UNIVERSIDADE FEDERAL DO PARANÁ

HENRIQUE NAOTO KATO
RAFAEL RZESCUTKO

SISTEMA DE GEOLOCALIZAÇÃO INDOOR PARA LOCALIZAÇÃO EM AMBIENTES E DE PESSOAS

CURITIBA

2019

HENRIQUE NAOTO KATO
RAFAEL RZESCUTKO

SISTEMA DE GEOLOCALIZAÇÃO INDOOR PARA LOCALIZAÇÃO EM AMBIENTES E DE PESSOAS

Trabalho de conclusão de curso de graduação, apresentado ao Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados da Universidade Federal do Paraná como requisito à obtenção do grau de Engenheiro Eletricista, Setor de Tecnologia da Universidade Federal do Paraná.
Orientador: Prof. Dr. Marcelo Eduardo Pellenz

CURITIBA

2019

Henrique Naoto Kato
Rafael Rzescutko
Sistema de Geolocalização Indoor para Localização em Ambientes e de Pessoas/ Henrique Naoto Kato
Rafael Rzescutko. – Curitiba, 2019-
114 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Marcelo Eduardo Pellenz

Trabalho de Conclusão de Curso – Universidade Federal do Paraná, 2019.

1. RSSI. Aplicativo. Servidor. Trilateração. Mínimos quadrados.
K-NN. Precisão. Probabilidade de sucesso. Comunicação.
I. Orientador: Prof. Dr. Marcelo Eduardo Pellenz.
II. Universidade Federal do Paraná
III.
IV. Sistema de Geolocalização Indoor para Localização em Ambientes e de Pessoas

CDU 02:141:005.7

TERMO DE APROVAÇÃO

HENRIQUE NAOTO KATO
RAFAEL RZESCUTKO

SISTEMA DE GEOLOCALIZAÇÃO INDOOR PARA LOCALIZAÇÃO EM AMBIENTES E DE PESSOAS

Trabalho de conclusão de curso, aprovado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, pela seguinte banca examinadora:

Prof. Dr. Marcelo Eduardo Pellenz
Orientador

Prof. Armando Heilmann
Departamento de Engenharia Elétrica
UFPR

Prof. Rodrigo Godinho Silva
Departamento de Engenharia Elétrica
UFPR

Curitiba, 05 de Dezembro de 2019.

Este trabalho dedico a Deus, as nossas famílias e amigos que sempre estiveram conosco nos momentos difíceis e que fizeram parte de tornar isso uma realidade.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo das nossas vidas, e não somente nestes anos como universitários, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

A esta universidade - Universidade Federal do Paraná, seu corpo docente, direção e administração que proveram a janela que hoje vislumbramos um horizonte superior, construído na confiança, no mérito e ética aqui presentes.

Ao professor Dr. Marcelo Eduardo Pellenz, pela orientação, apoio e confiança. Agradecemos a todos os professores por nos proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram, não somente por terem ensinado, mas por terem feito aprender.

Aos nossos pais, pelo amor, incentivo e apoio incondicional. Nossos agradecimentos aos amigos, companheiros de trabalhos e irmãos na amizade que fizeram parte da nossa formação que vão continuar presentes em nossas vidas com certeza juntos para novos desafios e a todos que fizeram parte direta e indiretamente de nossa caminhada.

“Em todas as coisas o sucesso depende
de uma preparação prévia, e sem tal
preparação o fracasso é certo.
(Confúcio, Kung-fu-tzu (551 a.C. - 479 a.C.)

RESUMO

A elaboração do projeto buscou desenvolver um sistema de localização *indoor* onde um usuário habilitado para a utilização do mesmo tivesse a possibilidade de conhecer a sua posição em um determinado ambiente apto. O sistema baseia-se na utilização da técnica de RSSI (Radio Signal Strength Indicator) para estipular a distância entre os dispositivos emissores, módulos *bluetooth*, e o dispositivo receptor, no caso o celular do usuário. O dispositivo celular é utilizado para mostrar os dados de posição para usuário, onde um aplicativo foi desenvolvido, e realizar a comunicação com um servidor que estará localizado na rede de dados. O servidor por sua vez será responsável por determinar a posição do usuário pela técnica de trilateração utilizando de duas técnicas de ajustamento para melhor predição da posição: a técnica dos mínimos quadrados e a técnica do K-NN. Estas duas técnicas são usadas para corrigirem erros de estimativa das distâncias entres as fontes emissoras e a receptora. Sobre as técnicas foram avaliadas três pontos principais: precisão, probabilidade de sucesso e tempo de resposta no sistema desenvolvido.

Palavras-chaves: RSSI. Aplicativo. Servidor. Trilateração. Mínimos quadrados. K-NN. Precisão. Probabilidade de sucesso. Comunicação.

ABSTRACT

The elaboration of the project sought to develop an indoor location system where a user qualified to use it had the possibility to know his position in a certain environment licensed. The system is based on the use of the Radio Signal Strength Indicator (RSSI) technique to determine the distance between the sending devices, bluetooth modules, and the receiving device, in this case the user's mobile phone. The mobile device is used to display position data for the user where an application was developed and to communicate with a server that will be located on the data network. The server in turn will be responsible for determining the user's position by the trilateration technique using adjustment techniques to better predict the position: the least squares technique and the K-NN technique. These two techniques are used to correct distance estimation errors between the sending and receiving sources. About the techniques, three main points were evaluated: accuracy, probability of success and response time in the developed system.

Key-words: RSSI. App. Server trilateration. Least squares. K-NN. Precision. Probability of success. Communication.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em muito se fala que tempo é dinheiro e que a perca de tempo é algo que não se pode ocorrer nos dias atuais ao fato do mundo de hoje ser tão dinâmico. As pessoas estão cada vez mais em um ritmo tão agitado que simples tarefas podem ser deixadas de serem feitas pela falta de tempo.

Uma outra característica do mundo moderno, está no fato deste ser, em muitas das vezes, altamente dinâmico a mudanças, tanto no que se diz ao ambiente bem como as pessoas. Como podemos exemplificar, uma pessoa pode estar em dois países diferentes no mesmo dia, isso é algo comum no mundo globalizado ou até mesmo feiras estarem em uma determinada região e se reconstruírem em outra em poucos dias, são alguns poucos exemplos do dinamismo no mundo atual.

Muitas tecnologias são implementadas e criadas a fim de ajudar e facilitar a vida das pessoas a fim de tentar minimizar os problemas do mundo moderno, sendo alguns dos exemplos como os citados.

Uma tecnologia bastante difundida e utilizada que se torna uma alternativa na resolução de alguns problemas, focando mais na parte de localização, é o GPS. O GPS foi uma tecnologia desenvolvida para uso militar, mas devido a sua facilidade de uso e implementação em dispositivos diversos, nos dias atuais, foi rapidamente distribuído e hoje se torna uma excelente alternativa de localização georreferenciada para a locomoção de pessoas em ambientes *outdoor*.

Como qualquer outra tecnologia o GPS não consegue, ou apresenta grande dificuldade, apresentar uma boa resposta em ambientes interiores, por exemplo, um prédio, pelo fato de ser um projeto desenvolvido para ambientes exteriores. Como mencionado anteriormente as tecnologias são desenvolvidas para suprir as necessidades e auxiliar a pessoas aos problemas existentes. Saber a sua localização em um ambiente e em muitos casos uma forma de auxiliar o usuário, da mesma forma que o GPS realiza, mas como esta tecnologia da forma que foi implementada não apresenta bons resultados de precisão quanto atuada em ambientes internos, foram se criados técnicas de localização, partindo de outras tecnologias e métodos para suprir estas necessidades.

A localização *indoor* é uma forma de auxiliar as pessoas na sua localização, indicar e encontrar caminhos ou rotas, encontrar pontos específicos ou até mesmo encontrar pessoas em um grande ambiente, sendo algum exemplos: um hospital, uma feira em um ambiente de eventos, um *campus* universitário, em um show, shopping e entre outros ambientes que se podem ser implementados.

Os projetos de localização *indoor* baseiam-se, geralmente, em um sistema de

comunicação sem fio entre o usuário e uma rede de dispositivos que utilizam protocolos específicos para essa comunicação e manipulação dos dados que deverão ser trocados, entre os dispositivos, para se conseguir de alguma forma estipular a posição do usuário no ambiente com base em um referencial.

Portanto os sistemas de localização *indoor* possuem a finalidade de auxiliar as pessoas em alguns pequenos problemas, como por exemplo, os citados anteriormente. Considerando uma situação onde um usuário que possui posse do sistema, poderia encontrar um determinado *stand* em uma feira de profissões ou uma determinada pessoa perdida em uma multidão de uma forma mais fácil do que os métodos utilizados para estes fins economizando tempo de procura.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral do presente trabalho é desenvolver um sistema de localização *indoor* para grandes ambientes, para que pessoas que tiverem posse do sistema possam localizar a sua posição neste grande ambiente desconhecido pelo usuário de uma maneira mais rápida e fácil.

1.1.2 Objetivos Específicos

Para a conclusão do trabalho foram assinalados os seguintes objetivos específicos, que serviram de guia do conteúdo que foi abordado e planejado no trabalho:

- Desenvolver um sistema em forma de aplicativo para a utilização do usuário;
- Desenvolver um sistema que possua uma boa usabilidade;
- Desenvolver um sistema que apresente um bom tempo de resposta e precisão da localização;
- Utilizar de técnicas de ajustamento de posição para melhorar a precisão do sistema;
- Utilizar de um sistema centralizado na rede para o processamento dos dados;

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CARACTERIZAÇÃO DE UM SISTEMA DE LOCALIZAÇÃO *INDOOR*

O sistema de localização *indoor* possui a mesma finalidade do *outdoor*, porém oferece dados mais precisos em ambiente confinado, tarefa complexa de ser executada em espaços abertos. As principais métricas do sistemas de localização são: precisão e probabilidade de sucesso (PUC, 2016). A primeira está relacionada a capacidade de estimar a localização com o menor erro possível, enquanto a probabilidade de sucesso se relaciona ao percentual de estimativas que localizam o objeto sob determinada precisão estipulada.

Na construção de um sistema de localização *indoor* existem três pontos, independente das técnicas utilizadas, característicos de todos os sistemas:

- Dispositivos sensoriais, ou que funcionarão como sensores, fornecendo os dados relevantes;
- Algoritmo ou técnica de localização que processa as métricas oriundas dos dispositivos sensoriais;
- Interface gráfica para a comunicação com o usuário.

A informação da localização pode ser disponibilizada através de representação física, aplicação de sistemas de coordenadas, absolutas ou relativas, ou simbólica, similar a linguagem coloquial lidando com pontos de referência como orientação ao usuário (PUC, 2016). Os sistemas *indoor* carregam no seu desenvolvimento características que aumentam a robustez, como:

- Segurança, privacidade e controle de acesso a informação de localização;
- Atraso ou tempo de resposta;
- Robustez e tolerância a falhas, onde mesmo quando a falha existir o dispositivo seja capaz de operar;
- Nível de complexidade utilizado, onde sistemas menos complexos que ocupam menos recursos são mais bem vistos;
- Área de cobertura;
- Quantidade de dispositivos conectados.

2.2 EFEITOS DA PROPAGAÇÃO DE ONDAS ELETROMAGNÉTICAS

Decorrentes dos efeitos da propagação de ondas *wirelees* a seguinte seção propõem exemplificar os efeitos mais comuns e significativos para as ondas interceptadas pelo receptor.

2.2.1 Perda de penetração

Característica associada a perda de potência da onda quando incide ou propaga no interior de um objeto (PUC, 2016). A tabela $Tabela$ apresenta alguns exemplos da perda de penetração para alguns objetos.

