 public void onLocationChanged(Location location) {
17   String latitude = String.valueOf(location.getLatitude());
18   String longitude = String.valueOf(location.getLongitude());
19   String accuracy = String.valueOf(location.getAccuracy());
20   /* (...) */
21 }
```

As medições foram realizadas em uma quadra poliesportiva (Figura 4.1) localizada no Centro Poliesportivo da USP - CEPEUSP, que fica instalado no bairro Butantã, da cidade de São Paulo, SP.

As duas próximas subseções descrevem duas atividades de campo, sendo a primeira utilizada para fazer uma comparação da qualidade do GPS em diferentes marcas e modelos de dispositivos móveis, e a segunda utilizada para verificar a qualidade dos dados derivados do mesmo (acurácia e azimute relativo entre duas coordenadas).

4.1.1 Atividade 1: Comparando dispositivos

Durante essa atividade realizada em campo, todos os dispositivos foram fixados lado a lado em uma plataforma paralela ao chão, separados uns dos outros por poucos centímetros. Levando em consideração que a plataforma na qual os aparelhos estavam fixos possui uma área retangular medindo 30 x 40 cm, trabalhamos com uma margem de erro muito pequena em relação à distância de um aparelho ao outro. A distância física de um dispositivo ao outro será sempre menor que:

TAXA DE ERRO
= Diagonal da plataforma
= RAIZQUADRADA((Largura da plataforma)^2 + $(Alturadaplataforma)^2$)
= $50cm$.

Todas movimentações desses dispositivos foram realizadas com o auxílio de um patinete, no qual foi fixada a referida plataforma, e foram feitas sobre as marcações da própria quadra, orientando-se principalmente pela linha mais externa da mesma, correspondente ao limite da quadra de futebol de salão. Foram feitas também marcações extras para identificação de pontos de parada com o auxílio de fita crepe e trena.

Nas atividades realizadas, procuramos contemplar uma variedade de situações possíveis de ocorrerem em relação ao manuseio do aparelho por um usuário. Cada situação, com os respectivos resultados e conclusões, está descrita nas subseções abaixo.

Figura 4.1: *Foto de satélite do ambiente nos quais foram feitos os testes de campo. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20*

Os resultados são expressos por gráficos de dispersão que registram as posições coletadas por GPS. Esses gráficos estão estruturados da seguinte forma:

- o eixo horizontal correspondente aos pontos de longitude;
- o eixo vertical correspondente aos pontos de latitude;
- em ambos os eixos seus pontos são separados por uma distância de $X*10^{-5}$ graus. O que equivale a uma distância de $X*1,1131949$ metros, conforme informação obtida através de métodos da classe Location da API do Android. Sendo X = Módulo((Latitude do ponto 1) - (Latitude do ponto 2)) ou X = Módulo((Longitude do ponto 1) - (Longitude do ponto 2)) Como exemplo, da latitude -23,56115 a -23,56110 temos $5*10^{-5}$ graus de diferença, equivalente a X = $5*1,1131949$ metros = 5,5659745 metros.
- cada curva plotada no gráfico equivale a um caminho percorrido;
- cada ponto equivale à posição na qual o dispositivo permaneceu parado por alguns poucos segundos. As paradas duraram cerca de 20s nos experimentos com GPS e cerca de 10s com a bússola.

Experimento 1: Movimento com paradas a cada 2 metros

O experimento em questão correspondeu ao deslocamento dos dispositivos em três linhas retas consecutivas, com paradas por alguns segundos a cada 2 metros, conforme esquema representado sobre a foto de satélite do local na Figura 4.2. Cada linha reta possui 8 pontos de parada, sendo de aproximadamente 2 metros a distância entre 2 pontos seguidos.

A figura 4.3 exibe gráfico com o resultado obtido a partir das posições geográficas coletadas. Importante notar que no referido gráfico não há plotagem de posições coletadas entre dois pontos de parada, portanto são eles que definem o formato da curva que descreve o movimento.

A melhor aproximação do trajeto real foi obtido pelo dispositivo Samsung Galaxy SII, cuja curva representante do percurso do mesmo (linha vermelha) pode ser desmembrada em três linhas correspondentes às linhas retas da quadra (Figura 4.2). Apesar de apresentar o melhor resultado, notam-se algumas imperfeições: a curva pode ser desmembrada em três linhas que não são retas; e existem alguns pontos de parada que não foram reconhecidos pelo programa.

Com os outros dispositivos nota-se uma maior disparidade em relação ao trajeto real. Por isso, escolhemos o *Samsung Galaxy SII* para ser utilizado nos dois próximos testes de campo.

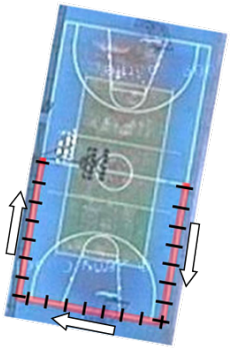

Figura 4.2: *Foto de satélite da quadra na qual foram feitos os testes de campo. Cada linha contínua vermelha equivale a um deslocamento e cada linha preta perpendicular à vermelha equivale a um ponto de parada. As setas indicam o percurso percorrido. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação*

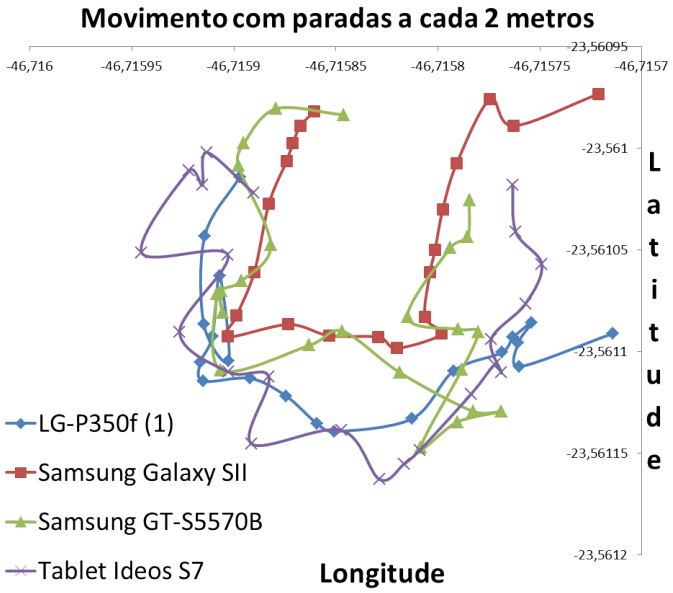

Figura 4.3: *Gráfico que descreve um movimento com paradas a cada 2 metros realizado por 4 dispositivos móveis, a partir de posições coletadas do GPS.*

Como entre dois pontos de parada consecutivos em uma mesma reta existe uma distância de 2 metros no trajeto real, pode-se levantar algumas estimativas. As Tabelas 4.1, 4.2 e 4.3 apresentam a média e o desvio padrão a cerca dessas distâncias para os 4 dispositivos que participaram da atividade. O valor lido pelo *Samsung Galaxy SII* também é apresentado nas tabelas. Cada tabela representa uma das retas com 8 pontos de parada nas quais os dispositivos foram deslocados, conforme Figura 4.2. Os pontos e retas estão ordenados conforme a sequência realizada no percurso.

Uma média razoável deveria ser algo em torno de 2 metros, o que atesta a falta de precisão das posições coletadas.

Experimento 2: Movimento de ida e volta

O experimento em questão correspondeu ao deslocamento dos dispositivos em três linhas retas consecutivas, nos dois sentidos (ida e volta), com apenas uma parada para a inversão do movimento, conforme esquema representado sobre a foto de satélite do local na Figura

1ª RETA			
Descrição	**Média(m)**	**Desvio padrão(m)**	**Samsung Galaxy SII(m)**
Distância do 1º ao 2º ponto	3,803333	2,110695	2,53
Distância do 2º ao 3º ponto	1,176666	0,39879	NÃO REGISTRADO
Distância do 3º ao 4º ponto	2,335	1,117691	1.07
Distância do 4º ao 5º ponto	2,183333	0,525769	NÃO REGISTRADO
Distância do 5º ao 6º ponto	1,466666	0,334714	2.68
Distância do 6º ao 7º ponto	1,616666	0,701522	2.27
Distância do 7º ao 8º ponto	1,9725	1,154682	3.33

Tabela 4.1: *Estatísticas sobre a distância entre pontos de paradas consecutivos para a 1ª reta percorrida pelos dispositivos durante o experimento 1*

2ª RETA			
Descrição	**Média(m)**	**Desvio padrão(m)**	**Samsung Galaxy SII(m)**
Distância do 1º ao 2º ponto	1,875	0,786066	0.93
Distância do 2º ao 3º ponto	2,583333	0,990168	1.90
Distância do 3º ao 4º ponto	2,295	1,560118	2.68
Distância do 4º ao 5º ponto	3,083333	1,452458	1.87
Distância do 5º ao 6º ponto	2,09	0,815721	0.76
Distância do 6º ao 7º ponto	3,5175	1,630488	0.78
Distância do 7º ao 8º ponto	2,7375	0,520857	0.91

Tabela 4.2: *Estatísticas sobre a distância entre pontos de paradas consecutivos para a 2ª reta percorrida pelos dispositivos durante o experimento 1*

4.4.

Na Figura 4.5 é apresentado um gráfico com o resultado obtido a partir das posições geográficas coletadas. A incompatibilidade das curvas torna clara a falta de precisão de uma determinada posição quando adquirida em momentos diferentes.

Experimento 3: Dispositivos parados

O experimento em questão correspondeu à permanência dos dispositivos em um mesmo local por um período aproximado de 20 minutos, conforme esquema representado sobre a foto de satélite do local na Figura 4.6.

3ª RETA			
Descrição	**Média(m)**	**Desvio padrão(m)**	**Samsung Galaxy SII(m)**
Distância do 1º ao 2º ponto	1,3775	1,125267	4,79
Distância do 2º ao 3º ponto	2,045	1,288216	1,63
Distância do 3º ao 4º ponto	2,94	1,185271	3,00
Distância do 4º ao 5º ponto	3,15	0,913126	1,90
Distância do 5º ao 6º ponto	1,12	0,550151	1,58
Distância do 6º ao 7º ponto	1,8875	1,1201	0,89
Distância do 7º ao 8º ponto	2,62	1,191554	1,70

Tabela 4.3: *Estatísticas sobre a distância entre pontos de paradas consecutivos para a 3ª reta percorrida pelos dispositivos durante o experimento 1*

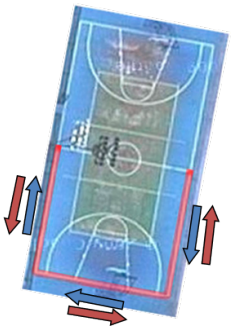

Figura 4.4: *Foto de satélite da quadra na qual foram feitos os testes de campo. A linha contínua vermelha equivale a dois movimentos, o primeiro no sentido das setas azuis (ida) e o segundo no sentido das vermelhas (volta). Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação.*

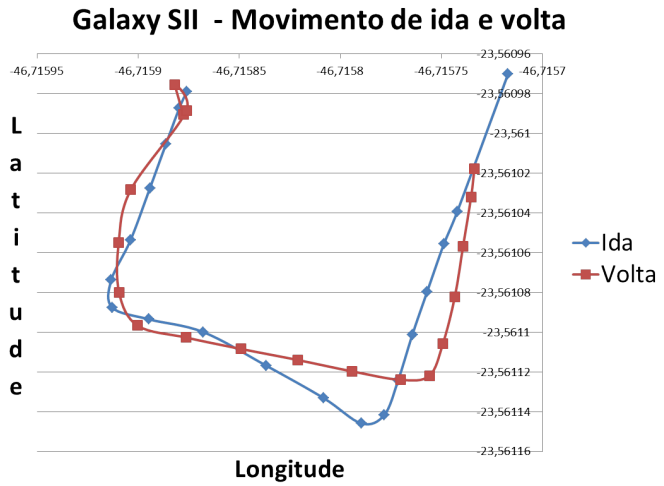

Figura 4.5: *Gráfico que descreve dois movimentos contínuos realizados pelo dispositivo Samsung Galaxy SII, a partir de posições coletadas do GPS.*

A Figura 4.7 apresenta um gráfico com o resultado obtido a partir das posições geográficas coletadas. Nesse último gráfico ocorre uma evidente dispersão entre as coordenadas que foram capturadas para uma mesma posição, o que denuncia uma falta de precisão preocupante para essa situação. O modelo S2, que apresentou as melhores curvas nos experimentos com movimento, foi o que apresentou mais variações de posicionamentos quando parado, tendo a distância entre os seus dois pontos mais extremos medindo 9,7 metros.

Essa discrepância entre os pontos é ainda mais nítida ao apontarmos a média e desvio padrão (ver Tabela 4.4) dos pontos do gráfico, lembrando que a distância entre dois aparelhos não ultrapassa 50cm (definido na Seção 4.1.1). É importante observar que o celular *LG-P350f* e o *tablet Ideos S7* apresentaram praticamente as mesmas leituras em todas as medições, por isso apenas um ponto é visível no gráfico da Figura 4.7 para esses dois dispositivos.

Latitude média: -23,560981253496181 graus
Longitude média: -46,715815001747865 graus
Desvio padrão da latitude: $2,86203424324607 * 10^{-5}$ graus ($\pm$ 3,18m)
Desvio padrão da longitude: $2,31325227761388 * 10^{-5}$ graus ($\pm$ 2,58m)

Tabela 4.4: *Estatísticas sobre as coordenadas geográficas presentes no gráfico da Figura 4.7*

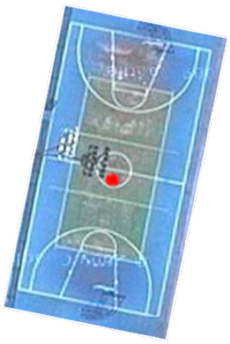

Figura 4.6: *Foto de satélite da quadra na qual foram feitos os testes de campo. O círculo vermelho indica o local no qual os celulares foram posicionados. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação.*

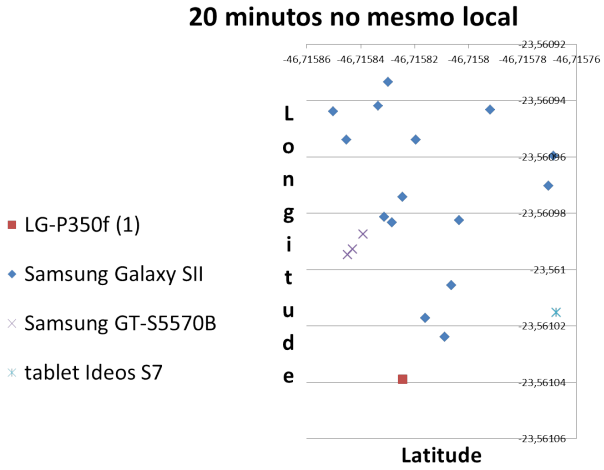

Figura 4.7: *Gráfico com as posições coletadas do GPS de 4 dispositivos móveis mantidos em um mesmo local físico por aproximadamente 20 minutos.*

4.1.2 Atividade 2: Analisando atributos

Foram realizados quatro experimentos com dispositivos móveis dos modelos *Samsung Galaxy SII* e *Samsung GT-S5570B*, com os seguintes passos em comum:

1. Um dos dispositivos foi mantido fixo;
2. O outro dispositivo é afastado do fixo, sequencialmente, em 0cm, 25cm, 50cm, 1m, 2m, 4m, 8m, 16m e 28m, mantendo a mesma direção;
3. Os dois dispositivos são sincronizados para coletarem os dados referentes a latitude, longitude e acurácia durante um certo intervalo de tempo, em cada posição discriminada no item acima.

A Tabela 4.5 apresenta a descrição de cada evento em relação aos passos acima descritos.

Atributo 1: Acurácia

Cada localização coletada pelo GPS de dispositivos Android possui uma estimativa de acurácia atrelada. Conforme documentação em [6], a acurácia corresponde a um raio de 68% de confiança, medido em metros. Isso significa que existe 68% de probabilidade de que a

Experimento	Dispositivo fixo	Dispositivo com múltiplas posições	Intervalo de tempo	Condições climáticas
1	*GT-S5570B*	*Galaxy SII*	2 minutos	Parcialmente nublado
2	*GT-S5570B*	*Galaxy SII*	2 minutos	Céu limpo na maior parte do tempo, com algumas nuvens carregadas passageiras
3	*Galaxy SII*	*GT-S5570B*	10 segundos	Céu limpo
4	*GT-S5570B*	*Galaxy SII*	10 segundos	Céu limpo

Tabela 4.5: *Descrição de experimentos para testar a qualidade do GPS*

verdadeira localização esteja contida no círculo formado pelo raio da acurácia e centrado nos pontos de latitude e longitude retornados.

Conforme trecho de código fonte presente no início desta Seção, a análise em questão trabalha sobre os dados coletados através da sentença

```
1 float accuracy = location.getAccuracy();
```

Conforme pode ser observado na Tabela 4.6, existem estimativas de acurácia distribuídas em um intervalo de 5 a 50 metros de raio, sendo que cerca de 87% dos valores mantém-se abaixo dos 10m e o valor mínimo é de 5m.

Acurácia(m)	ocorrências
5	573
7	831
10	683
12	199
15	39
17	34
20	12
22	21
25	4
35	1
40	2
45	2
50	3

Tabela 4.6: *Número de ocorrências de leituras de acurácia durante todos os experimentos*

É importante notar que na maioria das vezes que ocorreram estimativas de acurácia acima dos 25m, tratou-se de casos nos quais uma mesma coordenada geodésica (latitude x longitude) variou sua acurácia sem renovar sua localização global, ou seja, para uma mesma posição, foram estimados mais do que um valor de acurácia, conforme Tabela 4.7. Na amostra em questão, localizações com acurácias de raio grande estiveram, normalmente, associadas a raios de acurácia menores em momentos anteriores.

Estimativas de acurácia em apenas uma localização					
média	mínima	máxima	desvio padrão	Qtde. estimativas	ocorrências
5	5	5	0	1	554
7	7	7	0	1	830
7,5	5	10	3,535534	2	10
9,66	7	12	2,516611	3	1
10	5	15	5	3	2
10	5	15	7,071068	2	3
10	10	10	0	1	665
12	12	12	0	1	197
12,5	10	15	3,535534	2	1
15	5	25	9,128709	4	1
15	5	25	14,14214	2	1
15	15	15	0	1	31
17	17	17	0	1	33
17,25	12	25	5,560276	4	1
20	20	20	0	1	11
21,66	5	50	24,66441	3	1
22	22	22	0	1	21
27,5	5	50	23,27373	4	1
28,33	10	45	14,02379	6	1
40	40	40	0	1	1
50	50	50	0	1	1

Tabela 4.7: *Número de ocorrências de leituras de acurácia durante todos os experimentos da Tabela 4.5, mostrando estatísticas que envolvem casos com mais de uma estimativa de acurácia para uma mesma localização global.*

Conforme Tabela 4.8, os experimentos evidenciam também o quanto as condições climáticas podem interferir nos resultados do GPS. O experimento nº1, que teve a maior quantidade de nuvens durante a sua execução, apresentou uma acurácia pior do que o nº 2, registrando um raio médio de confiança com cerca de 3m a mais do que a do 2º experimento.

Acurácia (m)		
Média	Desvio padrão	Experimento
9,645073	3,175093	1
6,558639	2,179945	2

Tabela 4.8: *Estatísticas sobre dois experimentos semelhantes constantes na Tabela 4.5, com condições climáticas distintas.*

Atributo 2: Azimute relativo a duas coordenadas geodésicas

Os conceitos abordados nesta Seção fazem referência ao conteúdo abordado na Seção 3.3.3.

Através dos dois primeiros experimentos discriminados na Subseção anterior e referenciados na tabela 4.5 como experimentos 1 e 2, foram analisados o azimute entre as coordenadas das localizações dos dois dispositivos, conforme Tabelas 4.9 e 4.10.

Conforme descrição dos experimentos na Seção anterior, o dispositivo móvel é afastado do outro por uma linha reta. Como a direção permanece a mesma, então é esperado que o azimute também seja mantido. Além disso, o valor esperado do azimute nos dois experimentos é por volta de 146°, valor que foi apurado com a identificação manual das coordenadas geodésicas utilizadas no experimento, através da ferramenta *Google Maps* (https://maps.google.com/), e calculando o azimute entre os mesmos.

Azimute (graus)					
média	desvio padrão	leituras	posição (m)	dispositivo fixo	dispositivo móvel
101,78	53,26	118	0	*GT-S5570B*	*Samsung SII*
296,72	14,07	122	0,25	*GT-S5570B*	*Samsung SII*
52,85	27,50	122	0,5	*GT-S5570B*	*Samsung SII*
76,64	8,39	122	1	*GT-S5570B*	*Samsung SII*
75,5367	4,34	121	2	*GT-S5570B*	*Samsung SII*
121,93	2,00	68	4	*GT-S5570B*	*Samsung SII*
101,16	8,74	122	8	*GT-S5570B*	*Samsung SII*
118,12	5,63	122	16	*GT-S5570B*	*Samsung SII*
120,16	1,05	122	28	*GT-S5570B*	*Samsung SII*

Tabela 4.9: *Cálculo do azimute relativo entre dois dispositivos móveis, durante o "experimento 1"referenciado na Tabela 4.5.*

Azimute (graus)					
média	desvio padrão	leituras	posição (m)	dispositivo fixo	dispositivo móvel
194,58	6,17	121	0	*GT-S5570B*	*Samsung SII*
236,97	16,39	121	0,25	*GT-S5570B*	*Samsung SII*
193,04	1,97	121	0,5	*GT-S5570B*	*Samsung SII*
178,71	6,24	122	1	*GT-S5570B*	*Samsung SII*
179,00	17,71	121	2	*GT-S5570B*	*Samsung SII*
192,66	9,27	121	4	*GT-S5570B*	*Samsung SII*
145,07	6,58	121	8	*GT-S5570B*	*Samsung SII*
126,64	12,90	121	16	*GT-S5570B*	*Samsung SII*
149,93	3,83	121	28	*GT-S5570B*	*Samsung SII*

Tabela 4.10: *Cálculo do azimute relativo entre dois dispositivos móveis, durante o "experimento 2"referenciado na Tabela 4.5.*

Analisando as tabelas, conclui-se que o azimute se afasta bastante do valor ideal, 146°, quando os dispositivos estão afastados um do outro em até 2 metros de distância. Após 4 ou 8 metros, é nítida a melhor aproximação desse valor, principalmente em relação ao experimento 2 (Tabela 4.10), que apresentou condições climáticas mais favoráveis à qualidade do sinal do GPS.

4.2 Bússola digital no Android

A fim de realizar a persistência dos dados relativos à orientação espacial do dispositivo, que são vetores tridimensionais (dimensionados nos eixos x, y e z) construídos a partir de sensores de movimentação (acelerômetro ou giroscópio) e posicionamento (magnetômetro),

foi escrita uma aplicação que armazena no sistema de arquivos do celular o instante de tempo e os dados do vetor. Abaixo são apresentados trechos do código referentes à criação do vetor de orientação com auxílio do magnetômetro e acelerômetro, sensores que foram utilizados para todos os experimentos desta seção. O código completo está disponível em http://www.linux.ime.usp.br/~avila/mac499/codigo.html.

```
1 /*guarda a aceleração em m/s^2 relativas aos eixos físicos x, y e z*/
2 float[] gravityAxes = new float[3];
3 /*guarda o campo magnético em (10e^(-6)) Tesla relativos
4 aos eixos físicos x, y e z*/
5 float[] geomagneticAxes = new float[3];
6 /*guarda a matriz de rotação*/
7 float[] rotationMatrix new float[3];
8 /*guarda o vetor tridimensional relativo à orientação do dispositivo*/
9 float[] orientation = new float[3];
10
11
12 public void onSensorChanged(SensorEvent evt) {
13     /*tipo do evento*/
14     int type=evt.sensor.getType();
15
16     if (type == Sensor.TYPE_MAGNETIC_FIELD) {
17     geomagneticAxes[0] = evt.values[0]; /*x*/
18     geomagneticAxes[1] = evt.values[1]; /*y*/
19     geomagneticAxes[2] = evt.values[2]; /*z*/
20     ...
21     }
22     if (type == Sensor.TYPE_ACCELEROMETER) {
23       gravityAxes[0] = evt.values[0]; /*x*/
24       gravityAxes[1] = evt.values[1]; /*y*/
25       gravityAxes[2] = evt.values[2]; /*z*/
26     ...
27     }
28     /* (...) */
29     /*constrói a matriz de rotação*/
30     SensorManager.getRotationMatrix(rotationMatrix, null, gravityAxes,
           geomagneticAxes);
31     /*define a orientação a partir da matriz de rotação*/
32     SensorManager.getOrientation(rotationMatrix, orientation);
33 }
```

Foi criado um ambiente para medições composto por quatro celulares dos seguintes modelos: *LG-P350f* (duas unidades), *Samsung Galaxy SII* e *Samsung GT-S5570B*. Cada um deles foi sujeito aos seguintes experimentos:

1. Rotação do aparelho sobre uma plataforma fixa (Figura 4.8), com paradas temporárias a cada deslocamento de 15°, ao redor do eixo ortogonal à plataforma que cruza a origem do círculo de referência de movimentação (Figura 4.9). O movimento é composto por uma volta completa de 360°, com um total de 25 paradas, considerando 0° e 360° como duas paradas distintas.

2. Rotação do aparelho tal como descrito acima, só que agora com apenas um ponto de início (0°), um ponto final (360°) e um movimento contínuo entre os dois.

Nas seções a seguir são relatados os resultados obtidos com os experimentos.

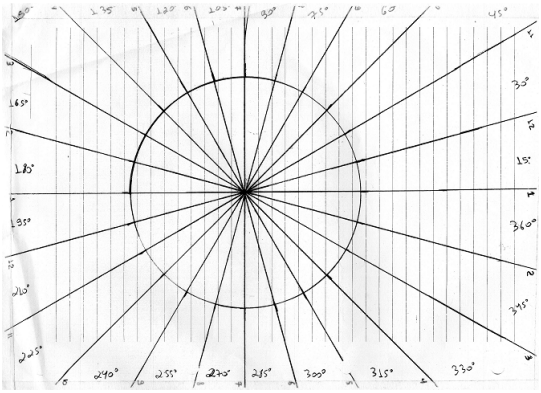

Figura 4.8: *Plataforma utilizada para controle do deslocamento dos dispositivos durante os experimentos com bússola digital.*

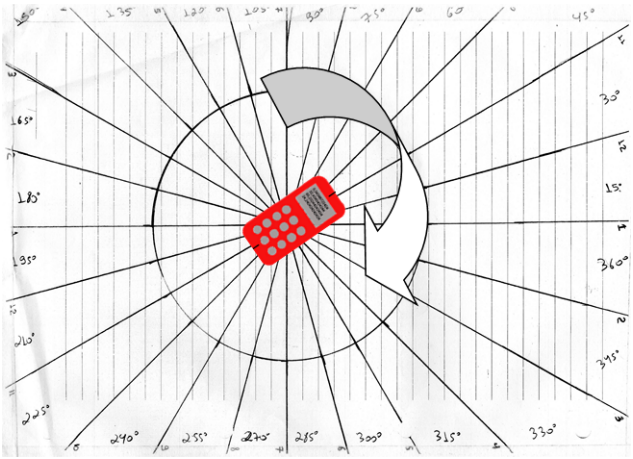

Figura 4.9: *Rotação do celular sobre a plataforma utilizada para controle do deslocamento dos dispositivos durante os experimentos com bússola digital.*

4.2.1 Análise da variação angular

A (Figura 4.10) apresenta gráficos relacionados ao experimento nº 1 que foi citado na seção anterior. Cada gráfico possui dois círculos:

- um círculo azul dividido em 24 setores cada, sendo cada setor a representação da diferença em graus de dois pontos de parada (linhas da plataforma) subsequentes. Cada linha diz respeito à direção apontada pelo celular ao fazer uma rotação de 15º em relação à linha imediatamente anterior;

- e um círculo vermelho tracejado dividido em 24 setores com 15º cada, que corresponde ao objetivo ideal de rotação do dispositivo apontado no item acima. Dessa forma, uma medição ideal seria aquela que posicionasse as linhas azuis exatamente sob as linhas vermelhas;

A Tabela 4.11 apresenta estatísticas referentes ao experimento número 1 para avaliar a precisão da bússola digital.

Conforme os dados da Tabela 4.11, é nítido que na maior parte dos casos existe uma margem de erro pequena na apuração das rotações em relação ao modelo ideal. O único caso alarmante dessa amostra foi o que ocorreu com o dispositivo *LG-P350f (2)*, que chegou a registrar uma abertura de quase 45º, quando se pretendia fazê-la em 15º, e por isso também

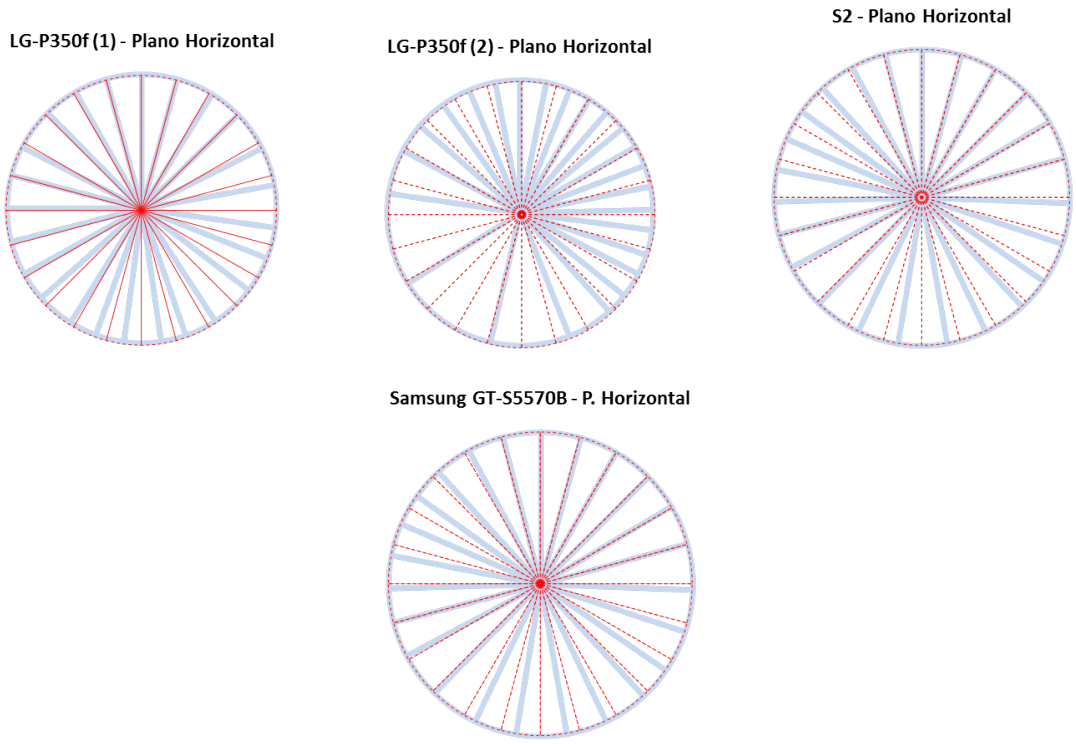

Figura 4.10: *Verificação da acurácia de rotação dos dispositivos em planos horizontais*

Modelo	ângulo ideal	média (graus)	desvio padrão (graus)
LG-P350f(1)	15º	14,32924605º	3,651347513º
LG-P350f(2)	15º	14,3781805º	10,4149795º
Samsung Galaxy SII	15º	14,43394106º	3,623642447º
Samsung GT-S5570B	15º	16,030757º	9,376393104º

Tabela 4.11: *Estatísticas referentes à ocorrência do experimento nº 1 com todos os dispositivos, no plano horizontal.*

teve o maior desvio padrão para os ângulos formados entre dois pontos de parada subsequentes. Assim como ocorreu com as análises com o GPS, o modelo *Samsung Galaxy SII* apresentou os melhores resultados, tendo a menor diferença em relação ao ângulo ideal.

Foram feitos também testes em planos inclinados. A Figura 4.11 apresenta os resultados relativos ao modelo *Samsung Galaxy SII*. A Tabela 4.12 apresenta as estatísticas referentes aos gráficos obtidos das medições em plano inclinado.

Conforme gráficos da Figura 4.11, podemos notar um maior afastamento do modelo ideal para os planos inclinados, que apresentam desvios de padrões maiores para os ângulos formados entre dois pontos de parada subsequentes. Isso se deve, principalmente, ao comportamento irregular do sensor magnetômetro quando não está mantido na horizontal, conforme abordado na Seção 3.2.2.

4.2.2 Análise da orientação do dispositivo com o magnetômetro descalibrado

Na seção anterior tomamos ciência como se deu a rotação (variação angular) de um ponto para o outro, mas não foi levada em consideração a direção apontada pelo dispositivo, que

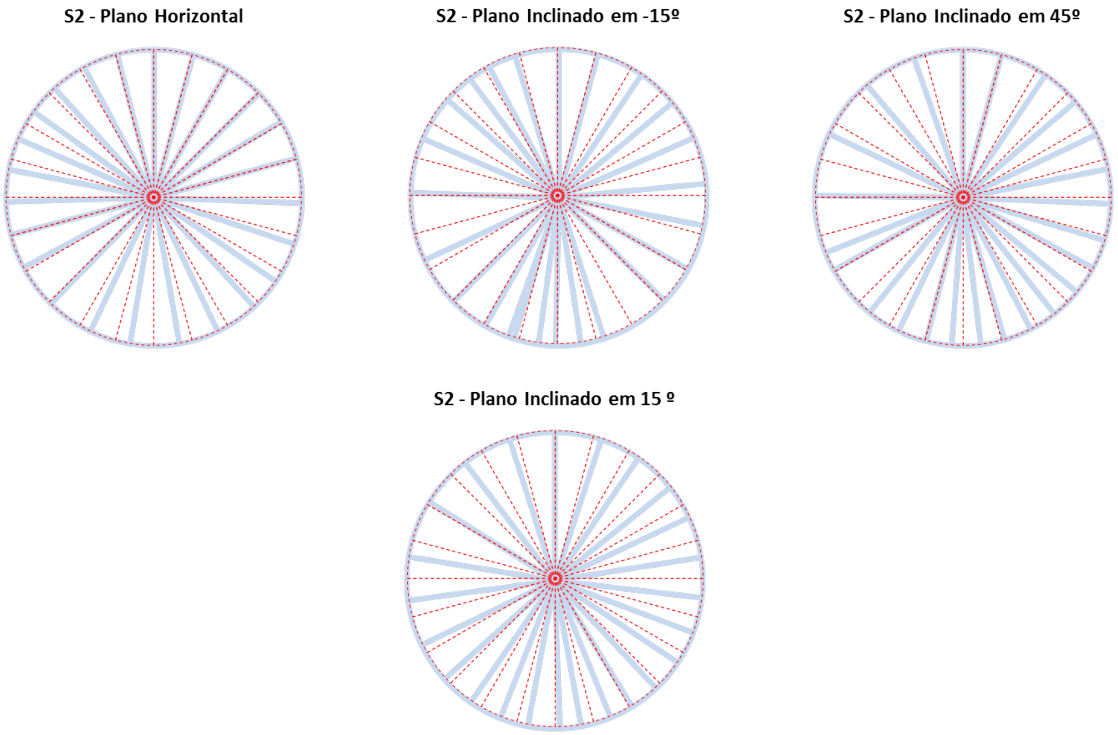

Figura 4.11: *Verificação da acurácia de rotação do* Samsung Galaxy SII *em planos inclinados*

Modelo	plano	média (graus)	desvio padrão (graus)
Samsung Galaxy SII	Horizontal	14,43394106º	3,623642447º
Samsung Galaxy SII	Inclinação de -15º	14,50806916º	6,654497978º
Samsung Galaxy SII	Inclinação de 15º	14,4178552º	4,578378876º
Samsung Galaxy SII	Inclinação de 45º	14,50889291º	5,72374275º

Tabela 4.12: *Estatísticas referentes à ocorrência do experimento nº 1 com o celular* Samsung Galaxy SII*, em planos inclinados.*

deve ser a mesma para cada ponto de parada. Essa direção equivale à variável x do trecho de código abaixo, que pode assumir valores no intervalo de -180º a 180º.

```
1 /*guarda o vetor tridimensional relativo à orientação do dispositivo*/
2 float[] orientation = new float[3];
3
4 public void onSensorChanged(SensorEvent evt) {
5       /* (...) */
6       /*define a orientação a partir da matriz de rotação*/
7       SensorManager.getOrientation(rotationMatrix, orientation);
8       /*coversão em graus*/
9       float x = (orientation[0]*180)/Math.PI
10 }
```

Ao posicionarmos diferentes celulares sobre o eixo horizontal da mesma plataforma, em uma mesma direção e no mesmo sentido, conforme Figura 4.12, os valores armazenados em x deveriam ser os mesmos, ou pelo menos muito próximos. Mas não é isso que acontece, conforme pode ser averiguado na Tabela 4.13 e no gráfico da Figura 4.13, que exibem os valores dessa variável para os quatro dispositivos posicionados nas mesmas condições: sobre o eixo ortogonal da mesma plataforma, na direção horizontal e no sentido para direita.

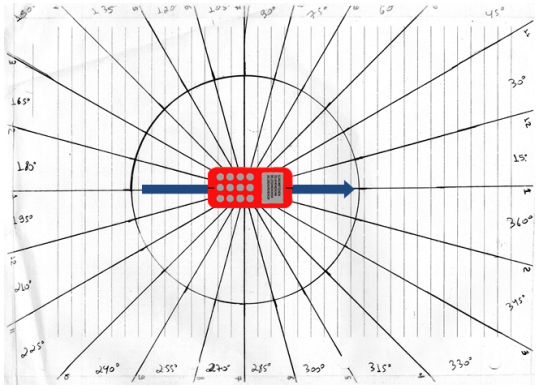

Figura 4.12: *Posicionamento do celular sobre o eixo horizontal da plataforma utilizada nos experimentos com bússola digital. Orientação do celular: Direção horizontal, Sentido para direita*

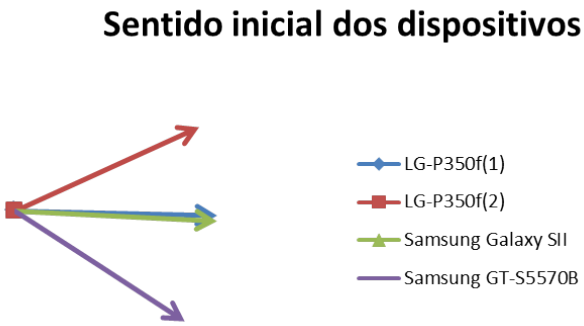

Figura 4.13: *Sentido inicial dos dispositivos posicionados tais como o esquema da Figura 4.12, estando os mesmos com seus sensores magnetômetros descalibrados.*

A partir das figuras 4.12 e 4.13 e da Tabela 4.13, nota-se que dois dispositivos apresentaram valores muito próximos, o *LG-P350f(1)* e o *Samsung Galaxy SII*. Isso não atesta, necessariamente, que esses dois estão corretos e os demais errados, pois a plataforma não foi alinhada com o polo norte. Ou seja, foi demonstrado apenas que a bússola digital de cada dispositivo identificou o sentido do polo norte com valores diferentes. Isso se deve, principalmente, ao fato de não ter sido feita uma calibragem do sensor magnetômetro, demonstrando a importância dessa ação para aplicações que façam uso da biblioteca. Conforme citado Seção 3.2, existe uma forma específica de calibração para cada tipo de sensor.

4.3 Comportamento da rede via *Wireshark*

Wireshark é um aplicativo que detalha o comportamento dos protocolos de rede [7], sendo possível analisar o tráfego dos pacotes que trafegam na rede em que o computador está conectado. Ou seja, é possível saber de onde um determinado pacote está vindo, para onde ele está indo e qual a informação contida nele. Com essa poderosa ferramenta foi feito um teste para analisar todo trajeto na rede de um pacote enviado pela biblioteca.

Modelo	sentido inicial (graus)
LG-P350f(1)	-1,258639939º
LG-P350f(2)	24,39133529º
Samsung Galaxy SII	-3,120302103º
Samsung GT-S5570B	-33,13623838º

Tabela 4.13: *Sentido inicial dos dispositivos posicionados tais como o esquema da Figura 4.12, estando os mesmos com seus sensores magnetômetros descalibrados.*

4.3.1 Detalhes do experimento

Para realização do experimento, foi criada uma aplicação que envia uma mensagem em broadcast e uma em unicast para um determinado endereço IP usando a biblioteca. Foram usados quatro celulares, já identificados no início deste capítulo, sendo o ponto de acesso correspondente ao modelo *Samsung Galaxy SII*. O *Wireshark* estava rodando em um computador que estava conectado à mesma rede que os celulares.

A relação celular x IP está representada na Tabela 4.14.

Celular	IP
LG-P350f(1)	192.168.43.190
LG-P350f(2)	192.168.43.177
Samsung GT-S5570B	192.168.43.158
Samsung Galaxy SII	192.168.43.1

Tabela 4.14: *Relação Celular x IP*

4.3.2 Resultados

O resultado obtido pode ser visto nas Figuras 4.14 e 4.15, que foram extraídas do *trace* do *Wireshark*.

A Figura 4.14 representa o experimento em que a mensagem foi enviada em modo broadcast. Como é possível identificar na figura, o IP de origem é o "192.168.43.158"(*Samsung GT-S5570B*) e o IP de destino é o "192.168.43.255", que corresponde ao endereço de broadcast dessa rede.

A figura 4.15 representa o experimento em que a mensagem foi enviada em modo unicast. O dispositivo LG-P350f(1) enviou a mensagem para o LG-P350f(2), ou seja, o IP fonte é "192.168.43.190"e o IP de destino é "192.168.43.177". Como é possível identificar na figura, a mensagem não passa pelo ponto de acesso para ir ao seu destino. Certificou-se através desse experimento que, para a comunicação entre dois dispositivos na rede, não é necessário que a mensagem passe pelo ponto de acesso, ou seja, é possível concluir que existe um enlace de comunicação entres os dispositivos na rede, não apenas entre eles e o ponto de acesso.

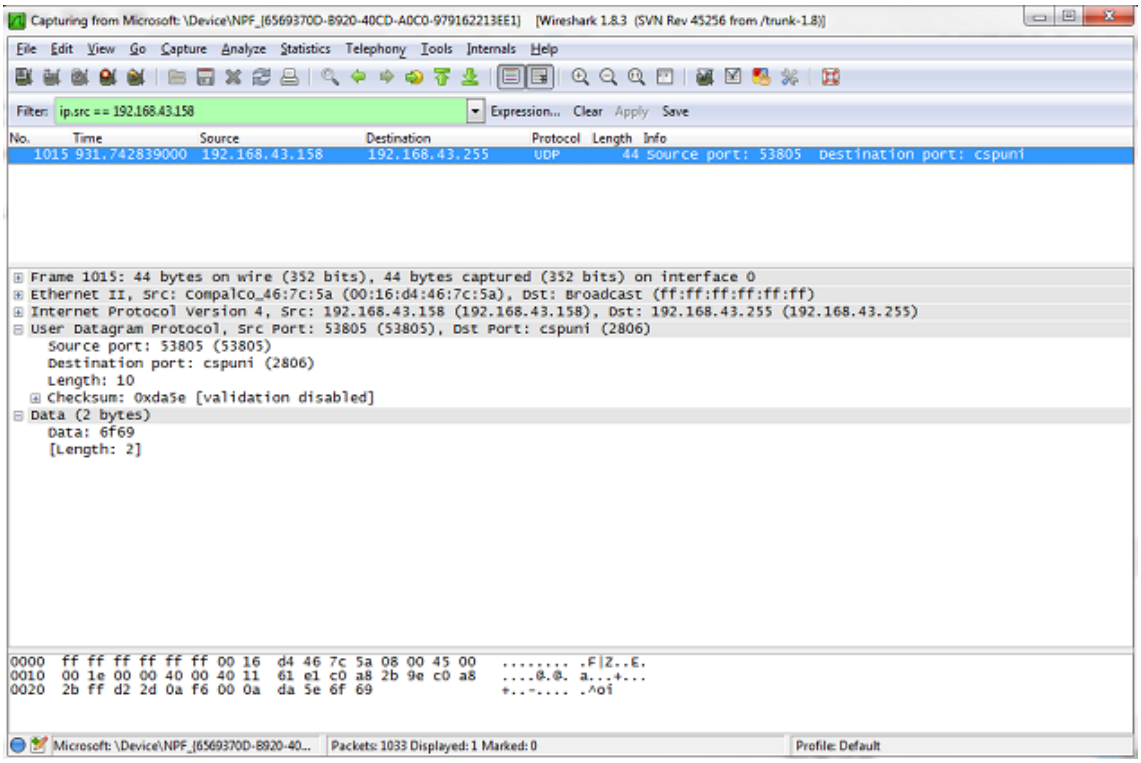

Figura 4.14: *Envio de mensagem em modo broadcast usando o protocolo UDP*

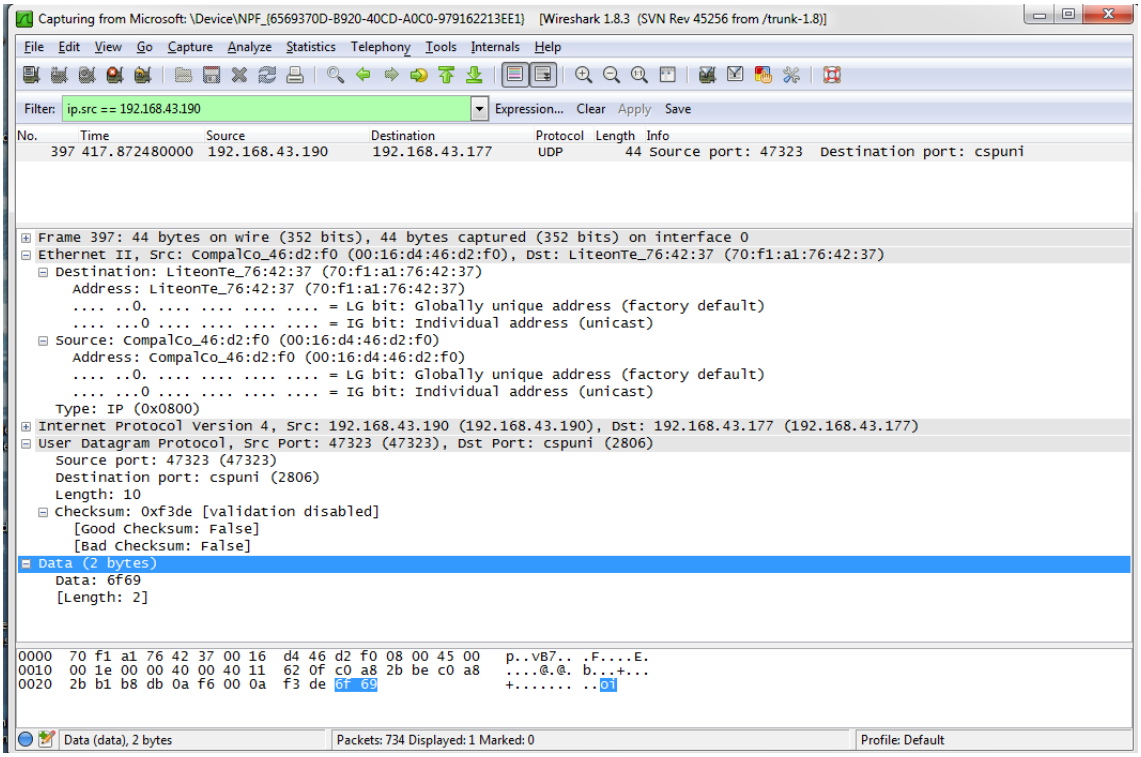

Figura 4.15: *Envio de mensagem em modo unicast usando o protocolo UDP*

Capítulo 5

Biblioteca

Nos capítulos anteriores foram apresentados conceitos fundamentais para a criação da biblioteca, que é o objetivo desse trabalho.

É importante salientar que biblioteca *GeoCommunication* foi escrita especificamente para Android e, portanto, usando a linguagem Java (ver Seção 3.1). A ideia dos autores foi criar uma biblioteca simples de usar, ou seja, mesmo um programador com pouca experiência em redes, envio de mensagem e sensores, deve conseguir usar a biblioteca sem grandes dificuldades. A documentação (Javadoc) pode ser encontrada na página dos autores, http://www.linux.ime.usp.br/~avila/mac499/doc/index.html.

Este capítulo apresenta os detalhes da implementação da biblioteca. A Seção 5.1 explica como fazer uso da biblioteca, através do diagrama de caso e uso e algumas restrições. A Seção 5.2 aborda os detalhes da implementação, apontando os pontos mais importantes do código.

5.1 Uso da biblioteca

O uso da biblioteca é apresentado através de suas restrições e do diagrama de caso de uso.

5.1.1 Requisitos de uso

Para desenvolver aplicações usando essa biblioteca *GeoCommunication* é necessário que o programador saiba desenvolver em Java e para a plataforma Android (Seção 3.1.2), porém é requerido, apenas, um conhecimento mínimo sobre gerenciamento de redes e sensores do Android.

A biblioteca foi programada para funcionar na API 2.2 ou superior, o que abrange cerca de 90% dos dispositivos android em 2012 [6]. Seu funcionamento não é garantido em versões de API inferiores. A única exceção atual para o funcionamento na referida API é para casos em que o programador opte pelo uso do filtro complementar como alternativa de geração do vetor de orientação, conforme abordado na Seção 3.4.1, o qual requer a API 2.3.

Para usá-la basta adicionar o arquivo `tcc_library.jar`, que pode ser encontrado no site dos autores http://www.linux.ime.usp.br/~avila/mac499/tcc_library.jar, no *classpath* da sua aplicação. É necessário, no entanto, adicionar algumas permissões de usuário no arquivo AndroidManifest.xml:

- CHANGE_WIFI_STATE
- CHANGE_NETWORK_STATE

- CHANGE_CONFIGURATION
- ACCESS_WIFI_STATE
- ACCESS_NETWORK_STATE
- INTERNET
- ACCESS_FINE_LOCATION

Apesar da necessidade da permissão para uso da internet, a mesma não é usada. Esta permissão é necessária para que um dispositivo envie uma mensagem na rede, independente do seu tipo.

5.1.2 Caso de uso

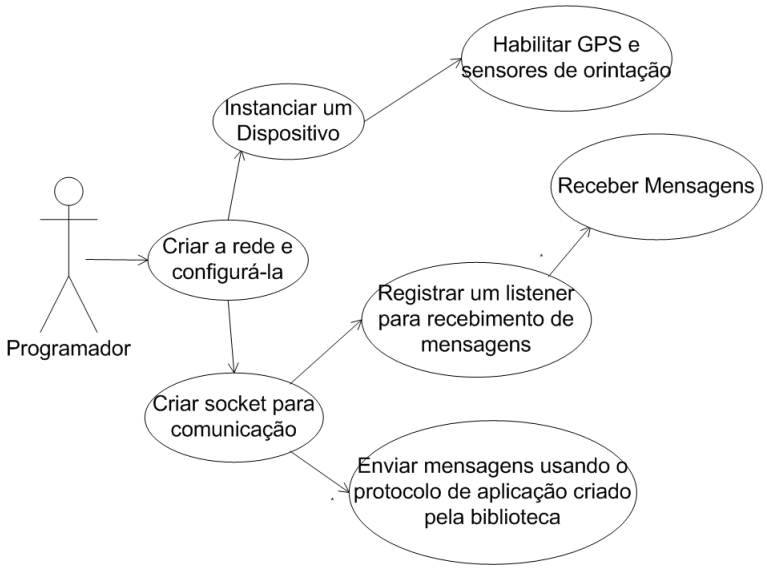

Figura 5.1: *Diagrama de caso de uso da biblioteca*

O diagrama de caso de uso, Figura 5.1, mostra uma sequência de ações que um programador deve seguir para usar a biblioteca. Primeiro é necessário criar a rede local entre os dispositivos. Feito isso, o programador deve realizar duas ações independentes, a primeira é instanciar um dispositivo, que é o objeto que a biblioteca usa para identificar cada nó na rede, e com isso habilitar a coleta das informações provenientes do GPS e sensores. A segunda ação é criar um *socket* para viabilizar a comunicação. A partir disso já é possível enviar mensagens na rede, sendo necessário registrar um "ouvidor"(*listener*) para recebê-las.

5.2 Implementação

Essa Seção objetiva apresentar os pacotes usados na biblioteca, focando na sua implementação.

5.2.1 usp.ime.gclib

Esse é o pacote principal da biblioteca, contendo, diretamente em sua raiz, todos os outros e uma única classe, `Device`, que é o principal vínculo entre eles.

O objeto `Device` representa um dispositivo móvel com os atributos essenciais para sua manipulação na biblioteca *GeoCommunication*:

- o nick, que corresponde ao identificador único do dispositivo;
- o endereço IP;
- sua identificação, se o dispostivo corresponde a um ponto de acesso;
- as informações sobre a localização geográfica;
- e as informações sobre a orientação do dispositivo no globo.

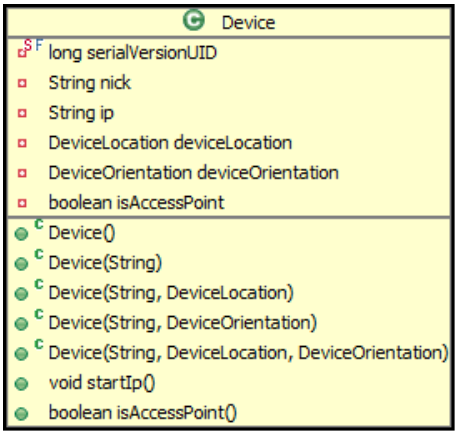

Figura 5.2: *Diagrama de classe do pacote usp.ime.gclib*

5.2.2 usp.ime.gclib.sensor.orientation

Esse pacote gerencia a formação do vetor de orientação do dispositivo, que é acoplado na classe `DeviceOrientation`. Um objeto desse tipo poderá ser associado ao ouvidor `OrientationSensorListener`, cuja responsabilidade é a de controlar a recepção dos sinais dos sensores acelerômetro, magnetômetro e giroscópio, direcionando o tratamento desses, através do método `sensorManager`, para um objeto que estenda a classe `DeviceOrientation`. Na versão 1.0 da biblioteca, existem três tipos de `DeviceOrientation`:

- `DeviceCompassOrientation`, que usa os sensores magnetômetro e o acelerômetro;
- `DeviceGyroscopeOrientation`, que é inicializado pelo objeto do item anterior e atualiza o vetor de orientação através do sensor giroscópio;
- `DeviceComplementaryFilterOrientation`, que utiliza os dois objetos acima, aplicando o filtro complementar sobre o resultado deles.

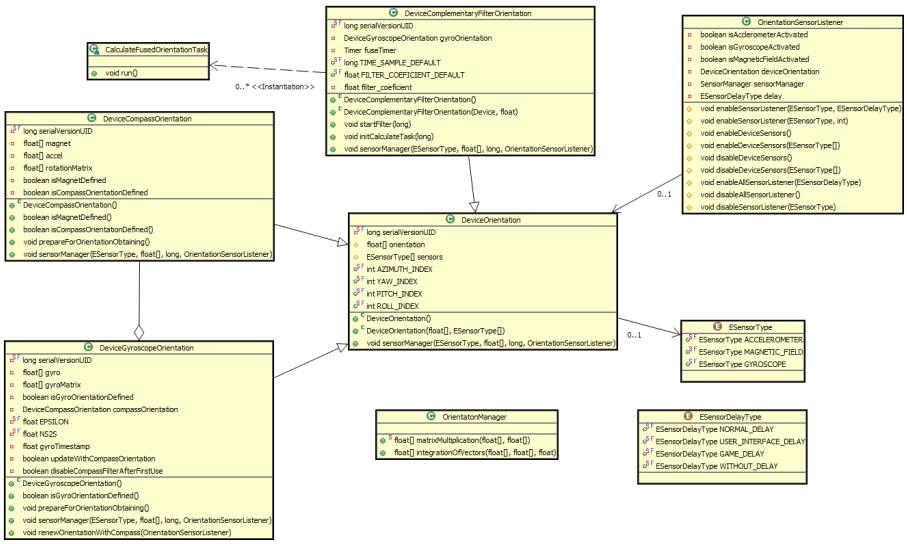

Figura 5.3: *Diagrama de classes do pacote usp.ime.gclib.orintation*

5.2.3 usp.ime.gclib.sensor.location

Esse pacote gerencia os dados de localização do dispositivo no globo, que são latitude e longitude e a acurácia da obtenção dessas coordenadas geodésicas. A classe `DeviceLocation` encapsula esses atributos e permite que os mesmos sejam definidos manualmente, tal como ocorre com o pacote `usp.ime.gclib.sensor.orientation`, ou atualizados através do ouvidor `LocationGpsListener`, usando um provedor de GPS.

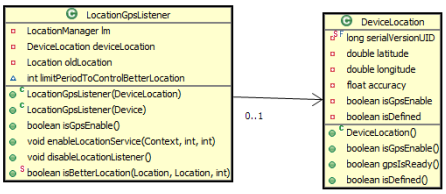

Figura 5.4: *Diagrama de classes do pacote usp.ime.gclib.location*

5.2.4 usp.ime.gclib.hit

Esse pacote é responsável pela gestão da comunicação entre os dispositivos baseada em suas coordenadas geográficas e orientação. Com o intuito de facilitar a compreensão, a nomenclatura utilizada na biblioteca e neste material segue a seguinte analogia: o envio de uma mensagem a um destinatário na rede corresponde a um tiro dado para acertar um determinado alvo. As características do tiro (Tabela 5.1) e do alvo (Tabela 5.2) são encapsuladas, respectivamente, pelas classes `ShootingRestrictions` e `TargetRestrictions`.

A apuração do acerto do alvo apoia-se em cálculos geodésicos, através dos quais é feita a verificação se o vetor de orientação, ou a área formada entre dois vetores, interseccionam o círculo formado pelo ponto de destino e seu raio de alcance. Seguem as etapas do cálculo para um tiro com apenas um vetor de orientação:

1. Verifica-se a distância entre o atirador e o alvo.

2. Constrói-se um círculo com o centro igual à posição do atirador e o raio definido no passo 1, contendo a posição do alvo na superfície;

Características do tiro	
Característica	**Descrição**
Ângulo de abertura	O tiro é formado por dois vetores com o ângulo de abertura atribuído.
Largura do tiro	O tiro é formado por dois vetores paralelos espaçados com a largura atribuída.
Alcance	Máxima distância do atirador ao alvo

Tabela 5.1: *Características referentes ao envio de mensagem baseado em coordenadas geográficas e realizado na biblioteca* GeoCommunication*, fazendo analogia da mensagem com um tiro.*

Características do alvo	
Característica	**Descrição**
Raio de acerto	Raio ao redor da posição geodésica (latitude x longitude) do dispositivo que define a área de acerto do alvo.
Permissão de Incremento do raio com a acurácia	Conforme [6], o valor da acurácia corresponde a um raio de confiança medido em metros. Portanto, o seu valor pode ser incrementado no raio do alvo.

Tabela 5.2: *Características referentes à identificação do destinatário baseada em coordenadas geográficas e feita pela biblioteca* GeoCommunication*, fazendo analogia do destinatário com um alvo.*

3. Calcula-se qual região do círculo o tiro acertou, gerando um "dispositivo virtual";
4. Verifica-se se o ponto definido no passo 3 está dentro da área de acerto do alvo, que corresponde a um círculo cujo centro e raio são equivalentes à posição do alvo e seu raio de acerto.

As Figuras 5.5 e 5.6 ilustram o cálculo acima, fazendo referência, respectivamente, a um acerto e um erro do alvo.

Figura 5.5: *Vista a aérea da condição de recepção pelo alvo de uma mensagem enviada.*

Figura 5.6: *Vista aéria da condição de NÃO recepção pelo alvo de uma mensagem enviada.*

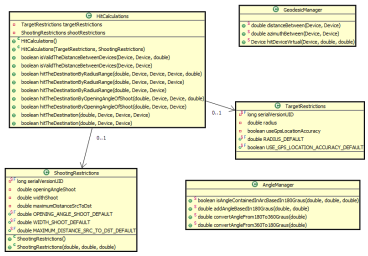

Figura 5.7: *Diagrama de classes do pacote usp.ime.gclib.hit*

5.2.5 usp.ime.gclib.net.protocol

Na Seção 2.2.1 foi apresentado o protocolo de aplicação HTTP e verificado que as mensagens trocadas são padronizadas, pois somente assim que o servidor e o cliente se entendem. Logo, foi criado um protocolo para a biblioteca *GeoCommunication* com algumas regras de sintaxe bem definidas.

As mensagens implementadas foram baseadas nas necessidades das aplicações, conforme Tabela 5.3. Seguem os tipos das mesmas: `ONLINE`, `ONLINEANSWER`, `GEOMSG`, `GEOACK`, `LIBCONFIG` e `APPDATA`, conforme definido na enumeração (*enum*) `EProtocolMessages`.

`Mensagem`	**Objetivo**
ONLINE	Informar a presença de um novo dispositivo na rede
ONLINEANSWER	Responder à mensagem do tipo ONLINE
APPDATA	Usado para passar dados da aplicação
LIBCONFIG	Usado para enviar as configurações da biblioteca a um determinado dispositivo
GEOMSG	Envio de mensagem baseado em coordenada geográfica e orientação
GEOACK	Usado para confirmar o recebimento de uma mensagem do tipo GEOMSG

Tabela 5.3: *Protocolo de aplicação*

As mensagens podem ser enviadas tanto usando UDP (Seção 2.3.2) quanto TCP (Seção 2.3.3) como protocolos da camada de transporte. Esses dois tipos estão definidos na

enumeração (*enum*) `EProtocolTranspLayer`.

Além de definir o protocolo de envio, é possível definir, também, para quem a mensagem será enviada. Os possíveis destinos estão definidos na enumeração (*enum*) `ESendTo`: o ponto de acesso (`GATEWAY`), todos conectados à rede (`ALL`) e uma lista de endereços IPs (`LISTIPS`).

A classe `AppProtocol` organiza as informações citadas nos parágrafos anteriores em um só objeto. Seus atributos são, portanto, um `EProtocolTranspLayer`, um `ESendTo`, um `EProtocolMessages` e uma lista de endereços IPs, que é usado apenas quando `ESendTo` for do tipo `LISTIPS`.

Para o envio da mensagem na rede é usado um objeto da classe `ProtocolInformation`, que possui como atributos:

- dispositivo fonte (`Device`);
- tipo da mensagem (`EProtocolMessages`);
- qual protocolo da camada de transporte foi usado, TCP ou UDP (`EProtocolTranspLayer`);
- a mensagem da aplicação (`byte[]` ou `Object`).

Para colocar informações adicionais, é necessário criar outra classe que estenda essa. É de responsabilidade do programador informar a instância correta para cada tipo de mensagem. Caso isso não seja feito, será lançado uma exceção de argumento inválido, `IllegalArgumentException`.

A mensagem do tipo `GEOMSG` é a que fundamenta a biblioteca *GeoCommunication*, pois é através do seu uso que ocorre a comunicação baseada em referências geográficas e orientação do dispositivo. A classe `ProtocolGEOMSGInformation`, que estende a classe `ProtocolInformation`, permite que, além das informações básicas, seja possível enviar, a outro nó na rede, as informações contidas na classe `ShootingRestrictions` (Seção 5.2.4).

É importante salientar que, apesar de ser possível o envio para uma lista de endereços IPs, a verificação do `GEOMSG` para validação do destinatário não é feita baseada no endereço de rede, mas sim em referencias geográficas e orientação do dispositivo.

Se for levada em consideração a implementação dos tipos de mensagens na biblioteca, não haveria necessidade de criar mais que um tipo, além do `GEOMSG`, visto que o uso delas é idêntico. Porém, o objetivo das mesmas é organizar a aplicação através de eventos, permitindo que o programador trate cada um deles separadamente.

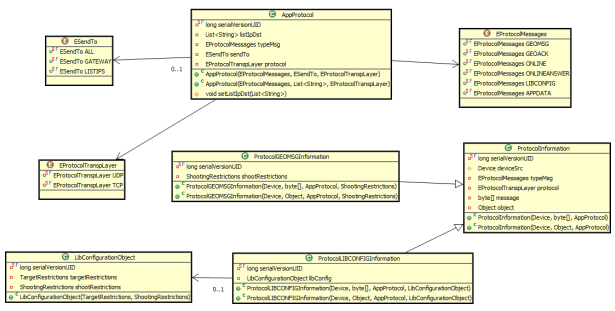

Figura 5.8: *Diagrama de classes do pacote usp.ime.gclib.protocol*

5.2.6 usp.ime.gclib.net.communication

Esse pacote tem como função principal a comunicação entres os nós (devices) pela rede. Ele contém classes responsáveis pelo envio e recebimento de mensagens usando o protocolo TCP (`TCPSender` e `TCPReceiver`) e o UDP (`UDPSender` e `UDPReceiver`). Essas classes, no entanto, não são manipuláveis pelo usuário (programador), sendo apenas para uso interno da biblioteca.

A interface `IReceiveListener` contém os métodos que serão chamados quando chegar uma nova mensagem, sendo obrigatória sua implementação pela aplicação para que a comunicação ocorra. Os métodos são:

- `onReceiveONLINE(ProtocolInformation))`
- `onReceiveONLINEANSWER(ProtocolInformation)`
- `onReceiveAPPDATA(ProtocolInformation)`
- `onReceiveLIBCONFIG(ProtocolLIBCONFIGInformation)`
- `onReceiveGEOMSG(ProtocolGEOMSGInformation)`
- `onReceiveGEOACK(ProtocolInformation)`

As classes `TCPReceiver` e `UDPReceiver` estendem a classe `Receiver`, que é responsável por verificar qual protocolo foi usado no envio da mensagem e, junto com os atributos de dispositivo fonte (*source device*), dispositivo de destino (*target device*) e restrições do alvo (*target restrictions*), identificar se ela é válida, ou seja, se ela pertence àquele dispositivo receptor, e, então, acionar o evento correspondente da interface `IReceiveListener`.

A interface com o usuário fica por conta da classe `CommunicationSocket`, que contém dois métodos, um para enviar a mensagem e um para recebê-la. Como é criada uma nova thread para receber as mensagens, o método de recepção é assíncrono. O método de envio recebe dois objetos como parâmetros, o `ProtocolInformation` e o `AppProtocol` (Seção 5.2.5). Este é usado para extração das informações para o envio, enquanto o primeiro é o objeto a ser enviado pela rede.

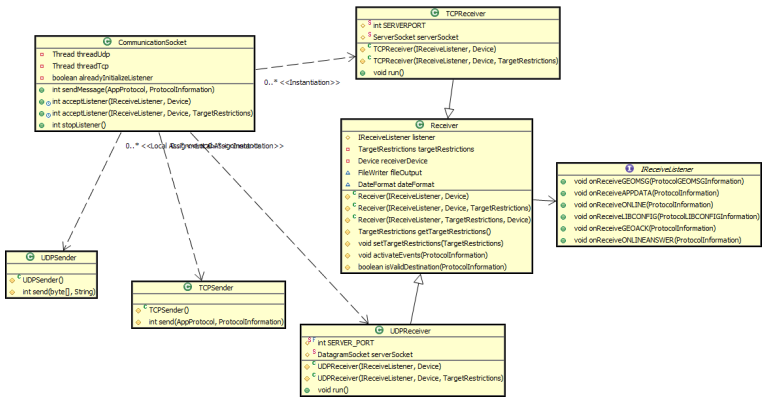

Figura 5.9: *Diagrama de classes do pacote usp.ime.gclib.communicatiom*

5.2.7 usp.ime.gclib.net.wifi

Esse pacote é responsável por todo processo da biblioteca que envolve manipulação de rede. Através da sua principal classe, `NetworkManager`, é possível se conectar a uma rede já existente, sabendo apenas o seu nome (ssid) e, caso possua, a senha. Para tal, essa classe requer permissão de acesso e mudança no estado do Wi-fi dos dispositivos Android que a usarem. A referida classe provê diversas outras informações e serviços, tais como: criar uma nova rede fazendo que o criador seja o ponto de acesso, buscar entre as redes disponíveis quais foram criadas pela biblioteca, remover a rede conectada do histórico do dispositivo.

A classe `NetworkConfiguration` serve como auxilio e contém apenas as configurações básicas de uma rede: o ssid, a senha e indica se ela é aberta (não possui senha). O prefixo **AND_CGLIB_** é adicionado no ssid para que sejam reconhecidas com facilidade as redes criadas pela biblioteca.

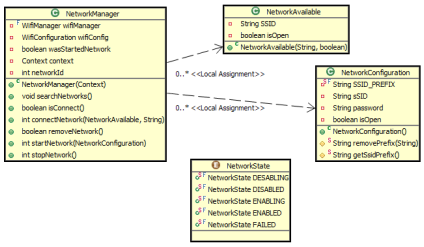

Figura 5.10: *Diagrama de classes do pacote usp.ime.gclib.wifi*

5.2.8 Biblioteca Geotools

A Biblioteca *GeoCommunication* utiliza as ferramentas de mapeamento OpenSource da biblioteca Geotools2, respeitando os termos de sua licença LGPL. Esta biblioteca disponibiliza código em JAVA para manipulação de dados geoespaciais, servindo para esse projeto devido ao seu suporte para manipulação e obtenção de direção, distância e coordenadas geodésicas.

Para tal finalidade, usa-se basicamente a classe `GeodeticCalculator` e seus vínculos no código fonte. Um objeto do tipo `GeodeticCalculator`, conforme referência contida em sua documentação interna [2], permite que os cálculos dos problemas geodésicos direto e inverso sejam realizados com base na teoria de T.Vincenty, conforme Seção 3.3.3, utilizando o método de Rainsford modificado com termos elípticos de Helmert.

Basicamente, as funções dessa classe permitem que sejam efetuados os cálculos geodésicos sobre um elipsoide. Por padrão é utilizado o modelo que representa o planeta Terra, o WGS84, conforme Seção 3.3.3. Em suma, é possível determinar a distância e azimute entre dois pontos, ou o ponto localizado a uma dada distância e azimute de outro ponto, conforme documentação da biblioteca GeoTools [2].

Capítulo 6

Resultados e caso de uso

Para avaliar a eficácia da biblioteca é imprescindível mostrá-la em funcionamento. Para isso foi escolhida uma aplicação que representa o famoso jogo "batata-quente".

A brincadeira tradicional consiste de um número de rodadas limitado pela quantidade de participantes. Em cada rodada, os participantes ficam trocando entre si uma batata até que a mesma "queime", momento esse normalmente definido pelo término de uma música. Quando isso ocorre, o jogador que segurava a batata é eliminado do jogo. O vencedor será aquele que restará após a eliminação de todos os outros participantes.

Na Seção 6.1 é apresentado o jogo Batata Quente através de sua interface de comunicação com o usuário. A Seção 6.2 apresenta os requisitos necessários para um usuário instalar esse jogo no seu dispositivo Android e poder usá-lo. A Seção 6.3 aponta como foi feita a implementação do caso de uso com recursos da biblioteca. A Seção 6.4 mostra os resultados obtidos do uso da aplicação.

6.1 Jogo batata quente

Após passagem pela tela de apresentação (Figura 6.1(a)), a busca pelo sinal de GPS é iniciada, com a finalidade de diminuir o tempo de espera do usuário pelo tempo de inicialização do GPS (*TTTF - Time To First Fix*), conforme discutido na Subseção 3.3.2.

Antes de prosseguir, o usuário é obrigado a definir um apelido (Figura 6.1(b)), que será utilizado para notificar aos demais usuários quem está com a posse da batata.

A tela de menu (Figura 6.1(c)) disponibiliza, além das opções tradicionais de um jogo: instruções, configurações, entrada e saída, a opção de criação do mesmo, que existe com a finalidade de restringir a participação de pessoas no jogo.

A tela de configurações (Figura 6.1(d)) oferece recursos relacionados à recepção de mensagens pelos dispositivos. Através dessa, o usuário pode optar por definir sua localização manualmente, atribuindo valores para latitude e longitude, o que é uma opção útil para situações em que os jogadores possam ficar fixos em determinadas posições e saibam previamente as mesmas e/ou para quando o sinal de GPS não exista ou esteja prejudicado. Além disso, caso não se queira usar o valor padrão da biblioteca, é possível definir também o raio de acerto daquele dispositivo.

Apenas um participante do jogo poderá ser o criador do mesmo. Ao criá-lo, atribuindo-lhe um nome e uma senha com no mínimo 8 caracteres, conforme Figura 6.1(e), o dispositivo desse jogador, automaticamente, tornar-se-á o ponto de acesso da rede, disponibilizando aos demais uma rede cujo nome será formado pelo prefixo AND_GCLIB_ (criado pela biblioteca *GeoCommunication*) concatenado ao nome do jogo.

Figura 6.1: *Telas da aplicação batata quente.*

Após esse procedimento, o criador será redirecionado para a tela de inicialização do jogo (Figura 6.1(f)), na qual deverá esperar pelo aceite de todos os demais participantes, pois, ao iniciá-lo, será negada a permissão de entrada para novos usuários.

Os demais participantes verificarão a criação do jogo e o aceitarão através da interface de listagem dos jogos disponíveis (Figura 6.1(g)), cujo acesso só será permitido caso o Wi-fi esteja ligado. Após seleção do jogo e informação de sua senha de acesso, o novo jogador aguardará pelo início do mesmo na tela de passagem da batata (Figura 6.1(h)).

Após iniciar o jogo, o criador será o dono da batata (Figura 6.1(i)), sendo responsável pelo primeiro envio, enquanto os demais dispositivos aguardam por sua recepção (Figura 6.1(j)). Para haver a troca da batata entre os dispositivos, basta que o dono da mesma aponte seu dispositivo móvel para o jogador que se deseja enviar e pressione a batata.

A duração total do jogo é definida aleatoriamente, entre um período de tempo mínimo (20s) e máximo (60s) estabelecidos por padrão. Em seguida, essa duração é divida em três intervalos aleatórios colocados em ordem decrescente de duração. Cada um deles equivalerá a uma etapa do jogo: batata fria (Figura 6.1(i)), batata morna (Figura 6.1(k)) e batata quente (Figura 6.1(l)), sendo que, quanto mais "quente"ela estiver, menos tempo ela ficará

na tela. Quando o tempo total acabar, a batata queimará (Figura 6.1(m)) e desclassificará o jogador que a "segurava"(Figura 6.1(n)).

A Figura 6.2 apresenta uma simulação de partida com duração de 1 minuto, em que a batata manteve-se por 40s na temperatura fria, por 14s na temperatura morna e por 6s na temperatura quente.

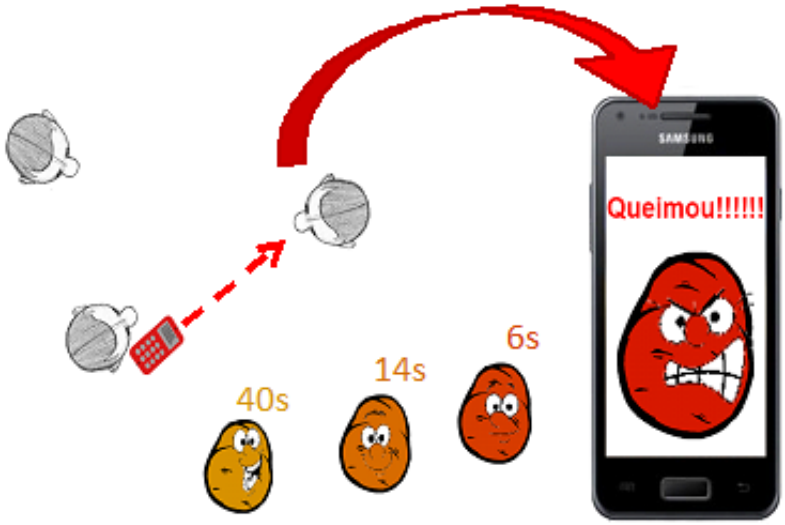

Figura 6.2: *Simulação de partida com duração de 1 minuto.*

6.2 Requisitos

Alguns requisitos precisam ser levados em consideração para a instalação, execução e uso dessa aplicação.

6.2.1 Instalação e execução

Esta aplicação não se encontra no *Android Market*, repositório de aplicações oficial do Android, então, para instalá-la, é necessário fazer o download do arquivo `batata_quente.apk`, que se encontra no site dos desenvolvedores, http://www.linux.ime.usp.br/~avila/mac499, e executá-lo em seu dispositivo Android, que deve estar previamente configurado para permitir instalações de aplicações não vinculadas ao *Android Market*.

Essa aplicação requer uma versão de API 2.2 ou superior. Como já foi dito na Seção 5.1.1, algumas permissões de usuário são necessárias. Além daquelas requeridas pela biblioteca, é preciso conceder também a permissão:

- VIBRATOR

Essa permissão é necessária para permitir que o dispositivo vibre quando a batata é enviada.

6.2.2 Usando a aplicação

Para uso da aplicação é necessário que o usuário:

- habilite o uso do GPS;

- ative o Wi-fi;
- calibre os sensores magnetômetro e acelerômetro;
- esteja localizado a pelo menos 10m de distância de outros jogadores.

6.3 Uso da biblioteca *GeoCommunication* na aplicação

Esta Seção se propõe a exemplificar a implementação dos recursos da biblioteca através do jogo "Batata Quente". Para isso, serão apresentadas as chamadas a funções da biblioteca usadas nas Atividades(Activities)[1] da referida aplicação.

StartGameActivity

No trecho de código que segue, é criado um objeto `NetworkConfiguration` com os dados passados pelo usuário (ssid e senha), passando-o como argumento para o método que inicia uma nova rede.

Código:

```
1 String ssid = ssidValue;
2 String password = passwordValue;
3
4 NetworkManager netMag = new NetworkManager(StartGameActivity.this);
5 NetworkConfiguration netConfig = new NetworkConfiguration();
6 netConfig.setSSID(ssid);
7 netConfig.setPassword(password);
8 netConfig.setOpen(false);
9
10 netMag.startNetwork(netConfig); //torna o dispositivo um ponto de acesso
```

ListGameAvailableActivity

No código abaixo são obtidas todas as instâncias de redes criadas pela biblioteca, gerando uma lista com os ssids das mesmas.

Código:

```
1 List< ItemListView > itens = new ArrayList< ItemListView >();
2
3 List<NetworkAvailable> nets = netMng.getAllNetworksAvailable();
4
5 for (NetworkAvailable net : nets) {
6         ItemListView item = new ItemListView(net.getSSID());
7         itens.add(item);
8 }
```

O código abaixo implementa a tentativa de conectar o dispositivo à rede especificada pelo seu ssid e senha.

Código:

```
1 String ssid = ssidValue;
2 String password = passwordValue;
3 NetworkAvailable netAv = new NetworkAvailable(ssid, false);
4
```

[1] Atividades ou, originalmente, *Activities* são classes da API do Android que representam ações que o usuário pode fazer [6]

```
5 int ret = netMng.connectNetwork(netAv, password);
6 if (ret == 0)
7   //connected
```

ConfigurationActivity

Abaixo é apresentada uma opção para caso o usuário queira usar uma localização fixa, definindo-a manualmente, ao invés de registrar um ouvidor de GPS.

Código:

```
1 Device device = new Device();
2 Double latitude = latitudeValue;
3 Double longitude = longitude Value;
4
5 device.getDeviceLocation().setLatitude(latitude);
6 device.getDeviceLocation().setLongitude(longitude);
```

A implementação que segue é executada quando o usuário deseja mudar o tamanho do seu raio de acerto:

Código:

```
1 TargetRestrictions targetRestrictions = new TargetRestrictions();
2 Double raio = getFromUser();
3
4 targetRestrictions.setRadius(raio);
```

TelaDeApresentacaoActivity

Criação do `Device` que será usado em todas as *Activities* da aplicação e inicialização dos ouvidores (*listeners*) dos sensores e do GPS. Para o cálculo do vetor de orientação do dispositivo, foi escolhido o objeto `DeviceCompassOrientation` (Seção 5.2.2), que utiliza os sensores magnetômetro e acelerômetro.

Código:

```
1 DeviceCompassOrientation compassOrientation = new DeviceCompassOrientation
      ();
2 Device device = new Device("nick", compassOrientation);
3 LocationGpsListener listenerLocation = new LocationGpsListener(device);
4 OrientationSensorListener listenerOrientation = new
      OrientationSensorListener(device);
5
6 listenerLocation.enableLocationService(getApplicationContext(), 0, 0);
7 listenerOrientation.enableSensorService(getApplicationContext());
```

BatataQuenteActivity

Essa classe implementa a interface IReceiveListener, implementando os seguintes métodos:

```
1 onReceiveGEOMSG(ProtocolGEOMSGInformation appInfo);
2 onReceiveAPPDATA(ProtocolInformation appInfo);
3 onReceiveONLINE(ProtocolInformation appInfo);
4 onReceiveLIBCONFIG(ProtocolLIBCONFIGInformation appInfo);
5 onReceiveGEOACK(ProtocolGEOACKInformation appInfo);
6 onReceiveONLINEANSWER(ProtocolInformation appInfo);
```

As sentenças abaixo iniciam o endereço IP do dispositivo, criam um socket e registram um ouvidor (*listener*) ao passar essa classe (BatataQuenteActivity) como parâmetro, pois a mesma implementa `IReceiveListener`.

Código:

```
1 device.startIp();
2 SocketCommunication socket = new CommunicationSocket();
3 socket.acceptListener(this, parameters.getDevice());
```

Seguem exemplos dos diversos usos de envio de mensagens presentes no código fonte da aplicação:

Código:

```
1 byte[] message = messageStream;
2 Device device;
3 Object potatoObject;
4 List<String> list = listObject; //Lista de ips de destino
5
6 //Exemplo de uso do APPDATA usando TCP e enviando um objeto (potatoObject)
7 AppProtocol app = new AppProtocol(EProtocolMessages.APPDATA, list,
      EProtocolTranspLayer.TCP);
8 ProtocolInformation info = new ProtocolInformation(device, potatoObject,
      app);
9 socket.sendMessage(app, info);
10
11 //Exemplo de uso do GEOACK usando TCP e enviando um byte[] (message)
12 AppProtocol app = new AppProtocol(EProtocolMessages.GEOACK, list,
      EProtocolTranspLayer.TCP);
13 ProtocolInformation info = new ProtocolInformation(device, message, app);
14 socket.sendMessage(app, info);
15
16 //Exemplo de uso do GEOMSG usando UDP e sendo enviado para todos os nós da
       rede
17 AppProtocol app = new AppProtocol(EProtocolMessages.GEOMSG, ESendTo.ALL,
      EProtocolTranspLayer.UDP);
18 ShootingRestrictions shootRestrictions = new ShootingRestrictions(10, 1,
      0);
19 ProtocolGEOMSGInformation info = new ProtocolGEOMSGInformation(device,
      message, app, shootRestrictions);
20 socket.sendMessage(app, info);
21
22 //Exemplo de uso do ONLINE usando TCP e sendo enviado para o GATEWAY
23 AppProtocol app = new AppProtocol(EProtocolMessages.ONLINE, ESendTo.
     GATEWAY, EProtocolTranspLayer.TCP);
24 ProtocolInformation protInfo = new ProtocolInformation(device, message,
      app);
25 socket.sendMessage(app, protInfo);
```

6.4 Resultados

Foram feitos diversos experimentos para validar a eficácia dessa aplicação. Inclusive, foi feito um com o orientador deste trabalho, o Professor Daniel Macêdo, no dia 08/02/2013, no bloco C do Instituto de Matemática e Estatística da Universidade de São Paulo, onde, o mesmo, pôde verificar o aplicativo em funcionamento.

Para comprovar, aqui, seu funcionamento foi criado um log usando os rescursos da plataforma de programação do Android.

O experimento foi feito com o auxilio de três dispositivos: *Samsung Galaxy SII*, *LG-P970*, *LG-P350f*. Cada um deles foi posicionado em uma das três coordenadas expostas na Figura 6.3. O dispositivo *Samsung Galaxy SII* foi posicionado em 1, o dispositivo *LG-P970* foi posicionado em 2 e o *LG-P350f* em 3.

Figura 6.3: *Localização no mapa dos dispositivos. Imagem extraída e modificada de http://maps.google.com.br*

A sequência dos eventos dessa demonstração foi:

1. O dispositivo *LG-P970*, localizado em 1, começa o jogo, portanto começa com a batata. A batata foi enviada desse dispositivo para o *LG-P350f*;
2. Em seguida, a batata foi enviada para o dispositivo *Samsung Galaxy SII*;
3. Enfim, a batata será enviada de volta para o *LG-P350f*.

O log detalhado de cada dispositivo encontra-se no Apêndice A.

Durante o experimento, a primeira tentativa de envio sempre foi apontando o dispositivo fonte para o receptor, variando o azimute nas tentativas subsequentes até acertar o receptor. Como se pode ver no log A, foram necessários mais de um envio em todas tentativas.

As Figuras 6.4 e 6.5 a seguir dão um exemplo de acerto e erro do alvo, respectivamente. O ponto azul é o dispositivo receptor, o verde é o fonte e o amarelo é o virtual. Para entendimento de como foram definidos os resultados, ver Seção 5.2.4.

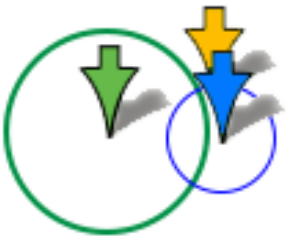

Figura 6.4: *O dispositivo verde acerta o dispositivo azul.*

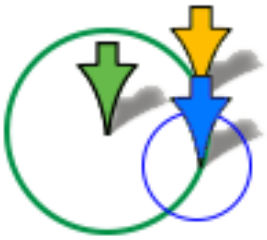

Figura 6.5: *O dispositivo verde erra o dispositivo azul.*

Capítulo 7

Conclusões

Ao longo desta monografia foi apresentado um conjunto de conceitos relacionados com a criação de uma biblioteca para o sistema operacional Android que dê suporte à comunicação entre dispositivos móveis baseada em coordenadas geográficas e que seja independente de um servidor externo.

Para garantir essa independência, a utilização de um dispositivo como um ponto de acesso mostrou-se uma ótima opção devido ao fato desse recurso estar disponível em grande parte dos dispositivos com sistema operacional Android e ser gerenciável pela API do mesmo, e por permitir que a troca de mensagens entre os dispositivos seja feita de forma ágil em uma sub-rede. A maior desvantagem desse recurso é o alto consumo de energia a que o ponto de acesso está submetido.

A alternativa adotada para viabilizar a comunicação por coordenadas geográficas envolve a integração de sensores de movimentação, de posicionamento e do GPS. Cada recurso apontado possui falhas relativas à sua própria natureza, conforme abordado na parte teórica deste trabalho e comprovado por experimentos. O filtro complementar apresenta uma solução de fusão dos sinais de saída dos sensores acelerômetro, magnetômetro e giroscópio, de forma que uns possam compensar o erro dos outros, permitindo resultados bastante satisfatórios para a formação do vetor de orientação de um dispositivo.

Porém, a utilização do GPS para identificação da localização do dispositivo no globo, apesar de ser o instrumento mais preciso para tal finalidade, mostrou-se bastante instável, variando sua qualidade entre diferentes modelos de dispositivos e dependendo de condições climáticas favoráveis e áreas abertas para o seu funcionamento. Tais características implicaram:

- Na manipulação mais cuidadosa do raio de acerto de um dispositivo, sendo que, quanto maior for o raio, maiores serão as chances de um dispositivo ser atingido por uma mensagem enviada em uma determinada direção;
- Em restrições de distância entre usuários que façam uso de aplicações criadas através da biblioteca;
- E no suporte da biblioteca para verificação se a identificação de uma localização é mais acurada que a outra, permitindo optar pelas melhores resultados.

O desenvolvimento do jogo batata-quente através dos recursos da biblioteca *GeoCommunication* demonstrou a facilidade do uso da mesma.

Além de prover as funções básicas para o objetivo proposto, a biblioteca também oferece diversas funcionalidades paralelas, como a escolha de um filtro para tratamento dos sensores, a possibilidade de atribuir uma velocidade no envio da mensagem, entre outros. Os inúmeros

trabalhos estudados e a variedade de possibilidades de recursos a serem implementados que explorem o envio de mensagens referenciadas geograficamente, garantem caráter o extensível da biblioteca.

Em suma, a biblioteca atingiu seu objetivo, mas implicará em restrições da forma de uso para as aplicações que a utilizem, principalmente devido às falhas com hardware, em especial o GPS. À medida que a tecnologia for evoluindo e as soluções aperfeiçoando-se, a eficácia das funções da biblioteca tenderão a aumentar, pois ela foi desenvolvida com módulos independentes do vínculo com o hardware.

Apêndice A

Log de validação de caso de uso

Para melhor compreensão do log, a Tabela A.1 mostra o significado de cada campo do log.

Envia	Recebe	
Campos	**Campos**	**Descrição**
hora	hora	hora que ocorreu o evento
evt	evt	tipo do evento, envia ou recebe.
ip_src	ip_src	endereço IP de quem está enviando a batata
lat_src	lat_src	latitude de quem está enviando a batata
long_src	long_src	longitude de quem está enviando a batata
azi_src	azi_src	azimute de quem está enviando a batata (variação de -180° a 180°)
	ip_dst	endereço IP de quem está recebendo a batata
	lat_dst	latitude de quem está recebendo a batata
	long_dst	longitude de quem está recebendo a batata
	azi_rel	azimute relativo entre o dispositivo que está enviando a batata e o que está recebendo
	lat_devVirtual	latitude do device virtual obtido
	long_devVirtual	longitude do device virtual obtido
	dist_virtual_dst	distância entre o dispositivo virtual e o destino
	raio_dst	raio do alvo, definido na classe `TagertRestrictions`
	acertou	esse valor será SIM se e, somente se, o dist_virtual_dst <= raio_dst, caso contrário será NAO

Tabela A.1: *Significado de cada campo do log usado para o experimento*

Log do Samsung Galaxy SII

Enviado do LG-P970 para o LG-P350f

hora: 09:18:48.414 evt: recebe ip_src: 192.168.43.1 lat_src: -23.425611293654512 long_src: -46.539445856882224 azi_src: 55.47786660887549 ip_dst: 192.168.43.100 lat_dst: -23.425384267098487 long_dst: -46.53941863227273 azi_rel: 6.31458610160877 lat_devVirtual: -23.425481846643745 long_devVirtual: -46.53924191913846 dist_virtual_destino: 21.046290336530866m raio_dst: 5.0m acertou: NAO

Enviado do LG-P350f para Samsung Galaxy SII

hora: 09:19:51.668 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535643 long_src: -46.53950898 azi_src: 73.9950456186299 ip_dst: 192.168.43.100 lat_dst: -23.425384267098487 long_dst: -46.53941863227273 azi_rel: 108.46377239503316 lat_devVirtual: -23.425332195225884 long_devVirtual: -46.53941742131611 dist_virtual_destino: 5.7682874645739535m raio_dst: 5.0m acertou: NAO

hora: 09:19:54.125 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535623 long_src: -46.53950982 azi_src: 68.82401868103996 ip_dst: 192.168.43.100 lat_dst: -23.425384267098487 long_dst: -46.53941863227273 azi_rel: 108.42773572392116 lat_devVirtual: -23.425324190469148 long_devVirtual: -46.53942019399251 dist_virtual_destino: 6.655400532700049m raio_dst: 5.0m acertou: NAO

hora: 09:19:59.985 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42538068 long_src: -46.53952767 azi_src: 69.28339671404751 ip_dst: 192.168.43.100 lat_dst: -23.425384267098487 long_dst: -46.53941863227273 azi_rel: 92.04178137349118 lat_devVirtual: -23.425345064007505 long_devVirtual: -46.53942561772334 dist_virtual_destino: 4.400043393230607m raio_dst: 5.0m acertou: SIM

Enviado desse dispositivo para LG-P350f

hora: 09:20:17.021 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -124.71070284759197

hora: 09:20:19.762 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -109.81261226769013

Log do dispositivo LG-P350f

Enviada do LG-P970 para esse dispositivo

hora: 09: 20: 25.991 evt: recebe ip_src: 192.168.43.1 lat_src: -23.425611293654512 long_src: -46.539445856882224 azi_src: 55.47786660887549 ip_dst: 192.168.43.158 lat_dst: -23.42536343 long_dst: -46.53949434 azi_rel: -10.231747359430093 lat_devVirtual: -23.42546855312627 long_devVirtual: -46.53922097586649 dist_virtual_destino: 30.266197892540095m raio_dst: 5.0m acertou: NAO

hora: 09: 20: 29.953 evt: recebe ip_src: 192.168.43.1 lat_src: -23.425611293654512 long_src: -46.539445856882224 azi_src: -9.811736643678922 ip_dst: 192.168.43.158 lat_dst: -23.42536344 long_dst: -46.53949447 azi_rel: -10.259005493161945 lat_devVirtual: -23.42536309736928 long_devVirtual: -46.53949237181784 dist_virtual_destino: 0.2177626821396704m raio_dst: 5.0m acertou: SIM

Enviada desse dispositivo para o Samsung Galaxy SII

hora: 09:21:28.955 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535643 long_src: -46.53950898 azi_src: 73.9950456186299

hora: 09:21:31.379 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535623 long_src: -46.53950982 azi_src: 68.82401868103996

hora: 09:21:36.477 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42538068 long_src: -46.53952767 azi_src: 69.28339671404751

Enviada do Samsung Galaxy SII para esse dispositivo

hora: 09:21:54.743 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -124.71070284759197 ip_dst: 192.168.43.158 lat_dst: -23.42537969 long_dst: -46.53953964 azi_rel: -87.65279157745806 lat_devVirtual: -23.425447905718297 long_devVirtual: -46.53951818876246 dist_virtual_destino: 7.866541586746193m raio_dst: 5.0m acertou: NAO

hora: 09:21:57.504 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -109.81261226769013 ip_dst: 192.168.43.158 lat_dst: -23.42538007 long_dst: -46.53954046 azi_rel: -87.86194707761561 lat_devVirtual: -23.425422398081263 long_devVirtual: -46.539533328431574 dist_virtual_destino: 4.74415393677668m raio_dst: 5.0m acertou: SIM

Log do dispositivo LG-P970

Enviada desse dispositivo para o LG-P350f

hora: 09:21:06.927 evt: recebe ip_src: 192.168.43.1 lat_src: -23.425611293654512 long_src: -46.539445856882224 azi_src: 55.47786660887549

hora: 09:21:10.839 evt: recebe ip_src: 192.168.43.1 lat_src: -23.425611293654512 long_src: -46.539445856882224 azi_src: -9.811736643678922

Enviada do P350f para o Samsung Galaxy SII

hora: 09:22:10.248 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535643 long_src: -46.53950898 azi_src: 73.9950456186299 ip_dst: 192.168.43.1 lat_dst: -23.425591221194498 long_dst: -46.53944533334948 azi_rel: 165.95588055487548 lat_devVirtual: -23.425289698393698 long_devVirtual: -46.539256869368195 dist_virtual_destino: 38.55011259703721m raio_dst: 5.0m acertou: NAO

hora: 09:22:12.607 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42535623 long_src: -46.53950982 azi_src: 68.82401868103996 ip_dst: 192.168.43.1 lat_dst: -23.42559809700652 long_dst: -46.539452895837314 azi_rel: 167.7467847988573 lat_devVirtual: -23.42526682251047 long_devVirtual: -46.53925971579713 dist_virtual_destino: 41.663305627540176m raio_dst: 5.0m acertou: NAO

hora: 09:22:17.729 evt: recebe ip_src: 192.168.43.158 lat_src: -23.42538068 long_src: -46.53952767 azi_src: 69.28339671404751 ip_dst: 192.168.43.1 lat_dst: -23.425603573204732 long_dst: -46.539467980833074 azi_rel: 166.1194534124102 lat_devVirtual: -23.4252994604957 long_devVirtual: -46.53929494806972 dist_virtual_destino: 38.0405987040766m raio_dst: 5.0m acertou: NAO

Enviada do Samsung Galaxy para o LG-P350f

hora: 09:22:35.785 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -124.71070284759197 ip_dst: 192.168.43.1 lat_dst: -23.42559097516114 long_dst: -46.5394780082857 azi_rel: -165.15425573198215 lat_devVirtual: -23.4255060382297 long_devVirtual: -46.539609131646785 dist_virtual_destino: 16.372630097648436m raio_dst: 5.0m acertou: NAO

hora: 09:22:38.563 evt: recebe ip_src: 192.168.43.100 lat_src: -23.425384267098487 long_src: -46.53941863227273 azi_src: -109.81261226769013 ip_dst: 192.168.43.1 lat_dst: -23.4255917262911 long_dst: -46.539477768850766 azi_rel: -165.26270199674968 lat_devVirtual: -23.425456976172136 long_devVirtual: -46.53963733793916 dist_virtual_destino: 22.10552235805319m raio_dst: 5.0m acertou: NAO

Referências Bibliográficas

[1] Biblioteca virtual da universidade de virgínia. http://www.lib.virginia.edu/scholarslab/resources/class/intro2GIS/introToGPS.pdf. Último acesso em 12/2012. 17

[2] Documentação oficial da biblioteca geotools. http://docs.geotools.org/. Último acesso em 12/2012. 49

[3] Google passes the 200,000 android activations/day mark. http://tech.fortune.cnn.com/2010/08/04/google-passes-the-200000-android-activationsday-mark/. Último acesso em 12/2012. 1, 11

[4] Gps vs agps: A quick tutorial. http://www.wpcentral.com/gps-vs-agps-quick-tutorial. Último acesso em 12/2012. 18

[5] Rfcs (*Request For Comments*). http://www.rfc-editor.org/. Último acesso em 12/2012. 5

[6] Site oficial - android developers. http://developer.android.com/index.html. Último acesso em 12/2012. 8, 11, 12, 13, 16, 29, 41, 45, 54

[7] Site oficial - *Wireshark*. http://www.wireshark.org. Último acesso em 12/2012. 37

[8] Site oficial - mobilewar. http://www.mobilewar.org/index.php/en/. Último acesso em 12/2012. 1

[9] Azzedine Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks*. Wiley-IEEE Press, Novembro 2008. 16

[10] Shane Colton. The balance filter. http://goo.gl/V0ziF. Último acesso em 12/2012. 21

[11] DEFENSE MAPPING AGENCY WASHINGTON DC. Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems. Technical report, Lancaster University, 1991. 19

[12] R. E. Deakin and M. N. Hunter. Geodesics on an ellipsoid - bessel's method. Technical report, RMIT University - School of Mathematical Geospatial Sciences, 2007. 19

[13] Vassilis Kostakos Denzil Ferreira, Anind K. Dey. Understanding human-smartphone concerns: A study of battery life. Technical report, Universidade de Madeira - Portugal, 2011. 12

[14] Rabie Khodr Jradi e Lasse Seligmann Reedtz. Ad-hoc network on android. Technical report, Technical University of Denmark, 2010. 8, 12, 16

[15] Lindsay Kleeman. Understanding and applying kalman filtering. http://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated3/kleeman_kalman_basics.pdf. Último acesso em 12/2012. 20

[16] Nisarg Kothari. Extended kalman filter for cell phone orientation tracking. http://www.sfonge.com/forum/topic/measuring-movement-accelerometer-and-gyroscope. Último acesso em 12/2012. 14, 20, 22

[17] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet.* Pearson Addison Wesley, terceira edition, 2006. 5

[18] Viktor Moell och André Horntvedt. Positioning for mobile phones using wlan and accelerometer data. Technical report, Lund University, 2012. 14, 15, 16

[19] Gabor Paller. Motion recognition at droidcon 2011. http://www.sfonge.com/forum/topic/measuring-movement-accelerometer-and-gyroscope. Último acesso em 12/2012. 14, 20

[20] B. Merminod Q. Ladetto. Digital magnetic compass and gyroscope integration for pesdestrian navigation. Technical report, 2002. 22

[21] Andy Rubin. 900,000 android devices activated each day. http://www.bgr.com/2012/06/11/android-activations-google-andy-rubin/. Último acesso em 12/2012. 1, 11

[22] Angel Rodriguez Ubejd Shala. Indoor positioning using sensor-fusion in android devices. Technical report, Kristianstad University, 2011. 13, 14

[23] Steven J. Vaughan-Nichols. Android/linux kernel fight continues. http://blogs.computerworld.com/16900/android_linux_kernel_fight_continues. Último acesso em 12/2012. 11

[24] T. Vincenty. Direct and inverse solutions of geodesics 0 the ellipsoid wlth application of nested equations. http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf. Último acesso em 12/2012. 20