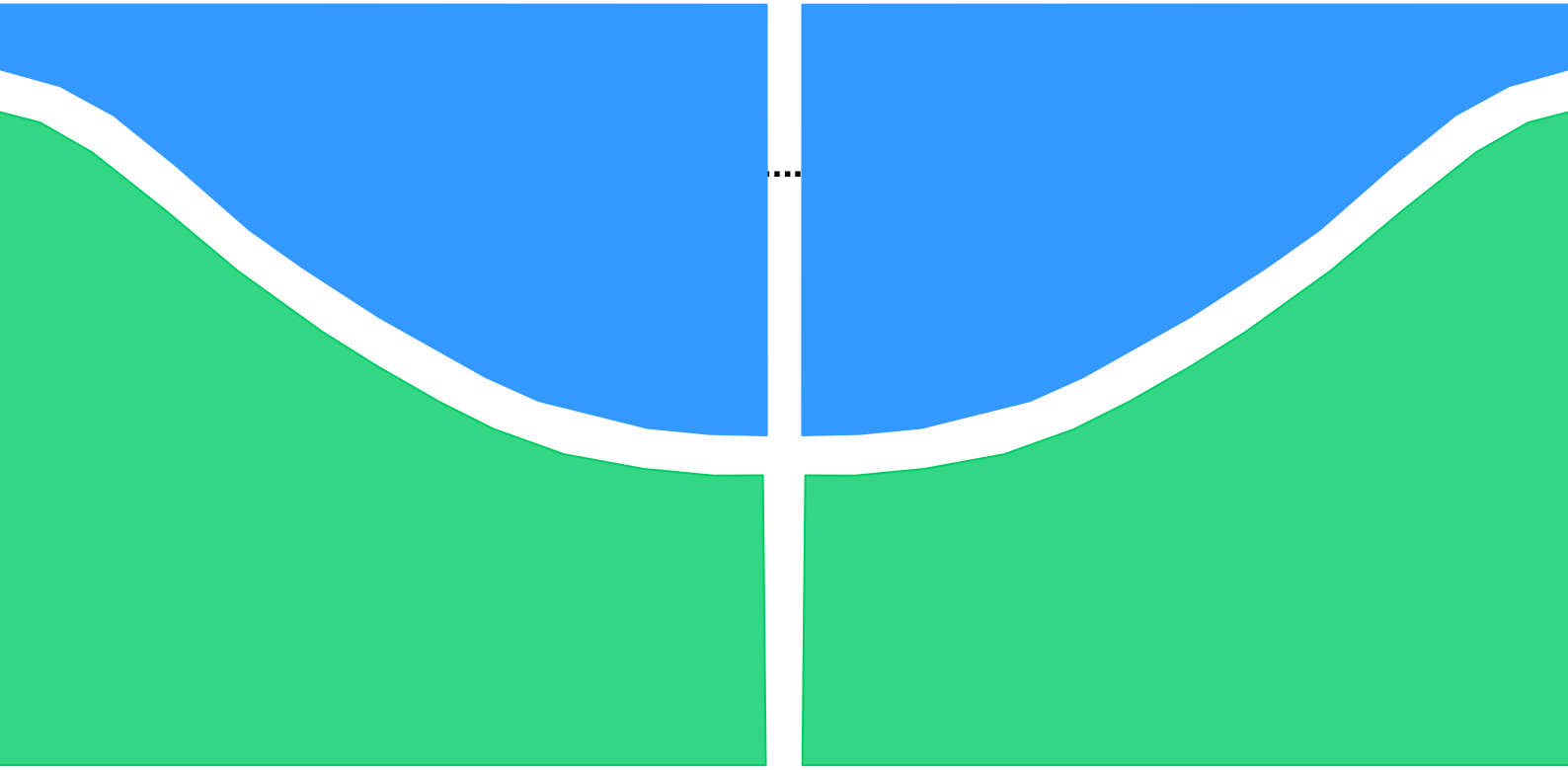


# TRABALHO DE GRADUAÇÃO

# ESTUDO E DESENVOLVIMENTO DE APLICATIVO PARA DETECÇÃO DE PROXIMIDADE PARA DISPOSITIVOS BLE

**Eduardo de Oliveira dos Santos**
**Pedro Souza Alves**

**Brasília, 03 de julho de 2018**

**UNIVERSIDADE DE BRASILIA**

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

# ESTUDO E DESENVOLVIMENTO DE APLICATIVO DE PROXIMIDADE PARA DISPOSITIVOS BLE

**Eduardo de Oliveira dos Santos**
**Pedro Souza Alves**

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

**Banca Examinadora**

Prof. Cláudia Jacy Barenco Abbas,PhD,UnB/ENE
(Orientador) _____

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

Prof. Rafael Timóteo de Sousa Jr, PhD, UnB/ ENE _____

**Dedicatória(s)**

*Primeiramente, dedico esse trabalho a Deus, que me iluminou e continua iluminando por todos momentos da minha vida. À minha família, essa conquista também é de vocês!*

*Pedro Souza Alves*

*Dedico esse trabalho primeiramente a Deus, por ser essencial a minha vida. Ao meu pai e minha família porque sem eles não seria capaz de fazer esse trabalho.*

*Eduardo de Oliveira dos Santos*

## Agradecimentos

*Agradeço a professora Cláudia, por ter confiado e nos orientado nesse trabalho, além de ter comprado os equipamentos necessários.*

*Agradeço a minha família, e em especial ao meu pai José Denizio dos Santos, por me proverem tudo que um ser humano precisa para viver.*

*Agradeço aos meus amigos que fiz no curso de Engenharia de Redes, sem eles não seria capaz de terminar o curso, e em especial ao co-autor desse trabalho, Pedro Souza, que além de parceiro de trabalho é um parceiro de vida*

*Agradeço, também, aos meus amigos que fiz no ensino médio que me acompanharam durante toda essa jornada de universidade*

*Não podia deixar de agradecer ao governo brasileiro por ter me dado a oportunidade de fazer um intercâmbio de forma gratuita.*

*Eduardo de Oliveira dos Santos*

*Após toda essa jornada na UnB, tenho total noção que se fosse apenas por mim as coisas não aconteceriam do jeito incrível que aconteceram.*

*Agradeço a todos os professores que tiveram a missão de transmitir não apenas conhecimento, mas também amor pelo que fazem. A cobrança de vocês fez com que eu me dedicasse 110% e hoje tenho orgulho de todo esforço que fiz. Em especial, gostaria de agradecer à professora Cláudia Jacy Barenco Abbas por toda a atenção e dedicação durante esse ano de trabalho. Também não posso deixar de lembrar do professor André Noll Barreto, que me deu a oportunidade de provar meu esforço num momento que poucos professores o fariam.*

*À minha família, que sempre me apoiou e entenderam os churrascos perdidos ou passados à frente do computador fazendo listas de exercícios e projetos. Todo o apoio não foi em vão e essa conquista é nossa!*

*Aos meus amigos conquistados na faculdade. Amigos, não colegas de curso. Tenho plena certeza que nos encontraremos no mercado de trabalho e gozando de todo conhecimento adquirido na UnB.*

*Ao Milf, dispenso textões. Estiveram comigo nos momentos bons e ruins. Que seja eterna enquanto dure essa amizade incrível.*

*Ao meu parceiro e co-autor desse trabalho, Eduardo. Não poderia ter escolhido um parceiro melhor. Sempre dedicado e prestativo e, por vezes, até genial. Obrigado por todas experiências compartilhadas dentro e fora da UnB. Seu sucesso é inevitável.*

*Vamos viver!*

*Pedro Souza Alves*

## RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo de detecção de proximidade para Android utilizando a tecnologia BLE. Analisa-se o comportamento do RSSI em uma rede Bluetooth para ajuste de parâmetros importantes na detecção de proximidade como tempo de *scan* e taxa de *advertising*. Também estuda-se o comportamento do RSSI em diferentes modelos de celulares e versões BLE. Especifica-se uma metodologia de detecção de proximidade baseada em *fingerprinting* e ao final desenvolve-se um aplicativo para celular e um aplicativo para dispositivos *RedBear* para validação da solução.



## ABSTRACT

This work presents the development of a proximity detection application for Android using BLE technology. The behavior of RSSI in a Bluetooth network is analyzed to adjust important parameters in the detection of proximity such as *scan* time and advertisement time. We also study the behavior of RSSI in different cell phone models and BLE versions. A fingerprint-based proximity detection methodology is specified and at the end a mobile application and a RedBear device application are developed for solution validation.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SÍMBOLOS

AD - *Advertisement Data Strucutures*
ADC - *Analogic- Digital Converter*
AoA - *Angle of Arrival*
API - *Application Programming Interface*
BER - *Bit Error Rate*
BLE - *Bluetooth Low Energy*
Bluetooth SIG - *Bluetooth Special Interest Group*
CRC - *Cyclic Redundancy Check*
FHSS - *Frequency Hopping Spread Spectrum*
GAP - *Generic Access Profile*
GATT - *Generic Attribute Profile*.
GFSK - *Gaussian Frequency Shift Keying*
GPS - *Global Positioning System*
LBS - *Location Based Service*
LTE - *Long Term Evolution*
MUSIC - *Multiple Signal Classification*
NFC - *Near Field Communication*
*IDE - Integrated Development Environment*
IEEE - *Institute of Electrical and Electronics Engineers*
IoT - *Internet of Things*
ISM - *Industrial Scientific Medical*
I²C - *Inter-Integrated Circuit*
PDU - *Packet Data Unit*)
PWM - *Pulse Width Modulation*
RSSI- *Received Signal Strength Indicator*
RF - *radiofrequência*
*SoC - System on Chip*
*SPI - Serial Peripheral Interface*
TOA - *Time of Arrival*
TDOA - *Time difference of arrival*
UART - *Universal asynchronous receiver-transmitter*
UUID - *Universally unique identifier*
UWB- *Ultra Wide Band*

# 1. INTRODUÇÃO

O desenvolvimento progressivo e a popularização de tecnologia móveis, incluindo *smartphones* e dispositivos de IoT (*Internet of Things)* mudaram a maneira do ser humano interagir e pensar. Os *smartphones* cada vez mais potentes permitem a criação de aplicações nunca antes imagináveis, os tamanhos cada vez menores dos dispositivos de IoT ampliam a área de atuação dessa tecnologia até a medicina. Seguindo essa tendência em um futuro próximo todos as "coisas", desde de carteiras até seres vivos, terão algum tipo de etiqueta ou algo similar que interaja com a internet. Essa realidade futura possibilita a criação de diversas funcionalidades, entre elas a gestão de posicionamento, isto é, saber exatamente onde se encontra a carteira, os óculos e até mesmo um filho.

Dentre todas as técnicas de localização existentes no mundo, uma técnica que pode-se destacar são as técnicas baseadas na potência do sinal recebido. Para utilização dessas técnicas é necessário a existência de algum rádio receptor que consiga medir a intensidade do sinal recebido e a partir dessa intensidade se calcula a distância. Dispositivos IoT cumprem esse requisito, facilitando a implementação de sistemas que utilizem dessas técnicas, visto que não é necessária alteração no *hardware* do dispositivo. Os padrões de redes sem fio atuais possuem um indicador da medida dessa potência, o RSSI (*Received Signal Strength Indicator)*.

O RSSI é um indicador de intensidade de um sinal recebido por um dispositivo e é medido em *dBm*. Por ser um valor de magnitude de uma onda eletromagnética, esse indicador tem relação direta com a perda de intensidade no caminho (*path loss*). Devido à essa perda de intensidade, é possível estimar a distância entre dois dispositivos. Essa estimativa é afetada pela interferência por vários fatores, incluindo a potência do transmissor, a sensibilidade do receptor, a atenuação do meio e ainda apresenta diferenças entre as leituras de diferentes dispositivos.

O trabalho apresentado analisará o comportamento desse indicador, bem como os parâmetros importantes para o mesmo como: frequência de *advertising*, o quão rápido o dispositivo deve enviar pacotes e a janela de *scan*, por quanto tempo o *smartphone* deverá escutar os pacotes enviados pelo dispositivo. Será analisado também o comportamento quanto a polarização das antenas e será desenvolvido um aplicativo para *smartphones*.

Além do fator de eficiência do aplicativo, outros aspectos devem ser levados em consideração, como o preço e a praticidade da solução final. Conforme constatado por Connell (2015), a utilização de UWB (*Ultra Wide Band*) para localização de dispositivos seria a maneira mais eficiente. Porém esse tipo de rede não é suportado por *smartphones*, além de ser uma solução cara por trabalhar em banda licenciada.

As duas redes mais populares disponíveis em celulares são o *Wi-Fi* e o *Bluetooth*. Faragher e Harle (2015) defendem que o *Bluetooth Low Energy* (BLE) é a melhor alternativa para esse tipo de aplicação, por não possuir muitos canais para serem *scaneados* e por não possuir *buffer*. A utilização de *buffers* em sistemas de localização não é apropriada, visto que é necessário que os pacotes sejam enviados no momento em que eles se encontram na posição e não segundos depois. Além de ser suportado por vários celulares, dispositivos BLE são baratos, pequenos e duram meses utilizando apenas uma pequena bateria.

O projeto será desenvolvido para ambientes outdoor, visando situações em que a utilização de GPS (*Global Position System)* seria muito onerosa e excessiva. Dispositivos equipados com a tecnologia *Bluetooth Low Energy* podem assumir tamanhos cada vez menores e mais baratos, o que facilitou a popularização dessa tecnologia e possibilita a adoção da mesma em qualquer tipo de objeto, desde um carro até uma carteira. Assim, abre-se o leque para criação de diversas aplicações integradas em ambientes *outdoor*.

## 1.1- DESCRIÇÃO DO PROBLEMA

O RSSI é algo intrínseco aos pacotes recebidos por um dispositivo BLE. Isso quer dizer que para todos os pacotes que chegam a um dispositivo, um valor de RSSI é determinado. Porém, esse valor sofre forte influência de fatores externos como obstáculos, tipos de antenas, multi-percurso e até mesmo a versão do BLE utilizada, causando muitas variações nesse valor. A dificuldade na utilização de detecção de localização baseado em RSSI se deve essas variações, que dificultam a modelagem matemática e aumentam a complexidade do proBLEma.

## 1.2- OBJETIVO GERAL

Desenvolver uma aplicação capaz de prover experiência de localização e/ou proximidade utilizando celulares e *hardware open-source* em ambientes outdoor.

## 1.3- OBJETIVOS ESPECIFICOS

- Escolher um hardware;
    - Estudo dos equipamentos no mercado a fim de encontrar dispositivos baseados na tecnologia BLE que custem pouco e que sejam de código aberto.
- Analisar comportamento do RSSI em diferentes fabricantes de celulares e versões BLE;
- Estudo de diferentes algoritmos de localização;
- Definição da metodologia a ser usada no projeto;
- Desenvolvimento de um aplicativo para celular Android;
- Modificação do firmware do dispositivo BLE adquirido;
- Realização de teste em campo;

## 1.4 JUSTIFICATIVA DO ESTUDO

O desenvolvimento de uma solução barata e de pequeno tamanho físico para detecção de proximidade utilizando a tecnologia BLE faz com que esta possa estar instalada em um número grande de objetos ou seres vivos podendo assim fazer parte de uma rede IoT que interage com as pessoas através de um celular.

Sob o ponto de vista acadêmico tem-se uma contribuição com o estudo do comportamento RSSI em diferentes versões BLE (4.0, 4.1 e 4.2), além da definição de uma metodologia baseada em *fingerprinting* e comprovação da efetividade desta metodologia em um ambiente outdoor através do desenvolvimento de um aplicativo.

## 1.5 ORGANIZAÇÃO DO TRABALHO

O trabalho é organizado em seis capítulos:

O capítulo 1 apresenta ao leitor uma visão geral sobre o que será tratado no decorrer do trabalho, em como os objetivos específicos e geral. O capítulo 2 expõe conceitos e definições descritos em livros, artigos, sites e outras fontes essenciais para o entendimento do assunto a ser tratado no trabalho, como: técnicas de localização, o funcionamento do *Bluetooth Low Energy* e a descrição dos equipamentos utilizados. O capítulo 3 fala sobre trabalhos que possuem a mesma motivação: localização de dispositivos. O capítulo 4 expõe as etapas de desenvolvimento do projeto de forma detalhada, são apresentados diversos testes de comportamento do RSSI do *Bluetooth* e são definidos horizontes para o

aplicativo final. O capítulo 5 apresenta a aplicação final desenvolvida, bem como suas limitações e seu funcionamento. O capítulo 6 discorre, de forma breve, sobre o desenvolvimento da proposta. Também apresenta os potenciais de utilização desta proposta em outros contextos, apresenta as dificuldades encontradas durante a execução do projeto e, por fim, apresenta sugestões de trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 TÉCNICAS DE LOCALIZAÇÃO

Esta seção expõe quatro técnicas diferentes de localização: AOA, TOA, TDOA e técnicas baseadas em RSS. São detalhados os funcionamentos, vantagens e desvantagens de cada técnica.

### 2.1.1 AOA

AOA, do inglês (*Angle of Arrival*) é uma técnica de localização empregada em serviços baseados em localização, conhecido como LBS (do inglês *Location Based Service*).

Em geral, algoritmos de estimação de AOA assumem algumas simplificações com relação às fontes dos sinais e arredores do arranjo de sensores. Uma das simplificações é considerar uma situação em que o arranjo está "distante o suficiente" e as frentes de onda são planas. Essa condição pode ser descrita pelas relações:

$$r > \frac{2D^2}{\lambda} \quad (1)$$

$$r \gg \lambda \quad (2)$$

Na Eq. (1) e na Eq. (2), r é a separação entre o arranjo e a fonte, D é a máxima dimensão do arranjo (largura ou altura), e λ é o comprimento de onda. Para ambientes outdoor essa condição de campo distante é plenamente satisfeita.

Porém, para ambientes indoor, a condição de campo distante não é atendida. Por isso, em ambientes indoor, deve-se considerar as frentes de onda como esféricas.

Uma outra simplificação é considerar todos sinais como se fossem complexos, ou seja, possuem amplitude e fase. Para sinais usados em comunicações móveis, isso realmente ocorre. Mas em sinais de ondas acústicas, o sinal, por ser real, só possui amplitude e não provê uma estimativa tão boa do espectro espacial.

Os métodos clássicos de estimação AOA são baseados em um modelo de arranjo. Nesse modelo, considera-se um sinal de entrada s(k) chegando em "m" sensores. Assim, um vetor é formado nas saídas dos sensores. Isso é expresso a seguir:

$$x(k) = a(\theta)\mathrm{s}(\mathrm{k}) + v(k) \quad (3)$$

$$x(k) = [x1(k) \dots xm(k)]^{\mathrm{T}} \quad (4)$$

$$v(k) = [v1(k) \dots vm(k)]^{\mathrm{T}} \quad (5)$$

Na equação (4), x(k) é um sinal recebido no k-ésimo sensor do arranjo e na Eq (5) o v(k) é o k-ésimo componente de ruído branco gaussiano no sensor. O vetor, conhecido como vetor direção, contém toda a informação do AOA.

Cada método de estimação de AOA extrairá e usará essa informação de uma maneira diferente. Métodos que utilizam hipóteses previamente estabelecidas de modo estatístico são denominados métodos paramétricos. Estes são mais precisos que os chamados métodos não paramétricos. (STOICA,MOSES, 1997)

Dos métodos não paramétricos se destacam os métodos Beamforming e o Capon. O método Beamforming funciona como um banco de filtros que maximiza somente o sinal de direção do AOA especificado, minimizando todos os outros por meio de "pesos". O método Capon é baseado no

Beamforming. Se diferencia no fato que ele avalia apenas os sinais da direção (x) e, por isso, seu desempenho é bem superior (COSTA DIAS et All., 2004).

Alguns sistemas de localização que utilizam AOA se baseiam na diferença de fase entre as frentes de onda que interceptam os múltiplos sensores do arranjo. São exemplos*: Beamforming*, *Capo*n, MUSIC (*Multiple Signal Classification*). Outros se baseiam em métodos TDOA, (do inglês *Time Difference of Arrival*) entre as frentes de onda e o AOA correspondente. A partir da interação dos sensores com os dados coletados, calcula-se o ângulo e o tamanho das arestas do triângulo formado em duas dimensões. Com esse ângulo, o arranjo também é capaz de calcular a posição da fonte ou determinar o ângulo da posição da fonte em relação à matriz. Ambas as capacidades são vantajosas para inúmeras aplicações. Para uma maior precisão ou para uma estimativa em três dimensões, e utiliza mais de três sensores de recepção.

Uma desvantagem dessa técnica é sua ineficiência em ambientes *indoor*, já que, em sua forma pura, necessita de uma linha de visão clara entre o dispositivo móvel e os sensores. Isso dificilmente ocorre nesse tipo de ambiente, já que existe uma grande probabilidade de existir interferências do tipo multipercursos. Nesse tipo de interferência, que ocorre devido à refrações e reflexões no percurso, são criados múltiplos caminhos para a onda, acarretando perda de informação e atenuação de sinal.

### 2.1.2 TOA

A técnica TOA (*Time of Arrival*) é baseado no tempo de chegada de um sinal ou um pacote. A estimação da distância entre dois nós se dá pela diferença entre o instante em que o pacote foi enviado e o instante em que ele chegou. Desse tempo, é descontado o tempo de atraso de processamento e resposta do pacote. Com essa informação e a velocidade de propagação das ondas, a distância é estimada.

Para garantir o funcionamento desse esquema, é necessário que os relógios dos nós estejam sincronizados e que sejam precisos, o que deixa o hardware do nó mais caro.

Uma dificuldade encontrada na utilização desse método é a variação do tempo de propagação e de resposta entre os diversos aparelhos dos fabricantes. Assim, a estimativa do tempo de chegada fica comprometida.

A distância entre um nó e uma âncora, um ponto fixo para comparar a posição do nó, pode ser calculada pela seguinte equação:

$$ri = ti * v \qquad i = 1.2.3 \ldots., n \tag{6}$$

Na Eq.(6), v corresponde à velocidade da luz e ti corresponde ao tempo de propagação do pacote.

No TOA o modelo utilizado é um modelo circular. Por isso, são necessários pelo menos três sensores para uma estimativa num plano de duas dimensões (Figura 2.1). Caso o número de sensores seja dois, existe a possibilidade dos círculos gerados a partir dos receptores não se interceptarem, e, portanto, não gerarem nenhuma solução possível.