Os seguintes efeitos são comuns:

- Objetos metálicos refletem grande parte do sinal incidido;
- Objetos sólidos não condutores tendem a refletir parte do sinal e propagar a outra parcela;
- Objetos úmidos absorvem grande parte do sinal incidente.

2.2.2 Multi-percurso

Fenômeno associado a três efeitos: reflexão, difração e espalhamento. A disposição dos objetos pode refletir o sinal e fazer com que percorra diferentes caminhos até o receptor (KUPPER, 2005). Os sinais que chegam em diferentes instantes de tempo sofrem interferência construtiva ou destrutiva. Dependendo do nível de sinais e suas diferentes fases, é necessário um receptor de maior qualidade.

2.2.3 Reflexão, refração, difração e espalhamento

O fenômeno de reflexão está associado quando uma onda eletromagnética incide em um objeto maior se comparado com seu comprimento de onda. Este efeito causa atenuação no sinal e o refletindo para várias direções. Já o fenômeno de difração está associado ao caso onde parte da onda incidente se propaga sobre o objeto e outra parte e refletida pelo mesmo, sendo em direções opostas (LAU, 2008).

O efeito de difração acontece quando a frente de onda é obstruída por uma superfície irregular. A difração está intrinsecamente associada ao comprimento de onda, efeito capaz de alterar a trajetória da oscilação.

O espalhamento ocorre quando o sinal incidente atinge um objeto cujas dimensões é menor que o seu comprimento de onda, fazendo que o sinal sofra atenuação e seja espalhado, como na reflexão, para diversas direções (LAU, 2008).

2.2.4 Umidade

A umidade é um efeito bastante preocupante, principalmente na questão do que se diz a atenuação do sinal. A umidade no ambiente pode aumentar a atenuação de um ambiente em até 10% em relação aos locais secos (PUC, 2016).

2.3 MODELOS DE PERDAS DE CAMINHO

O modelo de perda de caminho é a forma matemática de predizer o nível médio de potência no receptor e a variabilidade em torno dele.

Existem dois modos de se estimar as perdas do propagação do enlace de rádio: os modelos empíricos e os modelos teóricos (KUPPER, 2005). Os modelos empíricos, são baseados em medições reais no ambiente onde se deseja realizar a comunicação e assim é tomada uma equação que descreve o ambiente em questão. Já os modelos teóricos partem do pré-suposto de resolver as condições de contorno impotas pelo ambiente, ou seja, não possuem nenhum cunho experimental para a resolução do caso. Os modelos teóricos requerem um detalhamento preciso do ambiente para a sua boa resolução e geralmente requerem maior processamento para a sua predição, mas possuem a vantagem de não se precisar coletar amostras dos sinais no ambiente (KUPPER, 2005).

2.3.1 Propagação no espaço livre

O modelo de propagação em espaço livre é um dos modelos mais simples que se pode ter, sendo que ele considera que não existe nenhum objeto obstruindo a passagem do sinal entre o emissor e o receptor (HUANG; BARRALET, 2009). Este tipo de técnica leva em consideração apenas o fato da distância contribuir para a perca de potência do sinal emitido.

Considerando-se que uma antena transmissora possui um ganho G_T na direção da antena receptora e a potência de transmissão é dada por P_T, a densidade de potência a uma certa distância d é dada pela equação 2.1.

$$W = \frac{P_T G_T}{4\pi^2} \tag{2.1}$$

E sendo A a área efetiva da antena receptora dada pela equação 2.2 e G_R sendo o ganho da antena receptora.

$$A = \frac{\lambda^2 G_R}{4\pi} \tag{2.2}$$

E por fim a chega-se a equação do espaço livre, apresentada pela equação 2.3 , que é a combinação da densidade de potência transmitida sobre a antena receptora (HUANG; BARRALET, 2009).

$$\frac{P_R}{P_T} = G_T G_R [\frac{\lambda}{4\pi d}]^2 \tag{2.3}$$

Modificando a equação em termos logarítmicos e considerando L como a perda de potência transmitida em relação a recebida tem se a equação 2.4.

$$L(dB) = -10\log G_T - 10\log G_R + 20\log f_{MHz} + 20\log d_{Km} + 32,44 \tag{2.4}$$

2.3.2 Modelo log-distance

Em um ambiente *indoor* raramente o modelo de espaço livre será utilizado para se estimar a potência do sinal a uma determinada distancia, já que existem inúmeras variáveis, discutidas no capitulo 2.2, que contribuem para a atenuação do sinal.

O modelo log-distance é um modelo baseado no modelo de perda no espaço livre com alguns parâmetros extras que fazem uma melhor predição da atenuação do sinal no ambiente. Ele considera uma distância de referência d_0 e um fator de ambiente n que, em muitas vezes, é característico para cada ambiente (LEE, 2010).

A equação 2.5 , em escala logarítmica, estabelece uma relação de dois cenários para a atenuação do sinal somados. O primeiro cenário está relacionada a perda do sinal considerando o espaço livre que vai até ao ponto de referência d_0 e o segundo cenário que considera estabelece uma relação entre a distância de análise d e d_0 e o parâmetro do ambiente.

$$L_{TOT[dB]} = 10\log(\frac{4\pi d_0}{\lambda}) + 10n\log(\frac{d}{d_0}) \tag{2.5}$$

Os valores de n pode ser obtidos de maneira experimental para o ambiente em, mas existindo alguns padrões já testados para diferentes ambientes que são os mesmos demostrados na tabela 1 (LEE, 2010).

Tipo de ambiente	Valor do parâmetro n
Espaço Livre	2
Área urbana	2,7 à 3,5
Dentro de edifícios com linha visada	1,6 à 1,8
Dentro de edifícios com obstrução	4 à 6
Em fábricas	2 à 3

TABELA 1 – Valores típicos de n para os diversos ambientes

Para se estimar as perdas por sombreamento é mais interessante se acrescentar a equação 2.5 a variável X_σ (LEE, 2010) que representa as variações de média local em torno da média em área como demostrado na equação 2.6.

$$L_{TOT[dB]} = 10\log(\frac{4\pi d_0}{\lambda}) + 10n\log(\frac{d}{d_0}) + X_\sigma \tag{2.6}$$

2.4 ANÁLISE ESTATÍSTICA DA PERDA DO POTÊNCIA DO SINAL

Em comunicações *wireless* um bom planejamento de cobertura do enlace de radio é algo crucial para o com desempenho da rede. As topologias baseadas nesta tecnologia se tornam bastante complexas ao ponto que muitas das análises são probabilísticas e a medida que se deseja melhor determinação do cenário aumenta-se a complexidade do sistema de análise (SILVA; KOMATI, 2015). Pois diversos são os fatores que contribuem para a perda de potencia do sinal partindo da sua fonte emissora, principalmente em um ambiente *indoor* como: obstáculos fixos, paredes, pessoas e entre todos os objetos que contribuem para este fenômeno. Onde todos os parâmetros são relevantes para o melhor o melhor modelo de propagação condizente ao devido ambiente.

Para a determinação da perda de potencia e suas variações são usualmente utilizados três modelos: Rayleigh, Rice e log-normal (PUC, 2016). Que serão apresentados nos próximas sub-seções desta seção.

O canal de rádio para ambientes fechados é caracterizado por três características:

- Dependência com a distância;
- Variabilidade de larga escala;
- Variabilidade de pequena escala.

A dependência com a distancia está relacionada ao quanto o nível de potência do sinal diminui a medida que este avança no espaço. Todo sinal quando propagado em um canal reduz a sua densidade de potência devido ao espalhamento do sinal no espaço (KUPPER, 2005). Para o caso, como o analisado, de um canal sendo o ar e este sem nenhuma obstrução ou reflexões, empiricamente foi definido que o sinal sofre atenuação com relação quadrática da distância. Sendo que em ambientes com vários objetos que podem obstruir a passagem do sinal, está relação ser maior ainda. Assunto debatido na seção 2.3.

A variabilidade de larga escala está associada as flutuações recorrentes do nível de potência do sinal sobre o seu valor médio, muito devido a perda de potência causada por elementos presentes no próprio ambiente. Está variabilidade também é conhecida como efeito de sombreamento (KUPPER, 2005).

A variabilidade de pequena escala está muito associada a perda de potência causada pelo próprio sinal nele mesmo, um efeito muito comum em comunicação *wireless* em ambientes fechados (KUPPER, 2005). Este efeito é conhecido como efeito de múltiplos caminhos, onde as frequências mais elevadas são mais susceptíveis a este efeito. Em um ambiente fechado o sinal pode sofrer diversas reflexões e se construir diferentes formas de onda que chegam ao receptor em diferentes períodos de tempo. Estas ondas chegando em diferentes períodos de tempo causam interferência nelas mesmas.

2.4.1 Modelo de Rayleigh

O modelo de Reyleigh parte do princípio que o sinal que chega a antena receptora é uma combinação de sinais que sofreram diferentes atenuações e caminhos ao longo do percurso causando ao sinal recebido uma combinação de todos estes sinais, de maneira construtiva ou destrutiva, a onda recebida. Onde também é considerado que as fases das ondas são espalhadas de maneira independente umas das outras em uma distribuição uniforme de 0 a 2π. A distribuição de Rayleigh é mais aconselhada onde se deseja analisar melhor a interferência de múltiplos percursos no ambiente (PUC, 2016).

Portanto o sinal recebido obedece a distribuição segundo a distribuição de Rayleigh apresentada pela equação 2.7.

$$p(r) = \frac{r}{\sigma^2} \exp(-\frac{r^2}{2\sigma^2}) \tag{2.7}$$

Onde:

- σ - valor rms do sinal recebido antes da detecção do envelope;
- σ^2 - potência média no tempo do sinal recebido antes da detecção do envelope.

A probabilidade do envelope não exceder um determinado valor de R é é dada pela função de distribuição cumulativa seguinte a equação 2.8.

$$P(R) = P_r(r \leq R) = \int_0^R p(r)dr = 1 - \exp\,(-\frac{R^2}{2\sigma^2}) \tag{2.8}$$

O valor médio r_{mean}, a variância σ_r^2, e o valor mediano r_{median} da distribuição de Rayleigth são dados pela equação 2.9.

$$r_{mean} = 1,2533\sigma; \sigma_r^2 = 0,4292\sigma^2; r_{median} = 1,177\sigma \tag{2.9}$$

Como geralmente os sinais são representados em dBm, a equação de Rayleigh pode ser representada da sehuinte forma segundo a equação 2.10.

$$p_y(y) = \frac{1}{2\sigma^2}\frac{ln10}{10}10^{\frac{y}{10}}\exp(-\frac{10^{\frac{y}{10}}}{2\sigma^2}) - \infty < y < \infty \tag{2.10}$$

2.4.2 Modelo de Rice

Os modelos de comportamento estatístico do sinal podem atuar de maneira satisfatória em cada tipo de ambiente ou características. O modelo de Rice em comparado ao do Rayleigh é um desses exemplos. O modelo de Rayleigh considera que os percursos indiretos são dominantes em relação aos percursos diretos já em contrapartida o modelo de Rice, leva em consideração que o percurso direto, a onda visada, é dominante no sistema (KUPPER, 2005). O envelope do sinal é representado pela seguinte equação 2.11.

$$p(r) = \frac{r}{\sigma_r^2}\exp\left(-\frac{r^2+a^2}{2\sigma_r^2}I_0(\frac{ar}{\sigma_r^2})\right) \tag{2.11}$$

Onde $I_0(\frac{ar}{\sigma_r^2})$ é calculado pela seguinte equação 2.12 que é a função de Bessel modificada de ordem zero.

$$I_0(\frac{ar}{\sigma_r^2}) = \frac{1}{2\pi}\int_0^{2\pi}\exp(\frac{\arccos\theta}{\sigma_r^2})d\theta \tag{2.12}$$

Analisando a equação 2.11 pode-se notar que a medida que se anula a variável a a função se assemelha da equação de Raleigh.

Para a distribuição Riciana é definida um fator K, que nada mais é do que uma relação entre a potência do sinal dominante e a potência do sinal espalhado sendo definido pela equação 2.13.

$$K = \frac{a^2}{2\sigma_r^2} \tag{2.13}$$

A análise do fator K mostra dois resultados interessantes, quando K tende a zero a distribuição tende a equação de Rayleigh e quando K começa a assumir valores muito elevados a distribuição tende a uma Gaussiana, pois não existem componentes espalhadas (KUPPER, 2005).

Do mesmo modo que a equação de Rayleigh, muitas vezes é mais interessante se trabalhar em termos logarítmicos na equação de Rice, portanto assumindo $X = 20\log(r)$ e $M = 10/ln(10)$ tem-se a equação 2.14.

$$p(X) = \frac{1}{2M\sigma_r^2}\exp[\frac{X}{M} - \frac{1}{2\sigma_r^2}\exp\frac{X}{M}]\exp(-K)I_0[\frac{\sqrt{2K}}{\sigma_r}\exp(\frac{X}{2M})]; \quad (2.14)$$

2.4.3 Modelo Log-Normal

Como já mencionado, uma onda de radio ao atingir uma estação móvel ao longo de seu caminho sofreu diversos tipos de reflexões, atenuações, mudanças de fase sendo que cada objeto presente neste caminho possui uma contribuição para a onda que chega ao receptor (PUC, 2016). Para o modelo Log-Normal o sinal recebido R, medido em decibéis, possui a seguinte forma demonstrada pela equação 2.15

$$p(R) = \frac{1}{\sqrt{2\pi}\sigma_r}\exp[-\frac{1}{2}(\frac{R - M_R}{\sigma_r})^2] \quad (2.15)$$

Onde: M_r é a média de R e σ_r^2 sua variância.

A distribuição Log-Normal é mais usualmente utilizada em regiões ou ambientes em que o sinal sofra maiores interferências por causa da obstrução de objetos, locais de sombra (PUC, 2016).

A função de distribuição de probabilidade trás resultados interessantes no que se diz a respeito a probabilidade de um sinal recebido ser menor que um determinado valor, sendo está representada pela equação 2.15.

$$P(R_0) = prob(R \leq R_0) = \int_{-\infty}^{R_0} p(R)dr \quad (2.16)$$

2.5 TÉCNICAS PARA A MEDIÇÃO DE DISTÂNCIA ATRAVÉS DE ONDAS ELETROMAGNÉTICAS

Para se estabelecer uma posição relativa de um objeto a um outro é necessário se saber as posições destes objetos assim pode-se saber o quanto um objeto está distante de um outro. Sendo que também em contrapartida se é sabido a distância entre dois objetos a o angulo de sua posição em relação ao outro ou a distancia de um ponto especifico a outros objetos cujas posições são conhecidas, é possível se identificar a posição deste objeto de análise seguindo algumas regras da matemática cartesiana (BENSKY, 2008).

Em tudo este capítulo cuidará de apresentar técnicas mais comumente utilizadas para a obtenção de distâncias entre a estação móvel e as estações rádio base, para assim por meio de técnicas apresentadas no capítulo 2.6, apresentado posteriormente, sejam capaz de se encontrar a posição da estação móvel.

2.5.1 Tempo de chegada

O principio de funcionamento do ToA (*Time of Arrival*), se baseia no principio de calcular o tempo de propagação que o sinal leva para sair da fonte transmissora e chegar a fonte receptora, sendo assim sabendo a velocidade do sinal e se medido o tempo de transmissão se pode estimar a distância segundo regras da física (ALBACORE, 2011).

As ondas eletromagnéticas movem-se no ar próximas a velocidade da luz ($c = 3.10^8 m/s$). Para se conseguir possuir uma boa resolução da medida da diferença do tempo, os osciladores dos dispositivos usados devem ser capazes de operar a uma frequência maior que a segundo a regra estabelecida pela equação 2.17.

$$f_{DISP} > f = \frac{c}{\Delta.d.2} \tag{2.17}$$

Além de ser necessário para uma para uma boa resolução da distância um oscilador com alta frequência, o mesmo também deve possuir um baixo escorregamento (*clock drift*) (BENSKY, 2008).

Para o calculo da distância usando este método, como já se descrito, é levado em consideração a quantidade de tempo necessária para o sinal sair da fonte emissora a fonte receptora, ou seja, de uma forma bastante simples seria necessário se analisar a diferença de tempo de $t1 - t2$. Por isso que se faz necessário possuir uma ótima sincronização dos relógios da fonte emissora e da fonte receptora, pois senão haverão erros no sistema (BENSKY, 2008). Sendo este o principal problema desta técnica.

2.5.2 Diferença do tempo de chegada

O ToA (*Time Difference of Arrival*), é um variante do ToA que utiliza a variável tempo como o seu principal fator de medição. Possuindo a diferença que está técnica utiliza a diferença de tempo de chegada das estações rádio base para se basear a distância que estas se encontram da estação móvel. Está técnica também é conhecida por posicionamento hiperbólico tridimensional (ALBACORE, 2011).

O princípio físico de funcionamento se baseia na sequencia de passos a seguir: a estação móvel emite um sinal em *broadcast* para as estações rádio base conectadas a ela e fica esperando os seus retornos, as estações radio base recebem o sinal enviado processam o sinal e emitem um novo sinal a estação móvel, por fim a estação

móvel analisá o diferença de tempo de chegada dos sinais, podendo assim descobrir as distâncias d_1 e d_2 , uma vez conhecidos os tempos t_1, t_2 e a velocidade do sinal.

Este princípio é conhecido como técnica do posicionamento hiperbólico tridimensional pelo fato de se poder utilizar as equações hiperbólicas para a sua resolução. Sendo que tomando duas estações radio base, no exemplo considerando somente as duas, como sendo os focos da hipérbole a diferença da distancia entre as duas sendo $d_1 - d_2$ e sendo igual a Δd. Considerando as estações radio base em posições fixas e de diferença de distância física de $D/2$ (BENSKY, 2008), equação 2.18 descreve a equação da hipérbole em termos de Δd e D.

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \longrightarrow \frac{x^2}{(\frac{\Delta d}{2})^2} - \frac{y^2}{(\frac{D}{2})^2 - a^2} = 1 \tag{2.18}$$

Para o caso do TDoA são necessários no mínimo três estações radio base para se conseguir desmitificar todas as variáveis do sistema, já que x e y são desconhecidos.

Os problemas relacionados ao TDoA são os mesmos relacionados ao ToA: sincronização de relógios, osciladores de alta frequência e de baixo escorregamento (BENSKY, 2008). Fatores que contribuirão na precisão do sistema.

2.5.3 Angulo de chegada

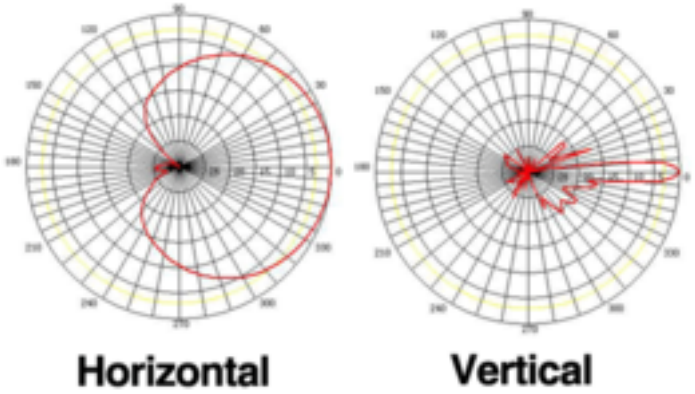
A técnica do AoA (*Angle of Arrival*) consiste em no principio de saber de onde (a direção) o sinal está atingindo a antena receptora. A grande vantagem desta técnica está no principio de não se precisar haver a sincronização entre os dispositivos e nem uma troca efetiva de dados entre os dispositivos, mas somente o angulo, como base a uma referencia feita na antena, do sinal recebido (ALBACORE, 2011).

Para a utilização desta técnica a dificuldade está em conhecer as propriedades técnicas da antena, sendo elas: diretividade, ganho, ângulo de meia potência e polarização (ALBACORE, 2011).

O principio para a obtenção da posição de localização utiliza-se da trigonometria. Onde sabido dois ângulos, os ângulos medidos pelas antenas, e a distância entre as antenas pode-se achar a posição da estação móvel, partindo do principio que se tem dois ângulos e um dos lados de um triângulo (PARK, 2010).

A figura 1 mostra uma antena com alta diretividade, e este é o maior problema desta técnica. Para se possuir uma boa resolução da posição da estação móvel a ser analisada é necessário um conjunto de antenas de boa qualidade que podem encarecer o hardaware com o nível de precisão requerido.

FIGURA 1 – Diagrama de uma antena com alta diretividade



Fonte: repositorio.ifpb.edu.br

2.5.4 Intensidade de sinal recebido

A técnica designada RSSI (*Received Signal Strength Indication*) é uma técnica bastante conhecida e usual principalmente na utilização de *Acces Points* (AP) para comunicação *wireless*. Ela utilizada da intensidade do sinal recebido para se estimar a distância que a fonte receptora se encontra da fonte emissora.

Um sinal que parte de sua fonte emissora e chega a fonte receptora parte com uma determinada potência e chega com uma potência menor do que a de partida. Conhecidos os modelos de ganho das antenas receptora e emissora, a intensidade do sinal de chegada e partida e o comprimento de onda a ser analisado, através de um modelo de perda de caminho adequado para o ambiente é possível ter uma aproximação da distância da estação rádio base a estação móvel (ALBACORE, 2011).

O modelo de RSSI é uma técnica da qual se extrai um valor relativo do nível de potencia recebida, ou seja, não apresenta um valor absoluto da potencia que chega ao dispositivo, mas o nível relativo de potência relativa entre a potência emitida e o quanto está sofreu atenuação ao longo do canal de comunicação.

Os modelos de RSSI possuem o grande problema no que se diz achar um modelo de perda de caminho ideal para o ambiente de aplicação. Pois como trabalham com a atenuação do sinal vários fatores devem ser considerados, como: atenuação causados por interferências, múltiplos caminhos, atenuação causado por objetos no ambiente que obstruem a onda visada entre as antenas e entre outros fatores (BENSKY, 2008). Diferentemente das outras técnicas a precisão do sistema está relacionada diretamente ao quão bem é feita o modelo de perdas de caminho estipulado para o ambiente, que pode ser bem complexa a sua análise.

2.5.5 Medição duplicada simétrica de mão dupla

A técnica de medição duplicada simétrica de mão dupla (do inglês *Symmetrical Double Sided Two Way Ranging - SDS-TWR*) é uma técnica que utiliza a medição do tempo de viagem do sinal de ida e volta nos dois sentidos, ou seja, primeiro é feita uma medida do tempo de viagem do sinal da estação móvel a estação radio base e voltando ao ponto de inicio e no outro momento é realizada a medida do tempo de viagem partindo da estação radio base a estação móvel e retornando a estação radio base (ALBACORE, 2011).

A grande vantagem deste tipo de técnica está no que se diz a respeito dos erros devido a deriva dos relógios dos dois sistemas, onde as derivas em cada caso são anuladas neles mesmos e o fato da eliminação da sincronização dos relógios nos pares de dispositivos (ALBACORE, 2011).

No entanto algumas questões devem ser levadas em consideração: que os pares de dispositivos (estação móvel e radio base) devem ser inteligentes ao ponto de se comunicarem em ambos os sentidos, a transmissão de dados se torna maior no sistema e a complexidade do software e sua boa análise se tornam um item critico para o sistema de localização (ALBACORE, 2011). Acrescentando alguns pontos a mais, mas sendo um método bastante similar ao ToA só que suprimindo as suas limitações.

2.6 ESTIMATIVA DA LOCALIZAÇÃO POR MEIO DAS DISTANCIAS ESTIMADAS DOS DISPOSITIVOS E A ESTAÇÃO MÓVEL

A distância encontrada entre uma estação radio base (por exemplo um modulo bluetooth) e uma estação móvel (por exemplo o celular de um usuário) é importante, mas essa medida de forma isolada não quer dizer muita coisa levando em consideração a localização referenciada. Se em um momento se tem a distância entre os dispositivos, deixando a estação rádio base como um objeto fixo, a estação móvel pode estar em qualquer posição que possua o mesmo raio de distância encontrado.

Assim como não se há precisão para se saber a real localização espacial especifica que o dispositivo (usuário) se encontra, foram desenvolvidas algumas técnicas que utilização, por meio das distancias entre as estações radio base, formas de se encontrar a posição espacial do usuário. Algumas destas técnicas serão apresentadas nas próximas seções.

2.6.1 Técnica de detecção unilateral

A técnica de unilaterização não é exatamente uma técnica usada eventualmente para se saber a localização de um usuário, mas sim saber se este usuário se encontra mais próximo de uma estação radio base ou de outra para que assim se possua uma

melhor comunicação entre os dois sistemas (BENSKY, 2008), os seja, o sistema irá se configurar para que a melhor comunicação possível se estabeleça entre as partes, já que está estará se comunicando somente com uma delas.

Este tipo de técnica é muito utilizada em sistemas de telefonia móvel. Em uma cidade, por exemplo, um dispositivo móvel pode estar recebendo sinais de diversas estações radio base (antenas) que se podem comunicar com o dispositivo em questão (LAU, 2008). O sistema realização uma estimativa do sinal, por seja qual método que se deseje, para se estimar a distancia do emissor ao receptor e assim estabelecer a melhor comunicação de dados.

2.6.2 Sensibilidade de proximidade

A técnica de sensibilidade de proximidade (do inglês, *proximity sensing*) é uma forma de localização bem simples, que não possui uma grande precisão de localização, mas que também não se é necessário a realização de cálculos matemáticos para se identificar a localização de um dispositivo e consequentemente o usuário.

O método de detecção baseia-se em uma localização simbólica da estação móvel, assim sabendo se o usuário está visível a estação rádio base ou não. Em outras palavras se existe uma comunicação efetiva entre o dispositivo móvel e a estação rádio base, quer dizer que os sistemas estão próximos um do outro, levando em consideração os limites de distancias de comunicação entre os dois dispositivos (BENSKY, 2008).

Este tipo de sistema é um dos mais simples que se pode obter de todas as técnicas, mas que também não trás informações precisas de localização. Sendo que em algumas aplicações o seu sistema pode ser implementado para algumas situações mais simples. Por exemplo, um campus universitário que possui diversos departamentos e um mesmo prédio e cada departamento é distribuído com uma certa quantidade de salas entre este prédio. Com o auxilio do sistema de localização de sensibilidade de proximidade o usuário poderia ter uma breve noção de qual bloco de departamento ele se encontra.

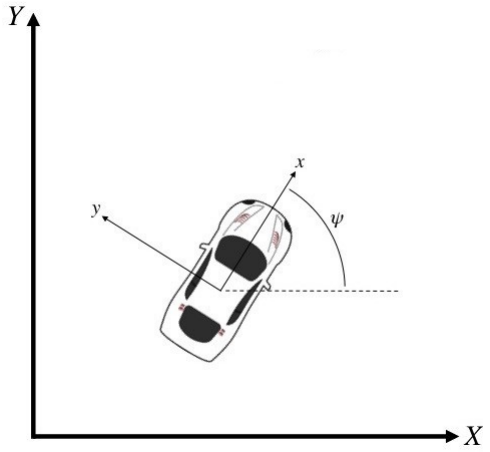
Uma outra utilização deste tipo de técnica, agora não voltado para a localização do usuário, é utilizada por algumas lojas para dispararem informações de marketing aos seus clientes. Onde por meio de aplicações e dispositivos de comunicação bluetooth, os usuários ao passarem perto a algumas lojas ou setores de uma grande rede, recebem informações de produtos, promoções dicas e entre outros dados.

2.6.3 Ponto morto

O método do ponto morto (que vem do inglês, *Dead reckoning*) é um sistema de determinação da posição futura levando em consideração a posição atual. Este

tipo de técnica é muito antiga, onde sua primeira utilização ocorreram durante as primeiras navegações. A utilização desta técnica ainda é usual, mas em algumas aplicações específicas como navegação de aviões, navios e automóveis juntamente a um sistema inteligente associado (BENSKY, 2008). Pode-se se exemplificar no casso dos automóveis a aplicação desta técnica, onde sabendo-se algumas variáveis de ambiente e prevendo o sua posição futura, pode-se prevenir uma colisão eminente e uma atuação do sistema para se evitar este caso.
Para exemplificar um pouco melhor está técnica, tomemos como base a figura 2.

FIGURA 2 – Utilização da técnica de ponto morto.



Fonte: SABUK, Abuk

A figura 2 mostra que sabendo-se a posição inicial, o angulo (α) a velocidade de deslocamento e se analisando para um tempo especifico pode-se descobrir a posição futura em que o objeto se encontrará, por meio da equação 2.19.

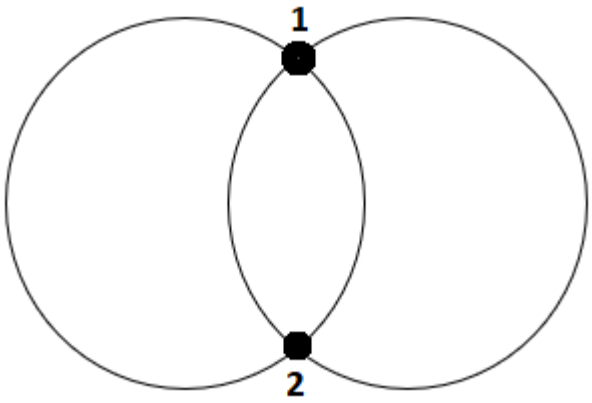
$$x_1 = x_0 + L.\cos\alpha, y_1 = y_0 + L.\sin\alpha \tag{2.19}$$

O grande problema desta técnica na utilização para a detecção de posição vem da questão que em muitas vezes a posição atualizada e estimada pode conter erros e estes são propagados aos próximos pontos, causando imprecisão ao sistema (PARK, 2010).

2.6.4 Lateração circular

A técnica de lateração circular é uma variação e evolução da técnica de unilateração. Na lateração circular (do inglês *circular lateration*) são necessárias duas estações rádio base para a detecção da posição da estação móvel. Está técnica em resumo cria um sistema de círculos, com base na distancia encontrada por alguma técnica, e a intersecção destes dois círculos será a posição aproximada da estação móvel (BENSKY, 2008), como mostra a figura 3.

FIGURA 3 – Técnica de lateração circular



Fonte: Fonte

A figura 3 mostra que sabido a distancia da estação móvel as estações rádio base, pode-se criar círculos virtuais de raios de distancia d_i para cada estação rádio base e as intersecções destes dois círculos virtuais pode causar uma boa aproximação da posição da estação móvel em meio a uma referencia estabelecida.

Para se saber a posição da estação móvel, é necessária a ajuda de cálculos matemáticos para se estimar a localização. Segundo o teorema de Pitágoras, se conhecidos os valores de x_i e y_i, que são as posições referenciadas das estações radio base em respeito as coordenadas cartesianas estabelecidas e obtida as distancias entre as estações radio base e a estação móvel, sendo d_i, pode-se calcular a posição aproximada da estação móvel conforme a equação 2.20.

$$d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2} \tag{2.20}$$

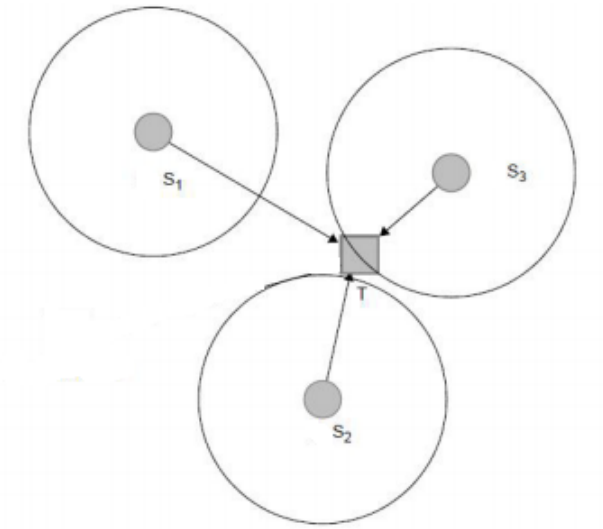
Para se saber a posição do dispositivo é necessário resolver a equação e o que no implica resolver a raiz quadrada. A técnica melhora os problemas de localização que existiam em outros modos, deixando agora a precisão relacionada a forma que se é obtida a distancia entre as estações rádio base e o dispositivo móvel, mas ainda havendo um pequeno problema para a distinção da real posição do usuário, pois na resolução da equação 2.20 se tem como resposta duas posições possíveis que ainda causam incerteza ao sistema (BENSKY, 2008).

2.6.5 Trilateração

Para se resolver o problema da lateração circular, onde ainda se havia a incerteza da posição da estação móvel pelo fato de se existirem duas raízes e proveniente duas posições, a técnica da trilateração com a medida de três distancias (entre estações radio base e a estação móvel) desmitifica a aparição de duas possíveis posições, como a figura 4 apresenta.

A utilização da técnica consiste na criação de três círculos virtuais, um a mais da técnica de lateração circular, para assim, por meio das distancias entre as estações radio base e a estação móvel, possa descobrir o ponto de intersecção entre os três círculos virtuais (BENSKY, 2008).

FIGURA 4 – Técnica de trilateração



Fonte: Adaptado de Albacore

Para a obtenção da posição do usuário dois métodos matemáticos são mais comumente utilizados sobre este tipo de técnica: a estimativa da máxima probabilidade (*maximum likelihood estimation - MLD*) ou a técnica da menor raiz (*least square - LS*) (BENSKY, 2008).

Neste caso de técnica a imprecisão do sistema não está mais relacionada em detectar a posição referenciada da estação móvel em um plano cartesiano, mas na técnica da detecção de distancia entre o dispositivo móvel e cada estação rádio base. Por isso uma boa estimativa da distância certamente causará uma boa precisão ao sistema (LAU, 2008).

2.6.6 Triangulação

Uma outra técnica bastante utilizada é a da triangulação (do inglês *triangulation*). Diferente dos métodos anteriores está técnica utiliza ângulos para a localização espacial do fonte móvel. Geralmente este método é utilizado com a técnica de AoA por se utilizar de ângulos para a obtenção da direção da fonte emissora.

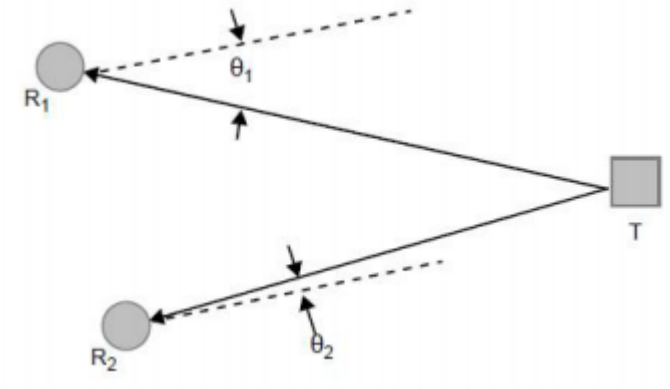
O principio da técnica é estabelecida nas leis da trigonometria plana, que onde se sabendo dois ângulos de um triangulo e um lado ou dois lados e um angulo as

outras incógnitas podem ser descobertas segunda a equação 2.21 (PARK, 2010).

$$\frac{A}{\sin a} = \frac{B}{\sin b} = \frac{C}{\sin c} \tag{2.21}$$

A figura 5 resume o método de localização por meio da triangulação. Para este tipo de sistema podem ser utilizadas duas diferentes técnicas. Uma delas consiste em se saber os ângulos θ_1, θ_2 e B e assim utilizar a equação dos senos, segunda a equação 2.21 e a lei dos cossenos segundo a equação 2.22 (PARK, 2010).

FIGURA 5 – Técnica da triangulação para a obtenção da posição



Fonte: Adaptado de Albacore

$$C^2 = A^2 + B^2 + 2.A.B.\cos c; B^2 = A^2 + C^2 + 2.A.C.\cos b; A^2 = B^2 + B^2 + 2.B.C.\cos c \tag{2.22}$$

E uma segunda maneira de se achar todas as variáveis do sistema é se saber as posições referenciadas das estações rádio base e os seus ângulos θ_1 e θ_2, assim aplicando-se as equação 2.23 e posteriormente a equação 2.24 pode-se encontrar as coordenadas referenciadas.

$$y_{EM} = \frac{(y_2.\tan\theta_2 - x_2)}{(\tan\theta_2 - \tan\theta_1)} \tag{2.23}$$

$$x_{EM} = y.\tan\theta_1 \tag{2.24}$$

Como mencionado está técnica é utilizada juntamente com a detecção da angulação de onde o sinal está chegando as antenas, por meio do auxilio do método da AoA. Devido a essa questão a precisão do sistema estará em direta relação na qualidade de precisão do sistema de antenas, onde antenas que consigam fazer uma boa detecção da origem do sinal contribuíram para uma melhor precisão do sistema (PARK, 2010).

2.7 AD-HOC

A rede ad-hoc, composta principalmente por dispositivos móveis, não depende da infraestrutura criada por outros sistemas de telecomunicação. Cada nó atua como transmissor e receptor, não existindo dispositivo distribuidor de pacotes, dotando a rede de comportamento auto-organizável (HEKMAT, 2006). As redes de tipo ad-hoc são comumente dividas em dois grupos:

- *Single Hop*: Distância de um nó entre transmissor e receptor;
- *Multi-hop*: Distância entre receptor e transmissor maior do que um nó.

A aplicação da técnica ad-hoc é justificável em situações em que não há infraestrutura presente ou o limite orçamentário do projeto é baixo (HEKMAT, 2006). Devido a natureza auto-configurável as redes ad-hoc *multi-hop* não são dependentes da linha de visada, efeito limitador em sistemas de comunicação, como o 3GPP ou o LTE (HEKMAT, 2006).

2.7.1 *Table-driven*

Algoritmos baseados em *table-driven* mantém todas as rotas possíveis mapeadas em diferentes tabelas, a medida que a rede muda sua configuração, as tabelas são atualizadas(YADAV; YADAV, 2007). Exemplo de protocolos do tipo *table-driven* são: DSDV, WRP, CGSR e STAR.

2.7.1.1 *Destination Sequenced Distance-Vector*

O algoritmo *distance vector* por si só gera loops, resultando em ciclos entre dois ou mais nós, sem que a mensagem atinja o destinatário. A proposta do DSDV é mitigar o problema de loops, presentes nas versões mais antigas de roteamento.

O protocolo DSDV (*Destination Sequenced Distance-Vector*), baseado no mecanismo *Distance Vector* de Bellman-Ford, depende de trocas periódicas das informações de roteamento entre os nós, a informaçao de conectividade de toda a rede é propagada a outros nós, assim, cada dispositivo da rede tem toda a informação de roteamento.

Para efetuar modificações na tabela de roteamento, o nó que recebe a atualização verifica se a informação sobre a nova configuração da rede é mais nova do que a armazenada no momento, caso sim, as informações são atualizadas. Caso os dados não sejam mais novos, só ocorre atualização se as rotas são melhores do que as registradas atualmente (ERGAWY, 2006).

2.7.2 *On-demand*

Sistemas de roteamento baseados em modelos *on-demand* não armazenam as rotas de comunicação com os dispositivos, nem participam na troca de tabelas de roteamento. Um nó só precisa descobrir e manter a rota até outro no momento que ambos necessitem trocar informações (PERKIN CHARLES; ROYER, 2002). O protocolo AODV (*Ad-Hoc on-demand distance vector*), algoritmo de roteamento do tipo on demand, funciona com o envio das mensagens:

- Requisições de rotas (RREQs);
- Requisições de respostas (RREPs);
- Erros de rotas (RERRs).

O nó solicitante usa seu endereço IP como o rementente da mensagem, enquanto o destinatário recebe o endereço 255.255.255.255, o que na camada de rede equivale ao endereço *broadcast* da rede, ou seja, todos os dispositivos receberão esta mensagem. O AODV atua no momento em que o destino da mensagem é novo, o nó transmite um RREQ, a fim de encontrar a rota.

O caminho está disponível quando a origem recebe uma mensagem RREP de volta a origem do RREQ (ARAUJO et al., 2007). Além do AODV também existem outros modelos on-demand, como: DSR, TORA e ARB.

2.8 *BLUETOOTH*

O *Bluetooth* é um padrão de tecnologia sem fio para troca de dados em distâncias curtas (usando ondas de rádio UHF de comprimento de onda curto, na faixa ISM de 2,4 a 2,485 GHz) de dispositivos fixos e móveis e construção de redes de área pessoal, ou *networking area personal* (PAN).

A tecnologia foi originalmente concebida como uma alternativa sem fio para cabos de dados RS-232 (ALBERT HUANG, 2005).

O IEEE o padronizou como IEEE 802.15.1. O *Bluetooth* SIG supervisiona o desenvolvimento da especificação, gerencia o programa de qualificação e protege as marcas registradas. A comunicação é baseada no princípio de mestre-escravo, cada mestre controla até sete equipamentos, a rede formada pelos equipamentos conectados recebe o nome de *piconet* formando uma conexão em estrela. O controlador administra a frequência e a alocação de canais, os escravos não podem estabelecer comunicação direta (LABIOD; AFIFI; SANTIS, 2007). A camada física do protocolo é dividida em unidades de tempo, conhecidas como *slots*, utilizadas para transmissão das mensagens.

No projeto será utilizado o *Bluetooth Low Energy (BLE)* ou Bluetooth 4.0 . O Bluetooth clássico e o BLE são muito similares, porém a principal diferença entre os dois é que o BLE envia menos dados e tem um baixo consumo de energia, já o Bluetooth clássico é mais focado para a transmissão de muitos dados.

2.9 MULTIPLEXAÇÃO DE CANAIS DE COMUNICAÇÃO

A técnica de multiplexação nada mais é que utilizar vários canais de informação em um único meio de transmissão. Canais de informação que não possuem relação entre si, em muitas das vezes, transmitindo a informação necessária entre as partes de maneira simultânea de modo que a interferência entre os canais de informação seja a mínima possível (MOECKE, 2017). A multiplexação é uma técnica que é adotada em dois casos: onde não se possuem uma grande quantidade de canais de informação ou quando mesmo que possível a existência de vários canais de comunicação a técnica se torna mais viável economicamente.

Diversas são as formas de multiplexação em sistemas de comunicação onde podem ser citadas as mais comuns:

- FDM - *Frequency Division Multiplexing*;
- TDM - *Time Division Mutiplexing*;
- WDM - *Wavelength Division Multiplexing*;

Na próxima sub-seção será desacorrida um pouco mais sobre o tipo de multiplexação TDM que utiliza a divisão do tempo para a utilização do canal de comunicação.

2.9.1 TDM (*Time Division Mutiplexing*)

Na multiplexação por divisão de tempo o canal transmissor é usado apenas por um canal de informação em um certo período de tempo, ou seja, as amostras de dados dos canais de informação em um determinado período de tempo são distribuídas periodicamente através de um processo de modulação por pulsos, onde estas amostras ficam intercaladas e periodicamente podem ser detectadas, tomando como base o mesmo canal de informação, a um período T definido (MOECKE, 2017).

A técnica de TDM faz com que os canais fiquem separados no tempo e e sobrepostos na frequência.

Um cuidado que se tem que deve se possuir a utilização desta técnica é o fato do receptor possuir a mesma sincronização de tempo de amostragem do transmissor, para que este possa reconstruir o sinal, dos vários canais de informação, de maneira

mais adequada possível, pois caso contrário sinais de dados de diferentes canais de informação serão computados como mesmos criando um dado errôneo, misturado ou interferente (MOECKE, 2017). Por isso são enviados constantemente sinais pilotos de sincronização.

Existem duas formas de multiplexação por divisão de tempo principais:

- **Multiplexação síncrona no tempo:** onde é alocada uma janela de tempo para o canal de informação independente se este possui dados para transmitir. Possui o desperdício na transmissão;
- **Multiplexação assíncrona no tempo:** onde é reservado o canal de transmissão conforme a demanda de dados dos canais de informação é necessária, onde envia primeiramente o endereço do canal de informação e posteriormente os dados.

A grande vantagem no utilização da técnica do TDM está na possibilidade da regeneração da informação transmitida durante a transmissão e uma maior imunidade a presença de ruído. Já a desvantagem do uso da técnica está associada ao fato desta possuir a necessidade de uma largura de banda maior se comparada aos outros sistemas.

2.10 APRENDIZADO DE MÁQUINA

Para se começar a pensar sobre aprendizado de maquina primeiramente deve se entender o que é aprendizado. Aprendizado é a capacidade de se adaptar, modificar e melhorar o seu comportamento e respostas seguindo os seguintes critérios (OSÓRIO, 2017):

- Adaptação - atender as mudanças do meio modificando o seu comportamento e evoluindo conforme estas mudanças ocorram;
- Correção - dos erros do passado, se criando alternativas para estes não serem cometidos no futuro novamente;
- Otimização - mudança de comportamento que não trazem bons resultados e gastam recursos desnecessariamente;
- Interação - o meio pode ser uma fonte de conhecimento;
- Representação - o sistema deve ser capaz de armazenar o conhecimento para que se possam ser explorado os novos dados fornecidos.

Tomado o conhecimento do que é aprendizado de uma maneira genérica, pode-se dizer que aprendizado de máquina nada mais é do que a implementação da definição de aprendizado de maneira digital, ou seja, se define como um sistema que pode modificar o seu comportamento atonamente com base no seu conhecimento adquirido com a miníma interferência humana possível, sendo que a tomada de decisões e modificação de comportamento é originada de um conjunto de regras e padrões estabelecidos (OSÓRIO, 2017).

Em muitas das vezes o aprendizado é um processo gradual é iterativo, onde o aprendizado é adquirido por meio de um conjunto de dados (dados de aprendizado) que funcionam como uma base de exemplos para as tomadas de decisões e sendo que cada dado pode possuir significâncias diferentes (pesos) para as tomadas de decisões. Os métodos de aprendizado de máquina podem ser de três maneiras (KROGH; VEDELBY, 1995):

- **Aprendizado supervisionado:** é um tipo de aprendizado onde o usuário ensina a rede em um primeiro momento. Este possui um conjunto de dados de referencia e os aplica a rede e esta é deve ser capaz de verificar o seu estado e aprendizado e o seu estado atual e corrigir os pesos de maneira a reduzir o erro;
- **Aprendizado semi-supervisionado:** neste tipo de aprendizado somente se tem informações qualitativas sobre o comportamento final desejado sem a possibilidade de se medir quantitativamente o erro;
- **Aprendizado não-supervisionado:** este tipo de análise se baseia na estatística para as tomadas de decisões, onde os pesos são rearranjados em função de critérios internos.

Existe uma linha tênue entre o aprendizado e o processamento de novos dados, onde o equilíbrio entre as duas partes é o mais indicado, tal fato conhecido como ponto ótimo (KROGH; VEDELBY, 1995). Onde uma grande quantidade de dados para o treinamento deixará o sistema mais robusto a novos dados, mas se requerera uma grande quantidade de tempo e dados para o processamento e um pequeno treinamento possuirá necessitará um grande processamento de dados para se chegar a resposta.

Na próxima sub-seção será tratado uma especie de aprendizado de máquina que será usado no projeto dentre uma grande gama de sistemas que poderiam ser utilizados dentro do contexto de aprendizado de máquina.

2.10.1 K-NN (*K-Nearest Neighbors*)

A técnica do K-NN (*K-Nearest Neighbors* é um método de aprendizado de máquina supervisionado, onde observando-se alguns pares de exemplos de entradas

e saídas, é se formada uma função que mapeia a entrada para a saída, ou seja, é mostrado ao sistema uma resposta correta na fase do seu treinamento para determinadas entradas escolhidas e quando o sistema já está treinado é disponibilizado outras entradas que resultam em determinadas saídas conforme seu treinamento foi construído (GRUS, 2016).

O K-NN é dos algorítimos mais simples de classificação que se pode encontrar, devido a grande complexidade de alguns algorítimos de classificação (GRUS, 2016). Este tipo de técnica consiste em um classificador onde o aprendizado é baseado na analogia de seus elementos, em outras palavras, é um sistema que usa de um conjunto de treinamento de exemplos e seleciona um conjunto de elementos do espaço que estão mais próximos dentro das mesmas características. E para se determinar um terceiro elemento que não faz parte do conjunto de elementos de treinamento, o classificador K-NN procura K elementos do conjunto de treinamento que possuam a menor distancia deste ponto (GRUS, 2016). Assim se designando de forma análoga de K-Vizinhos Próximos.

Algumas boas práticas para se utilizar o K-NN são sempre bem vindas, como:

- Um conjunto de exemplos de treinamento;
- Se utilizar de uma métrica para se calcular a distância entre os elementos;
- Definir o K para que o algorítimo saiba quantos vizinhos serão considerados no calculo.

Como se quer identificar um elemento desconhecido por meio de um conjunto de treinado de elementos, os passos a seguir devem ser realizados:

- Calcular a distância entre o ponto desconhecido e os exemplos do conjunto dos elementos;
- Identificar os vizinhos mais próximos, com a premissa do numero K estabelecida;
- Utilizar os valores dos vizinhos mais próximos para se estimar o ponto desconhecido.

Para se determinar a distância entre o ponto desconhecido e os exemplos do conjunto três técnicas são mais conhecidas como distância: Euclidiana, Manhattan e Minkowski, apresentadas pelas equações 2.25, 2.26 e 2.27 respectivamente,

considerando um conjunto de pontos $X = (x_1, X_2, ..., x_n)$ e $Y = (y_1, y_2, ..., y_n)$ (GRUS, 2016).

$$d(x, y) = \sqrt{(x_1 - y1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{2.25}$$

$$d(x, y) = |(x_1 - y1)| + |(x_2 - y_2)| + ... + |(x_n - y_n)| \tag{2.26}$$

$$d(x, y) = (|x_1 - y1|^q + |x_2 - y_2|^q + ... + |x_n - y_n|^q)^{\frac{1}{q}} \tag{2.27}$$

Se em alguns casos alguns pontos forem mais relevantes que outros, no sistema de aprendizado, pode se atribuir pesos aos pontos dentro das equações 2.25, 2.26 e 2.27.

O método do K-NN é um classificador livre que apresenta um único parâmetro livre o valor de K. Onde neste caso se escolhendo um valor de K muito pequeno a classificação fica sensível a ruídos e em escolhendo um valor de K muito grande, a vizinhança pode incluir elementos irrelevantes ao calculo desejado.

Para o Algoritmo do K-NN a precisão depende fortemente do modelo de dados e que na grande maioria das vezes necessita ser normalizados, para que determinados resultados não se sobre-saiam sobre outros (GRUS, 2016).

2.11 AJUSTAMENTO POR OBSERVAÇÃO

A realização de medidas, seja elas quais forem, estão susceptíveis a erros de medição. Estes erros podem ser decorrentes de diversos fatores e de diferentes formas dependendo do tipo de medida que se está realizando como: erro humano, imperfeições dos equipamentos, ruídos em alguns casos e entre os formas que podem causar imprecisão ao sistema (GEMAEL; MACHADO, 2015).

O sistema conterá erros, grandes ou pequenos, e não se levar em consideração este fato pode causar uma discrepância de resultados ainda maior.Por isso os métodos de ajustamento por observações cuidam de tratar este problema bem como de estimar a precisão da para a solução adotada, sendo a técnica dos mínimos quadrados uma destas técnicas.

2.11.1 Técnicas dos mínimos quadrados

Utilizando para meios explicativos o sistema dos mínimos quadrados para um sistema de referencia da distancia de dois pontos, o qual será o modelo que será

discutido em seções posteriores, dado pela equação 2.20 e acrescentando um ponto de resíduo na medida v_{ij}, pode se escrever a equação 2.20 da forma da equação 2.28.

$$l_{ij} + v_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2.28}$$

Onde j é o índice do ponto de que se está querendo saber a coordenada e i o índice do ponto observador. Portanto a equação 2.28 é uma função que leva em consideração a distância entre os dois pontos l_{ij} e as coordenadas dos pontos x_i, y_i, x_j e y_j podendo ser reescrita como a equação 2.29.

$$F(x_i, y_i, x_j, y_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2.29}$$

A equação 2.29 pode ser resolvida segundo a serie de Taylor de primeira ordem dado pela equação 2.30.

$$F(x_i, y_i, x_j, y_j) = F(x_{i0}, y_{i0}, x_{j0}, y_{j0}) + (\frac{\delta F}{\delta x_i})_0 dx_i + (\frac{\delta F}{\delta y_i})_0 dy_i + (\frac{\delta F}{\delta x_j})_0 dx_j + (\frac{\delta F}{\delta y_j})_0 dy_j \tag{2.30}$$

Onde x_{i0}, y_{i0}, x_{j0} e y_{j0} são os valores aproximados das incógnitas x_i, y_i, x_j e y_j, e $(\frac{\delta F}{\delta x_i})_0$, $(\frac{\delta F}{\delta y_i})_0$, $(\frac{\delta F}{\delta x_j})_0$ e $(\frac{\delta F}{\delta y_j})_0$ as derivadas parciais de F em relação a x_i, y_i, x_j e y_j. Já os valores dx_i, dy_i, dx_j e dy_j são as correções às aproximações iniciais, de modo que satisfaz a relação das equações 2.31 (GEMAEL; MACHADO, 2015).

$$x_i = x_{i0} + dx_i, y_i = y_{i0} + dy_i, y_j = y_{j0} + dx_j, y_j = y_{j0} + dy_j \tag{2.31}$$

E sendo as respostas das equações parciais sendo dadas pelas equações 2.32 e 2.33 (MENDONÇA et al., 2010). ,

$$\frac{\delta F}{\delta x_i} = \frac{x_i - x_j}{d_{IJ}}; \frac{\delta F}{\delta y_i} = \frac{y_i - y_j}{d_{IJ}} \tag{2.32}$$

$$\frac{\delta F}{\delta x_j} = \frac{x_j - x_i}{d_{IJ}}; \frac{\delta F}{\delta y_j} = \frac{y_j - y_i}{d_{IJ}} \tag{2.33}$$

Onde substituindo as equações 2.32 e 2.33 na equação 2.30, se resulta em uma equação de distâncias lineares segundo a equação 2.34 (GEMAEL; MACHADO, 2015).

$$\frac{x_{i0} - x_{j0}}{(d_{IJ})_0} dx_i + \frac{y_{i0} - y_{j0}}{(d_{IJ})_0} dy_i + \frac{x_{j0} - x_{i0}}{(d_{IJ})_0} dx_j + \frac{y_{j0} - y_{i0}}{(d_{IJ})_0} dy_j = k_{ij} + v_{ij} \tag{2.34}$$

Onde $k_{ij} = l_{ij} - (d_{IJ})_0$ e $d_{IJ} = \sqrt{(x_{j0} - x_{i0})^2 + (y_{j0} - y_{i0})^2}$.

Partindo do proposta que para se utilizar a técnica de trilateração necessita-se conhecer pelo menos três pontos e a estes sendo ajustados pelo método dos mínimos quadrados, pode-se escrever a equação 2.34 em sua forma matricial dada pela equação 2.35 que é regida por $AX + L = V$ (GEMAEL; MACHADO, 2015).

$$\begin{pmatrix} \frac{x_{u0}-x_a}{(d_{AU})_0} & \frac{y_{u0}-y_a}{(d_{AU})_0} \\ \frac{x_{u0}-x_b}{(d_{BU})_0} & \frac{y_{u0}-y_b}{(d_{BU})_0} \\ \frac{x_{u0}-x_c}{(d_{CU})_0} & \frac{y_{u0}-x_c}{(d_{CU})_0} \end{pmatrix} \begin{pmatrix} dx_u \\ dy_u \end{pmatrix} + \begin{pmatrix} (d_{(AU)})_0 - l_{au} \\ (d_{(BU)})_0 - l_{bu} \\ (d_{(CU)})_0 - l_{cu} \end{pmatrix} = \begin{pmatrix} vl_{au} \\ vl_{bu} \\ v_{cu} \end{pmatrix} \tag{2.35}$$

2.12 *ANDROID*

A plataforma *Android* é baseada no sistema Linux e atualmente é mantida pela google. O sistema é empregado principalmente em *smartphones* e devido ao alcance mundial se tornou um dos maiores ecossistemas vigentes para desenvolvimento de aplicações. Cada aplicativo é reconhecido como um usuário, sendo assim múltiplos usuários estão utilizando recursos do sistema. Os processos são identificados por números de identificação (ID) e a execução ocorre sobre máquinas virtuais, isolando eventuais falhas generalizadas (DEVELOPERS, 2018).

A fim de minimizar a interferência entre os processos executados pelo *hardware* o *Android* aplica o princípio do privilégio mínimo, em que os recursos alocados aos processos são os mínimos necessários, assim a disputa entre processos é mitigada. Os blocos básicos para a construção de aplicativos no ambiente são: *activity*, *services*, *content providers* e *transmission receptors*. As *activities* representam a telas de execução dos aplicativos, são basicamente as interfaces de usuário, seu ciclo de vida é representado pela Figura 6 e apresentam similaridade com as funções da linguagem C, pois são armazenadas em uma pilha do tipo FIFO (*First in First out*) à medida que são fechadas, ao tocar no botão *return* do *smartphone*, ocorre a saída da pilha e a destruição do processo pelo sistema operacional (DEVELOPERS, 2018).

A criação é realizada através do método *onCreate()* etapa que deve ser realizada as configurações básicas da *activity*, seguido pelo *onStart()* que realiza a preparação para o próximo método. O *onResume()* é utilizado em casos que a execução foi para o *background*, o campo verde da Figura 6 é o momento em que a *activity* está em execução no plano principal do usuário, ao deixar o plano principal o *onPause()* é executado, caso a *activity* saia da tela irá para o *onStop()* e se o usuário encerrar a execução a *activity* será destruída pelo *onDestroy()* (DEVELOPERS, 2018).

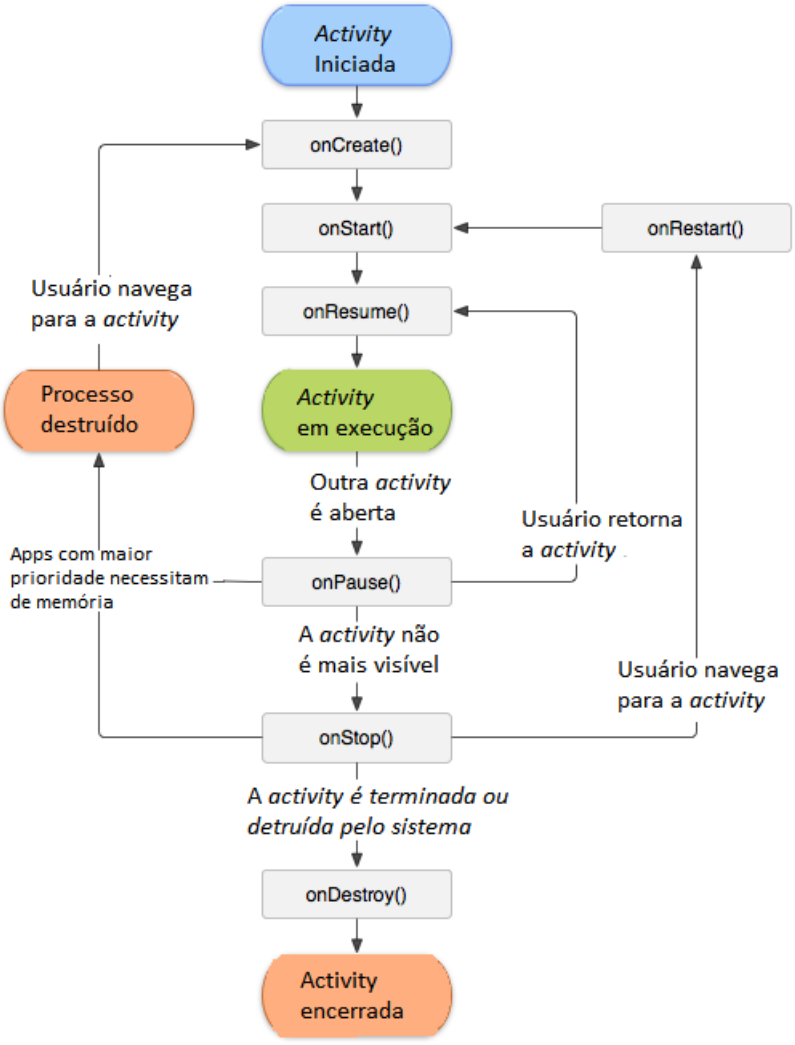
Os *services* são caracterizados por processos que rodam ao fundo sem que o usuário o veja. Os *content providers* são responsáveis pelo gerenciamento de conteúdo

compartilhado, enquanto que os *transmission receptors* lidam com as notificações do sistema (DEVELOPERS, 2018).

A vantagem do sistema operacional é a reutilização de ferramentas já criadas, como aplicativos que utilizam a câmera não necessitam escrever seu próprio algoritmo de gerenciamento, basta acessar os recursos do sistema, tal característica é possível devido ao uso da tecnologia de *intents* que se resumem a chamadas assíncronas ao sistema indicando a necessidade do uso de recursos, caso o aplicativo tenha a permissão necessária o sistema fará a ponte de comunicação entre os processos de diferentes apps (DEVELOPERS, 2018).

Dos quatro blocos de construção básicos a *intent* realiza a chamada de três deles: a *activity*, o *service* e o *transmission receptor*. O *content provider* só é chamado através de *querys* em objetos do tipo *content resolver* (DEVELOPERS, 2018).

FIGURA 6 – Ciclo de vida da *activity*



Fonte: Alterado de Android developers.

As formas de ocorrerem a iniciação dos blocos são representadas pelos tópicos

abaixo:

• *Activity*: Passar uma *intent* ao método *startActivity()* ou *startActivityForResult()*;

• *service*: Passar uma *intent* ao método *startService()* ou vincula-lo através da intente no método *bindService()*

• *transmission*: Passar uma *intent* aos métodos *sendBroadcast()*, *sendOrderedBroadcast()* ou *sendStickyBroadcast()*;

• *provider*: Consulta feita através da *query()* em um *content solver*.

O uso das *intents* é classificado como implícito ou explícito, caso o componente seja conhecido é passado como um dos parâmetros, tornando-a explícita, de forma contrária, o componente que executará a *intent* será definido pelo sistema operacional, tornando-a implícita, é comum que mais de um aplicativo seja capaz de lidar com a tarefa nesta ocasião a escolha será definida pelo usuário. Todo recurso, permissões, bibliotecas e APIs devem ser definidas no arquivo *manifest*, presente no diretório raiz do aplicativo, com exceção do *transmission receptor* que é declarado dinamicamente em tempo de execução (DEVELOPERS, 2018).

Além de codificação e componentes do ambiente, os aplicativos do sistema *Android* possibilitam a utilização de recursos exteriores, como imagens, áudios, leiautes, menus e animações o que tornam a experiência de usuário mais rica.

2.13 *BEACON*

O Beacon é um dispositivo de geolocalização que são utilizados em locais fechados para que usuário possa se localizar ou receber informações quando estiver com certa proximidade do aparelho. Os Beacons mais comuns no mercado são os Ibeacon e o Eddystone, sendo que o primeiro foi criado pela Apple e o posterior pela Google. Os dois são utilizados para o mesmo objetivo, porém possuem algumas diferenças citadas no quadro 2.

Quadro 1 – Diferentes dispositivos para localização Interna

	QRCode	NFC	GPS	Wifi	Beacon	Lidar	Ultrassom
Precisão de rastreamento	Alta	Alta	Baixa	Baixa	Alta	Alta	Alta
Esforço de manutenção	Fácil	Difícil	Difícil	Fácil	Fácil	Médio	Médio
Absorção da Industria	Médio	Médio	Baixo	Alto	Alto	Médio	Médio
Segurança e Privacidade	Médio	Alto	Alto	Médio	Alto	Alto	Médio
Custo de Instalação	Baixo	Baixo	Alto	Alto	Médio	Alto	Médio

Fonte: Adaptado de Ferreira, Resende e Martinho (2018)

Atualmente existe diversos dispositivos que podem ser utilizados para poder se localizar em um ambiente interno. O Beacon acaba sendo a melhor ferramenta para

ser utilizado em um local fechado, como pode ser observado no quadro 1.

Quadro 2 – Diferença entre iBeacon e Eddystone

	iBeacon(Apple)	Eddystone(Google)
Tecnologia	iBeacon é um protocolo de beacon que inicialmente foi colocado no iOS 7 da Apple e versões posteriores, permitindo assim que os iPhones e iPads escaneiem constantemente por dispositivos Bluetooth. O Beacon utiliza o *Bluetooth Low energy*, no qual faz parte da especificação Bluetooth 4.0	O Eddystone da Google é um protocolo de código aberto que pode ser produzido por qualquer empresa com o custo acessível.
Compatibilidade	É compatível com Android e iOS, mas é nativo somente no iOS	É compatível com qualquer plataforma que suporte o beacon BLE.
Perfil	É um software proprietário, portanto as especificações são controladas pela Apple.	É código aberto. A especificação é publicado abertamente no GitHub.
Facilidade de uso	Fácil de implementar	É flexível, porém necessita de uma codificação mais complicada quando se trota de integração, pois envia mais pacotes de informação que o iBeacon.
Pacotes transmitidos	Cada Beacon transmite informação que são identificados como um pacote. O iBeacon transmite apenas um pacote de publicidade que possui um número de identificação único composto por três partes - UUID, Maior e Menor.	Eddystone transmite três pacotes diferentes.

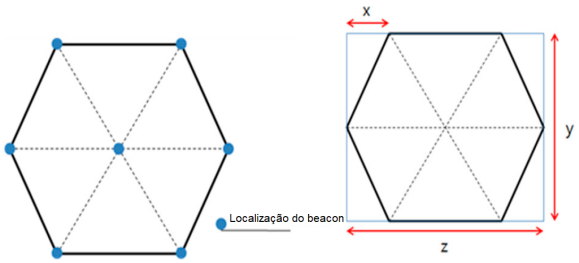
Fonte: Adaptado de Mittal (2016)

3 RESUMO ARTIGOS

3.1 AN INDOOR LOCATION-BASED CONTROL SYSTEM USING BLUETOOTH BEACONS FOR IOT SYSTEMS

Com o avanço da tecnologia, é descoberto com o tempo diversas maneiras de obter a localização indoor de um local com a utilização do bluetooth. Afim de obter uma precisão maior, é feito uma integração de várias métodos juntos. Um exemplo disto é apresentado no artigo "An Indoor Localization-Based Control System using Bluetooth Beacons for IoT Systems"(HUH; SEO, 2017). Eles utilizaram o método de Trilateração, *RSSI (Received Signal Strength Indication)*[1] e através dos localizadores (*beacons*) fazem o particionamento espacial do local.

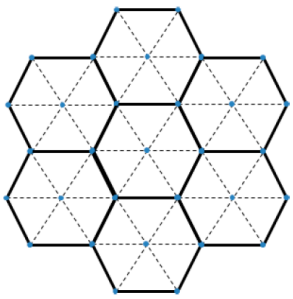
FIGURA 7 – Posiçãop dos Beacons



Fonte: Adaptado de Huh e Seo (2017)

Para que seja colocado os beacons em um ambiente, é determinado que cada área será em formato hexagonal (figura 7) para que possa ter uma melhor obtenção dos dados. Quando a área é grande, é feito a junção de vários hexagonos para que possa cobrir o espaço todo, assim como é apresentado na figura 8.

FIGURA 8 – Local dos Beacon



Fonte: Adaptado de Huh e Seo (2017)

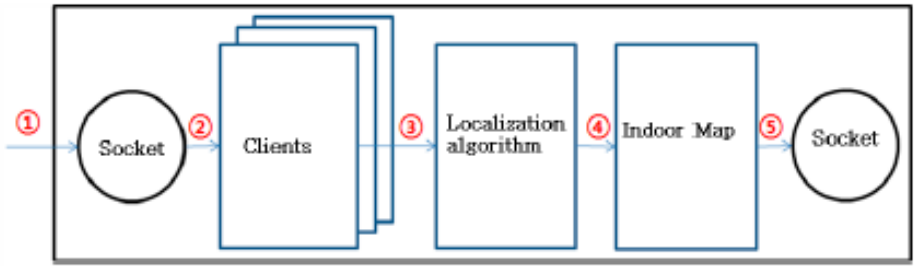
Para que fosse implementado o projeto foi criado um servidor para que fosse feito o controle de informação e os cálculos necessários para se obter a localização indoor do usuário. Inicialmente é feito uma comunicação do usuário com o servidor, no qual distingue os usuários através de um ID e realiza os cálculos. Após isto, através

[1] RSSI - Indicador de Potência do Sinal Recebido

do aplicativo e do desenho do local, a informação é repassada ao usuário pelo do aplicativo (figura 9).

Com a detecção da potência do sinal e com o método de trilateração ele obtém a distância (explicado melhor em 2.6.5). Antes de repassar a informação para o aplicativo, ele precisa passar por algumas condições de cálculo da trilateração, se não os valores são ignorados evitando com que obtenha resultados questionáveis.

FIGURA 9 – Desenho do projeto

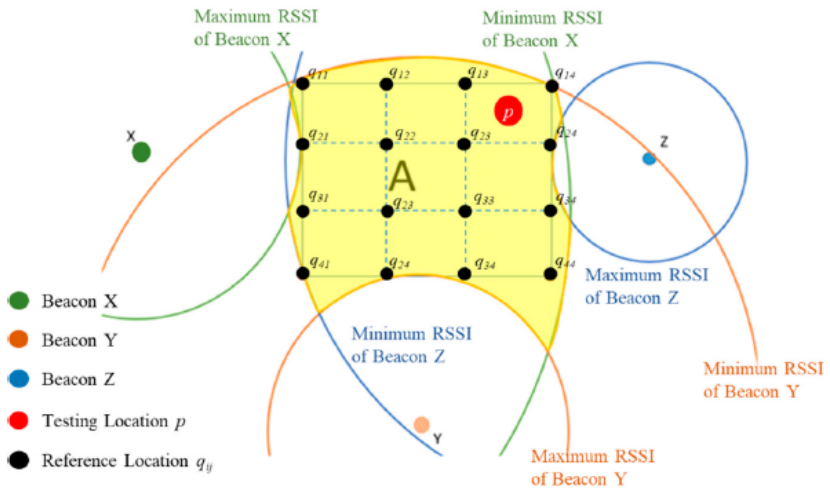


Fonte: Huh e Seo (2017)

3.2 DEVELOPING A BLE BEACON-BASED LOCATION SYSTEM USING LOCATION FINGERPRINT POSITIONING FOR SMART HOME POWER MANAGEMENT

Outro artigo que tenta melhorar a precisão dos beacons é o do Ke et al. (2018), no qual utiliza a triangulação com o algoritmo *fingerprint positioning* a partir dos valores RSSI. O algoritmo é baseado na potência do sinal recebido em diversos locais pré-determinados, assim ele associa com o local correspondente que foi armazenado na database. Na figura 10 mostra um exemplo de mapeamento de um local.

FIGURA 10 – Posição



Fonte: Ke et al. (2018)

Além de tentar se localizar em um ambiente interno, o aplicativo que é feito pelo Ke et al. (2018) também tem o objeto de monitorar remotamente o consumo de energia dos equipamentos. O aplicativo faz uma comunicação com o sistema inteligente de gestão de energia verde In-Snergy que assim que o usuário é posicionado em um ambiente, ele pode controlar remotamente algum equipamento, que neste caso foi um ventilador (figura 11. No aplicativo é possível configurá-lo para que quando chegar a um certo gasto, no qual é calculado com a fórmula 3.1, avisar o usuário de que o consumo está chegando ao seu limite e que o equipamento será desligado.

$$\text{Custo do consumo de energia} = \sum (\text{Custo de serviço ativo} + \text{Custo de serviço passivo}) \quad (3.1)$$

FIGURA 11 – Posição

Fonte: Ke et al. (2018)

O autor concluiu que em um sistema de localização utilizando Beacons é possível, porém necessita utilizar Beacons suficientes para aumentar a acuracidade da posição. Um ponto interessante que o autor traz é que nos testes realizados, quando o ponto de accesso do sinal de Wi-Fi é ligado o usuáriop não consegue obter sinal dos Beacons, porém quando é utilizado o Wi-Fi para conectar na internet não tem interferência (KE et al., 2018).

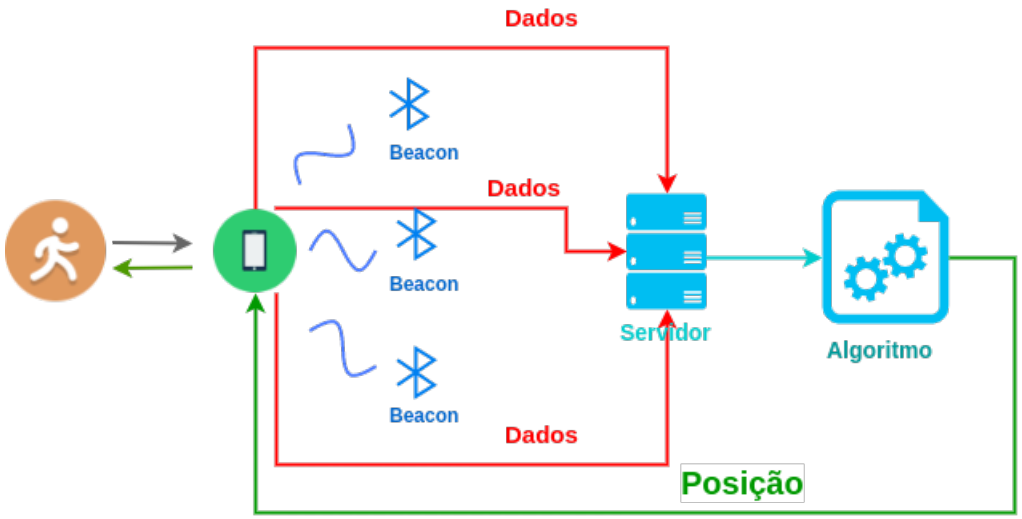
4 DESENVOLVIMENTO

4.1 DESCRIÇÃO DO PROJETO

O projeto a ser realizado diz a respeito de se desenvolver um sistema de localização *indoor*, para que um usuário habilitado para a utilização do sistema tenha a possibilidade de se identificar, com a ajuda de um aplicativo, a sua posição em um determinado ambiente.

O sistema possui três pontos principais: dispositivos de comunicação *bluetooth*, um dispositivo de capitação e um servidor onde serão processados os dados. O cenário a ser desenvolvido é mostrado na figura 12.

FIGURA 12 – Diagrama do projeto



Fonte: Autor

Discutindo um pouco sobre os três campos apresentados na figura 12, os dispositivos de comunicação *bluetooth* são os responsáveis por enviar as informações de RSSI para o sistema de captação, o qual é um dispositivo celular em que o usuário consta em sua posse. As fontes emissoras usadas no projeto são constituídas de módulos *bluetooth* usando-se uma configuração para estes dispositivos como *beacons*. Este modo de configuração faz com que os dispositivos fiquem emitindo informações a cada certos períodos de tempo, sendo que possuem também um protocolo de comunicação mais simplificado do que se comparado a protocolo de comunicação *bluetooth* convencional, além disso possuindo em seu protocolo já incluindo o valor de RSSI, cujo é o parâmetro que será usado para se estimar as distâncias entre a estação receptora e as emissoras.

O dispositivo de captação, como mencionado anteriormente, possui a finalidade de receber os valores de RSSI dos módulos *bluetooth* realizar um pequeno

pré-processamento inciar a comunicação com o servidor lhe enviando os dados necessários e posteriormente receber os dados de posição para a demostração ao usuário por meio de um aplicativo desenvolvido. Todos estes passos mencionados serão ocultos ao usuário, onde o mesmo visualizará, por meio da interface do dispositivo, a sua estimativa de posição.

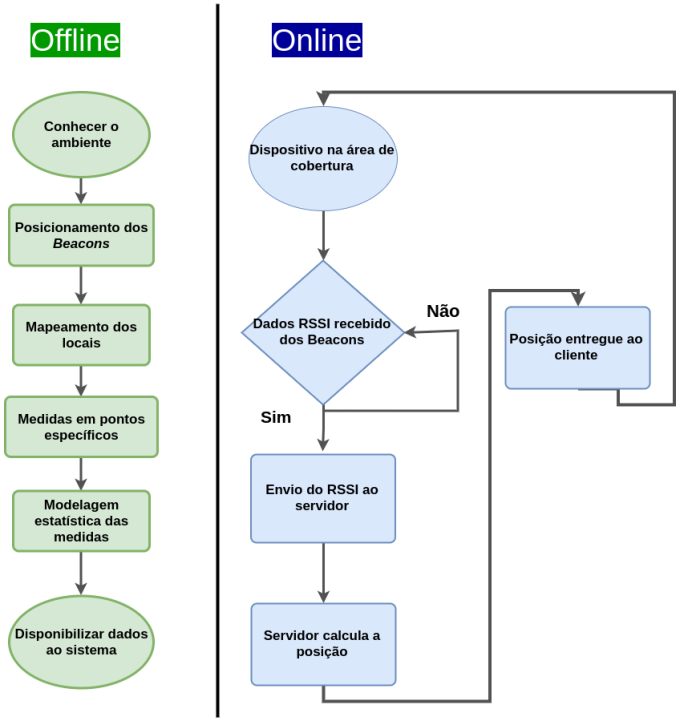
O servidor por sua vez trata-se do cérebro do sistema. No servidor irá conter o algorítimo que tratará de estimar a posição aproximada do usuário dentro do ambiente em questão. O servidor tratá de um computador que se encontra na rede onde através de um protocolo TCP/IP é possível se realizar a sua comunicação entre o dispositivo móvel do usuário. A fonte de captação envia ao servidor os dados necessário para o seu processamento e este também utilizando de dados já configurados sobre o ambiente, realizando o processamento através de um algorítimo desenvolvido para a aplicação e chegando ao resultado de uma posição referenciada ao plano cartesiano para o sistema. As coordenadas de posição processadas são retornadas ao dispositivo móvel que iniciou a comunicação e apresentadas ao usuário por meio da aplicação, fazendo assim com que o usuário possa conhecer a sua posição no ambiente.

Para a conclusão do projeto foram escolhidos alguns modelos de tecnologias, as quais serão apresentadas em mais detalhes as suas relevâncias ao projetos nas próximas seções, que mais se adaptaram ao sistema e que se gostaria de se desenvolver tanto por questões técnicas quanto por questões monetárias. Para se estimar a distância entre as fontes emissoras e o usuário foi utilizada a técnica do RSSI, a qual possui sua base em analisar a potência de emissão e o quanto ela sofreu de atenuação no ambiente. Onde utilizando este modelo e juntamente com um modelo de perda de caminho adequado se possui uma estimativa destas distâncias. Para a detecção da posição no ambiente foi utilizada a técnica de multilateração, uma variante da trilateração circular, onde por meio das distâncias obtidas pelos valores de RSSI, se torna possível encontrar uma estimativa da posição do usuário. Está técnica é juntamente utilizada com um modelo de ajuste matemático designado de ajuste por mínimos quadrados e com o modelo de K-NN modificado para atender melhor o sistema, sendo estas ultimas técnicas utilizadas para se conseguir uma melhor precisão.

4.2 DIAGRAMA DE FLUXO DO SISTEMA CONSTRUÍDO

Com o intuído de representar o modelo proposto e o seu fluxo de processo de uma maneira genérica, foi criado um modelo de fluxo, representado pela figura 13, onde demostra os principais passos que foram levados em consideração para a concretização do projeto, sendo este fluxo apresentado de maneira generalizada, onde serão discutidos de maneira mais especifica as partes apresentadas nesta seção no capítulo 4.3.

FIGURA 13 – