Uma biblioteca para comunicação ciente de localização geográfica entre dispositivos moveis

Renato Avila dos Santos e Tonny Costa Cordeiro

Orientador: Prof. Dr. Daniel Macedo 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, Dezembro de 2012

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 cumpre com essa funcionalidade e sem a necessidade de utilização de servidores externos. Para viabilizar a comunicação de forma padronizada entre as aplicações que usem os recursos da biblioteca, foi criado um protocolo de aplicação.

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

Palavras-chave: gps, sensores, android, biblioteca.

Abstract

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

Those technologies provide the necessary data for device communication through geographic landmarks but there are only a few applications that enable one device to send messages to another device simply by pointing at it. This work proposes the creation of a library to enable such functionality without the need of external servers. To accomplish a standardized communication between applications, a protocol was created.

The protocol's implementation and the basic functions for its use will be available as an open source library. In order to test it's functionality, we will use it in a simple game.
Keywords: gps, sensors, android, library.

Sumário

1 Introdução **1**

1.1 Motivação . . . 1
1.2 Objetivos . . . 2
1.3 Organização da monografia . . . 2

2 Conceitos sobre redes **3**

2.1 Arquitetura TCP/IP . . . 3
2.2 Camada de aplicação . . . 3
2.2.1 HTTP . . . 4
2.3 Camada de transporte . . . 5
2.3.1 UDP . . . 5
2.3.2 TCP . . . 5
2.4 Redes infraestruturadas e ad-hoc . . . 6
2.4.1 Rede ad-hoc . . . 6
2.4.2 Rede infraestruturada . . . 7
2.5 Envio de mensagens - sockets . . . 7
2.5.1 Socket . . . 8
2.5.2 Unicast . . . 8
2.5.3 Broadcast . . . 8
2.5.4 Multicast . . . 9
2.6 Tecnologias wireless . . . 10
2.6.1 Wi-fi . . . 10
2.6.2 Bluetooth e NFC . . . 10

3 Conceitos sobre sensores e Android **13**

3.1 Sistema operacional Android . . . 13
3.1.1 Algumas características . . . 13
3.1.2 Programação para Android . . . 13
3.1.3 Rede ad-hoc no Android . . . 14
3.1.4 Celular como ponto de acesso . . . 14
3.2 Sensores para Android . . . 14
3.2.1 Taxa de amostragem . . . 15

3.2.2 Magnetômetro . . . 16
3.2.3 Acelerômetro . . . 16
3.2.4 Giroscópio . . . 17
3.2.5 Vetor de orientação . . . 17
3.3 Sistemas de posicionamento . . . 18
3.3.1 GPS . . . 19
3.3.2 AGPS . . . 20
3.3.3 Cálculos geodésicos . . . 21
3.4 Filtro de erros dos sensores . . . 22
3.4.1 Filtro complementar . . . 22
3.4.2 Outras soluções . . . 24

4 Experimentos **25**
4.1 GPS no Android . . . 25
4.1.1 Movimento com paradas a cada 2 metros . . . 27
4.1.2 Movimento sem paradas . . . 29
4.1.3 Movimento de ida e volta . . . 30
4.1.4 Dispositivos parados . . . 30
4.2 Bússola digital no Android . . . 32
4.2.1 Análise da variação angular . . . 34
4.2.2 Análise da direção apontada pelo dispositivo . . . 36
4.3 Comportamento da rede via *Wireshark* . . . 37
4.3.1 Detalhes do experimento . . . 37
4.3.2 Resultados . . . 38
4.3.3 Conclusões . . . 39
4.4 Teste de confiabilidade . . . 39
4.4.1 Experimento . . . 40
4.4.2 Resultados . . . 40
4.4.3 Conclusões . . . 40

5 Biblioteca **41**
5.1 Estruturação da biblioteca . . . 41
5.2 Protocolo da camada de aplicação . . . 41
5.2.1 GCP - *GeoCommunication Protocol* . . . 42
5.3 Implementação . . . 44
5.3.1 *CommunicationSocket* . . . 44
5.3.2 *CommunicationPacket* . . . 44
5.3.3 *Device* . . . 44
5.3.4 *NetworkManager* . . . 44
5.3.5 *NetworkConfiguration* . . . 44

5.3.6 *NetworkAvailable* 44
5.3.7 *Interface IReceiveListener* 45
5.3.8 GeodesicManager 45
5.3.9 Biblioteca Geotools2 45
5.3.10 Filtro de erros DRNF 45
5.4 Restrições de uso 46

6 Resultados e caso de uso **49**
6.1 Jogo batata-quente 49
6.2 Requisitos 49
6.2.1 Instalação e execução 50
6.2.2 Usando a aplicação 50
6.3 Resultados 50

7 Conclusões **53**

Referências Bibliográficas **55**

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, se bem sucedido, 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 deixa mais clara 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 de novos usuários desse sistema operacional, por dia, aumentou de 200.000 para 900.000 [3, 22].

1.1 Motivação

A motivação para 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 do sensor acelerômetro de dispositivos móveis, esse aplicativo permite a criação de um ambiente interativo nos quais os celulares são simulados como armas. A biblioteca 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 das bússolas e GPSs de alguns celulares disponíveis no mercado brasileiro.

1.3 Organização da monografia

A monografia está organizada da seguinte forma: O Capítulo 2 e 3 descreve 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. No Capítulo 8 são apresentadas as análises subjetivas do curso de Bacharelado em Ciência da Computação por cada um dos autores dessa monografia. Informações adicionais são apresentadas no Apêndice.

Capítulo 2

Conceitos sobre redes

Este capítulo apresenta alguns conceitos básicos sobre redes para compreensão dos demais capítulos da monografia. 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 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 uma ilustração das camadas desta arquitetura. Cada camada sempre fornece serviços para a camada imediatamente superior.

Aplicação
Transporte
Rede
Enlace
Física

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

Nas Seções 2.2 e 2.3 serão apresentadas mais informações sobre as camadas de aplicação e transporte e como elas podem auxiliar na construção da biblioteca.

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. Parece para a aplicação que ela está se comunicando diretamente com a outra, sem saber e se preocupar com o que acontece entre envio e o recebimento das mensagens.

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

Há necessidade de criação de protocolos para que essas aplicações possam se comunicar com outras aplicações de um modo simples e único. Imagine duas pessoas conversando, digamos que em português, sobre algum determinado assunto. Isso é uma analogia à comunicação entre aplicações, onde cada pessoa é uma aplicação diferente e o idioma, português, é o protocolo usado para a 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, que é o objetivo dessa seção.

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 as sintaxes (regras) das mensagens para realizar a comunicação entre cliente e servidor.

Quando um cliente faz uma requisição de um recurso que está num determinado servidor, a mensagem deve conter 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 quer o conteúdo do recurso requerido, enquanto o método POST envia dados a serem processados pelo recurso (os dados são enviados no corpo da mensagem opcional). 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 processar essa requisição e responder ao cliente qual foi o resultado desse processo. Assim como na requisição, a mensagem de resposta tem um formato bem definido. A mensagem é 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 uma frase de justificativa que descreve o código de status.

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 uma resposta de uma requisição bem sucedida.

```
HTTP/1.1 200 OK
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
```

```
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

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. Os protocolos mais comuns dessa camada são o UDP e o TCP.

2.3.1 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 no 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.

Apesar dessas desvantagens, o UDP é um protocolo bem mais simples do que muitos outros, como o TCP, que será visto na Subseção 2.3.2. 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 2.5.3) e Multicast (Seção 2.5.4), 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.

A simplicidade do protocolo também pode ser percebida no cabeçalho da mensagem. O cabeçalho, que contém apenas oito bytes, contém apenas a porta de destino (dois bytes), o comprimento da mensagem (dois bytes), a porta de origem (dois bytes) e o checksum (dois bytes), sendo os últimos dois opcionais. Portanto, aplicações que usam o UDP como protocolo de transporte 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.

Aplicação na biblioteca

O protocolo é importante na identificação de um novo aparelho na rede, pois, ao se conectar na mesma, o dispositivo é desconhecido de todos os outros já conectados. Então, é sua função avisar a todos a sua presença. Como ainda não se sabe os endereços IPs dos outros dispositivos, é necessário o envio de uma mensagem em modo Broadcast que será encapsulada em segmentos UDP.

2.3.2 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.

A principal vantagem desse protocolo, em relação ao UDP, é a confiabilidade da entrega de mensagem. Durante o tráfego de pacotes, é de responsabilidade do TCP 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. 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 mensagem ser enviada.

Uma conexão TCP é ponto a ponto, portanto o único modo de envio é Unicast (Seção 2.5.2). Diferente do UDP que suporta os modos broadcast e multicast.

Oposto à simplicidade do cabeçalho do UDP, o cabeçalho do TCP, cujo tamanho varia de 20 bytes a 60 bytes, mais do que 7 vezes o tamanho do cabeçalho UDP, carrega muito mais informação.

Aplicação na biblioteca

Após um novo dispositivo ser identificado pelos outros aparelhos na rede, estes também devem ser identificados por ele. Então, ao receber uma mensagem informando a entrada de um novo usuário na rede, o receptor deve enviar uma mensagem de resposta ao novo membro para que este o inclua em sua lista de dispositivos. Nesse tipo de envio é conhecido o IP de destino, o que permite a utilização do protocolo TCP, garantindo a confiabilidade no envio e recepção dos pacotes trafegados. Portanto, se um dispositivo A encontra-se na lista de B, então B estará na lista de A.

2.4 Redes infraestruturadas 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. Não é necessário que haja uma estrutura física fixa para que os nós se comuniquem, portanto é a rede ideal para lugares com uma infraestrutura de suporte precária e quando for necessário manter a comunicação por um pequeno intervalo de tempo.

Apesar de ser um modo pouco conhecido e usado, a rede ad-hoc é uma solução para muitas aplicações, tais como as de cunho militar ou as que rodem em deserto ou florestas.

Esse tipo de rede serve até para computadores próximos um do outro, onde há necessidade de troca de informações entre eles mas não é desejável o uso da Internet.

Entretanto, há muitas desvantagens envolvendo redes ad-hoc. 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.

Aplicação na biblioteca

Era objetivo do projeto a implementação na biblioteca *GeoCommunication* de recursos que garantam a comunicação entre os dispositivos em redes ad-hoc. Isso se devia ao fato de uma rede desse tipo não precisar de uma infraestrutura externa, como a Internet, para troca de mensagens. Porém, devido a alguns fatores citados na subseção 3.1.3, houve alguns empecilhos para sua utilização, sendo abortado o seu uso na primeira versão da biblioteca.

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 os nós 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 nó, 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.

Aplicação na biblioteca

A opção pelo uso desse tipo de rede na nossa biblioteca deve-se a algumas facilidades. A principal delas é o fato do Sistema Operacional Android já prover, em diversos aparelhos, a possibilidade de transformar um celular num ponto de acesso. Quando os dispositivos se conectam na rede criada por um ponto de acesso, passam a pertencer à mesma sub-rede e compartilham o mesmo enlace (*wireless*), implicando na comunicação entre os nós de destino e origem de forma direta, sem passar pelo ponto de acesso ou demais nós na rede, conforme pode ser averiguado pelo experimento com o Wireshark descrito na Seção 3.3.

2.5 Envio de mensagens - sockets

Ao criar uma aplicação devemos escolher qual o protocolo de transporte que queremos usar. O TCP ou o UDP. Para tomar essa decisão devemos 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. Mas antes será introduzido o conceito de socket.

2.5.1 Socket

Socket é o canal de comunicação entre dois processos. Esse conceito é muito importante para entendermos 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" [18].

Um socket pode ser identificado por uma porta, e é por essa porta que 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 a 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.

Aplicação na biblioteca

Na biblioteca é feito o uso de Sockets em todo momento que se deseja enviar uma mensagem. *DatagramSocket* e *StreamSocket* são duas classes do pacote java.net que providenciam funcionalidades para lidar, respectivamente, com os protocolos UDP e TCP. Elas são encapsuladas na classe *CommunicationSocket* da biblioteca *GeoCommunication*, isentando o programador da preocupação de qual tipo de socket ele deva usar, cabendo a ele a definição da forma de envio, unicast ou broadcast .

2.5.2 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.

Aplicação na biblioteca

Unicast é usado na biblioteca para o envio de mensagens com o protocolo TCP, o que garante a entrega das mesmas, por exemplo, a mensagem ARVA do protocolo de aplicação criado, que será apresentado no Capítulo 5, usa esse tipo de envio. Para o uso desse modo, é necessário que a aplicação saiba previamente todos os endereços IP de destino.

2.5.3 Broadcast

É uma comunicação entre um único nó de origem e todos os outros nós da rede. Ou seja, quando for enviada uma mensagem em modo broadcast, todos os outros nós da rede a receberão. Portanto, não é necessário que no envio sejam especificados explicitamente todos os endereços IPs de destino, basta apenas indicar que a mensagem é do tipo broadcast e todos os nós da rede irão recebê-la.

Mas como identificar que a mensagem é broadcast? Como o nó deve aceitar a mensagem se o seu endereço IP não está como de destino? Simples. Toda rede criada já possui um endereço especial, o endereço de broadcast. Então, ao enviar uma mensagem, o endereço de

broadcast é atribuído ao endereço IP de destino e todos os nós dessa rede recebem a mesma. Portanto, todo nó deve aceitar mensagens com dois possíveis endereços de destino, o seu próprio endereço IP ou o endereço de broadcast da rede.

Apesar da ideia de broadcast ser simples, seus algoritmos de roteamento não são tão simples como o do unicast. Um dos seus principais desafios é enviar o menor número de pacotes pelos enlaces a fim de evitar a sobrecarga da rede.

Aplicação na biblioteca

Esse modo é usado para o envio de mensagens com o protocolo UDP. Apesar dessa forma de envio não ser confiável, foi feito um teste de confiabilidade na rede, conforme Seção 4.4, através do qual foi constatado um alto índice de confiabilidade na rede. Uma vez que a principal comunicação entre os dispositivos na rede é baseada em uma checagem na camada de aplicação para identificação do destino (é verificado se um vetor de direção atingiu um certo ponto geográfico ou não), e não na checagem de um endereço IP de destino na camada de rede, essa forma de envio é bastante adequada para aplicações a serem desenvolvidas através da biblioteca.

2.5.4 Multicast

É uma comunicação entre um único nó de origem e múltiplos destinatários. Esse modo é muito parecido com o broadcast, mas o fato de a mensagem ser enviada apenas a um grupo e não a todos os nós, torna essa transmissão especial.

Assim como no broadcast, o multicast também precisa de um endereço especial. 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, e 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, na maioria das vezes localizados em diversas sub-redes distribuídas.

Aplicação na biblioteca

Este modo de envio não é implementado nessa versão da biblioteca, pois a principio não há necessidade de criar grupos de celulares na rede e, portanto, não é necessário tal modo de envio.

Visto as diferenças entre unicast, broadcast e multicast, é desafio deste trabalho encontrar um ou mais modos entre esses que satisfaçam as necessidades da biblioteca proposta. A necessidade de envio de mensagens broadcast é evidente, porém esta usa o protocolo UDP, que não garante entrega dos pacotes. Portanto a solução que foi adotada mescla dois modos de envio: unicast via protocolo TCP, garantindo confiabilidade na troca de mensagens; e broadcast, que será usado para transmitir dados para todos os nós da rede.

2.6 Tecnologias wireless

Através das tecnologias *Wireless* é possível a comunicação entre dispositivos sem que haja cabo de comunicação ligando um ao outro. Existem diversas tecnologias wireless. Bluetooth, Wi-fi, infravermelho, 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 [15]) 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 do Wi-Fi, apresentaremos nessa seção também sobre 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, 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 [15], 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. Redes Wi-fi usam um canal de frequência para a comunicação, que pode sofrer interferência de outra rede Wi-fi se esta estiver usando o mesmo canal ou um canal que tenha intervalos de frequência em comum. Para minimizar este 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. Essas tecnologias são usadas em comunicação pareada de curta distância, enquanto o Wi-fi 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 se usar 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 desvantagem que torna impossível o uso dessas tecnologias para a biblioteca. A necessidade de um pareamento para que haja comunicação inviabiliza a ideia da comunicação ser feita baseada em referência geográfica, 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 na área de telefonia móvel, comprou a empresa em 2005 e 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, em 2010, eram vendidos mais de 200.000 aparelhos por dia [3], e recentemente, em junho/2012, segundo o Vice Presidente Sênior da Google, Andy Rubin, vende-se mais de 900.000 aparelhos por dia [22].

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. É um sistema baseado no kernel do Linux [24]. O código é aberto desde Outubro de 2008 e a licença é 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. A melhor opção para começar a programar para Android é instalando o Eclipse (IDE), o SDK do Android e o plugin do eclipse ADT (*Android Development Tools*) [6].

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

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

Recomendamos ao leitor mais interessado que leia as documentações básicas no site oficial dos desenvolvedores para Android [6] antes de começar a programar. Entender as funções principais e o fluxo da aplicação é essencial ao desenvolvedor iniciante.

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 quer dizer que não é possível criar uma rede desse modo. Para cria-la é preciso programar no Kernel do Linux e portanto o código deve ser escrito em C ou C++.

Para manipular o Kernel do Linux no Android, é necessário ter acesso *root* no aparelho. Como esse tipo de acesso depende de cada dispositivo e versão do SO [15], 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

Visto a dificuldade de criar uma rede ad-hoc, optou-se por criar uma rede infraestruturada. Para isso é necessário um ponto de acesso. Este ponto de acesso deve ser 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 isso torna o uso da biblioteca totalmente independente de qualquer infraestrutura externa, tal 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 [14].

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 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 calibragem de acordo com o tipo de sensor.

O framework para os sensores acelerômetro, giroscópio e magnetômetro usam 3 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 desta seção comentará mais a respeito dos sensores acelerômetro, giroscópio e magnetômetro.

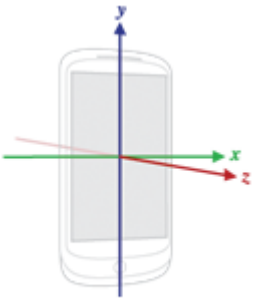

Figura 3.1: *Sistema de coordenadas do dispositivo conforme API de Sensores do Android. Figura extraída de [6].*

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. O tempo de atraso a ser definido pelo programador está atrelado ao tipo de atividade na API do Android: normal (0,2s), jogos (0,02s), interface de usuário (0,06s) ou "o mais rápido"(0s), podendo ser definido manualmente a partir da versão 3.0 do Android (nível de API 11). Conforme documentação em [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.

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 [23].

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 obtida em experimento com dispositivos ora em movimentos aleatórios ora parados. Tabela extraída de [23]*

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 feita 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 investigado em [17] a possibilidade de identificação de interferências magnéticas a partir da variação desses valores esperados.

3.2.3 Acelerômetro

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

Conforme Gabor Paller em [20], sua primeira utilização ocorreu em 2005 no celular Nokia 5500 "*sport device*", no SO Symbian. Trata-se de um sensor disponível desde a versão 1.0 do Android e presente amplamente nos diversos modelos com esse 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 [19], a utilização de uma tradicional forma para uso dos dados do acelerômetro conhecida como *Dead-reckoning* [1], quando

[1] cálculo da posição atual pelas anteriores

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 desse valor, através da multiplicação do mesmo pela diferença 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 [19]).

Dois erros muito comuns em giroscópios são o bias (polarização) e o drift (deriva). O primeiro corresponde à ocorrência de saídas 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 a 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 dos sensores magnetômetro e acelerômetro. Dessa matriz, é possível extrair diretamente o vetor de orientação do quadro de referência da Terra através do método `getOrientation` ou multiplicá-la pelo vetor de orientação obtido através do giroscópio (quadro de referência do dispositivo), tal como ilustrado na Figura 3.2 e Tabela 3.2.

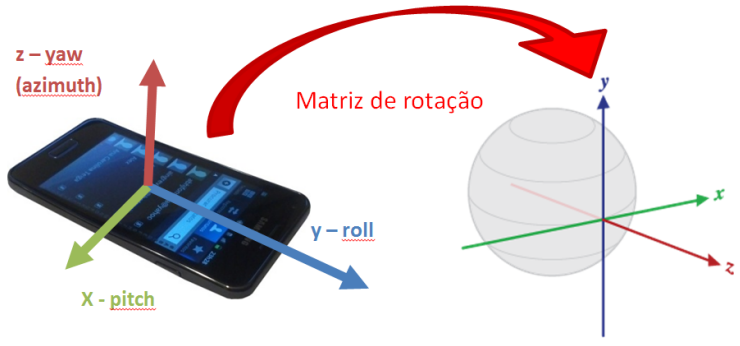

Figura 3.2: *Transposição do vetor de orientação do quadro de referência do dispositivo para o da Terra. Figura extraída e modificada de [6]*

	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*

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 [10], 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 [10] 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 [19] 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 rede. 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 irão fazer a posição estimada variar.

Ao utilizar o WIFI (rede wireless), a acurácia de posicionamento do dispositivo será similar ao alcance de acesso de um típico roteador WIFI (cerca de 300m em locais abertos, conforme [15]), que não necessita estar conectado a uma rede. Além da alta margem de erro em relação ao GPS (discutido a seguir), a cobertura de WIFI não é completa e varia de acordo com a localização.

As seções 3.3.1 e 3.3.2 descrevem os sistemas de posicionamentos implementados 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.

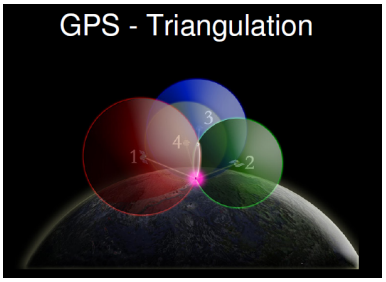

Figura 3.3: *Ilustração do algoritmo de obtenção do GPS. Imagem extraída de [1]*

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. Também deve-se levar em consideração as possíveis margens de erro de acurácia na apuração de um posicionamento. Conforme Tabela 3.3, podem ocorrer situações críticas, dependendo da aplicabilidade dos resultados.

Origem do erro	Erro
Efeitos ionosféricos	± 5 metros
Efeitos toposféricos	± 0,5 metros
Erros efemérides	± 2,5 metros
Erros no relógio do satélite	± 2 metros
Distorção em múltiplos caminhos	± 1 metro
Erros numéricos	± 1 metro

Tabela 3.3: *Acurácia relativa à obtenção de posicionamento por satélites*

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 e pelo instante de tempo constantes em satélites relevantes. 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.

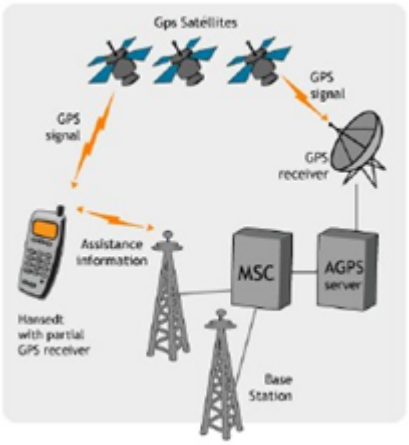

Figura 3.4: *Ilustração acerca do AGPS e sua infraestrutura de funcionamento. Imagem extraída de [16]*

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 descrevem tal como uma elipse, e que é utilizado por GPS, é o WGS84, correspondente ao sistema geodésico mundial criado em 1984, sujeito a revisões em anos posteriores.

A partir da Figura 3.5, são apresentados os conceitos comuns relacionados à ciência em questão. Uma coordenada geodésica corresponde a um ponto formado por latitude() e longitude() 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() 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.

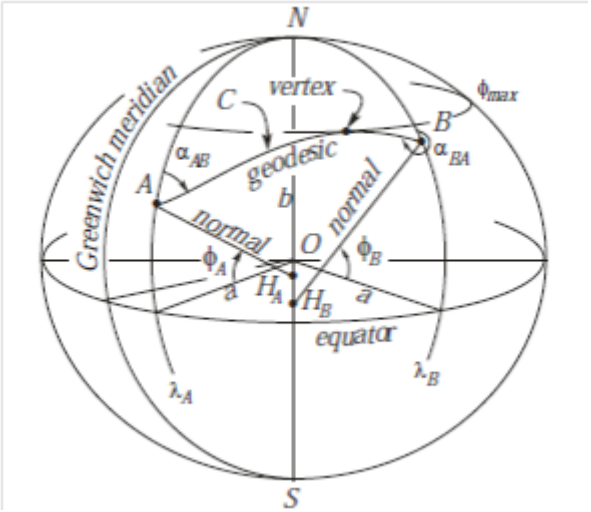

Figura 3.5: *Curva geodésica sobre um elipsoide. Figura extraída de [13]*

Os principais problemas 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 nã tem melhor performace. 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 [25], que apresentou uma solução bastante utilizada em computação, os erros na obtenção de azimutes são largos 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 devido ao erro de azimute, alcança um valor máximo de 0,3mm por unidade da 5ª casa decimal de segundos das medidas de latitude e longitude. Conforme [11], essa relação de medidas passou a ser utilizado como um padrão computacional de comprimento entre pontos de elipses.

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 [20], 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, o que não o inviabiliza, por possuir resultados 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 [17], 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 linearização.

Essas característica 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 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 [16], 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 de 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 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.

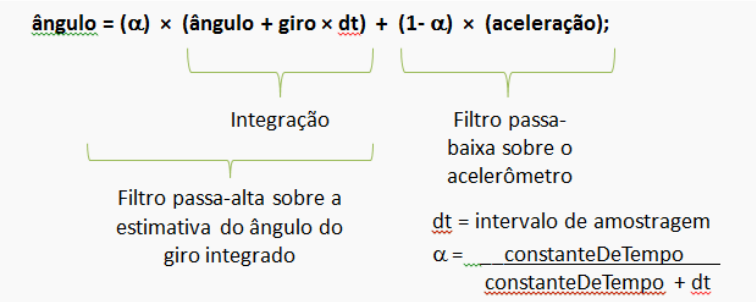

Figura 3.6: *Fórmula de obtenção do ângulo de orientação através do filtro Complementar*

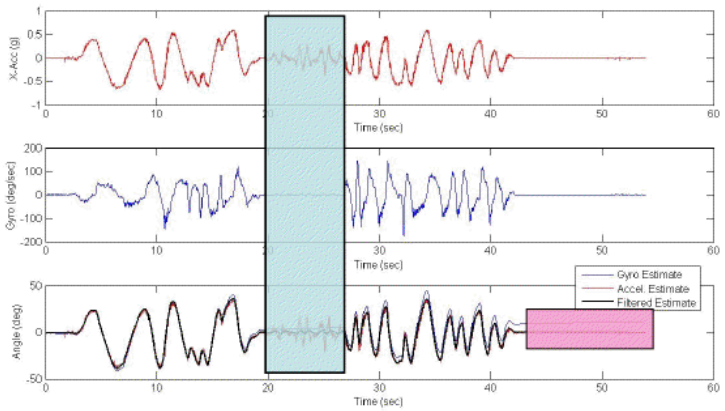

Figura 3.7: *Extraída de [12]. Gerenciamento dos erros pelo filtro Complementar.*

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 discute-se 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 [17]. 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 Euler. A Tabela 3.4 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 [21], 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 aproximado		
Teste	**acel + bussola**	**Giroscópio**	**Filtro Kalman**
rotação 360°	8°	2°	2°
rotações aleatórias (10s)	16°	10°	5°

Tabela 3.4: *Tabela extraída e traduzida de [17]. Comparação dos erros relativos a métodos de obtenção de rotação através de sensores.*

Capítulo 4

Experimentos

Antes de iniciar a implementação da biblioteca, foi necessário avaliar a precisão dos sensores dos celulares, já que a localização dos dispositivos será crucial para o correto funcionamento da biblioteca. 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 todas as informações lidas da bússola e do GPS;
- Utilização de 5 dispositivos móveis, sendo 4 celulares dos modelos LG-P350f (2 unidades), Samsung Galaxy SII e Samsung GT-S5570B; e um tablet do modelo Ideos S7.

Com respeito aos experimentos através dos quais buscamos a recuperação de dados a partir de sensores do celular (GPS, bússola digital, acelerômetro, entre outros), 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 trafego de pacotes na rede, com o auxílio do programa Wireshark. A Seção 4.4 apresenta um teste de confiabilidade para verificar a quantidade de mensagens perdidas no envio em broadcast.

4.1 GPS no Android

A fim de realizar a persistência dos dados relativos ao posicionamento geográfico (latitude e longitude) de diferentes celulares, foi escrita uma aplicação que armazena, no sistema de arquivos do celular, o instante de tempo e a posição atual do dispositivo. 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.

```
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   /* (...) */
20 }
```

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.

Durante a 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:

$$\begin{aligned} &\text{TAXA DE ERRO} \\ &= \text{Diagonal da plataforma} \\ &= \text{raiz quadrada}((Larguradaplataforma)^2 + (alturadaplataforma)^2) \\ &= 50\text{cm}. \end{aligned}$$

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.

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;

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*

- 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.

4.1.1 Movimento com paradas a cada 2 metros

O experimento em questão correspondeu ao deslocamento dos dispositivos por 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.

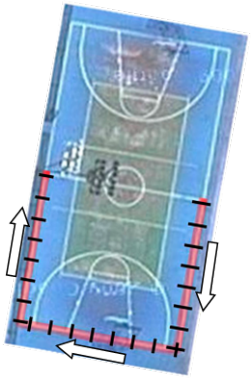

Figura 4.2: *Foto de satélite da quadra na qual foram feitos os testes de campo. Cada linha contínua vermelha equivale a um deslocamento e cada linha preta perpendicular à vermelha equivale a um ponto de parada. As setas indicam o percurso percorrido. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20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

A figura 4.3 exibe o resultado gráfico 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.

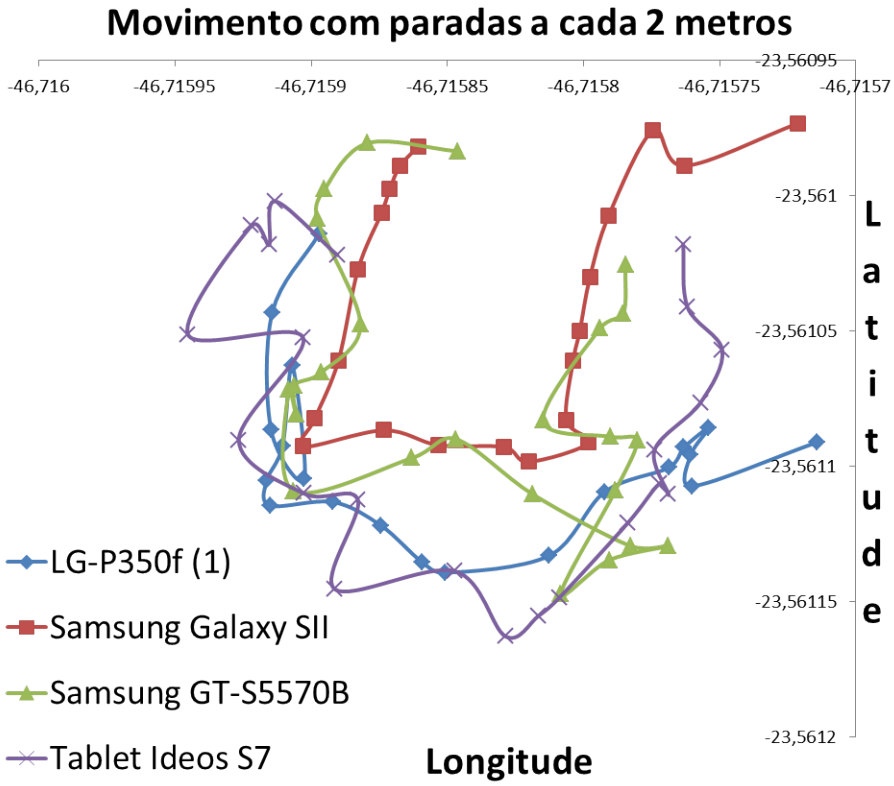

Figura 4.3: *Gráfico que descreve um movimento com paradas a cada 2 metros realizado por 4 dispositivos móveis, a partir de posições coletadas do GPS.*

Evidentemente, 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; existem alguns pontos de parada que não foram reconhecidos pelo programa.

Nos outros dispositivos a situação é um pouco mais alarmante devido à disparidade em relação ao trajeto real. Por isso, escolhemos o Samsung Galaxy SII para exemplificar unicamente os dois próximos testes de campo (Seções 4.1.2 e 4.1.3).

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. 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.

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

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

4.1.2 Movimento sem paradas

O experimento em questão correspondeu ao deslocamento dos dispositivos por três linhas retas consecutivas sem paradas, conforme esquema representado sobre a foto de satélite do local na Figura 4.4.

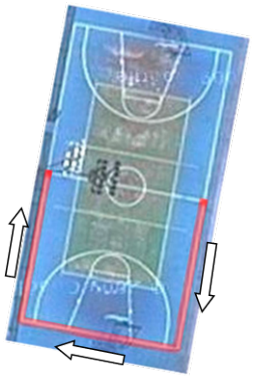

Figura 4.4: *Foto de satélite da quadra na qual foram feitos os testes de campo. A linha contínua vermelha equivale a um único movimento sem paradas. As setas indicam o percurso percorrido. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação.*

A Figura 4.5 apresenta o resultado gráfico obtido a partir das posições geográficas coletadas:

Claramente, a descrição do movimento, principalmente daquele relativo à primeira reta, destoa bastante do trajeto original.

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

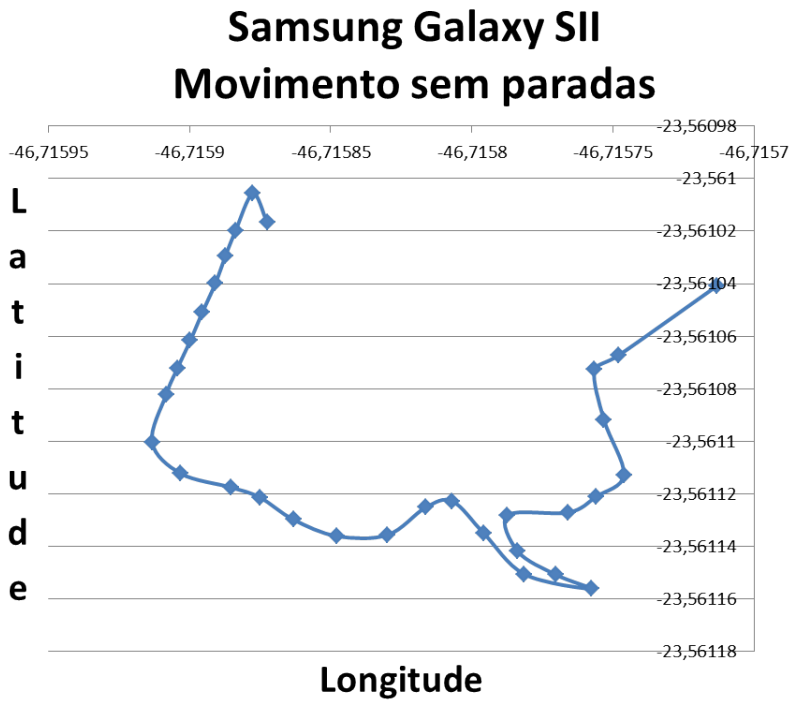

Figura 4.5: *Gráfico que descreve um movimento contínuo realizado pelo dispositivo Samsung Galaxy SII, a partir de posições coletadas do GPS.*

4.1.3 Movimento de ida e volta

O experimento em questão correspondeu ao deslocamento dos dispositivos por 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 4.6.

Na Figura 4.7 é apresentado o resultado gráfico 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.

4.1.4 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

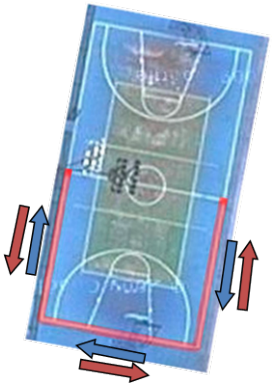

Figura 4.6: *Foto de satélite da quadra na qual foram feitos os testes de campo. A linha contínua vermelha equivale a dois movimentos, o primeiro no sentido das setas azuis (ida) e o segundo no sentido das vermelhas (volta). Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação.*

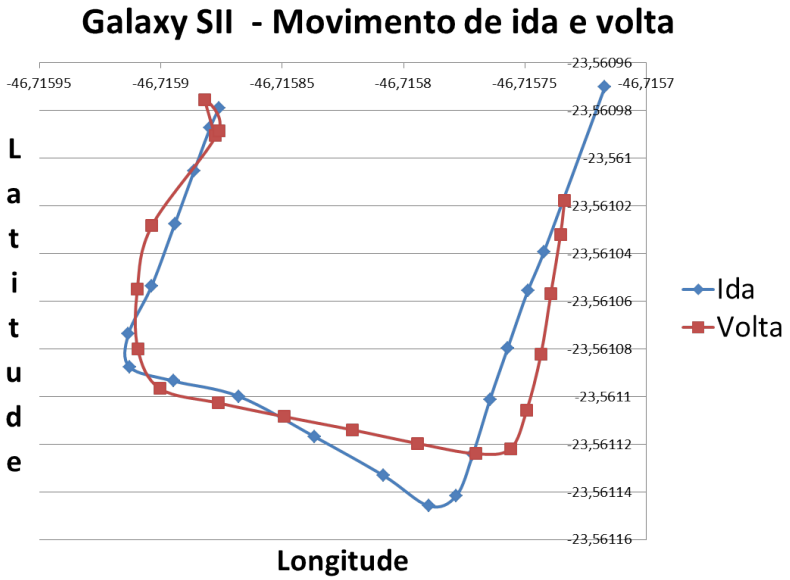

Figura 4.7: *Gráfico que descreve dois movimentos contínuos realizados pelo dispositivo Samsung Galaxy SII, a partir de posições coletadas do GPS.*

foto de satélite do local na Figura 4.8.

A Figura 4.9 apresenta o resultado gráfico 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 2 pontos mais extremos medindo 9,7 metros.

Essa discrepância entre os pontos é ainda mais nítida ao apontarmos a média e desvio padrão dos pontos do gráfico, lembrando que a distância entre dois aparelhos não ultrapassa 50 cm. 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 4.9 para esses dois celulares.

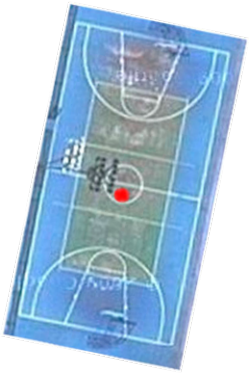

Figura 4.8: *Foto de satélite da quadra na qual foram feitos os testes de campo. O círculo vermelho indica o local no qual os celulares foram posicionados. Imagem extraída de http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20 e modificada para fins de representação.*

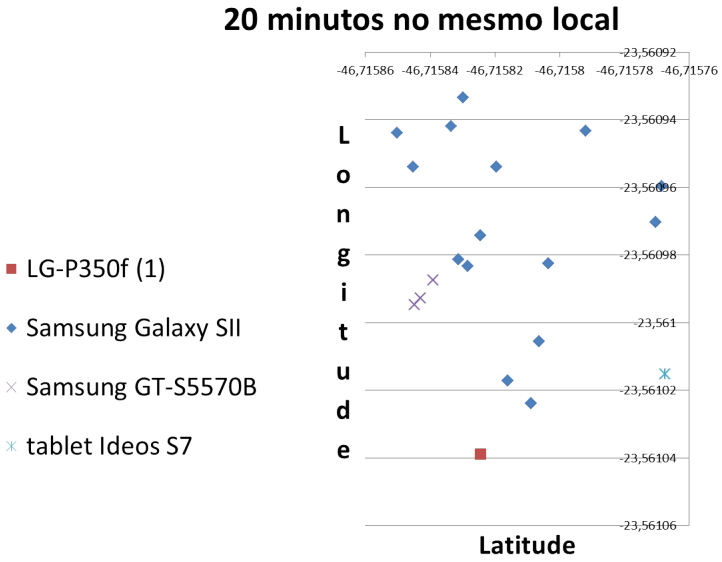

Figura 4.9: *Gráfico em que são plotadas as posições coletadas do GPS de 4 dispositivos móveis mantidos em um mesmo local físico por aproximadamente 20 minutos.*

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 dos sensores TYPE_ACCELEROMETER e TYPE_MAGNETIC_FIELD, 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. O código completo está disponível em http://www.linux.ime.usp.br/~avila/mac499.

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

```
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 4 celulares dos seguintes modelos: *LG-P350f* (2 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 4.10), com paradas temporárias a cada deslocamento de 15º, ao redor do eixo ortogonal à plataforma que cruza a origem do círculo de referência de movimentação (Figura 4.11). O movimento é composto por uma volta completa de 360º, com um total de 25 paradas, considerando 0º e 360º como duas paradas distintas.

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

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

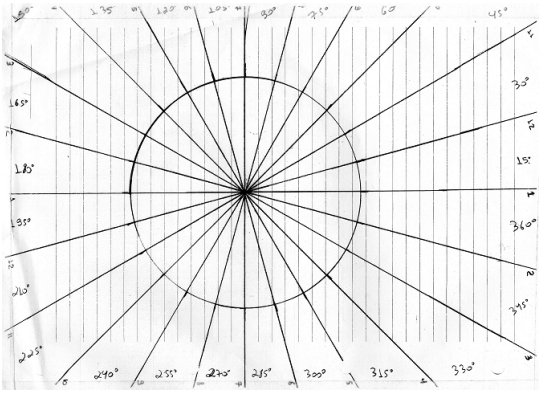

Figura 4.10: *Plataforma utilizada para controle da variação angular dos dispositivos durante os experimentos com bússola digital.*

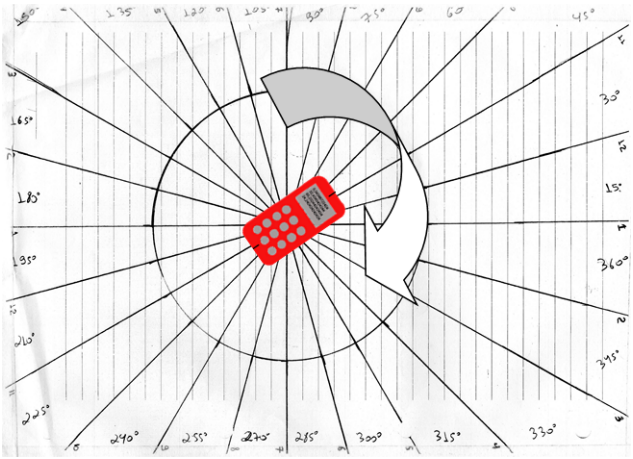

Figura 4.11: *Rotação do celular sobre a plataforma utilizada para controle da variação angular dos dispositivos durante os experimentos com bússola digital.*

4.2.1 Análise da variação angular

Seguem alguns gráficos (Figura 4.12) relacionados ao experimento nº 1 que foi citado na seção anterior. Cada gráfico possui 2 círculos:

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

- e um círculo vermelho tracejado dividido em 24 setores com 15º cada, que corresponde ao objetivo ideal de variação angular. Dessa forma, uma medição ideal seria aquela que posicionasse as linhas azuis exatamente sob as linhas vermelhas;

A Tabela 4.5 apresenta estatísticas referentes ao experimento número 1 para avaliar a precisão da bússola digital.

Conforme os dados da Tabela 4.5, 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 teve o maior desvio padrão para os ângulos formados entre dois pontos de parada subse-

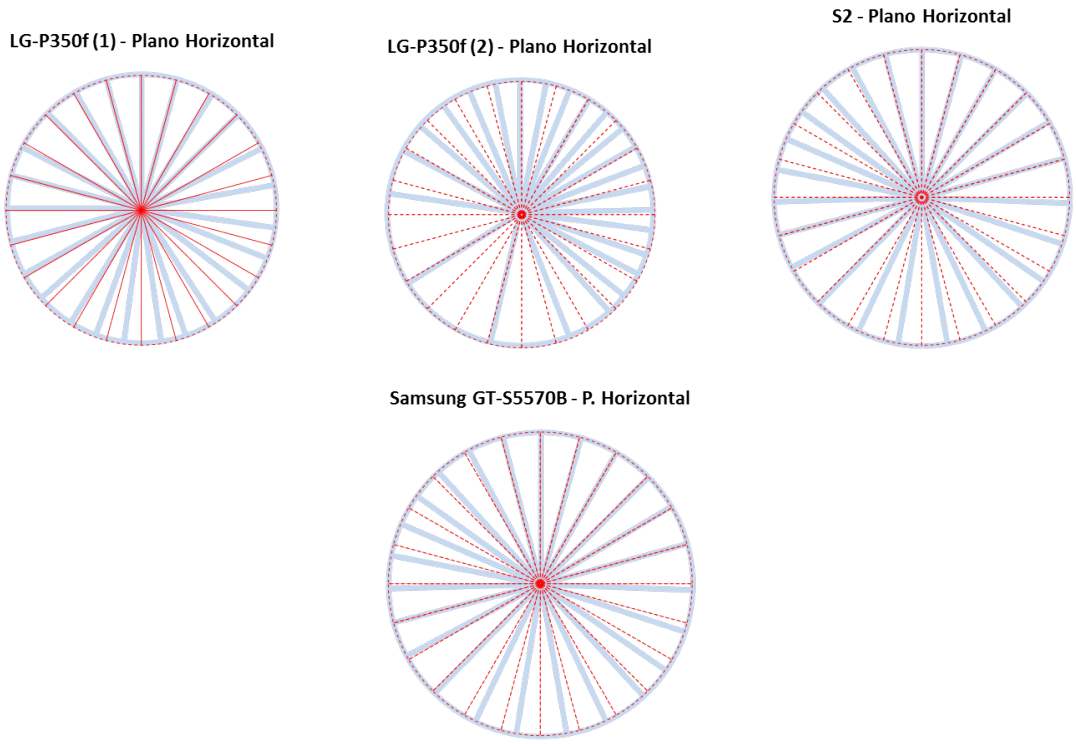

Figura 4.12

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 4.5

quentes. 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 4.13 apresentam os resultados relativos ao modelo *Samsung Galaxy SII*:

A Tabela 4.6 apresenta as estatísticas referentes aos gráficos obtidos das medições em plano inclinado.

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 4.6

Conforme gráficos da Figura 4.13, 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.

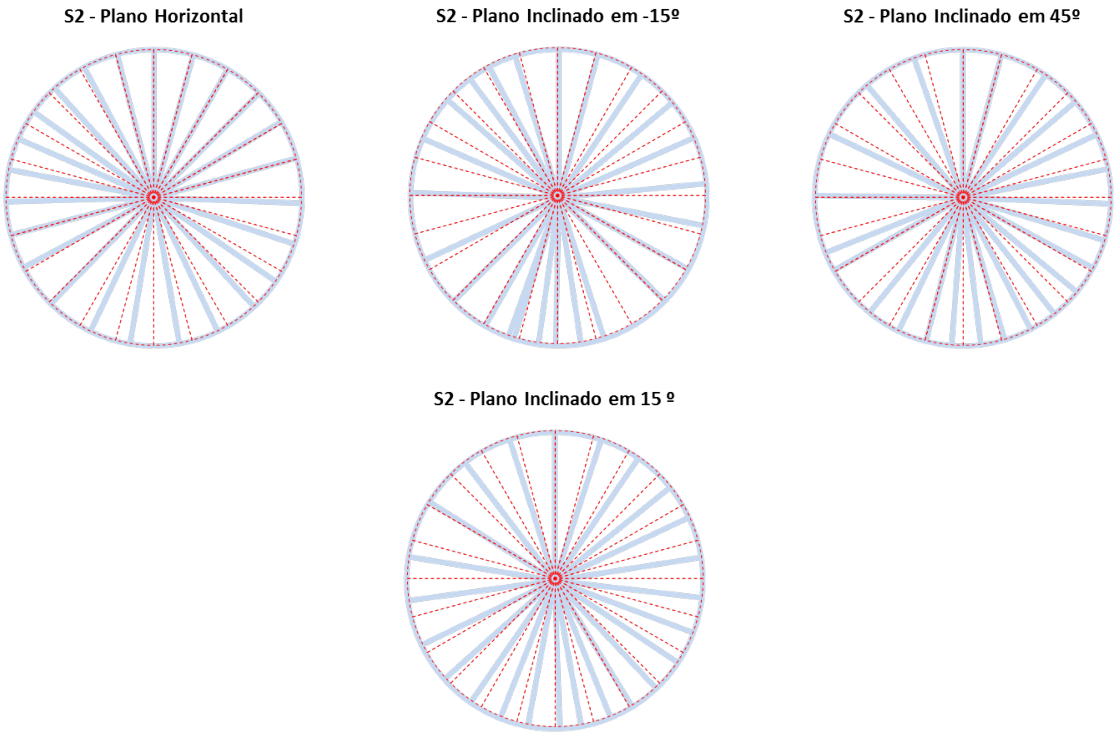

Figura 4.13

4.2.2 Análise da direção apontada pelo dispositivo

Na seção anterior tomamos ciência como se deu a variação angular de um ponto para o outro, mas não foi levada em consideração a direção apontada pelo dispositivo, que deve ser a mesma para cada ponto de parada. Essa direção equivale à variável x do trecho de código que segue e possui valores variando de -180 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 4.14, 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 4.7 e no gráfico da Figura 4.15, 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.

A partir das figuras 4.14 e 4.15 e da Tabela 4.7, notamos que 2 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. Por isso, essa situação também terá que ser tratada na implementação da biblioteca.

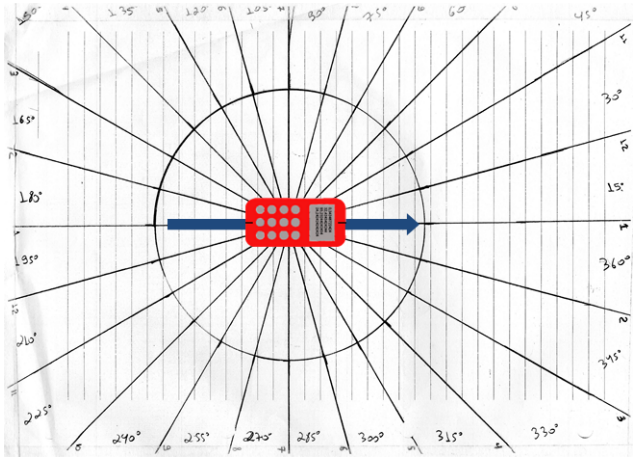

Figura 4.14: *Posicionamento do celular sobre o eixo horizontal da plataforma utilizada nos experimentos com bússola digital. Orientação do celular: Direção: horizontal, Sentido: Para direita*

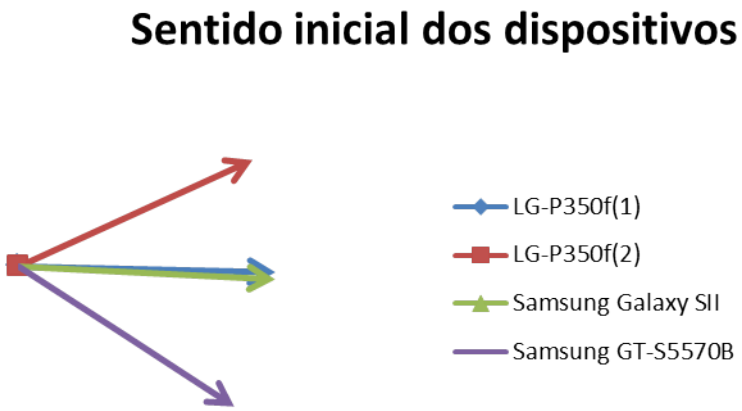

Figura 4.15: *Sentido inicial dos dispositivos posicionados tais como o esquema da Figura 4.14.*

4.3 Comportamento da rede via *Wireshark*

Wireshark é um aplicativo analisador de protocolos de rede [7]. É possível controlar o tráfego do que entra e sai da rede em que o computador está conectado. Esse programa intercepta todos os pacotes que trafegam na rede em que está conectado e os organiza por protocolos. 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.

4.3.1 Detalhes do experimento

Para realização do teste foi criada uma aplicação que envia uma mensagem em broadcast e uma em unicast para um determinado IP usando a biblioteca. Foram usados quatro celulares, já identificados no início deste capítulo, sendo que um deles é o ponto de acesso.

Para facilitar o entendimento desse teste, algumas considerações serão feitas. Considere como "celular 1"um celular LG-P350f , "celular 2"o outro LG-P350f, "celular 3"o Samsung

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 4.7

GT-S5570B e o "celular 4"o Samsung Galaxy SII. O celular 3 é o ponto de acesso.

A relação celular x IP está representada na Tabela 4.8.

Celular	IP
Celular 1	192.168.43.190
Celular 2	192.168.43.177
Celular 3	192.168.43.1
Celular 4	192.168.43.158

Tabela 4.8: *Relação Celular x IP*

4.3.2 Resultados

O resultado obtido dos testes pode ser visto nas Figuras 4.16 e 4.17. Figuras estas tiradas do trace do *Wireshark*.

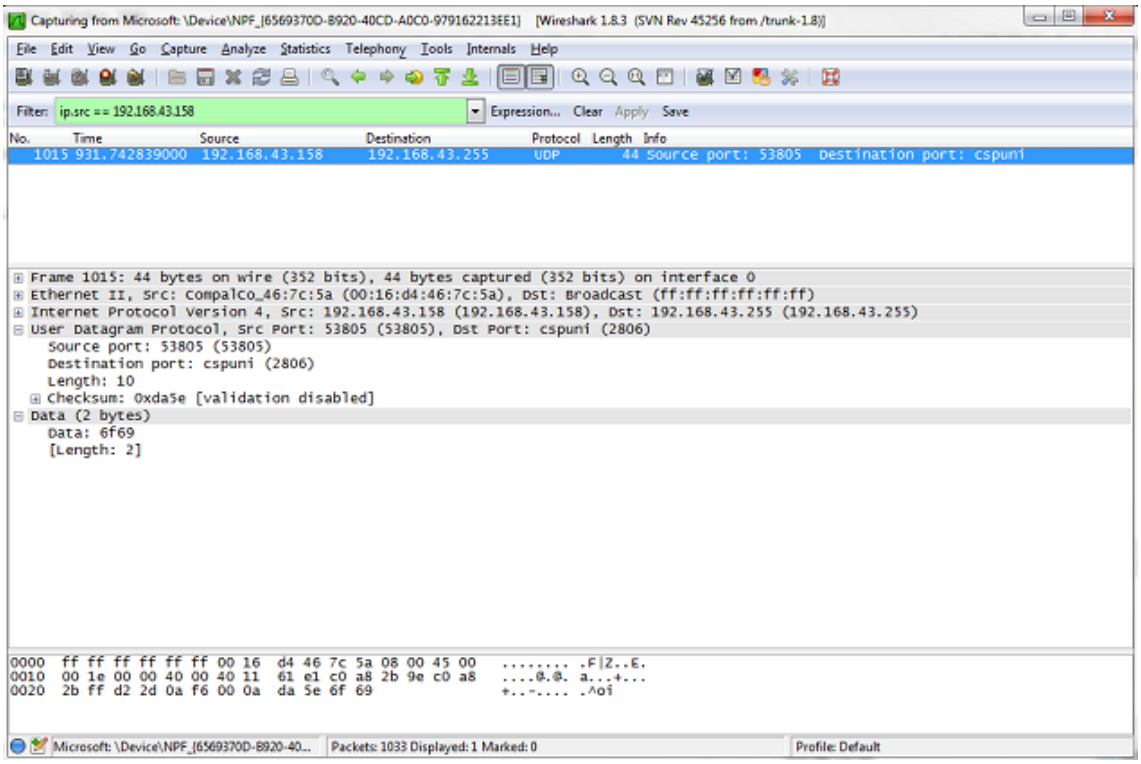

Figura 4.16: *Envio de mensagem em modo broadcast usando o protocolo UDP*

A Figura 4.16 representa o teste em que a mensagem foi enviada em modo broadcast. A mensagem foi enviada pelo celular 4. Como é possível ver na figura, o IP de destino é "192.168.43.255", que corresponde ao endereço de broadcast dessa rede.

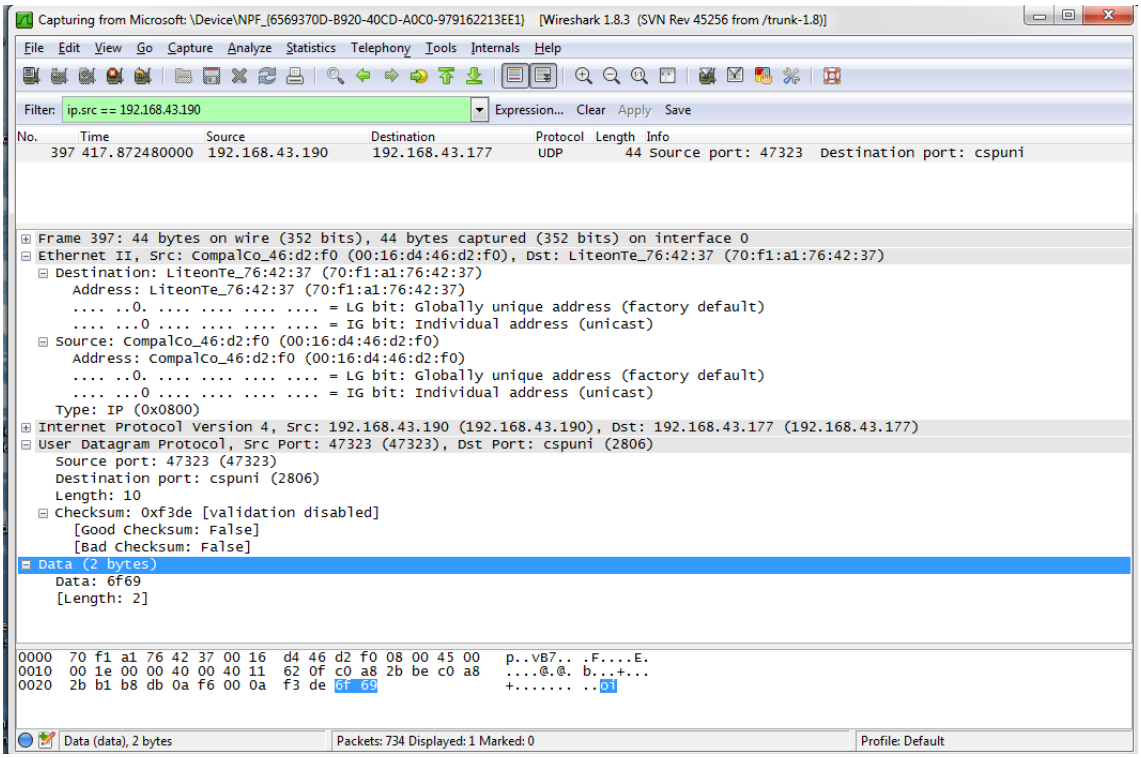

Figura 4.17: *Envio de mensagem em modo unicast usando o protocolo UDP*

A figura 4.17 representa o teste em que a mensagem foi enviada em modo unicast. O celular 1 enviou a mensagem para o celular 2, ou seja, o IP fonte é "192.168.43.190"e o IP de destino é "192.168.43.177". Como é possível ver na figura, a mensagem não passa pelo ponto de acesso para ir ao seu destino.

4.3.3 Conclusões

Através do primeiro experimento foi possível verificar na prática que quando não se sabe um endereço IP do destinatário, basta usar o endereço broadcast (ver Seção 2.5.3) que a mensagem é enviada a todos os nós da rede.

Certificou-se através do segundo 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 o ponto de acesso.

4.4 Teste de confiabilidade

Como é muito usado pela biblioteca o envio de mensagens em modo broadcast, usando o protocolo UDP, é necessário saber qual é a confiabilidade de entrega das mensagens nessa rede, ou seja, qual a porcentagem de chance de uma mensagem, que foi enviada, ser entregue corretamente. Apesar de ser um modo de envio não confiável, pode ser possível que, devido ao ambiente, ao pequeno número de dispositivos e a uma pequena área de cobertura, esse modo de envio tenha alta taxa de confiabilidade. Devido a essa incerteza foi feito esse experimento.

4.4.1 Experimento

Para realização desse teste foi criada uma aplicação que envia 500 (quinhetas) mensagens em broadcast uma seguida da outra. Nos celulares receptores foi armazenado no seu sistema de arquivos todas as mensagens recebidas e estas foram enumeradas para saber se alguma e qual mensagem foi perdida. Foi feito o uso dos quatro celulares mencionados acima, sendo um deles o ponto de acesso.

Considere como "celular 1"um celular *LG-P350f* , "celular 2"o outro *LG-P350f*, "celular 3"o *Samsung GT-S5570B* e o "celular 4"o *Samsung Galaxy SII*. O celular 3 é o ponto de acesso.

Durante o teste, o celular 1 é a fonte de envio, o celular 2 foi posicionado a 10 metros da fonte, o celular 3 a 20 metros e o celular 4 a 50 metros.

O experimento foi realizado em ambiente aberto, mais especificamente nas quadras do CEPE-USP, Centro de Práticas esportivas da Universidade de São Paulo[1],

4.4.2 Resultados

Os resultados podem ser exemplificados através da Tabela 4.9.

Celular	1	2	3	4
Distância da fonte	-	10m	20m	50m
Recebidas	500	407	500	364
% de acerto	100%	81,4%	100%	72,8%

Tabela 4.9: *Resultado do experimento*

4.4.3 Conclusões

Através da Tabela 4.9 pode-se ver que tanto a fonte de envio das mensagens quanto o ponto de acesso recebe todas as mensagens. Porém os outros celulares têm perda de pacotes. A média das mensagens recebidas pelos celulares é de 88,55%, neste caso.

Apesar do recebimento das mensagens não ter 100% de confiabilidade, a quantidade de mensagens perdidas é baixa. Um jeito simples da aplicação aumentar a confiabilidade dessa rede é adotar a técnica de reenvio de mensagens, enviando sempre duas vezes uma mesma mensagem. Para que uma mensagem seja perdida usando essa técnica ela tem que ser perdida na primeira tentativa de envio (ocorre em 11,45% das vezes) e ser perdida na segunda (ocorre em 11,45% das vezes) com isso a taxa de mensagens perdidas, neste caso, cairia de 11,45% para 11,45% * 11,45% = 1,31%.

[1] http://maps.google.com.br/maps?q=-23.560995,-46.715829&num=1&t=h&z=20

Capítulo 5

Biblioteca

Nos capítulos anteriores foram apresentados conceitos fundamentais para a criação da biblioteca, que, afinal, é o objetivo desse trabalho. Esses capítulos foram resumidos de forma a tornar mais clara a sua importância.

É importante salientar que essa biblioteca foi escrita especificamente para android e, portanto, usando a linguagem Java (ver Seção 3.1). A ideia dos autores era criar uma biblioteca simples de usar, ou seja, mesmo um programador com pouca experiência em redes, envio de mensagem e sensores, consegue usar a biblioteca sem grandes dificuldades.

Este capítulo esclarece os detalhes da implementação da biblioteca. A Seção 5.1 apresenta os detalhes da estrutura da biblioteca, assim como os diagramas de classe e uso. A Seção 5.2 apresenta o protocolo da camada de aplicação criado por essa biblioteca e como usá-lo. A Seção 5.3 entra em detalhes da implementação e os pontos mais importantes do código. Na Seção 5.4 são discutidos as restrições de uso da biblioteca, ou seja, quem pode usá-la e como.

5.1 Estruturação da biblioteca

A estrutura da biblioteca pode ser exemplificada com os diagramas de classe e de caso de uso.

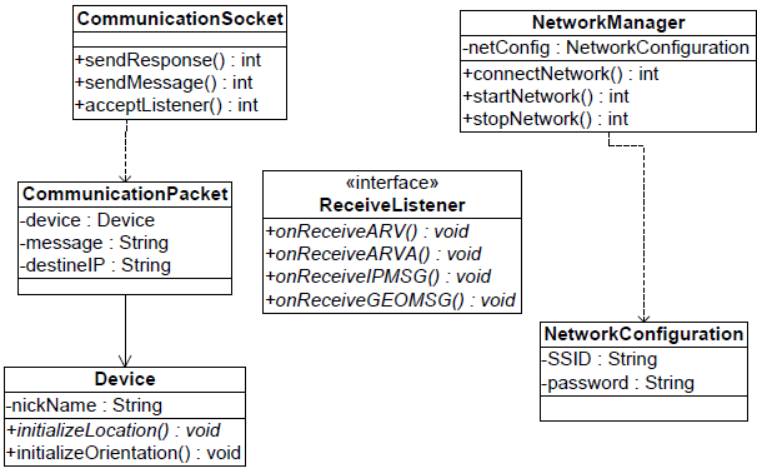

Figura 5.1: *Diagrama de classes da biblioteca*

O diagrama de classes, Figura 5.1, mostra as classes que o usuário (programador) pode usar na sua aplicação. As classes desse diagrama são descritas na Seção 5.3.

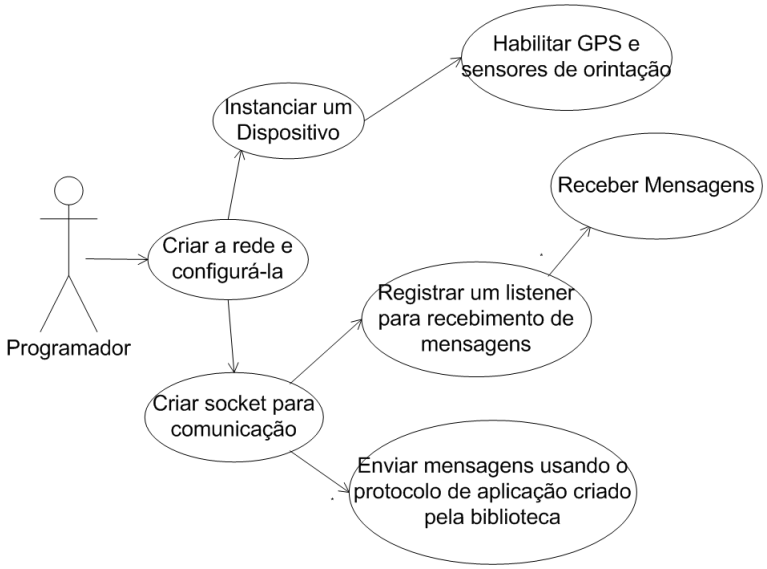

Figura 5.2: *Diagrama de caso de uso da biblioteca*

O diagrama de caso de uso, Figura 5.2, 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 e para receber mensagens é preciso primeiro registrar um "ouvidor"(*listener*).

5.2 Protocolo da camada de aplicação

Na Seção 2.2.1 foi apresentado o protocolo HTTP e foi visto que as mensagens trocadas são padronizadas, pois somente assim um servidor entende o que um cliente quer. Logo, é possível imaginar que o protocolo criado para usar nessa biblioteca tenha as mesmas características, ou seja, tenha mensagens com padrões bem definidos de forma que uma aplicação entenda a outra.

Mas como definir quais mensagens devem ser criadas? Que padrão deve-se seguir? Isso depende do tipo de aplicação que vai usar a biblioteca. Como o objetivo dessa biblioteca é prover informações sobre coordenadas geográficas e orientação de um dispositivo então se faz necessário mensagens que forneçam essas informações.

5.2.1 GCP - *GeoCommunication Protocol*

Baseado na ideia de criar mensagens pelas necessidades das aplicações foram criados quatro tipos de mensagens: ARV, abreviação de *Arrive* - Chegar, que informa a todos que o dispositivo remetente acabou de entrar na rede; ARVA, abreviação de ARV *Answer* - Resposta ao ARV, usado para responder a mensagem do tipo ARV informando que o dispositivo remetente recebeu a mensagem ARV do novo dispositivo; IPMSG, abreviação de IP *Message* - Mensagem IP, que pode ser usado pela aplicação para enviar mensagens onde o

dispositivo destino é identificado pelo endereço IP ao invés da localização geográfica dele; GEOMSG, abreviação de *Geographic Message* - Mensagem Geográfica, que tem como informações adicionais a latitude, a longitude e o azimuth do dispositivo de origem e pode ser usada para identificação de um dispositivo por coordenadas geográficas. A Tabela 5.1 resume as mensagens.

Mensagem	Modo de uso	Objetivo
ARV	ARV [dados da aplicação]	Informar a presença de um novo dispositivo na rede
ARVA	ARVA [dados da aplicação]	Responder uma mensagem do tipo ARVA
IPMSG	IPMSG [endereço IP de destino] [dados da aplicação]	Envio de uma mensagem baseado no endereço IP
GEOMSG	GEOMSG [latitude origem] [longitude origem] [*azimuth* origem] [dados da aplicação]	Envio de mensagem baseado em coordenada geográfica e orientação

Tabela 5.1: *Protocolo de aplicação GCP.*

Portanto o pacote encaminhado para a camada de transporte conterá as informações conforme a Tabela 5.2.

Tipo de mensagem	Informações do protocolo	Dados da aplicação

Tabela 5.2: *Cabeçalho do pacote da aplicação*

É importante ressaltar que as mensagens ARV e GEOMSG são enviadas em modo broadcast (Seção 2.5.3) usando o protocolo UDP (Seção 2.3.1) e, portanto, podem não ser entregues ao destino. É de responsabilidade da aplicação tratar esses casos. As mensagens ARVA e IPMSG não têm esse problema, pois, como o IP de destino é conhecido, são enviadas em modo unicast usando o protocolo TCP que tem entrega garantida(ver Seção 2.3.2).

5.3 Implementação

Diversas classes e pacotes foram criados para viabilizar a biblioteca. A importância das principais classes está escrita nas seções a seguir.

5.3.1 *CommunicationSocket*

É de responsabilidade dessa classe implementar a comunicação entre duas aplicações. Ela tanto envia mensagem quanto recebe. Para isso dois métodos são implementados, o método que envia a mensagem, `send(CommunicationPacket)` (ver Seção 5.3.2), e o método que aceita um "ouvidor"para habilitar o recebimento de mensagens, `acceptListener(IReceiveListener)` (ver Seção 5.3.7).

5.3.2 *CommunicationPacket*

Esta classe representa um pacote que é enviado para outra aplicação. Junto com esse pacote é acoplado os dados do protocolo de aplicação e com isso esse pacote é transmitido

na rede. Ao chegar no dispositivo de destino é extraído as informações do protocolo de aplicação e então é encaminhado o pacote original para a aplicação.

5.3.3 *Device*

Esta classe representa um dispositivo na visão da biblioteca. Assim cada dispositivo pode ser identificado na rede através das informações contidas nessa classe. É importante que o usuário habilite a coleta das informações do GPS e dos sensores usando, respectivamente, os métodos `initializeLocation()` e `initializeOrientation()`.

5.3.4 *NetworkManager*

Essa classe é responsável por criar uma rede local entre os dispositivos e o dispositivo que criar a rede será o ponto de acesso. Também é de responsabilidade dessa classe permitir a conexão de um dispositivo numa rede criada usando a própria biblioteca. Dois métodos são os principais, o `startNetwork(NetworkConfiguration)` (ver Seção 5.3.5) e o `connectNetwork(NetworkAvailable)` (ver Seção 5.3.6).

5.3.5 *NetworkConfiguration*

Esta classe é usada apenas para que a aplicação definir as configurações de uma rede. Seus principais atributos são: `ssid`, usado para definir um nome para rede criada; `isOpen`, usado para definir se a rede será aberta ou terá senha para acesso; e `password`, é a senha da rede, caso essa não seja aberta.

5.3.6 *NetworkAvailable*

Esta classe define um objeto que representa uma rede criada pela biblioteca. Ela contem apenas três atributos, o ssid, a senha e um atributo para verificar se a rede é aberta ou não.

5.3.7 *Interface IReceiveListener*

Essa interface é necessária ser implementada pela aplicação caso esta deseje receber mensagens de outras aplicações. Sempre que uma mensagem do protocolo de aplicação chegar a um dispositivo a mensagem será encaminhada para ao método correspondente. A relação [tipo de mensagem] - [método encaminhado] está representada abaixo. (ver Seção 5.2 para mais detalhes do protocolo de aplicação).

- ARV - `onReceiveARV(CommunicationPacket)`
- ARVA - `onReceiveARVA(CommunicationPacket)`
- IPMSG - `onReceiveIPMSG(CommunicationPacket)`
- GEOMSG - `onReceiveGEOMSG(CommunicationPacket)`

É apenas necessário implementar esses quatro métodos.

5.3.8 GeodesicManager

A classe `GeodesicManager` apoia-se em cálculos geodésicos para fazer a verificação se um vetor de orientação consegue atingir um ponto de destino com um certo raio, ambos projetados no planeta Terra. São facultativas a definição do alcance desse vetor e a possibilidade de substituição do mesmo por um feixe de vetores, definindo uma área de acerto.

5.3.9 Biblioteca Geotools2

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].

5.3.10 Filtro de erros DRNF

Dentre as inúmeras possibilidades de tratamento de erros, optou-se por implementar o DNRF (*Drift Noise Removal Filter*) na biblioteca.

O DNRF utiliza o filtro Complementar para fundir os sensores acelerômetro, magnetômetro e giroscópio, de forma a minimizar os drifts e ruídos no vetor de orientação a ser gerado.

Conforme modelo de funcionamento do referido filtro esquematizado na Figura 5.3, após a formação das matrizes de rotação da bússola digital e do giroscópio, essas são sujeitas a correções de erros numéricos que podem violar sua propriedade de ortogonalidade. O DNRF corrige dois tipos de erros:

- **Erro de integração**: gerado devido de a uma taxa de rotação constante atribuída aos intervalos de tempo de amostragem.
- **Erro de quantização**: ocorre quando nem todos os bits são obtidos da conversão de analógico para digital devido à representação finita do sinal digital.

Após a correção da matriz de rotação, outro procedimento efetuado pelo modelo escolhido é a remoção de Gimbal Lock, que corresponde ao problema gerado quando o ângulo de rotação pitch assume o valor de 90^{o}, fazendo a rotação do yaw e roll ocorrer efetivamente sobre o mesmo eixo vertical.

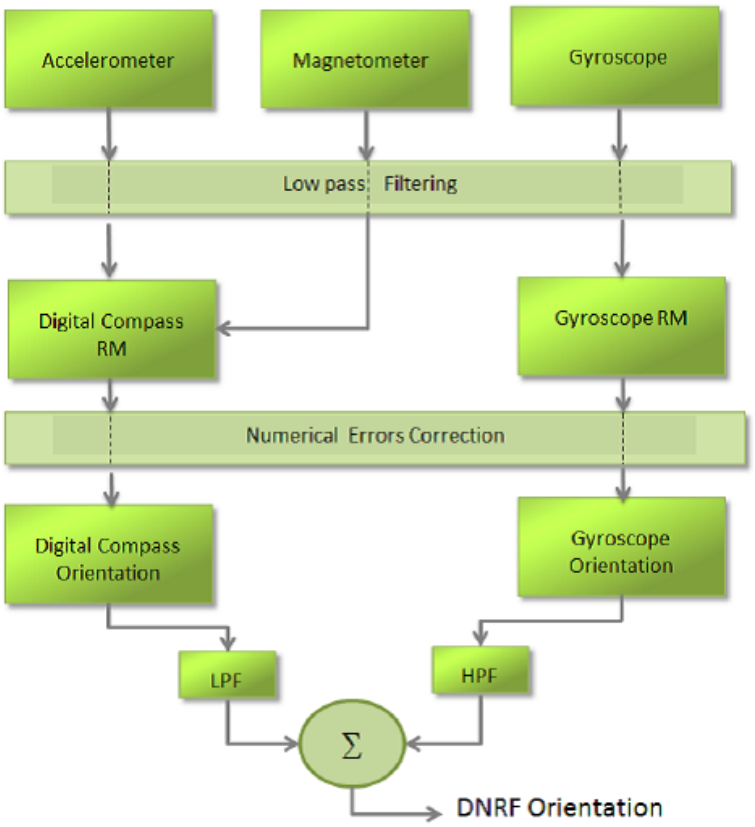

Fig. 3. DNRF Algorithm Flow Diagram

Figura 5.3: *Diagrama de fluxo do Algoritmo DNFR. Figura extraída de [9]*

5.4 Restrições de uso

Para desenvolver aplicações usando essa biblioteca é necessário que o programador saiba desenvolver em Java e para a plataforma android (ver Seção 3.1.2), porém é apenas necessário um conhecimento mínimo, pois a biblioteca abstrai toda implementação especifica do android, deixando para o usuário apenas a função de criar a aplicação.

A biblioteca foi programada para funcionar na API 2.2 ou mais, o que abrange cerca de 90% dos dispositivos android em 2012 [4]. Seu funcionamento não é garantido em versões de API inferiores.

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, no *classpath* da sua aplicação. É necessário, no entanto, adicionar algumas permissões de usuário no arquivo AndroidManifest.xml, são elas:

- CHANGE_WIFI_STATE
- CHANGE_NETWORK_STATE
- CHANGE_CONFIGURATION
- ACCESS_WIFI_STATE
- ACCESS_NETWORK_STATE
- INTERNET

- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION

Apesar da necessidade da permissão para uso da internet, a internet não é usada. Esta permissão é necessária para que um dispositivo mande uma mensagem na rede e o android não diferencia se essa mensagem está sendo mandada numa rede local ou na internet (uma rede mundial).

Capítulo 6

Resultados e caso de uso

Para avaliar a eficácia da biblioteca é imprescindível mostrá-la em funcionamento. Por isso foi criado uma aplicação que faz uso da biblioteca.

A aplicação escolhida foi a representação do 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 serão mostrados detalhes do jogo, da implementação e de como a biblioteca foi importante para sua criação. 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 mostra os resultados obtidos do uso da aplicação.

6.1 Jogo batata-quente

Um dos jogadores, o criador, deve ser responsável por definir as configurações iniciais do jogo e disponibilizar a rede aos demais participantes. É ele também que deverá fazer a primeira jogada, passando a batata virtual ao participante que estiver na direção apontada por seu dispositivo.

A partir de então, todos os jogadores poderão fazer o mesmo procedimento: apontar e lançar a batata. Uma vez que estiver com a mesma na tela do dispositivo, o jogador terá no máximo 3 segundos para livrar-se dela. Caso esse limite seja ultrapassado, ela queimará e o jogador será desclassificado.

A duração total do jogo é definida aleatoriamente, não ultrapassando o intervalo máximo definido pelo criador. Em seguida, essa duração é divida em 3 intervalos aleatórios colocados em ordem decrescente de duração. Cada um deles equivalerá a uma etapa do jogo. Quão mais próximo do esgotamento do tempo uma partida estiver, mais quente a batata estará. Quando o tempo total acabar, a batata queimará e desclassificará o jogador que a "segurava". A Figura 6.1 apresenta uma simulação de partida.

6.2 Requisitos

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

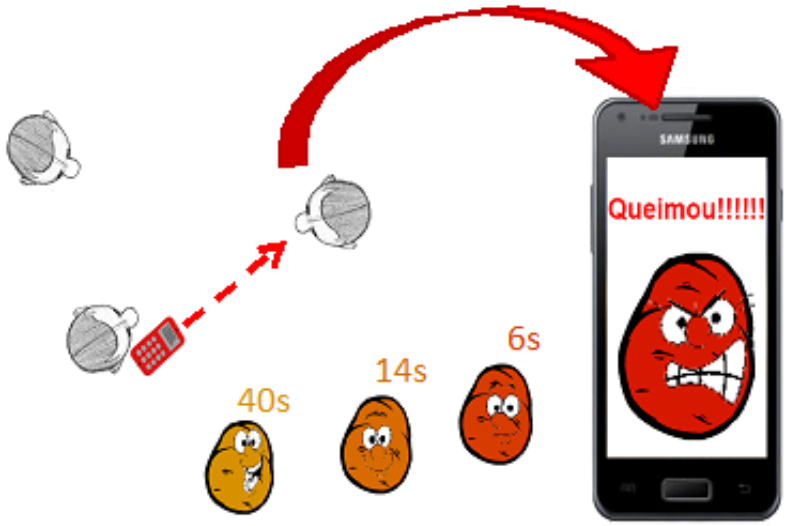

Figura 6.1: *Simulação de partida com duração de 1 minuto.*

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 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 5.4, algumas permissões de usuário são necessárias. Além daquelas requeridas pela biblioteca, é preciso conceder também a permissão:

- WRITE_INTERNAL_STORAGE

Essa permissão é necessária para persistir dados do usuário, como, por exemplo, o seu nome e histórico.

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 o magnetômetro (bússola digital);
- esteja localizado a pelo menos 20m de distância de outros jogadores.

6.3 Resultados

Conforme estudo apresentado neste trabalho, existem 2 partes críticas para criação de uma aplicação a partir da biblioteca *GeoCommunication*: a comunicação entre os dispositivos na rede e a identificação de um dispositivo via coordenada geográfica.

A aplicação possui implementado um mecanismo de segurança para garantir a confiabilidade de troca de mensagens na rede. Portanto, quando há o envio da mensagem que carrega o status da batata pela rede, a imagem da batata só sairá do visor do dispositivo do jogador que a está lançando, quando houver uma confirmação (via TCP) de recepção do pacote por parte da aplicação do jogador que foi apontado.

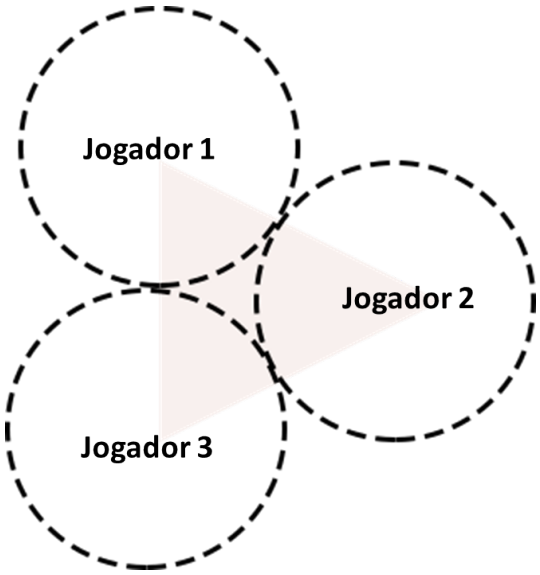

Figura 6.2: *Cenário do jogo*

Com relação à identificação do dispositivo na rede, o jogo só pôde ser viabilizado através da construção de um cenário ideal: apenas 3 jogadores podem jogar e eles devem estar a uma distância de 20 metros uns dos outros de forma a compor um triângulo equilátero em que eles são os vértices, tal como exemplificado na Figura 6.2. Se fôssemos levar em consideração apenas as falhas típicas dos sensores, não necessitaríamos criar um esquema de jogo tão específico, pois o filtro DNRF implementado na biblioteca possui uma boa acurácia na formação do vetor de orientação. Porém, as margens de erro gritantes do GPS apontados nas seções anteriores, que chegou a variar quase 10 metros no experimento relativo à Seção 4.1.4, determinou a criação dessa estrutura em que os jogadores possuem um raio de 10 metros ao seu redor, aumentando bastante a sua área de acerto.

Capítulo 7

Conclusões

Ao longo desta monografia foi apresentado um conjunto de conceitos relacionados com a criação de uma biblioteca para Android com o objetivo de realizar comunicação entre dispositivos móveis baseada em coordenadas geográficas. Se houver erros no uso do GPS e sensores, a identificação de um dispositivo via protocolo de aplicação da biblioteca torna-se prejudicada, pois essas tecnologias são usadas exclusivamente para essa finalidade.

Através dos experimentos realizados com o sistema de posicionamento escolhido, o GPS, é claro a notável discrepância de comportamento e qualidade entre os dispositivos de diferentes marcas e modelos. Somado a isso, ainda há a margem de erro para a definição do posicionamento global intrínseca ao próprio sistema, conforme tabela 3.3 e que foi comprovado por experimentos, conforme Figura 4.3. Esses são um dos principais motivos dos resultados negativos quanto ao uso da biblioteca em pequenas áreas.

Os sensores do Android também mostraram-se imprecisos, sendo a bússola digital que apresentou resultados mais críticos, por sofrer grande interferência de aparelhos que emitem ondas eletromagnéticas. Mas, com o uso do giroscópio, pode-se obter apenas a primeira amostra da bússola para criar o vetor de orientação do dispositivo no quadro de referência da terra. Depois disso, as rotações do vetor de orientação podem ser obtidas através dos sensores acelerômetro e giroscópio integrados pelo filtro complementar, solução essa adotada pela biblioteca. Portanto, os sensores não são um problema para o uso da biblioteca.

Com o desenvolvimento do caso de uso, o jogo batata-quente, ficou claro a facilidade de desenvolver uma aplicação que utiliza a biblioteca GeoCommunication. Apesar dos resultados negativos em pequenas áreas, a aplicação se mostrou eficaz em grandes áreas, como por exemplo, um campo de futebol.

Resumindo, foram encontrados diversos problemas referentes a hardware ao longo do trabalho e diversas referências na literatura para minimizar esses problemas. À medida que os hardwares forem evoluindo e as soluções aperfeiçoando-se, a eficiência da biblioteca tenderá a aumentar, pois ela foi desenvolvida com módulos independentes do vínculo com o hardware.

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. 19

[2] Documentação oficial da biblioteca geotools. http://docs.geotools.org/. Último acesso em 12/2012. 45

[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, 13

[4] Gps vs agps: A quick tutorial. http://www.wpcentral.com/gps-vs-agps-quick-tutorial. Último acesso em 12/2012. 20

[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. 10, 13, 14, 15, 17

[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] Shahid Ayub, Alireza Bahraminisaab, and Bahram Honary. A sensor fusion method for samart phone orientation estimation. Technical report, Lancaster University. 46

[10] Azzedine Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks*. Wiley-IEEE Press, 2008. 18

[11] Earl F. Burkholder. The 3-d global spatial data model - foundation of the spatial data infraestructure. http://goo.gl/pAsCJ. Último acesso em 12/2012. 22

[12] Shane Colton. The balance filter. http://goo.gl/V0ziF. Último acesso em 12/2012. 23

[13] R. E. Deakin and M. N. Hunter. Geodesics on an ellipsoid - bessel's method. Technical report, RMIT University - School of Mathematical Geospatial Sciences, 2007. 21

[14] Vassilis Kostakos Denzil Ferreira, Anind K. Dey. Understanding human-smartphone concerns: A study of battery life. Technical report, Universidade de Madeira - Portugal, 2011. 14

[15] Rabie Khodr Jradi e Lasse Seligmann Reedtz. Ad-hoc network on android. Technical report, Technical University of Denmark, 2010. 10, 14, 18

[16] 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. 23

[17] 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. 16, 22, 24

[18] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet.* Pearson Addison Wesley, terceira edition, 2006. 8

[19] Viktor Moell och André Horntvedt. Positioning for mobile phones using wlan and accelerometer data. Technical report, Lund University, 2012. 16, 17, 18

[20] Gabor Paller. Motion recognition at droidcon 2011. http://www.sfonge.com/forum/topic/measuring-movement-accelerometer-and-gyroscope. Último acesso em 12/2012. 16, 22

[21] B. Merminod Q. Ladetto. Digital magnetic compass and gyroscope integration for pesdestrian navigation. Technical report, 2002. 24

[22] 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, 13

[23] Angel Rodriguez Ubejd Shala. Indoor positioning using sensor-fusion in android devices. Technical report, Kristianstad University, 2011. 15, 16

[24] Steven J. Vaughan-Nichols. Android/linux kernel fight continues. http://blogs.computerworld.com/16900/android_linux_kernel_fight_continues. Último acesso em 12/2012. 13

[25] 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. 22