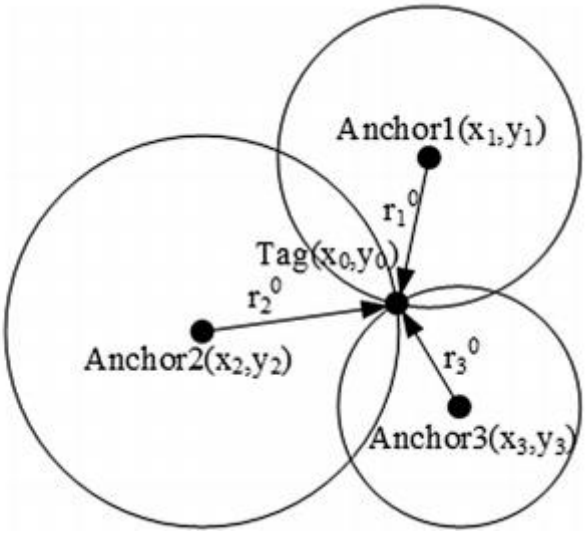



Figura 2.1 - Modelo de localização circular. Fonte: (WANG et al., 2016)

### 2.1.3 TDOA

A técnica TDOA (*Time Difference of Arrival*) é baseado no tempo de chegada de um sinal específico em estações de recepção (sensores) fisicamente separadas. Assim como o TOA, os relógios dos nós também devem estar sincronizados e precisos. Porém, como essa é a única exigência da técnica, o TDOA requer um equipamento de hardware mais simples e de menor custo se comparado à técnica TOA. Se comparado com o RSSI, também leva vantagem por ser menos suscetível em relação a fatores como temperatura e mudanças nos terminais de recepção.

Os métodos de estimação de localização baseados em TDOA exploram a correlação existente entre pares de sensores para estimar melhor o tempo de chegada. O cálculo dos atrasos entre um par de sensores pode ser feito utilizando uma correlação cruzada entre os pares de sensores.

A partir da distância ri obtida pela Equação (3), pode-se calcular a diferença do tempo dos sinais:

$$ri - rj = rij = tij * v \quad i = 1,2,3,\dots,n \tag{7}$$

O TDOA se utiliza de um modelo hiperbólico (Figura 2.2), muito mais complexo que o modelo circular do TOA. Cada sensor, que já tem coordenadas conhecidas, estimará a distância para a estação de acordo com a Eq. (8). Em seguida, é possível calcular a coordenada da estação através da Eq. (9), apenas com 4 ou mais sensores.

$$\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2} - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = r21 \tag{8}$$

$$\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = ri1 \tag{9}$$

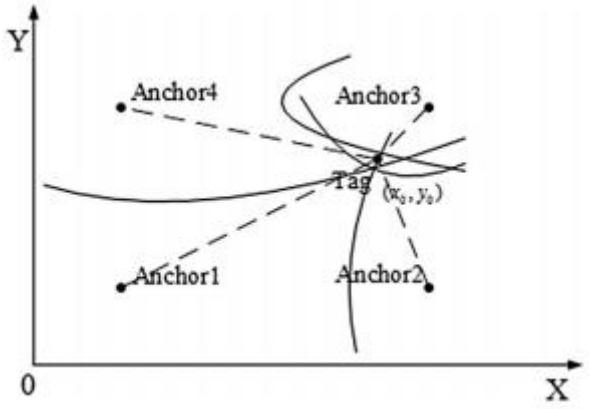



Figura 2 .2 - Modelo de localização hiperbólica. Fonte: (WANG et al., 2016)

Entretanto, os métodos baseados em TDOA não assumem os multipercursos dos sinais que chegam ao arranjo de sensores. Por isso, suas aplicações ficam limitadas quando multipercursos são considerados.

### 2.1.4 Técnicas baseadas em RSS (*Received Signal Strength)*

As técnicas de localização baseadas em potência de sinais são muito utilizadas em ambientes *indoor*, onde os sistemas de localização não conseguem oferecer muita acurácia (MADUSKARTAPASWI, 2017). O funcionamento dessas técnicas se baseiam em medir a potência do sinal recebido no ponto desconhecido com relação a posições conhecidas. A partir do valor da potência do sinal recebida, o ponto desconhecido consegue estimar a sua posição. Existem dois modos do ponto desconhecido estimar a sua posição.

O primeiro modo se dá pela conversão da potência recebida em distância utilizando equações de propagação de ondas. Sabendo a distância em que o ponto se encontra de pontos conhecidos, é possível estimar a sua posição utilizando um algoritmo de multilateração. Esse tipo de abordagem requer uma grande complexidade no algoritmo mas provê alta acurácia[(MADUSKARTAPASWI, 2017).

Para se estimar a distância entre o dispositivo transmissor e o receptor, primeiramente deve-se escolher um modelo de estimação de distância que utiliza a potência do sinal. O modelo mais utilizado é o Log-Normal Shadowing model, que, de modo simplificado, é descrito na equação abaixo:

$$\Pr(d)\,[dBm] = \Pr(d0)\,[dBm] - 10nlg(d/d0) \tag{9}$$

Onde "d" é a distância entre o receptor e o transmissor, "d0" é a distância referência. Consequentemente, Pr(d) é a potência de recepção e Pr(d0) é a potência de recepção para a distância referência. A variável "n" é um valor de atenuação exponencial, que varia de ambiente para ambiente. Por padrão, é utilizado 1 metro para a distância "d0". Com isso, obtemos a seguinte equação:

$$\mathrm{RSS}[dBm] = \Pr(d)\,[dBm] = A - 10n\lg(d) \tag{10}$$

Designa-se "A", o valor de potência do sinal recebido a uma transmissão de 1 metro de distância.

O outro modo de estimar a posição é considerar a potência do sinal diretamente. Para esse tipo de abordagem é necessário realizar o mapeamento do ambiente, onde são realizadas medições da potência do sinal em vários pontos do ambiente e as medidas são gravadas em um banco de dados (FAGUNDES, 2008). Posteriormente, quando o ponto desconhecido quer estimar a sua posição, ele compara o valor da potência recebida com os valores previamente mapeados e toma uma decisão baseada em que posição mapeada ele está mais próximo. Esse modo de operação requer um algoritmo menos complexo e em mapeamentos menores não sofre com os proBLEmas provenientes dos multipercursos que o sinal percorre. Apesar disso é necessário o mapeamento prévio (MADUSKARTAPASWI, 2017). Esse esquema comumente recebe o nome de *fingerprinting* pela literatura.

Os sistemas baseados em RSS(*Received Signal Stregnth)* normalmente baseiam-se numa infraestrutura preexistente de rede, como Wi-Fi ou Bluetooth. Isso diminui bastante os seus custos de implantação

## 2.2 TECNOLOGIA BLUETOOTH

O Bluetooth (IEE 802.15) é um padrão de tecnologia usado para comunicações sem fio de curto alcance. Ele foi desenvolvido no fim dos anos 90 com a intenção de unir, de forma barata, os mundos da comunicação e da computação, sem perder o foco em questões como confiabilidade, segurança e baixo consumo de energia.

Seu uso é bastante disseminado entre aparelhos de uso pessoal e periféricos, como fones de ouvido, impressoras e smartphones. Pelas informações contidas nesses tipos de dispositivos, o Bluetooth virou alvo de uma série de ataques, incluindo man-in-the-middle (HAATAJA e TOIVANEN, 2010) e ataques de negação de serviço.

A comunicação entre dispositivos Bluetooth ocorre em uma rede denominada *piconet*. Nessa rede, são definidos quem são os dispositivos mestres e quem são os dispositivos escravos. Os dispositivos mestres requisitam os serviços, além de organizar e comandar a transmissão e recepção de dados. Cabe então aos escravos prover os serviços requisitados pelo dispositivo mestre. São definidas de acordo com a quantidade de mestres e escravos, as topologias *Piconet* Simples, *Piconet Multi-slave* e a *Scatternet* (Figura 2.2).

Na topologia *Piconet* simples, a conexão é ponto a ponto, com apenas um mestre e um escravo, que segue a "*hopping sequence*" do mestre nos sinais. A topologia *Piconet multi-slave* conta com um mestre e até 7 escravos ativos. Esses escravos podem entrar em "*sleep mode*" para liberar conexões. O conjunto de várias topologias *piconets* é denominada *Scatternet*, em que um dispositivo mestre de uma topologia pode ser um escravo de outro dispositivo situado em outra *piconet*. Na Scatternet, não é possível ser mestre de mais de uma *piconet*, porém é possível um escravo pertencer a duas *piconets* diferentes.

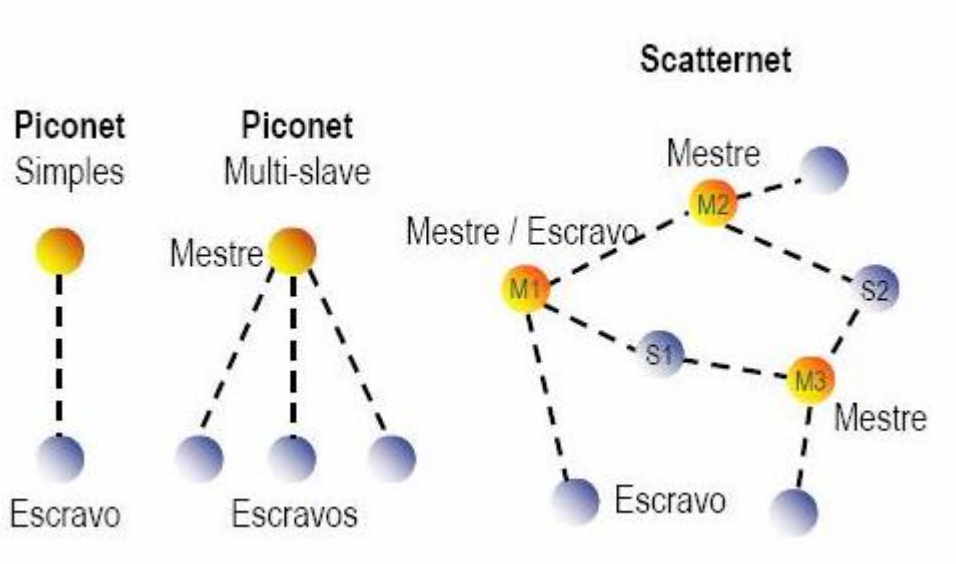



Figura 2.3 – Arquitetura Piconet. Fonte: (DIAS, 2018)

### 2.2.1 Bluetooth 4.0

Em 2010 foi lançado o Bluetooth na sua quarta versão, o chamado Bluetooth 4.0, que teve um grande avanço na eficiência do consumo de energia. Isso foi possível graças à tecnologia BLE *(Bluetooth Low Energy*) que garante um consumo consideravelmente inferior às suas versões anteriores, utilizando como alimentação baterias do tamanho de moedas. Essas características geram inúmeras possibilidades de aplicações, principalmente na utilização de dispositivos em redes de sensores sem fio, que são base para redes IoT (*Internet of Things)* e para a indústria *health care*.

O BLE se utiliza de mensagens de curta duração para reduzir o consumo de bateria (FARAGHER, HARLE, 2015). Essas mensagens, que podem ser tanto mensagens de dados ou mensagens de algum tipo de anúncio, conhecidas como *advertisement messages*. Esse tipo de mensagem é controlado pelo GAP (*Generic Access Profile*), e é a única maneira de se transmitir mensagens para mais de um dispositivo de uma só vez (*broadcasting*).

Beacons são mensagens de anúncio que ocorrem de modo broadcast. Quando essas mensagens são recebidas por um dispositivo compatível, que consegue tratar essas informações, este dispositivo é capaz de estimar com uma boa precisão a localização do dispositivo emissor. Por isso, são amplamente implementados em dispositivos periféricos de clientes, que, ao mandar dados para uma aplicação, conseguem gerar um rastreio do cliente ou notificações baseadas em check-in. Para que dispositivos periféricos consigam estabelecer uma conexão com um dispositivo central a fim de trocar e armazenar dados, foi desenvolvido o padrão GATT (*Generic Attribute Profile*).

O GATT define que um dispositivo periférico pode se comunicar com apenas um dispositivo central, como um smartphone (Figura 2.4). Assim que um dispositivo estabelecer conexão com um

dispositivo central, ele parará de enviar mensagens de anúncio, e por isso, outros dispositivos não conseguirão enxergá-lo e muito menos estabelecer uma conexão com ele até que sua conexão atual seja terminada. Caso um dispositivo periférico precisar trocar dados com outro dispositivo periférico, eles deverão se conectar a um dispositivo central que faça esse encaminhamento de dados.

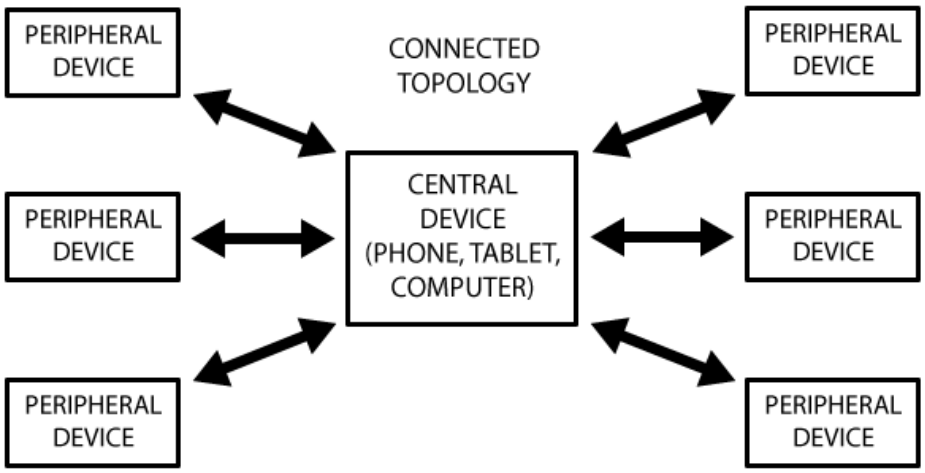



Figura 2.4 –Topologia de conexões GATT. Fonte: (DIAS, 2018)

Nas comunicações GATT, o dispositivo periférico é tratado como GATT server e o dispositivo central é tratado como GATT *client*. Todas as transações de dados são iniciadas pelo GATT *client*, e então, respondidas pelo GATT server. As transações são baseadas em objetos aninhados, chamados perfis, serviços e características.

A característica é o ponto de nível mais baixo dessas transações, e encapsula um único ponto , ou array, de informação. Os serviços são responsáveis por dividir os dados em unidades lógicas, que utilizam essas informações. Um exemplo de serviço é a Frequência cardíaca, Heart Rate Service, que conta com a característica medida de frequência cardíaca. Os perfis são conjuntos de serviços utilizados pelo Bluetooth SIG (Bluetooth Special Interest Group) e por desenvolvedores periféricos. O perfil de frequência cardíaca, por exemplo, combina serviços da Frequência Cardíaca e do serviço de Informação de Dispositivos.

### 2.2.2 Bluetooth 4.1

O Bluetooth 4.1 é tido como uma revisão da versão 4.0. Anunciado em 2013, essa versão teve como objetivo deixar as conexões mais rápidas e mais estáveis. A usabilidade foi melhorada ao poder assimilar perfeitamente recursos com diferentes tecnologias de células, tal como o LTE (*Long Term Evolution*). Assim, a chamada *plug and play* acontece, com reconexões manuais menos frequentes e uma troca de dados mais eficiente.

Uma outra melhoria destacada é a maior flexibilidade à criação de dispositivos que utilizam a topologia Bluetooth, por meio das topologias *dual-mode* e *link-layer*.

Com base no conceito de IoT, que prevê uma crescente variedade de dispositivos permanentemente online, e consequentemente assim, um exponencial crescimento do tráfego, a versão 4.1 do Bluetooth habilita essas possibilidades estabelecendo o IP como base de conexão. Isso permite que os dispositivos possam criar um canal dedicado entre si.

### 2.2.3 Bluetooth 4.2

O Bluetooth 4.2 apresentou vários novos recursos que melhoraram, principalmente, a taxa de transmissão das conexões e a privacidade dos usuários. Os novos recursos de privacidade implicam que para que um *beacon* seja recebido um dispositivo, este previamente deve dar permissão para esse rastreamento. Isso impede que rastreiem um dispositivo sem permissão através de uma conexão Bluetooth. Os novos recursos

A transferência de dados é até 250% mais rápida que nas versões anteriores. O aumento da velocidade de transferência de dados e a capacidade de pacotes em 1000% reduzem a possibilidade de erros de transmissão e também o consume de bateria, o que resulta numa conexão muito mais eficiente. ("New Bluetooth Specifications EnaBLE IP Connectivity And Deliver Industry Leading Privacy And Increased Speed | Bluetooth Technology Website", 2018)

### 2.2.4 Bluetooth 5

A versão mais atualizada do Bluetooth foi apresentada em 2017. Ela trouxe melhorias bastante significativas, especialmente para modos de operação com baixa energia. A motivação principal para esse esforço continua sendo o IoT. Porém, em 2018, a grande maioria dos smartphones ainda utiliza versões anteriores da tecnologia. Os aparelhos conectáveis como fones de ouvido, alto-falantes, mouses e teclados também precisam possuir hardware compatível com o Bluetooth 5. Um ponto positivo é a compatibilidade reversa com as versões anteriores da tecnologia.

Já que se sacrificou largura de banda para se obter um alcance maior, o Bluetooth 5 oferece também a possibilidade de dobrar a largura de banda às custas de energia. Com essa configuração, as transmissões suportam 2 Mbps a 20 dB no modo de baixa energia. Assim, existem duas possibilidades de acordo com o foco da operação: para longas distâncias ou para maiores quantidades de dados.

### 2.2.5 Camada Fisica do Bluetooth 4.0 ,4,1 e 4.2

O *Bluetooth Low Energy* opera na banda *Industrial Scientific Medical* (ISM) de 2,4 GHz, possui 40 canais de radiofrequência (RF) com espaçamento de 2 MHz. Existem dois tipos de canais BLE RF: canais de *advertisement* e canais de dados. Os canais de *advertisement* são usados para a descoberta de dispositivos, o estabelecimento de conexões e a transmissão em *broadcast*, enquanto os canais de dados são usados para comunicação bidirecional entre dispositivos conectados.

Existem três canais de *advertisement*, que possuem frequências centrais que minimizam a sobreposição com os canais IEEE 802.11 1, 6 e 11, que são comumente usados em vários países de forma a minimizar a interferência do IEE802.11 no BLE, são eles: o 37, 38 e 39. A Figura (2.5) mostra a disposição desses canais.

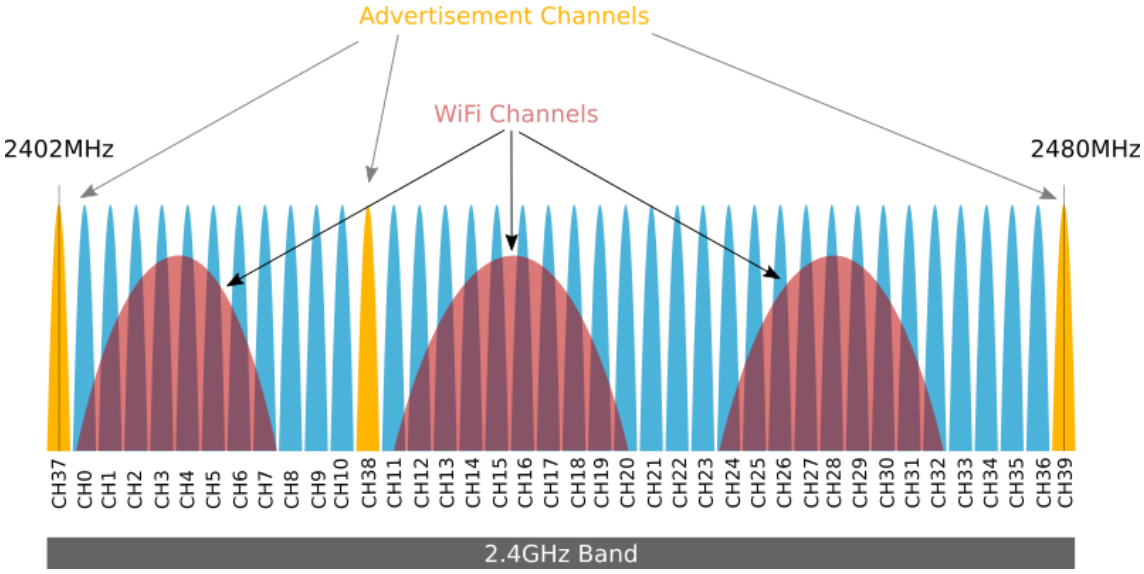



Figura 2.5- Canais de Advertisement e canais do WIFI. Fonte: ("A BLE Advertising Primer · Argenox Technologies", 2018)

O FHSS (*Frequency Hopping Spread Spectruim)* é utilizado na parte superior dos canais de dados para enfrentar proBLEmas de interferência e propagação sem fio, como *fading* e *multipath*. Este mecanismo seleciona um dos 37 canais de dados disponíveis para comunicação durante um determinado intervalo de tempo.

Todos os canais físicos usam a modulação *Gaussian Frequency Shift Keying* (GFSK), que é simples de implementar. O índice de modulação está na faixa entre 0,45 e 0,55, o que permite reduzir o consumo de energia de pico. A taxa de dados da camada física é de 1 Mbps. A sensibilidade do receptor é definida no BLE como o nível de sinal no receptor para o qual uma taxa de erro de bit (BER) de $10^{-3}$ é alcançada. A especificação BLE exige uma sensibilidade melhor ou igual a -70 dBm. A faixa de cobertura é tipicamente em várias dezenas de metros. (GOMEZ *et All* 2012)

### 2.2.6 Tipos de pacotes BLE

A camada física do *Bluetooth Low Energy* possui apenas um formato de pacote, que é usado tanto nos canais de *advertising* como nos canais de dados. Cada pacote consiste de quatros campos: o preâmbulo, o endereço de acesso, o PDU (*Packet Data Unit*) e o CRC (*Cyclic Redundancy Check*). A Figura (2.6) mostra o formato do pacote.

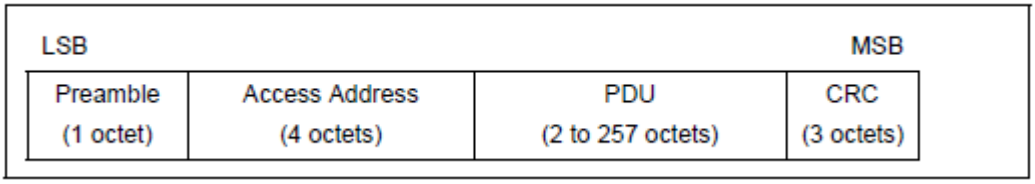



Figura 2.6 – Formato do pacote BLE. Fonte: (BLUETOOTH SPECIFICATION Version 4.2[Vol 6], 2014)

O preâmbulo é utilizado para realizar a sincronização de frequência e estimação de tempo de bit. Em pacotes nos canais de *advertising*, o preâmbulo deve ser 10101010b. Nos pacotes nos canais de dados o preambulo depende do último bit do endereço de acesso: caso ele seja 1, o preambulo é 01010101b, caso 0, o preambulo é 10101010b.

O endereço de acesso é um número de 32 bits aleatórios, que é gerado durante a conexão entre dois dispositivos. Como os pacotes dos canais de *advertising* são enviados antes de estabelecida qualquer conexão, o endereço nesse caso é 0x8E89BED6.

O CRC é calculado em cima do PDU. O tamanho máximo do PDU era de 39 bytes até o Bluetooth 4.1, porém na versão 4.2 esse tamanho foi estendido até 257 bytes. Esse aumento por sua vez, foi apenas em pacotes transmitidos nos canais de dados. Nos pacotes transmitidos nos canais de *advertising* o tamanho continuou o mesmo (39 bytes). Além do tamanho, o PDU dos canais de dados possui um campo de 4 bytes denominado MIC (*Message Integrity Check).*

O PDU de canais de *advertising*, possui dois campos majoritários (Figura 2.7): Um cabeçalho de 16 bits que especifica o tipo de *advertising* PDU e o tamanho do *payload*. O cabeçalho é formado pelo PDU Type, dois campos reservados para o futuro (RFU) e mais dois campos (TxAdd e RxAdd), que são usados de acordo com o tipo de *advertising* PDU (Figura 2.8). O tamanho do *payload* varia de acordo com o tamanho especificado no cabeçalho.

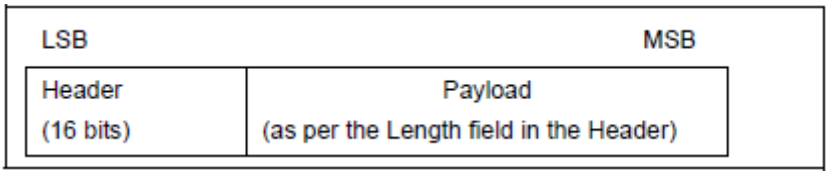



Figura 2.7- PDU dos canais de advertising Fonte: (BLUETOOTH SPECIFICATION Version 4.2[Vol 6], 2014)

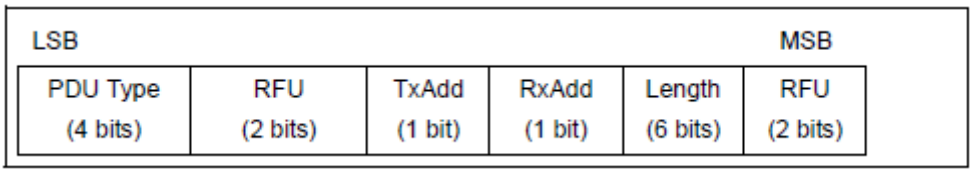



Figura 2.8- Cabeçalho do PDU. Fonte: (BLUETOOTH SPECIFICATION Version 4.2[Vol 6], 2014)

Existe um tipo de *advertising* PDU para cada situação, são eles:

- ADV_IND: quando qualquer outro dispositivo pode se conectar. Por exemplo: um dispositivo *slave* é ligado pela primeira vez.
- ADV_NONCONN_IND: quando não se pode conectar no dispositivo, mas ele pode ser detectado por qualquer um. Por exemplo: *beacons*.
- ADV_*SCAN*_IND: quando um dispositivo pode ser conectado, porém ele deseja informar mais informações do que apenas o tamanho do *payload*.
- ADV_DIRECT_IND: quando um dispositivo quer se conectar apenas com outro dispositivo já conhecido.

Os três primeiros *advertising* PDU são utilizados para comunicação broadcast, já o último é utilizado para restabelecer uma comunicação. O formato do *payload* para os PDUs utilizados para comunicação *broadcast* são iguais, a Figura. (2.9) mostra o formato dele.

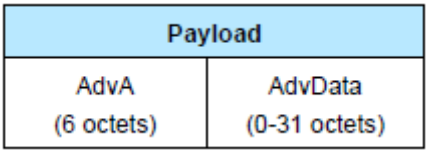



Figura 2.9 – Formato do *payload* para comunicação *broadcast*. Fonte: (BLUETOOTH SPECIFICATION Version 4.2[Vol 6], 2014)

O *payload* é formado pelo AdvA, que é o endereço do dispositivo que está transmitindo esse pacote, e o AdvData, que é formado por 1 ou mais *Advertisement Data Strucutures (*AD). Estes são formados pelo campos: tamanho (1 *byte*), tipo (1 *byte*) e a data em si, que pode chegar até os 29 *bytes*. Existem vários tipos de *Advertisement Data Strucutures*, como citado em 2.2.1 o GAP que gerencia essas estruturas. Um AD bastante utilizado é o *Manufacturer Specific Data*, os iBeacons utilizam esse AD para informar seus dados.

### 2.2.7 Operação do BLE

Um dispositivo BLE pode operar em três modos diferentes, dependendo da funcionalidade necessária: *advertising*, *scan* e inicialização. Um dispositivo no modo de *advertising*, transmite periodicamente informações de *advertisement* nos três canais destinados a isso. Como mostra a Figura (2.9), um dispositivo no modo *advertising* permanece enviando PDUs em sequência sobre cada um dos três canais em um *Advertising Event*, que é composto por um tempo chamado de *AdvInterval* fixo (representado por $\tau$AI na figura 2.9) e um tempo chamado de *AdvDelay* aleatório (representado por $\delta$). (CHO et al., 2014)

O *AdvDelay* é um valor aleatório para separar os *Advertising Events* quando tiver mais de um dispositivo em modo *advertising*. Como existe esse valor aleatório, as transmissões dos PDUs nos três canais se tornam completamente assíncronas. (CHO et al., 2014)

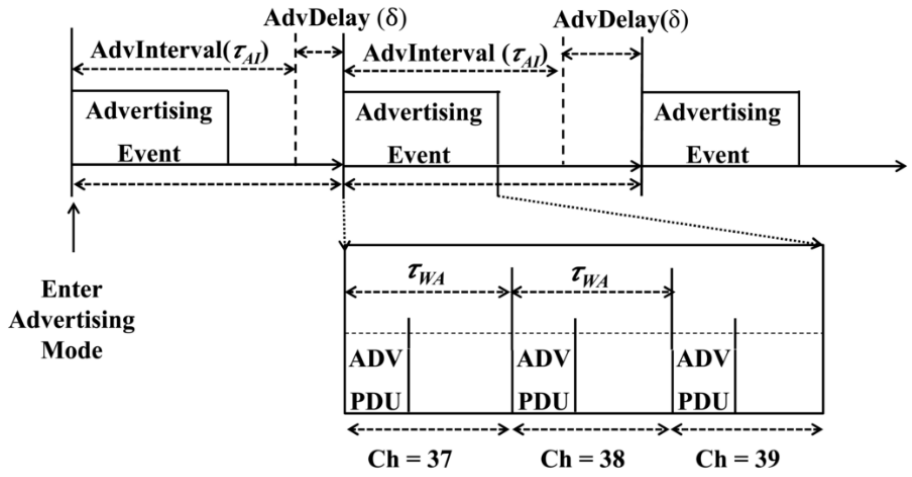



Figura 2.9 – Advertising Event. Fonte: (CHO et al., 2014)

O *AdvInterval* deve ser um inteiro múltiplo de 0,625 ms no intervalo de 20 milissegundos a 10,24 segundos, sendo que caso o *Advertising* PDU for do tipo ADV_NONCONN_IND ou ADV_*SCAN*_IND, o mínimo para esse intervalo é de 100 ms. Por esse motivo que caso se necessite de uma frequência maior que 10 Hz o tipo de PDU tem que ser alterado para o ADV_CONNECTABLE_UNDIRECTED. *O AdvDelay* deve estar dentro do intervalo de 0 a 10 milissegundos. De acordo com a especificação, um período de *advertising* para cada canal (denotado por τWA) deve ser menor ou igual a 10 ms. (CHO et al., 2014)

Analogamente, o dispositivo operando no modo *scan,* percorre periodicamente os 3 canais de *advertising* . Durante cada *scanInterva(* τSI na figura 2.10)*l,* um canal de *advertising* diferente é escutado durante uma *scanWindows(*τSW na figura 2.10). (CHO et al., 2014)

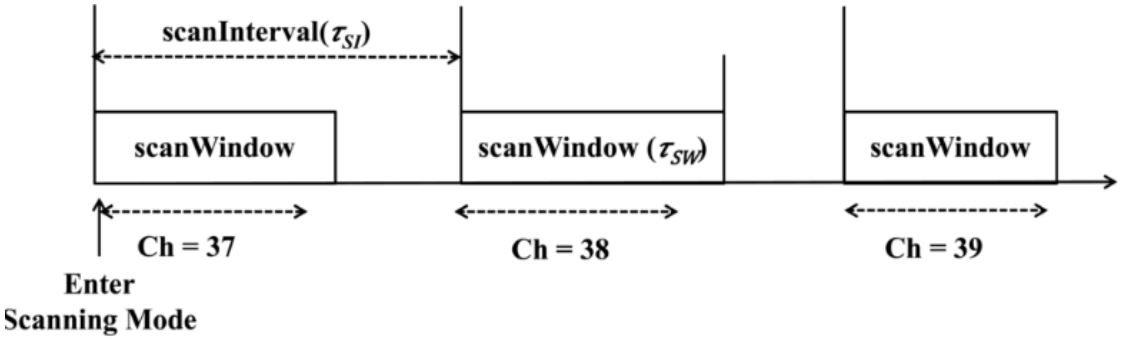



Figura 2.10- Operação de *Scan*. Fonte: (CHO et al., 2014)

O *ScanWindow* e o *ScanInterval* devem ser valores entre 2.5 e 10.24 segundos. A tabela 2.1 sintetiza os valores dos parametros importantes para a operação do *scan.*

Tabela 2.1 - Parâmetros importantes para o *scan* . Fonte: (CHO et al., 2014)

| Notação | Significado | Especificação recomendada |
|---|---|---|
| τWA | Periodo de *advertising* por canal | ≤ 10 ms |
| τAI | *Advertisiment Interval* | Inteiro múltiplo de 0,625ms entre [20~10,24]ms |
| δ | *AdvDelay* (Intervalo randômico escolhido entre [0, $\delta_{max}$] | [0, $\delta_{max}$] |
| $\delta_{max}$ | Valor máximo que o *AdvDelay* pode assumir | ≤ 10 ms |
| τSI | *Scan Interval* | Inteiro múltiplo de 0,625ms entre [2,5~10,24]ms |
| τSW | *Scan Window* | Inteiro múltiplo de 0,625ms entre [2,5~10,24]ms τSW≤ τSI |

### 2.2.8 iBeacons

Como visto na seção 2.2.1, os *beacons* são como faróis que enviam sinais de luz para barcos. Os iBeacons são dispositivos *Bluetooth Low Energy* desenvolvidos pela Apple que apenas enviam um sinal num formato específico. Embora desenvolvido pela Apple, o iBeacon também é suportado por outros tipos de dispositivos, tais como Android e BlackBerry. Por isso, o termo iBeacon está rapidamente se tornando um termo genérico para o BLE *beacon*. O padrão definido para o BLE *Advertising* pela Apple consiste em 4 partes de informação:

- UUID: É uma string de 16 bytes usada para diferenciar um grande grupo de *beacons* relacionados. Por exemplo, se a Coca-Cola mantivesse uma rede de *beacons* em uma cadeia de mercados, todos os *beacons* da Empresa X compartilhariam o mesmo UUID. Isso permite que o aplicativo de smartphone dedicado da Coca-Cola saiba quais anúncios de *beacon* vêm de *beacons* de propriedade da Empresa X.
- *Major*: É uma cadeia de 2 bytes usada para distinguir um subconjunto menor de *beacons* dentro do grupo maior. Por exemplo, se a Empresa X tivesse quatro *beacons* em um

determinado mercado, todos os quatro teriam o mesmo *major*. Isso permite que a Empresa X saiba exatamente em que loja está seu cliente.

- *Minor*: É uma cadeia de 2 bytes destinada a identificar *beacons* individualmente. Mantendo o exemplo da Empresa X, um *beacon* na frente da loja teria seu próprio Minor exclusivo. Isso permite que o aplicativo dedicado da Empresa X saiba exatamente onde o cliente está na loja.
- *Tx Power*: É usado para determinar a proximidade (distância) do farol. *Tx Power* é definido como a intensidade do sinal a exatamente 1 metro do dispositivo. Isso precisa ser calibrado e codificado antecipadamente. Os dispositivos podem usar isso como uma linha de base para fornecer uma estimativa de distância aproximada.

Por padrão, esses dispositivos têm tamanho bastante reduzido, tanto que podem ser alimentados por uma simples *coin cell*.

Muitas das aplicações atuais acontecem com smartphones se comunicando com outros smartphones. Isso ocorre pela popularização desse tipo de dispositivo, presente no mundo inteiro nos mais diferentes preços e modelos. Porém, existe uma quantidade altíssima de objetos que não possuem essa capacidade de comunicação entre si. Objetos como relógios, portões, malas e geladeiras, se integrados com iBeacons, podem se comunicar entre si e também com smartphones. Essa integração é capaz de gerar infinitas experiências interativas, como notificações de lojas e restaurantes. Com os iBeacons, esses estabelecimentos comerciais são capazes de cruzar dados e sugerir promoções e mercadorias baseadas no perfil do usuário e em sua localização.

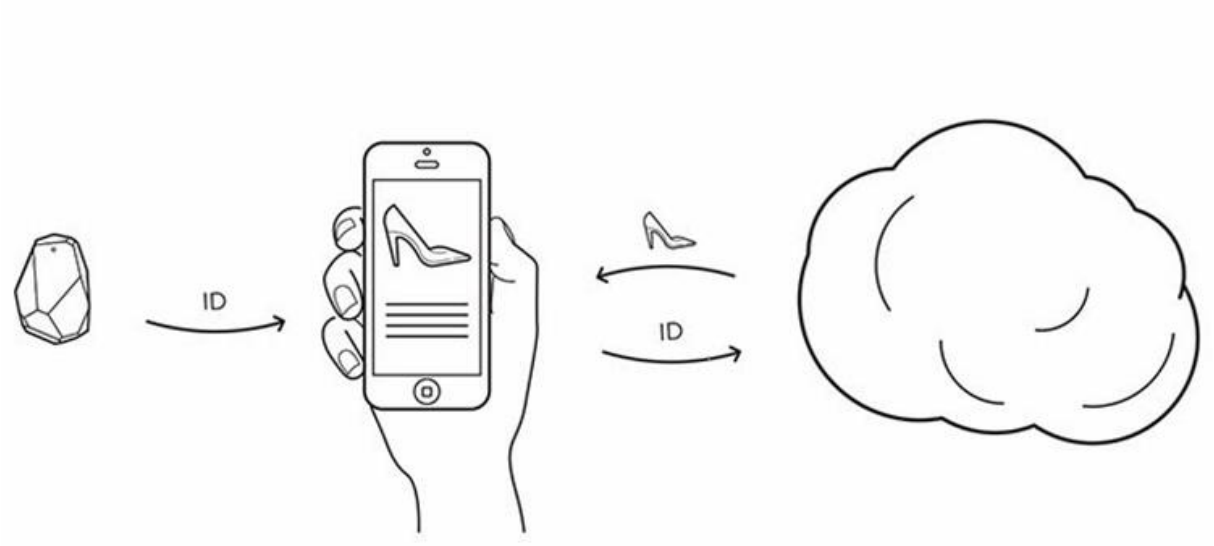



Figura 2.11- Cruzamento de dados entre iBeacon e usuário. Fonte: (CAMERON, 2018)

A questão da privacidade e da segurança dos dados coletados pelos *beacons* são configuráveis pelo usuário. Se o cliente tiver fornecido muitas informações pessoais e já tiver dado permissão para usarem esses dados (ou seja, um aplicativo de banco de varejo, membros de um esquema de fidelidade de loja de varejo), os dados coletados poderão ser analisados em um nível mais profundo. Se o aplicativo não requerer nenhuma informação pessoal, os dados coletados serão anônimos e a análise dos dados será mais limitada à exibição de passos.

Oferecer a dois dispositivos a capacidade de se comunicarem uns com os outros quando estão muito próximos é o desafio comercial que a tecnologia NFC (*Near Field Communication*) tem tentado resolver. Os iBeacons da BLE também podem resolver esse desafio, mas com o benefício adicional de que o dispositivo não precisa ser segurado fisicamente contra um sensor, pois ele pode ficar no bolso ou na bolsa o tempo todo.

A tecnologia GPS é ótima para uso externo, mas os sinais de satélite são significativamente menos eficazes dentro de um edifício. Os iBeacons oferecem uma solução econômica para os serviços de localização tanto em edifícios quanto em *outdoor*, com a vantagem adicional de serem baratos de se implantar e de drenarem significativamente menos a bateria de um smartphone do que a tecnologia GPS.

## 2.3 EQUIPAMENTO REDBEAR BLE NANO 2 E AMBIENTE DE DESENVOLVIMENTO

O *Redbear BLE Nano 2* é uma plataforma de desenvolvimento, que possui o *Bluetooth Low Energy SoC(System on Chip)* nRF52832 da *Nordic SemiConductor*. O processador do nRF5282 é um *Cortex-M4f que* possui um clock de 64 MHz, e foi construído para executar as tarefas o mais rápido possível e voltar ao modo de conservação de bateria.

As aplicações para esse *SoC* podem ser programadas usando a *Nordic's SDK,* plataforma de desenvolvimento para micro controladores Nordic, *mbed.org*, que é um site para desenvolvimento de dispositivos *IoT* , ou utilizando a plataforma de desenvolvimento de aplicações do Arduino.

A pequena plataforma de desenvolvimento conta com 12 pinos de entrada/saída digital, 12 pinos PWM (*Pulse Width Modulation)* e 6 pinos com ADC (*Analogic- Digital Converter)*, além disso: 2 pinos para I²C (*Inter-Integrated Circuit)*, 8 pinos para SPI *(Serial Peripheral Interface),* 2 pinos para UART (*Universal asynchronous receiver-transmitter)*. Para a compilação de programas, ele possui uma porta SWD (*Serial Wire Debug)*. A Figura (2.11) mostra a relação dos pinos no *BLE Nano 2*

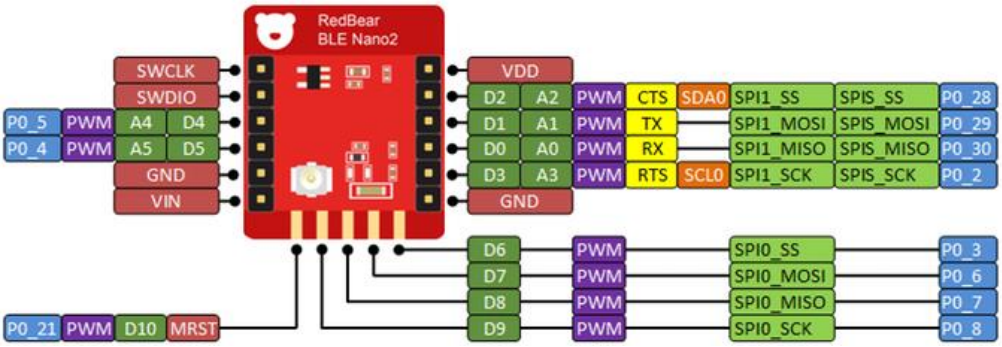



Figura 2.11- Relação dos pinos no RedBLEar Nano 2. Fonte: ("redbear/nRF5x", 2018)

Além de possuir todas essas funcionalidades supracitadas, ele opera com tensões entre 1,8 V ~ 3,6 V. Ele pode ser alimentado com tensões entre 1,8 V até 13 V, sendo que tensões entre 3,3 V até 13 V devem ser ligadas no pino VIN, que possui um conversor. E tensões entre 1,8 V até 3,3 V devem ser ligadas no pino VDD.

Uma das desvantagens dessa plataforma de desenvolvimento é a necessidade mais um hardware para a compilação de programas, o DAPlink. O DAPlink é um projeto open source desenvolvido pelo ARM mbed team, que é utilizado para compilar programas utilizando a porta SWD. Ele conta com uma CPU ARM Cortex-M3 MCU e uma porta USB. A figura 2.12 retrata um DAPLink e sua relação dos pinos e a Figura. 2.13 mostra um DAPLink com o RedBear BLE Nano 2.

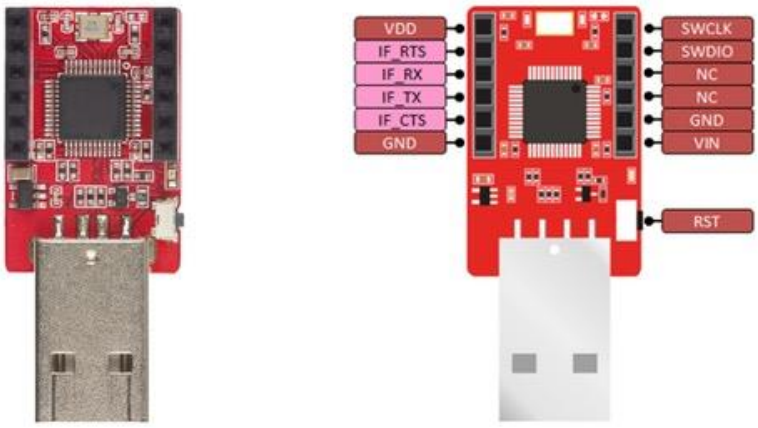



Figura 2.12- DAPlink e relação dos pinos. Fonte: ("redbear/nRF5x", 2018)

Figura 2.13 – RedBear BLE Nano 2 e DAplink Fonte: ("redbear/nRF5x", 2018)

### 2.3.1 Arduino IDE

Arduino IDE é um ambiente de desenvolvimento integrado *open-source* utilizada para desenvolvimento para dispositivos embarcados. A linguagem de programação utilizada para programar nessa IDE é baseada em C, e possui milhares de bibliotecas para utilização de módulos externos (*Shields).* A IDE possui suporte para a instalação de compiladores de várias placas de desenvolvimento, dessa forma além de se aproveitar das facilidades da IDE ainda se pode e aproveitar as diversas bibliotecas já criadas para Arduino. O compilador do RedBear BLE Nano 2 pode ser instalado na Arduino IDE, além de possuir uma biblioteca criada pelos desenvolvedores com alguns exemplos.

### 2.3.2 Bateria CR-2032 e *holder*

A bateria-botão CR-2032 é a responsável pela alimentação do RedBearBLE Nano 2. Ela é uma bateria de lition com tensão de saída de 3 V, e capacidade de 220mAh, tornando-a suficiente para alimentação do dispositivo. O peso dela é de 3.2 gramas, o diâmetro é de 20 mm e a altura de 3.2mm. Na Figura 2.11 é apresentado a CR-2032.

Como *RedBear BLE Nano2 n*ão possui entrada para a bateria-botão, o *LilyPad Coin Cell Baterry Holder* é usado para fornecer as saídas de tensão necessária para alimentar o dispositivo. Esse *hardware* possui 4 pinos de saídas e uma chave *on-off* para ligar e desligar o circuito. Dos 4 pinos, 2 são utilizados para alimentar. A Figura 2.11 mostra o *holder* com os 4 pinos.

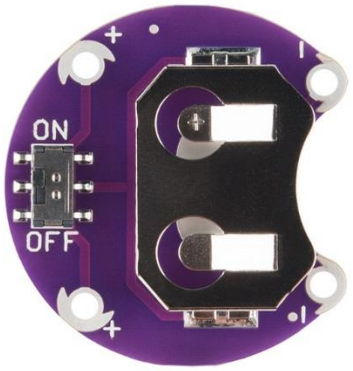



Figura 2.11- LilyPad Coin Cell Battery Holder. Fonte:("LilyPad Coin Cell Battery Holder - Switched - 20mm - DEV-13883 - SparkFun Electronics", 2018)

## 2.4 Asus Zenfone Zoom 3

O Asus ZenFone Zoom 3 é um *smartphone* da marca ASUS, lançado em agosto de 2016. O sistema operacional dele é o *Android Nougat 8.0  Oreo* e é equipado com a versão 4.2 do *Bluetooth*, abaixo algumas informações relevantes sobre esse aparelho.

- Sistema Operacional: Android 8.0 Oreo
- CPU :Octa-core 2.0 GHz Cortex-A53
- Memória RAM: 4 GB RAM
- Bluetooth: 4.2, A2DP, EDR, LE

## 2.5 Samsung Galaxy S6

O *Samsung Galaxy S6* é um *smartphone* lançado pela Samsung em abril de 2015. O Sistema operacional dele é o *Android 7.0 Nougat* e a versão do Bluetooth presente é a 4.1. Abaixo algumas informações relevante sobre o *smartphone.*

- Sistema Operacional: Android 7.0 Nougat
- CPU :Octa-core (4x2.1 GHz Cortex-A57 & 4x1.5 GHz Cortex-A53)
- Memória RAM: 3 GB RAM
- Bluetooth: 4.1 A2DP, LE, aptX

## 2.6 Moto G 2nd Gen

O Motorola *Moto G 2nd Gen* é um *smartphone* da Motorola lançado em Setembro de 2014. O Sistema operacional é o *Android Marshmallow* e a versão do *Bluetooth* presente é a 4.0.

- Sistema Operacional: Android 6.0 Marshmallow
- CPU: Quad-core 1.2 GHz Cortex-A7
- Memória RAM: 1 GB RAM
- Bluetooth: 4.0, A2DP, LE, aptX

# 3 TRABALHOS CORRELATOS

Sistemas de localização de dispositivos utilizando técnicas baseadas em RSS (Received Signal Strength) são objetos de estudo bastante novos e que ganharam muita força com a chegada do *Bluetooth Low Energy*.

Em 2012, Chuenurajit *et al.*, propuseram um sistema de localização para redes de sensores sem fios *indoor* utilizando RSSI no padrão do *ZigBee*. Eles utilizaram 8 módulos *ZigBee* e utilizaram um algoritmo de min-max para estimar a posição do nó desconhecido. Ao fim, o erro médio encontrado no esquema foi de 2 metros.

Já em 2013, Wang *et al.* desenvolveram um sistema de localização *indoor* utilizando Bluetooth 2.1, combinando técnicas de triangulação com RSSI, o sistema foi testado em um ambiente pequeno e alcançou precisão de 3 centímetros em determinados casos.

No fim de 2015, Faragher e Harle, desenvolveram um esquema de localização utilizando *beacons* de *Bluetooth Low Energy*. O esquema desenvolvido utilizava um método de *fingerprinting* para localizar os dispositivos. O estudo foi realizado em um ambiente *indoor*, e com vários *beacons*. Foram obtidos erros de menos de 2,6 metros durante 95% do tempo.

Maduskar e Tapaswi (2017) desenvolveram um software que se utiliza de um método de *fingerpriting* para localização indoor, utilizando Wi-Fi. O software foi desenvolvido para Android e utilizava de vários AP's(*Acess Point)* para ter precisão. Com a utilização de 4 APs obteve-se os melhores resultados, de 9 testes realizados em um ambiente com 2 andares, 5 foram localizados precisamente e nos outros 4 o maior erro foi de 4 metros.

Diferentemente de todos os trabalhos supracitados, o projeto apresentado nesse relatório tem o foco em ambientes *outdoor*, e será utilizado *Bluetooth Low Energy*. Adicionalmente, em todos os trabalhos citados, o dispositivo estima a localização em relação aos pontos conhecidos (celulares). Nesse trabalho os pontos conhecidos estimam a posição do dispositivo. O sistema desenvolvido utilizará dos princípios de métodos de *fingerprinting* e da abordagem de definição de níveis de proximidades como a apresentada por Shashidhara e Poornima (2018), na qual é definido intervalos de valores de RSSI e classificados como: Perto, longe ou Muito Longe. Além de imprimir uma estimativa de distância, o aplicativo desenvolvido informará se a direção está correta em relação ao dispositivo.

Atualmente, existem algumas aplicações que já utilizam do mesmo princípio desse trabalho, como: as coleiras inteligentes de cachorros criadas pela *SportDog*, na qual é possível ser guiado até os cachorros utilizando a coleira, utilizando GPS("SportDOG", 2018); a empresa *ustwo* está trabalhando juntamente com a o London Society for Blind People (RLSB) para utilização de *iBeacons* como substituição de cachorros de cego, eles estão colocando vários equipamentos em pontos públicos, como metrô e informando via áudio orientações de qual caminho seguir. ("Can iBeacons Be Used To Help The Visually Impaired Navigate Public Transport?", 2018).

# 4 METODOLOGIA

## 4.1 Delimitação do tema

Nesse trabalho será criada uma aplicação para localização de dispositivos equipados com *Bluetooth Low Energy* utilizando apenas *hardware* e *software open-source*. Será utilizado um *smartphone* como o radar que irá localizar o dispositivo, que será o RedBear BLE Nano 2. Esse dispositivo foi escolhido por ser *open-source*, pequeno e já preparado para a versão 5 do Bluetooth, além de ser barato.

A fim de atingir esse objetivo, serão adotadas 3 etapas, mostradas no fluxograma da Figura. (4.1)

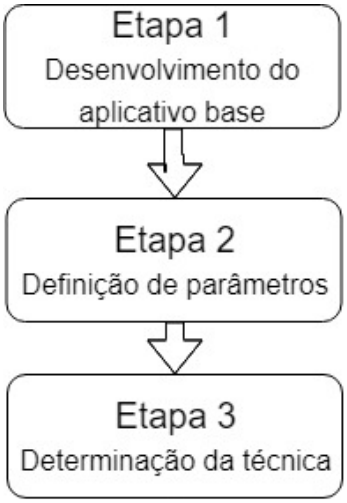



Figura 4.1 – Fluxograma do projeto.

- Etapa 1: Desenvolvimento do aplicativo base

Esta etapa consiste na criação de um aplicativo base para o Android que possa prover estatísticas das coletas Bluetooth LE como vazão de rede, média dos valores RSSI obtidos, desvio padrão, além de salvar os valores RSSI de cada pacote que chegue no *smartphone*. Além modificação de um *firmware* do RedBear BLE Nano 2 que permita a mudança de parâmetros, como o intervalo de *advertising*.

- Etapa 2: Definição de parâmetros de configuração baseado no comportamento do RSSI

Nesta etapa serão realizados testes a fim de definir, da melhor forma, os parâmetros de: frequência de *advertisement* e janela de *scan*, além da definição de qual versão do *Bluetooth Low Energy* utilizar.

- Etapa 3: Determinação da técnica

A partir dos resultados obtidos na etapa 2, será definido que técnica será utilizada para a localização do dispositivo desejado.

Ao final dessas etapas, será desenvolvido o aplicativo final utilizando todos os resultados obtidos durante essas etapas. O desenvolvimento do aplicativo, bem como o teste do mesmo serão mostrados no capitulo 5.

## 4.2 DESENVOLVIMENTO DO APLICATIVO BASE

### 4.2.1 Preparação do ambiente de desenvolvimento do RedBear BLE Nano 2

Foi escolhida a Arduino IDE para desenvolver para o Red Bear BLE Nano 2. Além de ter fácil instalação, a linguagem de programação é fácil de ser entendida. Porém, são necessárias algumas configurações na IDE antes de desenvolver para o dispositivo.

A Arduino IDE pode ser baixada gratuitamente no site oficial do Arduino. Após a finalização do download e a da instalação, foi necessária a instalação do compilador do Red Bear BLE Nano 2. Para realizar isso, foi adicionado o link (https://redbear.github.io/arduino/package_redbear_nRF5x_index.json) no menu de preferências "*Preferences"* nas configurações referente a *Additional Boards Managers URLs*, conforme mostrado na Figura. (4.2). O menu *Preferences* é acessível por meio da navegação no menu superior *File> Preferences*.

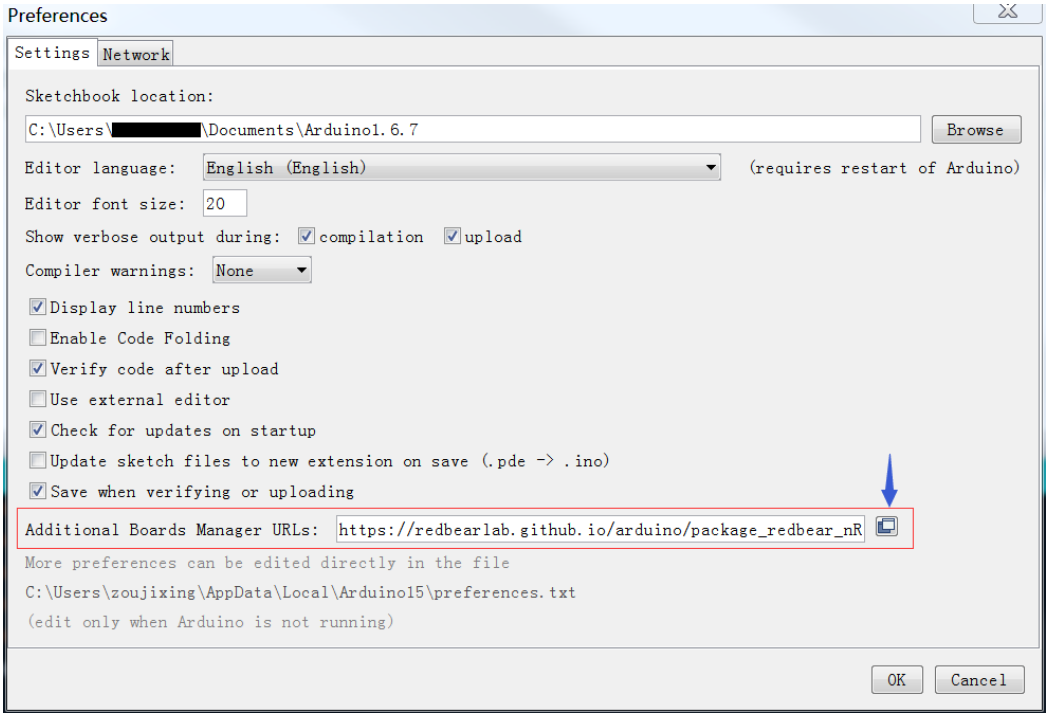



Figura 4.2 – Configuração da Arduino IDE, parte 1.

Após isso, foi necessária a instalação do compilador e da biblioteca da placa de desenvolvimento do projeto. Para isso, no menu *Board Manager*, o pacote *RedBear nRF52832 Boards* foi instalado conforme Figura. (4.3). O menu *Board Manager* é acessível por meio de *Tools>Board> Board Manager*

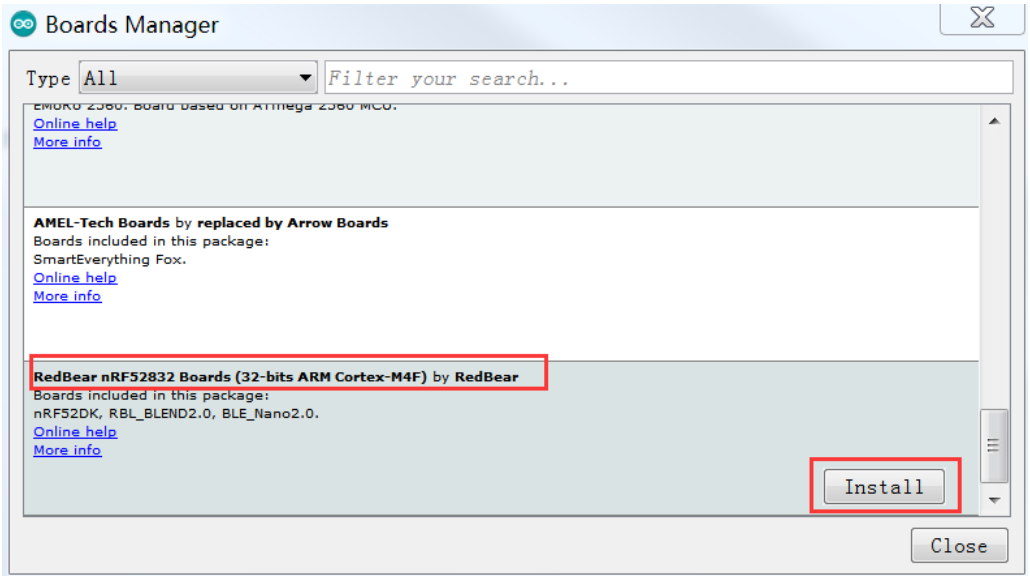



Figura 4.3 - Configuração da Arduino IDE, parte 2.

Depois desses passos, a Arduino IDE ficou pronta para o desenvolvimento para o RedBear BLE Nano 2

### 4.2.2 *Firmware Base do Red Bear BLE Nano 2*

O Código base utilizado para a criação do *firmware* do RedBear BLE Nano 2 foi o exemplo "BLE_beacon.h" disponível na biblioteca instalada na seção (4.2.1). Esse código apenas simula um *iBeacon* e o dispositivo não responde a nenhum pedido de empareamento, assim como os beacons definidos na seção 2.2.5.

Na primeira parte do código (Figura. 4.4), é definido o *payload* do beacon no estilo do *iBeacon*. Como nesse projeto o que é estudado é o valor do RSSI, o conteúdo do pacote não importava, portanto, esse *payload* não foi alterado.

```
const static uint8_t beaconPayload[] = {
  0x4C,0x00,                                                  // Company Identifier Code = Apple
  0x02,                                                       // Type = iBeacon
  0x15,                                                       // Following data length
  0x71,0x3d,0x00,0x00,0x50,0x3e,0x4c,0x75,0xba,0x94,0x31,0x48,0xf1,0x8d,0x94,0x1e, // Beacon UUID
  0x00,0x00,                                                  // Major
  0x00,0x00,                                                  // Minor
  0xC5                                                        // Measure Power
};
```

Figura 4.4 – Código do *firmware*, parte 1.

Na segunda parte do código (Figura. 4.5), o pacote de *advertisement* é criado, as *flags* são configuradas (a) e o modo de *advertising* é escolhido. O modo ADV_CONNECTABLE_UNDIRECTED (b)foi escolhido por ser o modo em que é possível atingir os menores valores de *advertising Interval*. Depois, os valores de *advertising Interval* e *advertising Timeout* foram escolhidos (c), sendo que o primeiro é o valor que controla a taxa de *advertising*, por isso teve seu valor alterado diversas vezes durante a realização dos testes.

```
ble.init();
// set advertisement
ble.accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE);
ble.accumulateAdvertisingPayload(GapAdvertisingData::MANUFACTURER_SPECIFIC_DATA, beaconPayload, sizeof(beaconPayload));
// set advertise type
//  ADV_CONNECTABLE_UNDIRECTED
//  ADV_CONNECTABLE_DIRECTED
//  ADV_SCANNABLE_UNDIRECTED
//  ADV_NON_CONNECTABLE_UNDIRECTED
ble.setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED);

// ble.setAdvertisingInterval(160);
ble.setAdvertisingInterval(100);
// set adv_timeout, in seconds

ble.setAdvertisingTimeout(0);
// start advertising
ble.startAdvertising();
```

(a) (b) (c)

(a)Configuração das flags, (b)Escolha do modo de *advertising*, (c)

Figura 4.5 – Código do *firmware*, parte 2.

Ao final o código foi compilado e instalado no RedBear BLE Nano 2. Para verificar como o pacote estava sendo gerado, foi utilizado um *sniffer* de *Bluetooth Low Energy* para realizar uma varredura e o resultado está mostrado na Figura. (4.6). Foi possível identificar o modo de *advertisement* escolhido (a), as *flags* habilitadas (b) e o *payload* do beacon.

```
> Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
v Nordic BLE Sniffer
    Board: 3
  > Header, Packet counter: 23730
    Length of packet: 10
  > Flags: 0x01
    Channel: 39
    RSSI (dBm): -34
    Event counter: 0
    Delta time (µs end to start): 649
    [Delta time (µs start to start): 729]
v Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  v Packet Header: 0x2440 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Random)
      .... 0000 = PDU Type: ADV_IND (0x0)
      ...0 .... = RFU: 0
      ..0. .... = Channel Selection Algorithm: #1
      .1.. .... = Tx Address: Random
      0... .... = Reserved: False
      Length: 36
    Advertising Address: f0:a8:c3:89:a8:b2 (f0:a8:c3:89:a8:b2)
  v Advertising Data
    v Flags
        Length: 2
        Type: Flags (0x01)
        000. .... = Reserved: 0x0
        ...0 .... = Simultaneous LE and BR/EDR to Same Device Capable (Host): false (0x0)
        .... 0... = Simultaneous LE and BR/EDR to Same Device Capable (Controller): false (0x0)
        .... .1.. = BR/EDR Not Supported: true (0x1)
        .... ..1. = LE General Discoverable Mode: true (0x1)
        .... ...0 = LE Limited Discoverable Mode: false (0x0)
    v Manufacturer Specific
        Length: 26
        Type: Manufacturer Specific (0xff)
        Company ID: Apple, Inc. (0x004c)
      v Data: 0215713d0000503e4c75ba943148f18d941e00000000c5
        > [Expert Info (Note/Undecoded): Undecoded]
    CRC: 0xfcf2ae
```

(a) (b) (c)

(a) PDU do pacote,(b) Flags, (c)*payload* do Beacon

Figura 4.6 – Captura do pacote gerado.

### 4.2.3 Configuração Física do RedBearBLE Nano 2

Para facilitar o manuseio, o dispositivo foi montado em cima de uma *protoboard*: a alimentação foi realizada por de uma bateria de botão CR-2032, acoplado em a um *holder*. São utilizados dois *jumpers* para ligar a bateria com as entradas de alimentação do RedBearBLE Nano 2. A Figura (4.7) mostra como ficou a aparência final do dispositivo conectado.

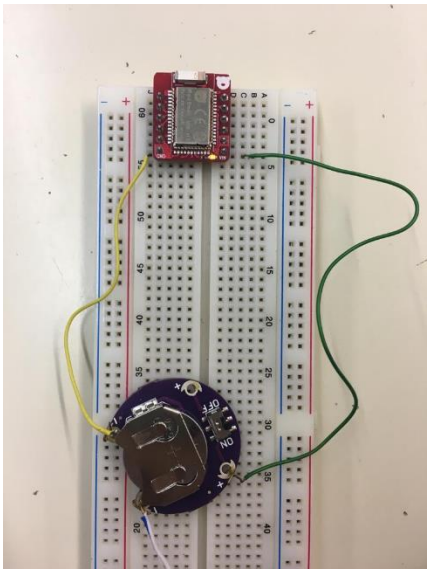


Figura 4.7 – Configuração Física do dispositivo

### 4.2.4 Preparação do ambiente de desenvolvimento para Android

Para desenvolver o aplicativo Android foi escolhida a IDE oficial do Android, a Android Studio IDE. Ela pode ser baixada de forma gratuita no site oficial da Android Studio IDE.

Como os dispositivos nos quais os aplicativos seriam utilizados eram respectivamente o *Moto G2*, o *ZenFone Zoom 3* e o *Samsung Galaxy S6*, e eles possuem o Android na versão 6.0 *Marshmallow*, 7.0 *Nougat* para o *Samsung Galaxy S6* e 8.0 (Oreo) para o *ZenFone*, a API utilizada neste aplicativo foi a API 26 do Android. Essa API foi desenvolvida para o Android Marshmallow, e não conta com vários recursos novos que outras versões da API possuem, como diferenciar de qual canal da camada física o quadro é proveniente.

Existem várias bibliotecas para o *Bluetooth Low Energy* (BLE) para Android, sendo as mais conhecidas: *SweetBlue* e e *AltBeacon*. Porém, apesar das bibliotecas oferecem facilidades na hora do desenvolvimento, elas "escondem" os recursos de mais baixo nivel do BLE no Android. Por isso, foi utilizado a biblioteca padrão do BLE para Android.

A biblioteca padrão do BLE para Android foi introduzida na API 18 do Android, e foi desenvolvida para a descoberta de dispositivos BLE e transferência de pequenas quantidades de dados. Nessa biblioteca existe um objeto que é responsável pelo *scan*, o *BluetoothLeScanner*. Esse objeto se relaciona diretamente com o *hardware* do BLE existente no dispositivo celular e é por meio de dois métodos dele que a varredura BLE é realizada. São eles: *BluetoothLeScanner.startScan(List<ScanFilter> filters, ScanSettings settings, ScanCallback callback)* e *BluetoothLeScanner.stopScan(ScanCallback callback). ("Bluetooth low energy overview | Android Developers", 2018)* Depois que o método *startScan* é utilizado quando houver resultados, uma função é invocada para tratamento das respostas, essa função é chamada de *callback*. Os objetos *ScanFilter* e *ScanSettings* são criados para as configurações da varredura. No primeiro, é possível estipular filtros para a varredura como: endereços MAC, UUIDs, nomes dos dispositivos, entre outros. Já no segundo, é possível especificar as configurações da varredura como o modo da varredura (baixa energia, baixa latência ou balanceado) e o modo de detecção de sinal, que é a configuração que especifica como o *callback* deve ser disparado.

Por mais baixo nível que a biblioteca padrão do BLE possa chegar, não é possível alterar o *scan window* e o *scan interval*, definidos na seção 2.2.2. Existem três possíveis valores para essas variáveis podendo ser alteradas com a troca do modo de varredura. O modo com os menores valores é o *SCAN_MODE_LOW_LATENCY* e o maior *SCAN_MODE_LOW_POWER*, a Tabela (4.1) apresenta os valores que essa variável pode assumir.

Tabela 4.1- Modo de varredura e os parâmetros de *scan*. Fonte[(KWONKI, 2018)]

| Modo de Varredura | *SCAN WINDOW* (ms) | *SCAN INTERVAL* (ms) |
|---|---|---|
| *SCAN_MODE_LOW_POWER* | 500 | 5000 |
| *SCAN_MODE_BALANCED* | 2000 | 5000 |
| *SCAN_MODE_LOW_LATENCY* | 5000 | 5000 |

Dessa maneira, como pode ser observado na seção 2.2.5, pode ser concluído que no *SCAN_MODE_LOW_LATENCY* , o intervalo de *scan(scan interval*) é totalmente composto por uma *scan window.*

### 4.2.5 Desenvolvimento do aplicativo base

Concluída a definição de biblioteca a ser utilizada e do nível da API escolhida, foi criado um novo projeto, e a primeira coisa a ser feita para poder usar o Bluetooth do celular é a definição de permissão no arquivo manifesto do aplicativo. Paralelo a isso, foi adicionada também a permissão de gravação de arquivos. As seguintes linhas foram adicionadas Figura. (4.8)

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Figura 4.8 – Linhas do manifesto

Sendo as duas primeiras necessárias para acessar o rádio do *Bluetooth*, as duas subsequentes para que o *scan* tenha resultado ("Bluetooth low energy overview | Android Developers", 2018) já as outras duas para poder salvar arquivos.

A partir da API 23 do Android, é necessário pedir as permissões em período de execução[fonte permissão]. Logo foram criadas duas funções que fazem esse papel: a *pedirBluetooth()*(Figura (4.9)) e a *pedirArquivo()(*Figura(4.10))

```
public void pedirBluetooth(){

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        // Android M Permission check
        if (this.checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
            builder.setTitle("This app needs location access");
            builder.setMessage("Please grant location access so this app can detect beacons in the background.");
            builder.setPositiveButton(android.R.string.ok,  listener: null);
            builder.setOnDismissListener((dialog) -> {
                    requestPermissions(new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                            PERMISSION_REQUEST_COARSE_LOCATION);
            });
            builder.show();
        }
    }
}
```

Figura 4.9 – Função pedirBluetooth()

```
void pedirArquivo(){

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            // Android M Permission check
            if (this.checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
                final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
                builder.setTitle("This app needs location access");
                builder.setMessage("Please grant location access so this app can detect beacons in the background.");
                builder.setPositiveButton(android.R.string.ok,  listener: null);
                builder.setOnDismissListener((dialog) -> {
                        requestPermissions(new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                                PERMISSION_REQUEST_COARSE_LOCATION);
                });
                builder.show();
            }
        }

}
```

Figura 4.10- Função pedirArquivo().

Fonte permissão https://developer.android.com/training/permissions/requesting?hl=pt-br

Após isso, foram criados os objetos que realizam a interface do rádio do Bluetooth com o *software* (Figura 4.11)

```
BluetoothManager bluetoothManager;
BluetoothAdapter mBluetoothAdapter;
BluetoothLeScanner mBluetoothLeScanner;

bluetoothManager = (BluetoothManager) getSystemService(BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();
mBluetoothLeScanner = mBluetoothAdapter.getBluetoothLeScanner();
```

Figura 4.11- Criação dos objetos do Bluetooth

Finalmente, foi possível chamar a função que realiza o *scan*, como citado em 4.2.3, a função *BluetoothLeScanner.startScan(List<ScanFilter> filters, ScanSettings settings, ScanCallback callback)*, recebe 3 parâmetros de entrada. Para o aplicativo utilizado foi criado um filtro com o endereço MAC do dispositivo BLE (Figura (4.11a), dessa forma a aplicação ignora pacotes vindo de outro dispositivo BLE Já para o modo de varredura, foi escolhido o modo que maximiza a detecção de pacotes, como descrito na seção 2.2.7, o *SCAN*_MODE_LOW_LATENCY(Figura (4.12c). Como o foco da aplicação é a detecção dos pacotes de *advertisement* enviados pelo dispositivo, o modo de detecção foi escolhido foi o Match_Mode_Aggresive(Figura (4.12d), que acusa o recebimento de pacotes mesmo com sinais mais fracos. Para a configuração de disparo do *callback* foi escolhido o modo CALLBACK_TYPE_ALL_MATCHES (Figura (4.12b), que dispara o *callback* com qualquer correspondência.

```
ScanFilter macAddres = new ScanFilter.Builder().setDeviceAddress("F0:A6:C3:89:A8:B2").build();  (a)
final List<ScanFilter> filters = new ArrayList<>();
filters.add(macAddres);

final ScanSettings settings = new ScanSettings.Builder()
        .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES) (b)
        .setReportDelay(0).setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)  (c)
        .setMatchMode(ScanSettings.MATCH_MODE_AGGRESSIVE). (d)
                build();
mBluetoothLeScanner.startScan(filters, settings, callback);
```

Figura 4.12 – Configuração do *scan* no Android.

O *callback* criado para receber as requisições guarda o valor de RSSI do pacote recebido e incrementar um contador.

Para parar o *scan* no tempo desejado, o método *BluetoothLeScanner.stopScan(ScanCallback callback)*, é colocado dentro de um novo thread que dispara essa função após um tempo de janela estipulado. Ao fim do *scan* é executado algum método de acordo com o teste a ser realizado.

Adicionalmente foram criados alguns botões para iniciar o *scan*, estipular o tempo de *scan*, para limpar os resultados e salvar os resultados. A interface do aplicativo criado pode ser visto na Figura. (4.12).

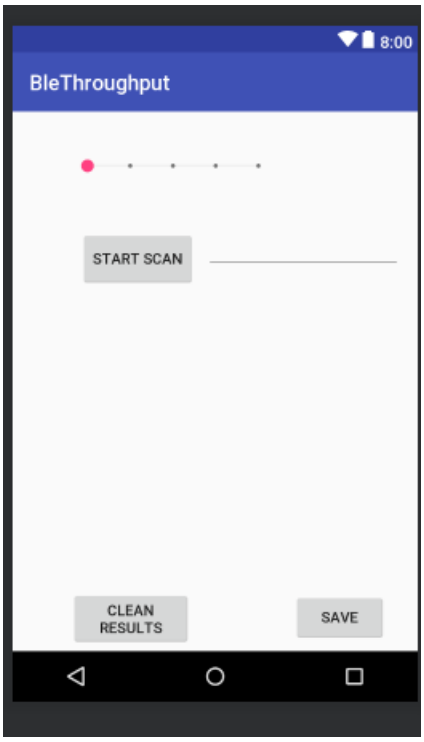



Figura 4.12 - Interface do aplicativo.

## 4.3 TESTES PARA DEFINIÇÃO DOS PARAMETROS

O objetivo dessa etapa é a definição dos parâmetros que influenciam em um *scan*, como frequência de *advertisement* do dispositivo BLE, janela de *scan* do celular, *scan window* e a definição da versão do Bluetooth que será usado. O ambiente utilizado para realização de todos os testes foi *outdoor* sem nenhuma rede Wi-Fi de 2.4GHz por perto, nem nenhum outro dispositivo Bluetooth além dos que compõe os testes.

### 4.3.1 Métrica dos testes

- Janela de *scan*

A janela de *scan* definida nesse trabalho é o tempo entre o início do *scan* e do final do mesmo. A diferença de tempos entre as chamadas dos métodos *BluetoothLeScanner.startScan()* e o *BluetoothLeScanner.stopScan().* Esse tempo é diferente da *scan window* definido na seção 2.2.2, conforme citado na seção 4.2.4 esse tempo é fixado.

- Frequência (taxa) de *advertisement*

O valor da frequência de *advertisement* é o inverso de um *Advertisement Event*. Conforme visto na seção 2.2.2, um *Advertising Event* é composto por um *AdvInterva*l acrescido de um *AdvDelay*. Como o *AdvDelay* é um valor aleatório que pode variar entre 0 e 10 ms, neste trabalho, a frequência de *advertisement* será apenas o inverso do *Advertisement Interval* (*AdvInterv*al), tornando-a uma aproximação.

### 4.3.2 Teste Geral

O primeiro teste realizado foi para verificar o comportamento do RSSI do *Bluetooth Low Energy*. Nesse teste foi variado a distância para observar o comportamento do RSSI em três modelos de

celulares diferentes: Um *Moto G 2nd Gen*, um *Samsung Galaxy S6* e um *Asus Zenfone Zoom 3*, cada celular possui uma versão diferente de Bluetooth, conforme explicitado nas seções 2.4, 2.5 e 2.6. Para esse teste, ao fim do *scan* o aplicativo calculava a média e o desvio padrão e salvava esses resultados para exportação.

As distâncias entre os celulares e o RedBear Nano 2 utilizadas para o teste começaram em 10 centímetros e atingiram 1 metro, variando de 10 centímetros em 10 centímetros. Essas distâncias eram marcadas num tecido, que servia de referência. Foi considerada uma distância de até apenas 1 metro com base nos estudos de Faragher e Harle(2015) que comprovam que a acurácia para distâncias maiores que aproximadamente 2 metros cai drasticamente, ou seja: a probabilidade de várias distâncias assumirem um mesmo valor RSSI aumenta.

Também se levou em consideração uma variável que poderia influenciar diretamente no resultado: o alinhamento das antenas. Por isso, estudou-se como funcionavam as antenas dos celulares Android presentes no mercado, incluindo os que estavam à disposição para a implementação do projeto: Dois celulares *Motorola Moto G 2nd Gen*, um *Samsung Galaxy S6* e um *Asus Zenfone Zoom 3*. Do estudo, foi apontado que a própria fórmula utilizada para se relacionar o RSSI com a distância (equação 9 da introdução teórica) mostra que "n" e "A" são parâmetros que dependem do meio, e por isso, dependem também do ângulo das antenas. Como afirmado por Polese *et all*(2014), uma pequena alteração no ângulo entre as antenas acarretaria em enormes variações de valores RSSI e, portanto, influenciaria bastante o cálculo da distância. Por se tratar de uma antena omnidirecional, e para ter o maior alcance possível, a posição escolhida do celular para se realizar os testes seguintes foi a vertical, com as costas viradas para o BLE e ao nível da superfície do solo, conforme pode ser visto na Figura. (4.13).

Figura 4.13- Foto do ambiente de teste.

Para este primeiro teste, fixou-se o tempo da janela de *scan* e taxas de *advertisement* em 30 segundos e 50 Hz, respectivamente. Foram escolhidos esses valores para que, após serem analisados os resultados, sejam alterados de acordo com a finalidade do projeto.

A superfície utilizada para apoiar o tecido com a referência em distância foi o asfalto, e cada celular foi testado alternadamente na posição marcada. Tendo assimilado essas informações, o celular e o BLE foram fixados na posição correta.

Para a análise dos resultados do RSSI, foram desconsiderados os primeiros pacotes recebidos, visto que esses pacotes foram capturados durante o alinhamento do celular com o RedBear BLE Nano 2. Os resultados estão apresentados abaixo:

Os valores de RSSI obtidos no teste, para os três celulares estão apresentados nas Figuras, (4.14, 4.15 e 4.16):

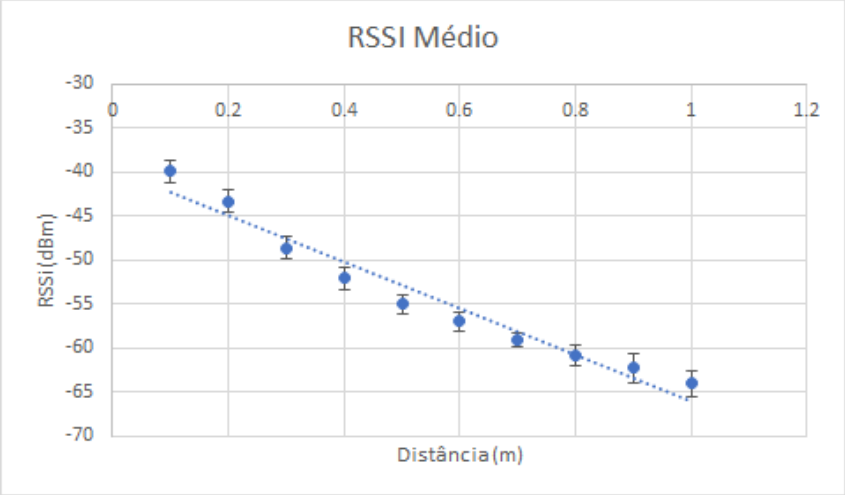



Figura 4.14-Valores de RSSI obtidos pelo *Asus Zenfone. Zoom 3*.

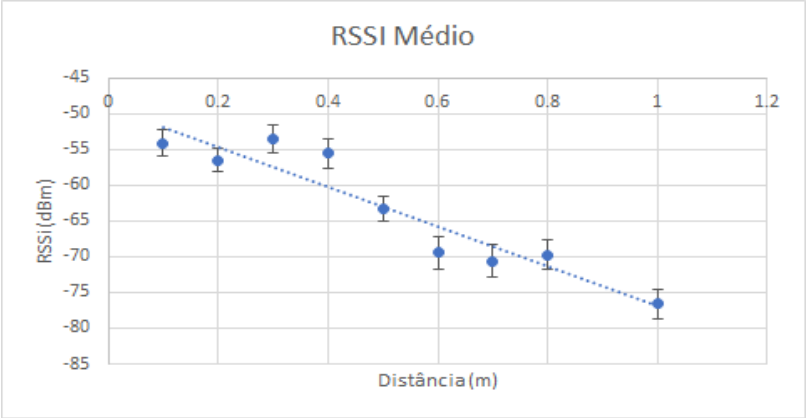



Figura 4.15-Valores RSSI obtidos pelo *Moto G2*.

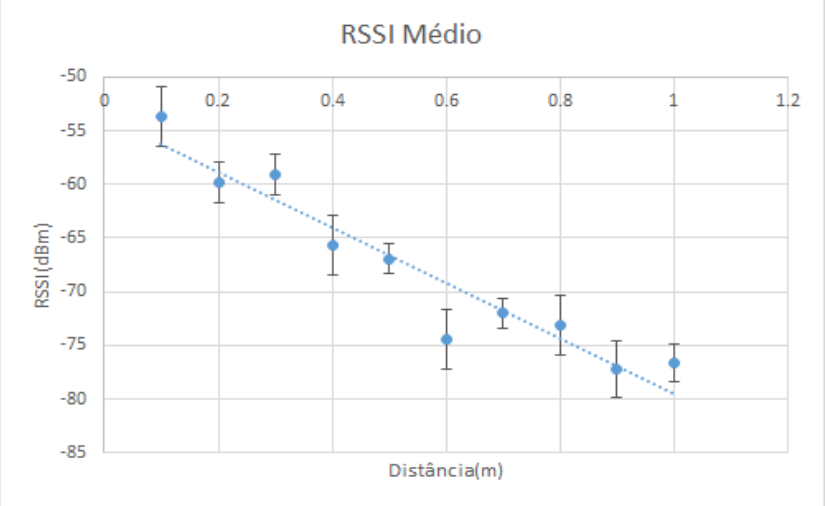



Figura 4.16 - Valores do RSSI obtidos pelo Samsung Galaxy S6.

Os gráficos dos valores de RSSI, obtidos nesse teste, apresentam uma relação de proporção inversa entre o RSSI e a distância: quanto maior a distância menor a potência de sinal recebida. Essa relação pode ser observada pela linha de tendência presente nas três figuras. Essa relação já era esperada, visto que ela pode ser modelada pela Eq. (9), na qual a distância é uma variável.

Os maiores valores de RSSI foram obtidos pelo *Zenfone Zoom 3*, juntando isso ao fato de que a amplitude dos desvios padrão obtido por esse celular foi menor do que nos outros dois (Anexo I), pode-se concluir que a antena de recepção desse celular apresenta o melhor ganho entre eles. Além de apresentar o maior ganho entre as antenas, o *Zenfone Zoom 3* possui a versão 4.2 do Bluetooth, a mais nova entre todos os celulares testados. Essa versão apresenta diversas vantagens em relações as outras, conforme relatado em 2.4.2. Por apresentar os melhores resultados, optou-se por dar prosseguimento no trabalho utilizando apenas esse celular.

### 4.3.3 Teste para definição do tempo de janela de *scan* e taxa de *advertisement*

Tendo em vista o comportamento geral do BLE, foi realizado um teste a fim de determinar a melhor combinação entre tempo de *scan* do celular e a taxa de *advertisement* do RedBear BLE Nano 2, independentemente da distância. Conforme afirmado por Faragher e Harle (2015), a taxa de *advertisement* influencia na definição de tempo da janela de *scan*, pois uma boa janela de *scan* deve receber ao menos um pacote em cada um dos 3 canais de *advertising*, além de ter que levar em conta a mobilidade do que deseja encontrar, quanto mais mobilidade mais rápido tem de ser a taxa de *advertising*.

Para os valores da janela de *scan* foram definidos os seguintes valores: 5 segundos, 15 segundos e 30 segundos. Já para a taxa de *advertisement*, foram escolhidos: 10 Hz, 30 Hz e 50 Hz. Todas essas combinações entre janela de *scan* e frequência de *advertisement* foram testadas as seguintes distâncias: 30 cm, 60 cm, 1 m, 2,5 m, 5 m e 10 m. Dessa maneira pode-se verificar o comportamento do RSSI em todas combinações. As Figuras (4.17), (4.18) e (4.19) apresentam os resultados obtidos nesse teste

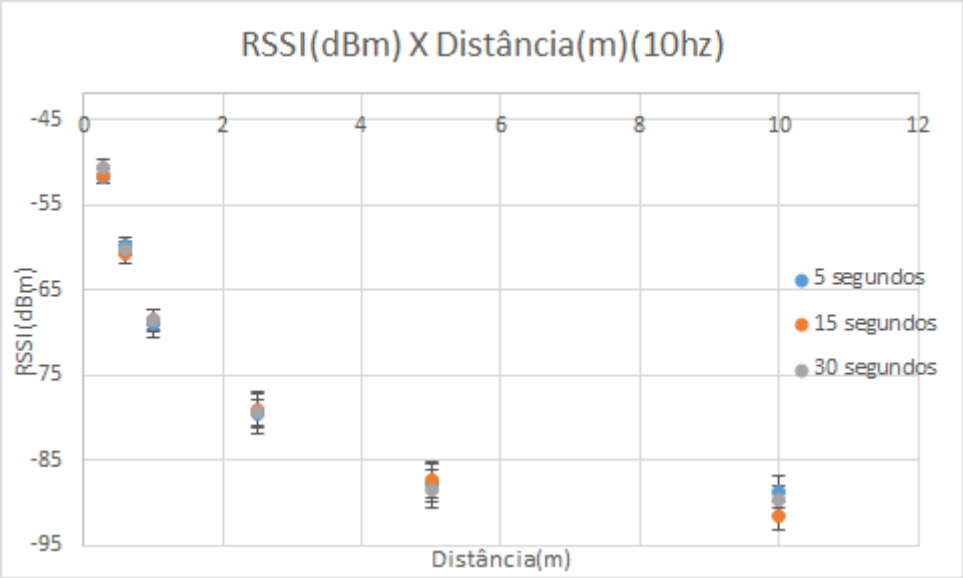



Figura 4.17 - RSSI x Distância (Frequência de *advertisement*:10 Hz).

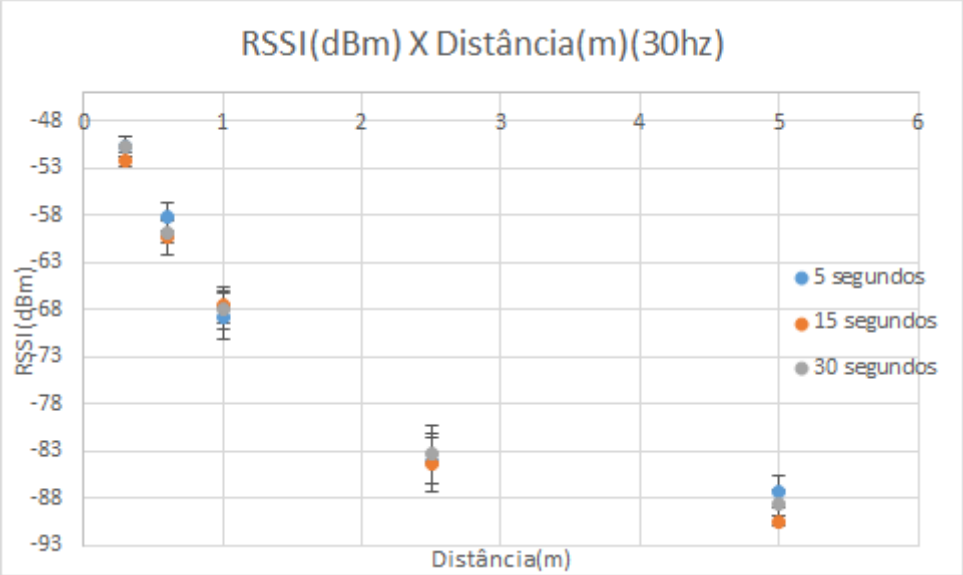



Figura 4.18 - RSSI X Distância, frequência de *advertisement:* 30 Hz*).*

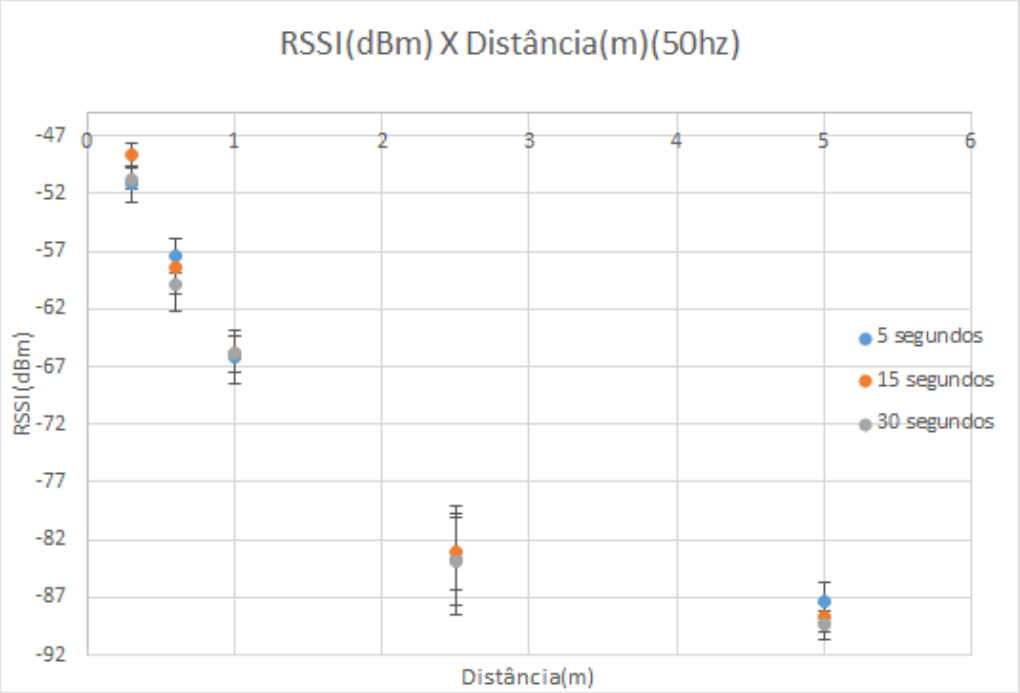



Figura 4.19 - RSSI X Distância, frequência de *advertisement:* 50 Hz*).*

A médias dos valores de RSSI ficaram muito próximas umas das outras em todos os casos. Já os desvios padrões foram crescendo com o aumento da frequência, o que leva a crer que, nessa situação, o aumento dos números de pacotes que chegam ao celular implica em mais imprecisão no modelo, nessa situação.

Outro comportamento que pode ser observado foi que apenas com a frequência de *advertisement* de 10 Hz (Figura. 4.17) pode ser recebido pacotes a 10 metros. Comparando os valores de RSSI com a distância para as 3 frequências (Figura., 4.20), percebe-se que os valores obtidos a 10 Hz sofrem menos perda com o aumento da distância nesse caso especifico.

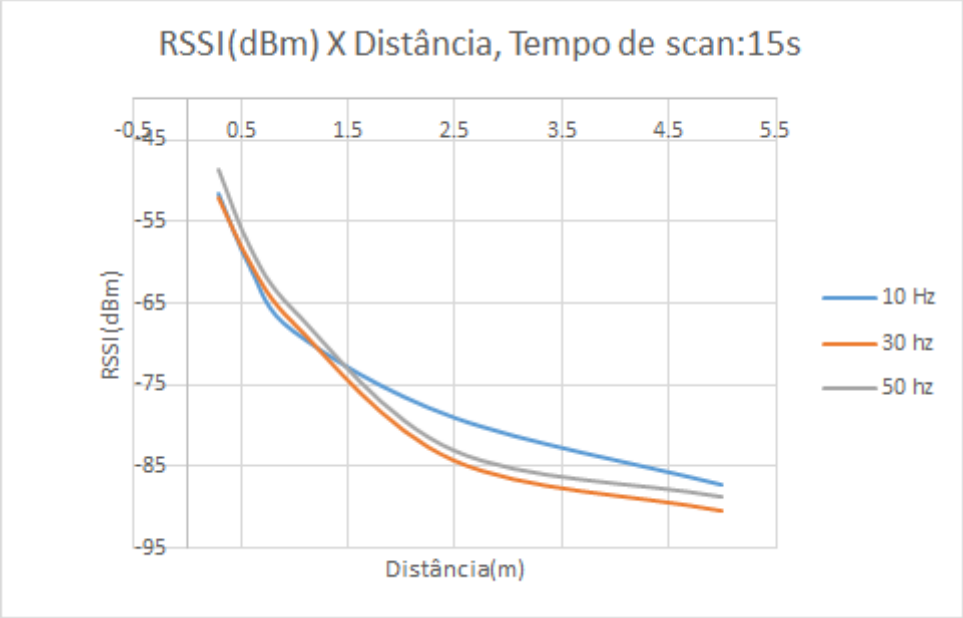



Figura 4.20 - Comportamento do RSSI a diferentes taxas de *advertising.*

A fim de encontrar qual melhor combinação entre tempo de *advertisement* e tempo de *scan*, foi calculada a amplitude dos desvios padrão da medida do RSSI em todas as distâncias. O resultado desse cálculo pode ser observado no Anexo I. A melhor combinação nesse caso foi 10 Hz e 15 segundos, como deseja-se ter a maior precisão possível e o dispositivo não se move, esses valores serão utilizados ao longo do trabalho.

## 4.4 DETERMINAÇÃO DA TÉCNICA

Após ajustados os parâmetros que melhoraram a recepção e a distância de alcance, foram realizados testes a fim de escolher a técnica de localização que melhor se encaixa na situação.

### 4.4.1 Teste de *fingerprinting*

Foi realizado um teste preliminar de um esquema de *fingerprintg* com apenas um celular. Um sistema de *fingerprinting* necessita de um mapeamento preliminar, a chamada fase *offline*, e depois uma fase *online* onde os resultados são comparados com o do mapeamento. Então para simular esse modelo foram marcados 9 pontos num tecido, com a separação de 30 centímetros entre eles em todas as direções. Essa distância foi escolhida por ser o erro aceitável nesse projeto. O celular foi posicionado de forma ficar a 30 cm do começo do *grid.* A Figura (4.21) demonstra como ficou à disposição dos pontos e a localização do celular.

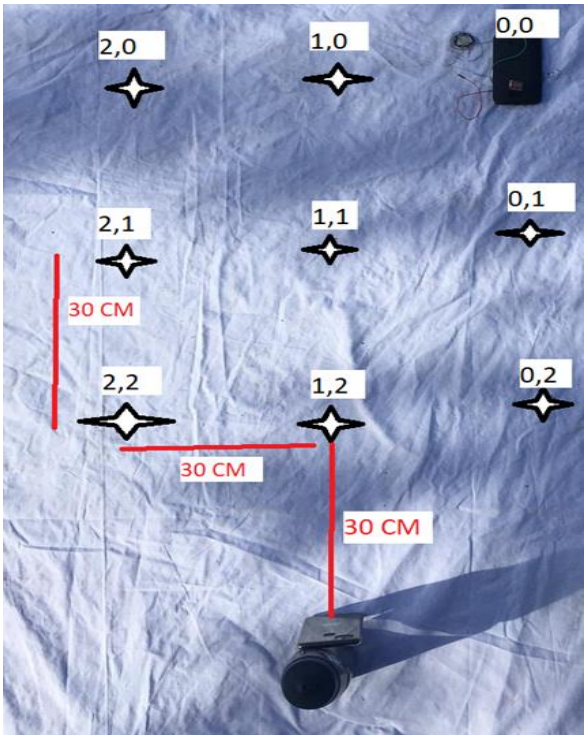



Figura 4.21- *Grid* do teste.

Todo o processamento foi realizado após a realização dos testes, de forma *offline*, para ter mais controle dos resultados. O filtro utilizado para a agregação dos valores *raw* do RSSI foi a média simples. A Tabela (4.2) mostra os valores em cada posição do *grid*

Tabela 4.2 – Valores do Grid mapeado

| x,y | 0 | 1 | 2 |
|---|---|---|---|
| 0 | -72.18 | -61.4 | -60.31 |
| 1 | -59.18 | -55.33 | -57.42 |
| 2 | -55.00 | -46.99 | -53.96 |

Após o mapeamento, foi realizada uma simulação da fase *online*. O dispositivo *BLE* foi colocado em todos os pontos e os valores RSSI foram coletados novamente. Depois disso, os valores foram comparados com o valor do mapeamento, a posição que apresentasse o valor mais próximo seria a posição estimada do dispositivo. As distâncias lineares entre os pontos do grid e o celular foram calculados usando a fórmula de distância euclidiana (nome da forma do triângulo equilátero). Os resultados estão apresentados no Anexo III, e graficamente na Figura. 4.22, na quais as posições circuladas de verde são as posições em que foi acertada:

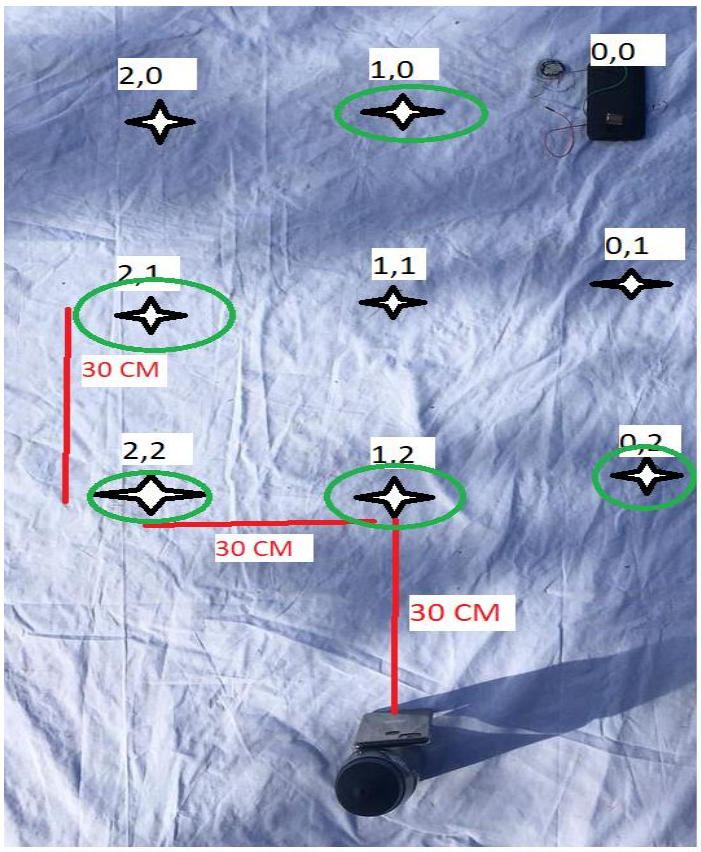



Figura 4.22- Resultado do mapeamento.

Com essa configuração, foi possível acertar 5 dos 9 testes, o que corresponde a 55% de acertos. Os acertos ficaram concentrados na primeira "linha"[x=2],onde todos os pontos foram acertados. A distância entre o celular e os pontos do grid que essa linha se encontram é de 30 centímetros para o celular do centro e de 42,42 centímetros para os outros dois pontos daquela linha. O ponto mais longe que foi localizado foi o [1,0], que se encontra a 90 centímetros do celular. Vale ressaltar que o ponto [1,0] se encontra em linha reta do celular, na "coluna" [y=1]. O ponto mais distante localizado corretamente que não se encontra na coluna [y=1] foi o ponto [2,1], que se encontra do lado esquerdo do celular.

Os valores médios do RSSI mapeados do lado esquerdo, [y=2], apresentaram valores maiores que os valores mapeados do lado direito, [y=0]. Além disso, o erro para o ponto [2,0], que foi o único ponto não encontrado do lado esquerdo, foi pequeno, de apenas 2,61dB. Como já supracitado, as posições das antenas influenciam no resultado do RSSI. Assim, podemos constatar que provavelmente a polarização da antena do celular utilizado no teste se encontra mais ao lado esquerdo do celular, o que faz com que os valores RSSI dos pontos neste lado sejam maiores.

Os resultados desse teste não foram satisfatórios, 55 % de acerto é um número muito baixo para um esquema de localização em ambientes *outdoor*, onde não há interferências significativas. Essa precisão baixa se deve pela utilização de apenas um celular, esquemas de *fingerprinting* necessitam de mais de um celular, ou de mais *beacons*.

Apesar disso, o princípio do *fingerprinting*, de realizar um mapeamento prévio pode ser reaproveitado como uma forma de se ter uma noção de localização visto que valores da mesma linha apresentaram valores próximos entre eles.

### 4.2.2 Teste de abertura

Com a intenção de verificar o comportamento do RSSI em relação a distância e ângulo, foi realizado o teste a seguir.

Primeiramente, para identificar o lado do RedBear BLE Nano 2 que apresenta o RSSI menor, foram realizadas capturas em todos os sentidos do dispositivo. O RedBear foi posicionado no centro de um círculo com raio de 20 cm e foram realizadas medidas nos 4 pontos cardeais. A Figura (4.23) mostra a disposição dos pontos em relação ao dispositivo

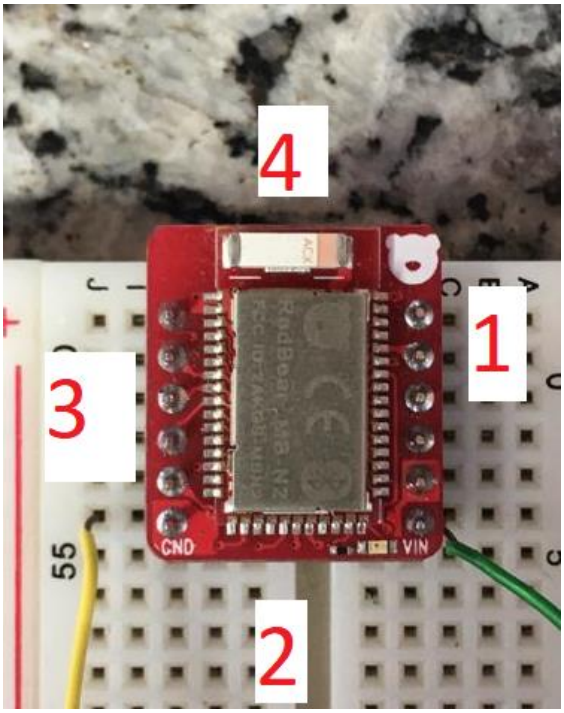



Figura 4.23- RedBear e as posições.

O celular foi posicionado em todos os pontos marcados com a parte de trás (câmera) apontado para o RedBear. Os resultados obtidos estão apresentados na Tabela (4.3).

Tabela 4.3 - Valores Médios do RSSI em cada posição

| Posição | RSSI médio(dBm) |
|---|---|
| 1 | -46,95 |
| 2 | -63,97 |
| 3 | -40,32 |
| 4 | -52,94 |

Os menores valores foram obtidos nas posições 3 e 1, e os maiores valores foram encontrados nas posições 2 e 4, como era de se esperar visto o diagrama de potência 3D apresentado na Figura (4.24).

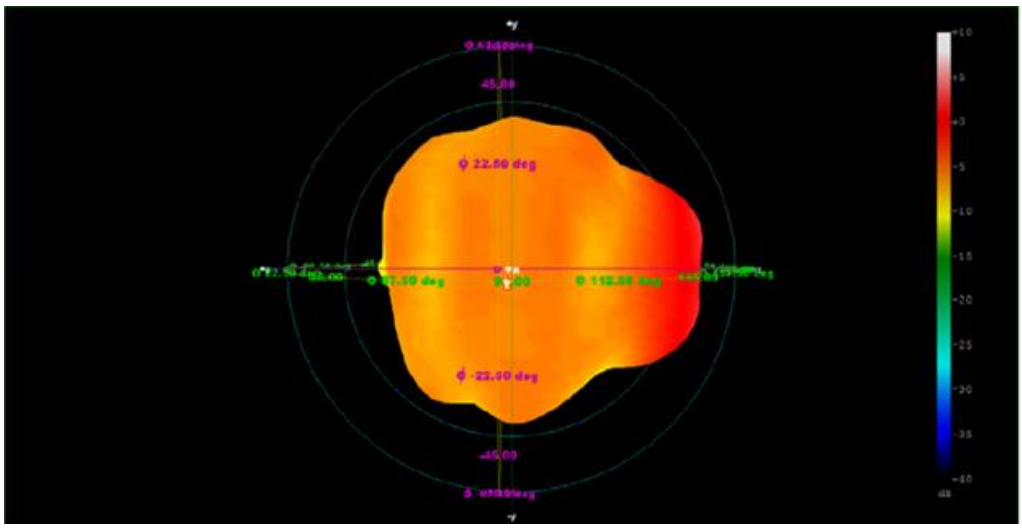


Figura 4.24- Diagrama de radiação 3D. Fonte: ("redbear/nRF5x", 2018)

Como o diagrama apresenta a maior potência é encontrada na posição correspondente a posição 3 do nosso modelo. As pontas apresentam menor potência assim como encontrado no nosso experimento. O lado com maior potência irradiada foi utilizado como referência para os próximos testes

Após a definição do melhor lado do *RedBear BLE Nano 2*, foi realizado um teste para verificar a posição em que o celular recebe maiores valores de RSSI. Para isso, o celular foi posicionado no centro de um círculo com raio de 20 cm, e em outro de 1 metro. O *RedBear* foi posicionado no perímetro dos círculos, e variando de 45 em 45 graus, para poder ter 8 posições equidistantes. Para ficar mais intuitivo, foram atribuídos números para as posições. A Figura (4.25) mostra como ficou a numeração de cada posição. O celular estava com a traseira (câmera) apontada para a posição numerada com o 7.

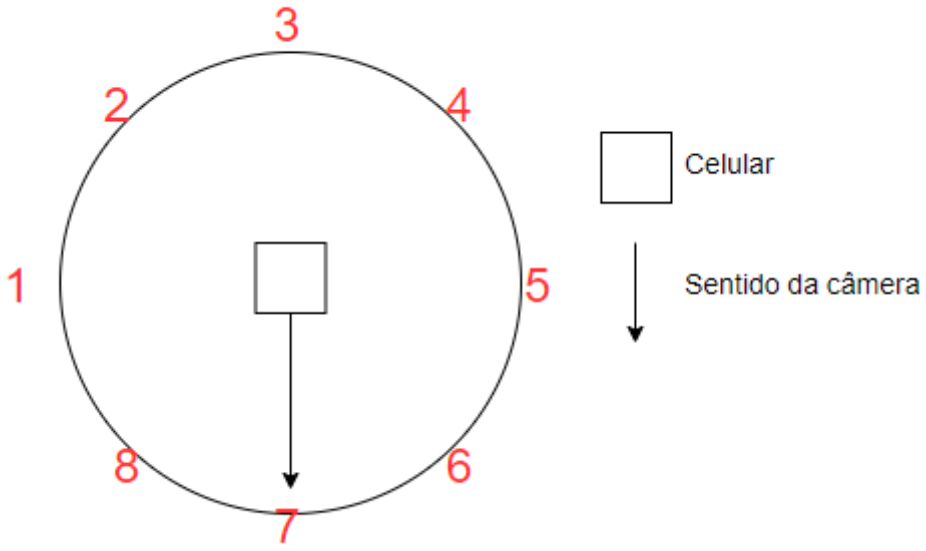



Figura 4.25 – Posições relativas ao celular.

Tabela 4.4- Resultado do teste do lado do celular

| Posição | 1 m | Desvio | 20 cm | Desvio |
|---|---|---|---|---|
| 1 | -71,75 | 2,70 | -49,29 | 0,89 |
| 2 | -76,76 | 1,95 | -57,43 | 1,71 |
| 3 | -62,63 | 0,99 | -48,14 | 1,88 |
| 4 | -64,27 | 1,26 | -46,18 | 1,32 |
| 5 | -58,77 | 0,61 | -48,44 | 1,18 |
| 6 | -60,08 | 0,77 | -44,91 | 0,76 |
| 7 | -62,51 | 0,59 | -47,96 | 1,49 |
| 8 | -77,40 | 5,75 | -48,44 | 1,18 |

Os resultados obtidos nas duas distâncias apresentaram semelhanças, as posições com maiores valores de RSSI estão sempre à esquerda do celular, e as menores sempre a direita. Vale ressaltar que todos os pontos estão à mesma distância do celular e os resultados apresentaram grandes diferenças, o que confirma a teoria de que a posição das antenas tem influência direta com o RSSI, e que a antena do celular tem mais potência para o lado esquerdo do mesmo.

As diferenças entre posições “vizinhas” não foram tão grandes. No caso dos 20 centímetros, a maior diferença ficou entre as posições 1 e 2 que foi de aproximadamente 8 dB, as duas posições estão localizadas a direita do celular. Já para um metro a maior ficou entre a posição 7 e 8, na qual a diferença ficou em 14,89 dB.

De maneira geral, é possível perceber que os resultados localizados ao lado esquerdo do celular apresentam valores mais confiáveis, isto é, estão mais próximos uns dos outros e os seus valores tem maior magnitude. Dessa forma, conclui-se que é melhor limitar o ângulo de operação da possível aplicação a um ângulo de 180 (iniciando na posição 7 e terminando na posição 3 seguindo o sentido anti-horário).

Outro comportamento dos valores do RSSI que podem ser constatado nesse teste, é que não há como descobrir a direção em que o dispositivo se encontra apenas com o valor do RSSI. Conforme pode ser constatado na Tab. (4.8), onde o valor da posição 2 a 20 cm é de -57,43 dBm, e o valor da posição 5 a 1 metro é de -58,77 dBm, a diferença entre os dois valores é menor que 1 db, porém a diferença entre as distâncias é de mais de 80 cm.

### 4.2.3 Técnica escolhida

Após os dados coletados através dos testes acima, decidiu-se tratar de proximidade em vez de localização do dispositivo BLE. Essa proximidade estaria relacionada a intervalos de distância com valores RSSI previamente medidos (*Fingerprinting*) de modo radial. Em seguida, uma nova captura seria feita para ser comparada aos valores RSSI previamente medidos, e assim classificar o dispositivo num certo intervalo. Isso implicaria que os valores RSSI das posições assumidas como mais confiáveis, numa mesma distância, deveriam gerar um resultado de proximidade igual.

### 4.2.4 Teste de eficiência da técnica escolhida

Com os resultados do teste anterior e, considerando apenas as posições entre 4 e 7, todos os valores obtidos foram acrescidos ou subtraídos do desvio padrão, de maneira que o menor de todos os

valores fosse definido como limite inferior para aquela posição e o maior de todos os valores como o limite superior para aquela posição.

A fim de testar esses limites estabelecidos, após cada coleta o valor do desvio padrão era acrescido no limite inferior (linf) ou subtraído no limite superior (lsup) do valor da média, formando um "intervalo". Esse intervalo era comparado com os limites definidos anteriormente, e de acordo com a comparação dos valores, a localização era dada. A Tabela (4.5) mostra as posições e os valores limite (*thresholds*), mostrados em módulo para facilitar o entendimento, visto que o valor RSSI em dB é dado em números negativos. Assim, quanto menor o valor do limite, menor a distância esperada.

Tabela 4.5- Limites de cada posição

| Localização | Limites (em módulo) |
|---|---|
| Menor que 20 cm | lsup<=44,15 |
| Perto de 20 cm (por baixo) | lsup>=44,15 e linf<44,15 |
| Com certeza 20 cm | lsup<=49,6 e linf>=44,15 |
| Perto de 20 cm (por cima) | lsup>49,6 e linf<=49,6 |
| Entre 20 cm e 1 metro | lsup>49,6 e linf>49,6 |
| Perto de 1 metro (por baixo) | lsup>=58,15 e linf<58,15 |
| Com certeza 1 metro | lsup<=65,.53 e linf>58,15 |
| Perto de 1 metro (por cima) | lsup>65,53 e linf<58,15 |
| Maior que 1 metro | lsup>65,53 e linf>65,53 |

Foram realizados testes a 10 centímetros, 20 centímetros, 60 centímetros, 80 centímetros, 1 metro e 20 centímetros e 5 metros. Os resultados, para cada posição, estão apresentados no Anexo III.

Para esse teste, foram considerados errados os resultados em que a localização não condiz com a localização estimada. Por exemplo, para 10 centímetros, os resultados que o aplicativo deveria apresentar seria "Menor que 20 centímetros", para 20 centímetros "Com certeza 20 cm", para 60 centímetros "Entre 20 cm e 1 metro", para 80 cm "Perto de 1 metro pra baixo", para 100 cm "Com certeza 1 metro" e para 120cm "Perto de 1 metro pra cima" e para 5 metros "Maior que 1 metro" r. Dessa forma, foram obtidos 8 acertos de 28 resultados.

Os resultados mostram que essa categorização utilizando "Perto de X por cima" ou "Perto de X por baixo" e "Com certeza X" é muito imprecisa, não sendo possível obter tamanha precisão usando apenas um celular. Porém, é possível estabelecer limites menos granulares, de forma com que gere resultados mais acurados.

Então, da mesma maneira que foi proposta uma nova delimitação de limites, considerando apenas se a distância em que o dispositivo se encontra é menor do que as delimitadas: 20 centímetros e 1 metro, para esse teste. Os valores, em módulo, que definem os limites são: 49,61dBm para 20 centímetros e 65,53 dBm para 1 metro. Assim, a Tabela 4.6 foi criada.

Tabela 4.6- Novos limites

| Localização | Limite (em módulo) |
|---|---|
| No raio de 20 cm | <49,61 |
| No raio de 1 m | >49,61 e <65,53 |
| No raio de mais de 1 m | >65,53 |

Aplicando esses limites, foram obtidos os resultados demonstrados no Anexo III.Com os novos limites definidos, os erros foram reduzidos de 20 para 6, comprovando a eficácia dessa maneira menos granular de caracterizar as posições.

O mesmo teste foi realizado novamente, porém, com diferentes limites. Os limites das novas posições foram calculados da mesma maneira: foram coletados previamente os valores de RSSI das novas distâncias nas quatro melhores posições (4,5,6,7), que em seguida foram acrescidos ou subtraídos do desvio padrão. O maior valor à cada distância se tornou o limite. Dessa maneira, obteve-se os limites apresentados na Tab. (4.7).

Tabela 4.7- Limites alterados

| Localização | Limite (módulo) |
|---|---|
| Menor que 1 metro | <65,53 |
| Entre 1 metro e 2 metros | >65,53 e <78,84 |
| Entre 2 metros e 5 metros | >78,84 e <91,95 |
| Entre 5 metros e 10 metros | >91,95 e <94,93 |
| Maior que 10 metros | >94,93 |

Com esses novos limites, foram realizados testes a 50 centímetros, 150 centímetros, 350 centímetros, 750 centímetros e 1250 centímetros, de modo com que cada limite fosse testado. Além disso, as distâncias foram escolhidas para serem equidistantes entre dois limites. Por exemplo, o ponto com distância de 150 centímetros fica a 50 centímetros de 1 metro e a 50 centímetros de 2 metros. Os resultados obtidos neste teste estão apresentados no Anexo IV.

O aplicativo errou em apenas um caso, mostrando- se bastante acurado para essa situação. Dessa maneira, conclui-se que ao se utilizar apenas um celular com *Bluetooth Low Energy* na versão 4.2 em ambientes *outdoor*, a experiência de proximidade numa distância de aproximadamente 10 metros é bastante efetiva. Para isso, é necessário que haja um mapeamento prévio para definição de limites (*fingerprinting*).

# 5 RESULTADOS

Como a localização exata é uma tarefa complicada para se determinar apenas com um celular, o aplicativo desenvolvido guia a pessoa até o dispositivo. O aplicativo analisando o RSSI recebido, consegue mostrar direções até chegar ao dispositivo. O nome do aplicativo desenvolvido é *BLEGuide*, uma alusão ao *Bluetooth Low Energy* e ao termo em inglês *guide*, de guia no português.

## 5.1- APLICATIVO FINAL

### 5.1.1 – Descrição do aplicativo

Para a criação do aplicativo final, todas as funções apresentadas na seção 4.2.5, bem como o manifesto foram reaproveitados. Algumas premissas tiveram que ser realizadas para a criação do aplicativo. São elas:

- A parte traseira do celular (câmera) aponta sempre para a direção do deslocamento, isto é, o celular não é movimentado para os lados.
- Em um raio de menos de um metro, o dispositivo já é visível e pode ser achado.

Com isso, a lógica do aplicativo foi desenvolvida para que quando o dispositivo se encontrar a 1 metro ele acuse que foi localizado, e quando o aplicativo mande mudar de posição o celular mude junto.

A base do funcionamento do aplicativo é que quando o valor do RSSI aumente, o dispositivo se encontra mais perto. Como não foi possível estabelecer uma relação entre o RSSI e a direção conforme relatado na seção 4.2.2, para encontrar a direção na qual o dispositivo se encontra foi desenvolvido um algoritmo que caso ao se mover o valor do RSSI diminua, é solicitado que se mude de direção até que se encontre uma posição na qual o valor do RSSI volte a aumentar.

Ao realizar o primeiro *scan*, o valor do RSSI obtido é guardado como referência (RSSIref). O aplicativo, então, imprime uma estimativa de distância de onde o dispositivo se encontra, e solicita a pessoa que dê um passo para frente. Ao chegar na nova posição, o *scan* deverá ser realizado novamente e caso o valor do RSSI (RSSIn), for maior que o valor de referência, o aplicativo manda a pessoa seguir nessa mesma direção. Caso o valor do RSSI for menor que o valor de referência, o aplicativo manda a pessoa voltar para a referência (Pref) e dar um passo em outra direção. Esse processo é repetido por no máximo quatro vezes, para contemplar as quatros direções (norte, sul, leste e oeste). Se em nenhuma das direções o valor for maior que a referência, o aplicativo manda a pessoa seguir a posição em que o RSSI(RSSIdec) foi o maior. Esse pequeno algoritmo vai sendo repetido até que o dispositivo se encontre a menos de 1 metro do celular. Toda essa lógica está mostrada na Figura. (5.1).

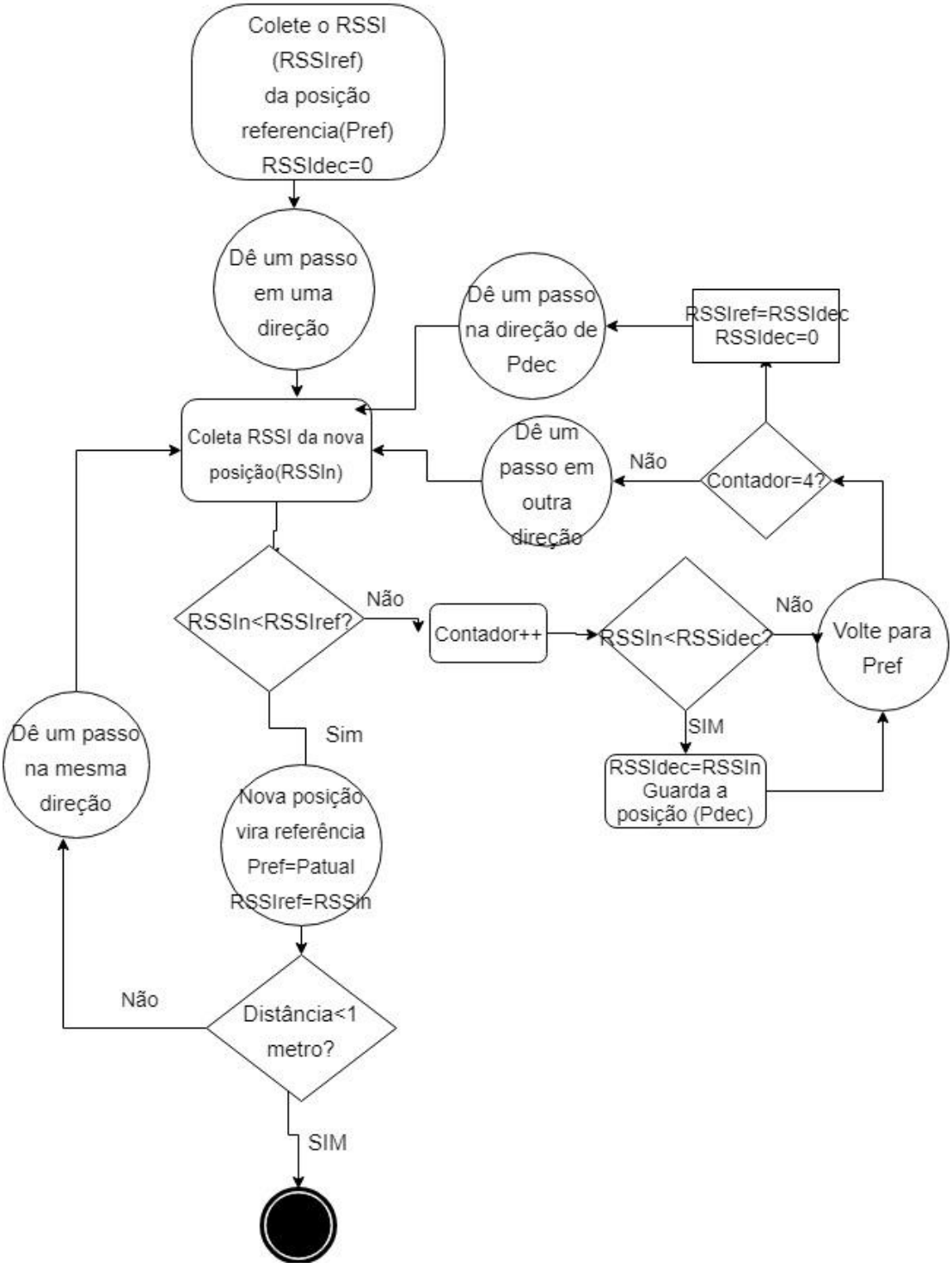



Figura 5.1 – Fluxograma do aplicativo criado.

O *Bluetooth Low Energy* possui limitação de alcance, conforme constatado na seção 4.2.1, na qual foi possível atingir a distância máxima de 10 metros utilizando o celular com as costas apontadas para o dispositivo, constatada como uma das melhores posições (seção 4.2.2). Dessa maneira, durante a utilização do aplicativo, o dispositivo pode parar de ser detectado. Nessas situações, muito provavelmente, a direção em que está sendo seguida está afastando o usuário do dispositivo. A fim de lidar com essas limitações, sempre que o dispositivo parar de ser detectado, o usuário será orientado a tentar fazer um outro *scan*. Caso este *scan* resulte novamente em "dispositivo não encontrado", o usuário será orientado a voltar para a posição referência (Pref) e trocar de direção para à direita do usuário. Caso esse passo seja repetido nas 4 direções, o dispositivo será considerado fora de alcance do celular. Estas

medidas também serão realizadas caso o resultado for “dispositivo não encontrado” desde o primeiro *scan*.

Para implementar esse algoritmo foi criada a classe *Controladora.java*, que controla toda a lógica. Para controlar os passos foi desenvolvida um sistema de turnos, no qual cada *scan* é considerado como um turno. Para se criar uma instância da mesma, é necessário passar três objetos do tipo *TextView*, que são objetos responsáveis pela apresentação do resultado na tela do aplicativo. Além desses três objetos, a classe contém as variáveis de controle apresentadas na Figura. (5.1). O construtor, bem como as variáveis de controle, está apresentado na Figura. (5.2).

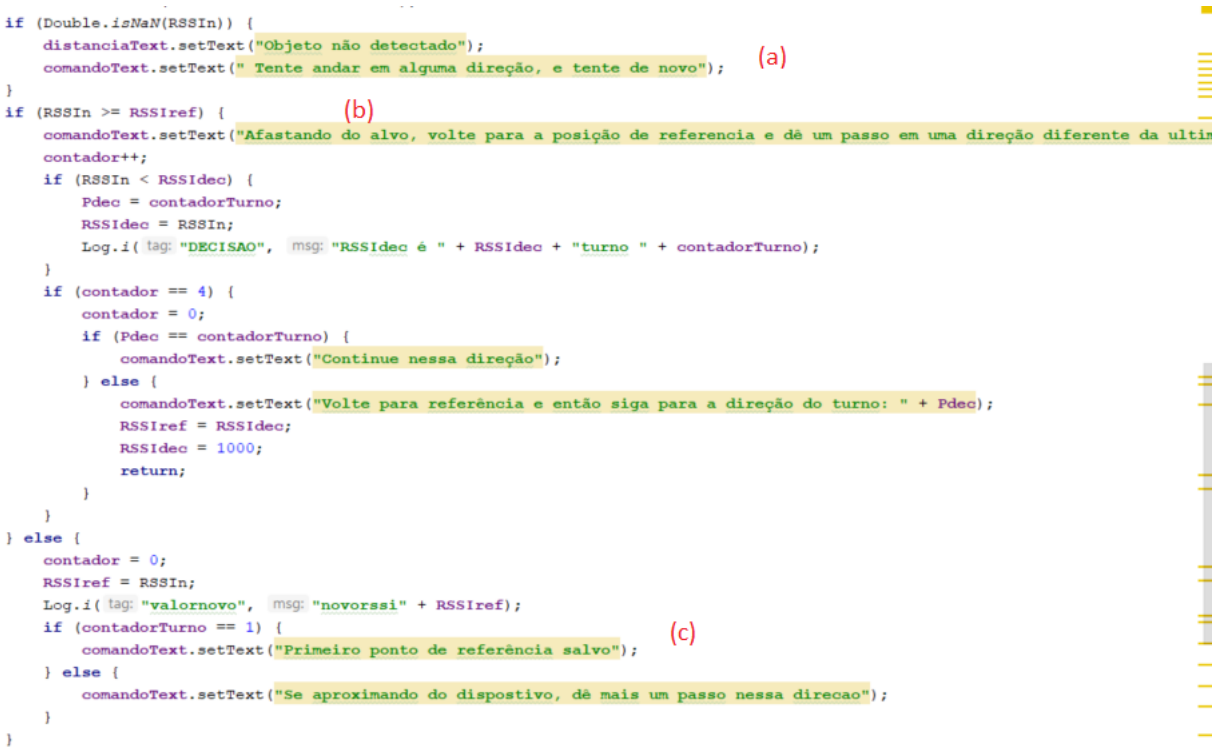
```
public class Controladora {
    int contadorTurno;
    int Pdec;
    double RSSIref;
    int contador;
    double RSSIdec;
    String log=" ";
    TextView comandoText, turnoText, distanciaText;


    public Controladora(TextView comandoText, TextView turnoText, TextView distanciaText){
    this.contadorTurno=0;
    this.RSSIref=1000;
    this.RSSIdec=1000;
    this.comandoText=comandoText;
    this.turnoText=turnoText;
    this.distanciaText=distanciaText;

    }
```

Figura 5.2 – Variáveis e construtor da classe *Controladora.java*.

Ao fim de cada *scan*, a função *Controladora.calcularDistancia()* é chamada. Essa é a função que implementa todo o algoritmo criado. Primeiramente, a média e o desvio padrão são calculados. Logo depois, o RSSIn é calculado como sendo o módulo do valor de RSSI médio é acrescido do desvio padrão. A partir desse ponto começam as primeiras verificações. Primeiro é verificado se algum pacote foi recebido (Figura. 5.3 a), depois é verificado se o valor de RSSIn é maior que o valor de RSSIref(Figura. 5.3b), e também se é o primeiro turno(Figura. 5.3 c). Após cada verificação são realizadas as operações descritas na Figura. 5.1, e dependendo de cada situação, são apresentadas as instruções na tela do aplicativo.




```
if (Double.isNaN(RSSIn)) {
    distanciaText.setText("Objeto não detectado");
    comandoText.setText(" Tente andar em alguma direção, e tente de novo");
}
if (RSSIn >= RSSIref) {
    comandoText.setText("Afastando do alvo, volte para a posição de referencia e dê um passo em uma direção diferente da ulti
    contador++;
    if (RSSIn < RSSIdec) {
        Pdec = contadorTurno;
        RSSIdec = RSSIn;
        Log.i( tag: "DECISAO",  msg: "RSSIdec é " + RSSIdec + "turno " + contadorTurno);
    }
    if (contador == 4) {
        contador = 0;
        if (Pdec == contadorTurno) {
            comandoText.setText("Continue nessa direção");
        } else {
            comandoText.setText("Volte para referência e então siga para a direção do turno: " + Pdec);
            RSSIref = RSSIdec;
            RSSIdec = 1000;
            return;
        }
    }
} else {
    contador = 0;
    RSSIref = RSSIn;
    Log.i( tag: "valornovo",  msg: "novorssi" + RSSIref);
    if (contadorTurno == 1) {
        comandoText.setText("Primeiro ponto de referência salvo");
    } else {
        comandoText.setText("Se aproximando do dispostivo, dê mais um passo nessa direcao");
    }
}
```

Figura 5.3 – Primeiras decisões do método *calcularDistancia()*.

Depois disso, o método estima a distância em que o dispositivo se encontra utilizando os limites definidos na Tab. (4.13). Os valores estimados são apresentados na tela do aplicativo para o usuário.

### 5.1.2 – Interface do aplicativo

Por se tratar de um protótipo de aplicativo, o *layout* do aplicativo é bem simples. Existem dois botões, um responsável por realizar a busca do dispositivo e outro responsável pelo reinicio da procura. Há três campos em que são mostradas as informações referentes ao turno, o comando e a distância aproximada do dispositivo. Além disso, existe uma *seekbar* responsável pela definição do tempo de procura, são três opções: A procura mais eficiente, meio termo e a mais rápida. Na opção “mais eficiente”, é utilizado o valor de 15 segundos como o tempo de *scan*, que foi definido na seção 4.3. Já para o “meio termo”, o tempo utilizado é de 7 segundos e para o “mais rápido” foi definido o tempo de 3 segundos.  A Figura (5.4) apresenta o *layout* do dispositivo.


Figura 5.4 – *Layout* do aplicativo.

### 5.1.3 – Limitações do aplicativo

A versão em que o aplicativo se encontra durante a escrita desse relatório apresenta uma série de limitações, como:

- O aplicativo só funcionará com o dispositivo criado nesse trabalho, isso porque conforme citado na seção 4.2.5 , o filtro do *scan* despreza todos os pacotes que não originarem do endereço MAC do dispositivo criado;
- Como foi necessária a criação de um mapeamento prévio, a distância aproximada só será acurada se o celular for o mesmo do utilizado durante o período de teste, o *ASUS Zenfone Zoom 3*.

### 5.1.4 – Funcionamento do aplicativo

Ao iniciar o aplicativo, é necessário especificar a velocidade dos *scans* que serão realizados durante a captura. Por padrão o aplicativo começa com “Mais eficiente”. Os três valores possíveis são: “Mais eficiente”, “meio termo” e “mais rápido”, eles podem ser alternados durante o tempo de execução do aplicativo. Para alternar entre os valores é só mudar a posição da *seekbar*. A figura (5.5) mostra como é apresentado as informações no aplicativo.


Figura 5.5 – Ajustes disponíveis de velocidade da procura.

Após definida a velocidade, pode-se apertar o botão “Procurar dispositivo”. Na primeira vez que o botão for pressionado, o valor de referência será criado, uma mensagem sobre isso informará ao usuário e, além disso, uma estimativa da distância será informada. A primeira captura pode ser vista na Figura. (5.6).

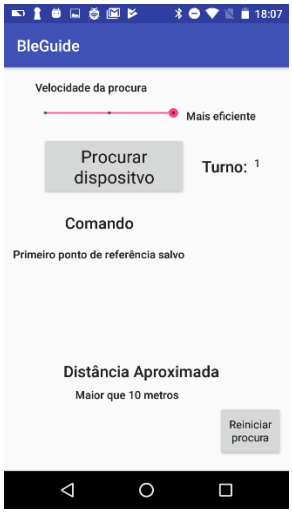



Figura 5.6 – Primeira execução do aplicativo.

Depois da primeira captura, o algoritmo entrará em funcionamento e dependendo da circunstância, aparecerão mensagens direcionando o usuário e a distância aproximada para o dispositivo BLE. O contador de turno aumentará a cada vez que for pressionado o botão “Procurar dispositivo”. As respostas que podem ser obtidas pelo aplicativo estão apresentadas na Tab. (5.1).

Tabela 5.1 – Comandos que podem ser solicitados pelo aplicativo

| Comando | Motivo |
|---|---|
| Se aproximando do dispositivo, dê mais um passo nessa direção. | A nova captura de RSSI resultou num resultado mais próximo do dispositivo ( RSSIn<RSSiref) |
| Afastando do alvo, volte para a posição de referência e de um passo em uma direção diferente da ultima | A nova captura de RSSI resultou num resultado mais distante do dispositivo ( RSSIn>RSSiref) |
| Objeto não detectado | O celular não conseguiu receber pacotes do dispositivo BLE |
| Volte para referência e então siga para a direção do turno: x | Foram testadas todas as direções, e a direção que resultou no melhor resultado foi a X. |
| Objeto localizado | Quando a distância do dispositivo e o celular é menor que 1 metro |

Em paralelo aos comandos mostrados pelo aplicativo, é informada a distância aproximada do dispositivo com o celular. As mensagens que podem ser apresentadas estão expostas na Tab. (4.13).

## 5.2- TESTE FINAL

A fim de validar a funcionalidade do aplicativo, foram realizados alguns testes. O ambiente em que foi realizado o teste foi o mesmo do que todos os outros realizados no capitulo 4. Para esse teste, foi utilizado apenas o modo “Mais eficiente”, e o celular utilizado foi o *ASUS Zenfone Zoom 3*. Em todos os testes, cada “passo” foi numerado e padronizado para medir 1 metro.

Para a visualização dos trajetos, foi desenvolvido um sistema de cores. Os significados são:

- Vermelho: o aplicativo está guiando o usuário por um percurso errado, se afastando do dispositivo.
- Laranja: o aplicativo percebeu que o usuário está indo por um percurso errado, troca de direção.
- Azul: O aplicativo está guiando o usuário pelo percurso correto, se aproximando do dispositivo.
- Preto: O aplicativo detecta que o celular não foi capaz de receber pacotes do dispositivo.

O primeiro teste realizado foi o teste mais simples possível. Para isso, se posicionou o celular a 8 metros de distância do dispositivo e com a câmera virada para ele. Assim, caso o algoritmo funcione, o aplicativo daria orientações para seguir em frente até que seja detectada uma distância menor que 1 metro, que é a margem para que o dispositivo seja localizado. O trajeto desse teste pode ser visto na Figura. (5.8), onde a letra C representa a posição inicial do celular e a letra B representa a posição do dispositivo BLE.

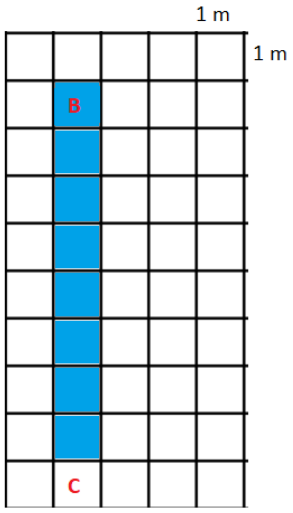



Figura 5.8 – Percurso feito em linha reta.

Para o segundo teste, o dispositivo foi posicionado a 8 metros do celular, mas com a câmera do celular direcionada para outra direção (esquerda). Dessa maneira, o aplicativo mandou seguir 3 metros no sentido errado (Passo 3), até cair na decisão de troca de direção. Ao comparar que o RSSIn é maior que o RSIDec, o aplicativo orientou seguir numa outra direção. Após voltar para a Pref e dar um passo na nova direção (Passo 4), foi dada mais uma instrução para mudança de direção. Mais um passo foi dado até o aplicativo novamente orientar uma mudança de direção. Após comparar o RSSIn das quatro direções, o aplicativo fez a instrução na direção correta (Passo 6). Após isso, o aplicativo mandou seguir mais sete passos em linha reta, até orientar uma nova mudança de direção. Ao dar um passo nessa nova direção, o aplicativo detectou uma distância menor que 1 metro, o que resulta no comando “Objeto localizado”. A distância final em que o celular ficou do dispositivo, foi de 44 centímetros. Essa distância é consequência dos erros experimentais ao dar os “passos”, que não foram feitos de maneira totalmente perpendicular. A Figura (5.9), onde a letra C representa a posição inicial do celular e a letra B representa a posição do dispositivo BLE, mostra o trajeto que foi percorrido nesse teste

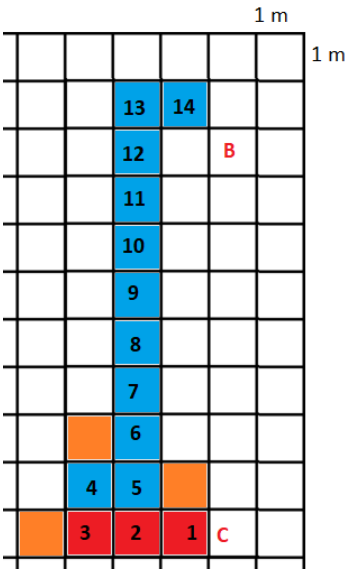



Figura 5.9 – Percurso com celular apontado para a esquerda.

De maneira semelhante, foi posicionado o celular a 8 metros do dispositivo e com a câmera traseira posicionada para direita. Após o passo 2, o aplicativo entrou na decisão de troca de direção. Ao virar para a direita, percorreu-se dois passos. Após o passo 4, o celular não pôde mais detectar o dispositivo. Seguindo os comandos propostos, voltou-se para a Pref (Passo 4) e trocou-se a direção do deslocamento. Seguiu-se nessa direção por 3 metros, até o celular, novamente, não conseguir detectar o dispositivo. Voltou-se 1 metro (Passo 6) e houve uma mudança de direção. Do passo 6 ao passo 16, o algoritmo funcionou corretamente e guiou o usuário numa linha reta até resultar no comando "Objeto localizado" com uma distância de 57 centímetros. Novamente, essa distância é consequência de mudanças de direção não perpendiculares. O percurso descrito acima está representado na Figura (5.10), onde a letra C representa a posição inicial do celular e a letra B representa a posição do dispositivo BLE.

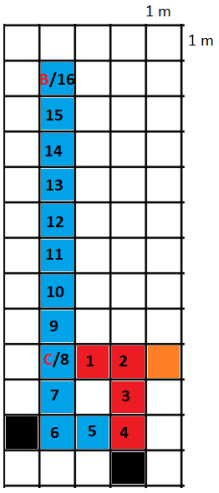



Figura 5.10 – Percurso com celular apontado para a direita.

Foi realizado outro teste com o dispositivo posicionado a 12 metros do celular. Nesse teste, o aplicativo não foi capaz de localizar o dispositivo desde o primeiro *scan*. Isso comprova os dados teóricos de alcance do Bluetooth na versão 4.2, de apenas 10 metros.

# 6 CONCLUSÃO

Neste trabalho foi desenvolvido um aplicativo de detecção de proximidade de dispositivos utilizando *Bluetooth Low Energy*. Para atingir esse objetivo, foram necessárias várias etapas, desde a definição da versão do Bluetooth que seria utilizada até a definição do *fingerprinting*, juntamente com definição de intervalos de espaço, como técnica de localização utilizada. Foi necessário realizar vários testes de comportamento do RSSI, já que não havia muitos estudos do comportamento dele em ambientes outdoor.

Primeiramente foi observado o comportamento do RSSI em ambientes outdoor, e pode ser concluído que o RSSI varia de forma inversa à distância, assim como era esperado pela teoria. Verificou-se também o comportamento do RSSI com em relações as diferentes versões do Bluetooth e nos testes realizados a versão que apresentou o comportamento mais adequado à proposta foi a versão 4.2. O alinhamento das antenas foi um fator que influenciou bastante nos estudos e, como forma de minimiza-lo, o dispositivo e o celular nos testes foram posicionados no mesmo nível de maneira a fazer com que as antenas se mantivessem sempre com o mesmo alinhamento.

A relação entre a janela de *scan* e a frequência de *advertisement* também foi estudada. Como o dispositivo não se movia, não tinha necessidade do tempo de janela de *scan* ser muito pequena, nem da frequência de *advertisement* ser muito rápida. Com isso, foram realizados testes, e pode ser concluído que, nessa situação, a utilização de 10 Hz como taxa de *advertisement* permitiu alcançar distâncias maiores. O tempo de janela de *scan* que permitiu medidas dos valores de RSSI mais estáveis independentemente da distância foi 15 segundos.

Após isso, tentou-se obter uma relação do RSSI com a distância e ângulo das antenas, e, infelizmente, não foi possível detectar nenhuma relação entre elas, visto que as informações sobre a polarização da antena dos celulares não são divulgadas pelos seus fabricantes. Esse ponto limitou bastante o projeto e mostrou que obter a localização de um dispositivo com apenas um celular não é algo fácil. Além disso, uma outra justificativa é o fato de que os valores de RSSI são sempre valores inteiros, o que limita muito a diferenciação de um ponto para outro.

Ao final, foi decidida a técnica de localização que seria utilizada no aplicativo final. Determinar a localização de um dispositivo com apenas um celular se mostrou ineficaz. A acurácia alcançada no teste utilizando *fingerprinting* foi de 55% e no teste utilizando a primeira divisão em limites foi de 28%. Essa ineficácia pode ser explicada pela natureza instável do RSSI e de não ter mais celulares para gerar mais mapeamentos para o método de *fingerprinting*.

Porém, detectar proximidade é possível, como provado neste trabalho no qual foi detectado apenas um erro. Foi criado um algoritmo, que analisando o RSSI e suas variações nas vizinhanças e utilizando o conceito de detecção de proximidade, consegue contornar o problema de não conseguir extrair informações de direções do RSSI. Como resultado, foi criado um aplicativo que consegue guiar o usuário até o dispositivo e apontar a proximidade entre ele e o dispositivo.

O aplicativo tem muito a evoluir para possuir uma versão comercial. O principal problema é que ele está muito atrelado ao hardware utilizado nos testes. Logo, o principal trabalho futuro seria a possibilidade de utilização desse aplicativo com qualquer dispositivo, utilizando algum algoritmo de regressão para atualizar o mapeamento de RSSI com qualquer celular. Outra melhoria seria o estudo com o Bluetooth 5.0 que apresenta várias vantagens em relação ao Bluetooth 4.2, como a possibilidade da utilização de 37 canais para *advertisement*.

Essa funcionalidade do aplicativo pode ser utilizada em várias aplicações, como: substituição de cães-guias para cegos, guias para encontrar dispositivos perdidos em casa, pode ser implementado em robôs para realização de tarefas domésticas que necessitam ser guiados até determinado cômodo.

# 7 REFERÊNCIAS


**A BLE Advertising Primer** · Argenox Technologies. Disponível em: <http://www.argenox.com/a-BLE-advertising-primer>. Acesso em: 20 abr. 2018.

**Android Beacon Library**. Disponível em: <https://altbeacon.github.io/android-beacon-library/>. Acesso em: 20 fev. 2018.

**Arduino - Software**. Disponível em: <https://www.arduino.cc/en/Main/Software>. Acesso em: 20 fev. 2018.

**Asus Zenfone 3 Zoom ZE553KL -** Full phone specifications. Disponível em: <https://www.gsmarena.com/asus_zenfone_3_zoom_ze553kl-8509.php>. Acesso em: 20 fev. 2018

**Bluetooth low energy overview** | Android Developers. Disponível em: <https://developer.android.com/guide/topics/connectivity/bluetooth-le>. Acesso em: 20 jun. 2018.

**BLUETOOTH SPECIFICATION Version 4.2[**Vol 6]. Traducao . [s.l.] Bluetooth SIG, 2014.

**Bluetooth SIG Introduces New Bluetooth 41 Specification | Bluetooth Technology Website.** Disponível em: <https://blog.bluetooth.com/bluetooth-sig-introduces-new-bluetooth-4-1-specification>. Acesso em: 28 fev. 2018.

CAMERON, N. **Chatswood Chase reveals customer insights from iBeacon trial.** Disponível em: <https://www.cmo.com.au/article/551328/chatswood_chase_reveals_customer_insights_from_ibeacon_trial/>. Acesso em: 28 jun. 2018.

CONNELL, C. **What's The Difference Between Measuring Location By UWB, Wi-Fi, and Bluetooth?.** Disponível em: <http://www.electronicdesign.com/communications/what-s-difference-between-measuring-location-uwb-wi-fi-and-bluetooth>. Acesso em: 20 jun. 2018.

CHO, K. et al**. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks**. Sensors, v. 15, n. 1, p. 59-78, 2014.

**Compensation of the Antenna Polarization Misalignment in the RSSI Estimation**. Proceedings of the 3rd International Conference on Sensor Networks, 2014.

COSTA DIAS, M.; DE CASTRO, R.; APOLINÁRIO JR., J. **On the perfomance of new and classical approaches AOA estimation for near-field acoustical waves**. 2004.

FAGUNDES, L. **Técnicas de localização de dispositivos móveis em redes WiFi - TDOA**. Universidade Federal do Rio Grande do Sul. Instituto de Informática. Curso de Especialização em Tecnologias, Gerência e Segurança de Redes de Computadores., 2008.

FARAGHER, R.; HARLE, R**. Location Fingerprinting With Bluetooth Low Energy Beacons**. IEEE Journal on Selected Areas in Communications, v. 33, n. 11, p. 2418-2428, 2015.

GOMEZ, C.; OLLER, J.; PARADELLS, J. **Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Sensors**, v. 12, n. 9, p. 11734-11753, 2012.

HAATAJA, K.; TOIVANEN, P**. Two practical man-in-the-middle attacks on Bluetooth secure simple pairing and countermeasures.** IEEE Transactions on Wireless Communications, v. 9, n. 1, p. 384-392, 2010.

**iBeacon** - Apple Developer. Disponível em: <https://developer.apple.com/ibeacon/>. Acesso em: 20 fev. 2018.

KWONKI, L. Android AOSP - **Definition of *scan* interval and *scan* window in android source code.** Disponível em: <http://ai2inventor.blogspot.com/2017/06/android-aosp-definition-of-*scan*.html>. Acesso em: 20 jun. 2018.

**LilyPad Coin Cell Battery Holder - Switched - 20mm** - DEV-13883 - SparkFun Electronics. Disponível em: <https://www.sparkfun.com/products/13883>. Acesso em: 20 jun. 2018.

MADUSKAR, D.; TAPASWI, S. **RSSI based adaptive indoor location tracker.** Scientific Phone Apps and Mobile Devices, v. 3, n. 1, 2017.

**Meet Android Studio** | Android Developers. Disponível em: <https://developer.android.com/studio/intro/>. Acesso em: 20 fev. 2018.

**Motorola Moto G Dual SIM (2nd gen)** - Full phone specifications. Disponível em: <https://www.gsmarena.com/motorola_moto_g_dual_sim_(2nd_gen)-6648.php>. Acesso em: 20 fev. 2018.

**New Bluetooth Specifications EnaBLE IP Connectivity And Deliver Industry Leading Privacy And Increased Speed | Bluetooth Technology Website**. Disponível em: <https://www.bluetooth.com/news/pressreleases/2014/12/03/new-bluetoothspecifications-enaBLE-ip-connectivity-and-deliver-industry-leading-privacy-and-increased-speed>. Acesso em: 28 fev. 2018.

NEWMAN, N. **Apple iBeacon technology briefing. Journal of Direct, Data and Digital Marketing Practice,** v. 15, n. 3, p. 222-225, 2014.

POLESE, D. et al. **Compensation of the Antenna Polarization Misalignment in the RSSI Estimation**. Proceedings of the 3rd International Conference on Sensor Networks, 2014.

**redbear/nRF5x**. Disponível em: <https://github.com/redbear/nRF5x/tree/master/nRF52832>. Acesso em: 20 abr. 2018.

STOICA, P.; MOSES, R. **Introduction to spectral analysis**. Traducao . Upper Saddle River: Prentice Hall, 1997.

**Samsung Galaxy S6 - Full phone specifications.** Disponível em: <https://www.gsmarena.com/samsung_galaxy_s6-6849.php>. Acesso em: 20 fev. 2018.

SHASHIDHARA, D.; POORNIMA, M. BLE: **BluePrint Finder through Android Application. International Journal of Innovative Research in Computer and Communication Engineering**, v. 6, n. 4, 2018.

**SweetBlue Android Bluetooth Smart Library** | iDevices LLC. Disponível em: <https://idevicesinc.com/sweetblue>. Acesso em: 20 fev. 2018..

XIN-YU LIN et al. **A mobile indoor positioning system based on iBeacon technology**. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015.

# ANEXOS

## ANEXO I

Tabela 8.1- Valores do desvio padrão de cada celular.

| | *Zenfone Zoom 3* | | *Samsung Galaxy S6* | | *Motorola Moto G 2* | |
|---|---|---|---|---|---|---|
| Distância(m) | RSSI Médio(dBm) | Desvio Padrão | RSSI Médio(dBm) | Desvio Padrão | RSSI Médio(dBm) | Desvio Padrão |
| 0,1 | -39,91 | 1,20 | -53,69 | 2,78 | -54,05 | 2,02 |
| 0,2 | -43,29 | 1,29 | -59,87 | 1,91 | -56,91 | 2,82 |
| 0,3 | -48,63 | 1,29 | -59,08 | 1,91 | -54,03 | 2,49 |
| 0,4 | -52,06 | 1,28 | -65,71 | 2,77 | -55,73 | 2,27 |
| 0,5 | -55,05 | 1,04 | -66,94 | 1,42 | -63,28 | 2,43 |
| 0,6 | -56,98 | 1,11 | -74,43 | 2,72 | -69,36 | 2,39 |
| 0,7 | -59,06 | 0,78 | -72 | 1,43 | -71,08 | 2,54 |
| 0,8 | -60,78 | 1,15 | -73,12 | 2,82 | -70,68 | 4,41 |
| 0,9 | -62,32 | 1,61 | -77,15 | 2,61 | -75,63 | 2,15 |
| 1 | -64,06 | 1,47 | -76,63 | 1,71 | -76,64 | 2,10 |

## ANEXO II

Tabela 8.2 – Amplitude do desvio padrão em relação a taxa de *advertising* e tempo de *scan*.

| Taxa de Advertising | Tempo de *Scan* | Desvio Padrão Máximo | Desvio Padrão Mínimo | Amplitude do desvio padrão |
|---|---|---|---|---|
| 10 hz | 5 segundos | 2,26 | 0,88 | 1,38 |
| 10 hz | 15 segundos | 2,09 | 1,00 | 1,08 |
| 10 hz | 30 segundos | 2,25 | 0,87 | 1,37 |
| 30 hz | 5 segundos | 2,53 | 0,98 | 1,54 |
| 30 hz | 15 segundos | 3,05 | 0,5 | 2,55 |
| 30 hz | 30 segundos | 3,02 | 1,14 | 1,88 |
| 50 hz | 5 segundos | 4,68 | 1,47 | 3,21 |
| 50 hz | 15 segundos | 3,36 | 1,00 | 2,36 |
| 50 hz | 30 segundos | 3,81 | 0,96 | 2,85 |

## ANEXO III

Tabela 8.3 – Resultado do esquema de grid.

| Ponto | Distância(cm) | RSSI mapeado (dBm) | RSSI fase online(dBm) | Posição encontrada |
|---|---|---|---|---|
| 0,0 | 94,87 | -72,19 | -62,21 | 1,0 |
| 0,1 | 67,08 | -59,18 | -62,82 | 1,0 |
| 0,2 | 42,43 | -55,01 | -55,38 | 0,2 |
| 1,0 | 90,00 | -61,44 | -62,51 | 1,0 |
| 1,1 | 60,00 | -55,00 | -63,01 | 1,0 |
| 1,2 | 30,00 | -46,99 | -49,13 | 1,2 |
| 2,0 | 94,87 | -60,32 | -62,96 | 1,0 |
| 2,1 | 67,08 | -57,42 | -56,60 | 2,1 |
| 2,2 | 42,43 | -53,96 | -53,99 | 2,2 |

## ANEXO IV

Tabela 8.4 – Resultados do teste com limites granulares.

| Posição | 10 cm | 20 cm | 60 cm | 80 cm | 100 cm | 120 cm | 500 cm |
|---|---|---|---|---|---|---|---|
| 4 | Perto dos 20 cm para cima | Perto dos 20 cm para cima | Perto de 1 m para baixo | Com certeza 1 m | Com certeza 1 m | Maior que 1 m | Maior que 1 m |
| 5 | Com certeza 20 cm | Perto dos 20 cm para cima- | Entre 20 cm e 1 m | Com certeza 1 m | Com certeza 1 m | Perto de 1 m pra cima | Maior que 1 m |
| 6 | Com certeza 20 cm | Entre 20 cm e 1 m | Perto de 1 metro para baixo | Com certeza 1 m | Com certeza 1 m | Maior que 1 m | Maior que 1 m |
| 7 | Com certeza 20 cm | Perto dos 20 cm para cima | Com certeza 1 m | Com certeza 1 m | Maior que 1 m | Maior que 1 m | Maior que 1 m |

Tabela 8.5 – Resultados com o novos limites.

| Posição | 10 cm | 20 cm | 60 cm | 80 cm | 100 cm | 120 cm | 500 cm |
|---|---|---|---|---|---|---|---|
| 4 | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de mais de 1 m | No raio de mais de 1 m |
| 5 | No raio de 20 cm | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de mais de 1 m | No raio de mais de 1 m |
| 6 | No raio de 20 cm | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de mais de 1 m | No raio de mais de 1 m |
| 7 | No raio de 20 cm | No raio de 1 m | No raio de 1 m | No raio de 1 m | No raio de mais de 1 m | No raio de mais de 1 m | No raio de mais de 1 m |

Tabela 8.6 – Resultados com os limites até 10 metros.

| Posição | 50 centímetros | 150 centímetros | 350 centímetros | 750 centímetros | 1250 centímetros |
|---|---|---|---|---|---|
| 4 | Menor que 1 metro | Entre 1 metro e 2 metros | Entre 2 metros e 5 metros | Entre 5 metros e 10 metros | Maior que 10 metros |
| 5 | Menor que 1 metro | Entre 1 metro e 2 metros | Entre 2 metros e 5 metros | Entre 5 metros e 10 metros | Entre 5 metros e 10 metros |
| 6 | Menor que 1 metro | Entre 1 metro e 2 metros | Entre 2 metros e 5 metros | Entre 5 metros e 10 metros | Maior que 10 metros |
| 7 | Menor que 1 metro | Entre 1 metro e 2 metros | Entre 2 metros e 5 metros | Entre 5 metros e 10 metros | Maior que 10 metros |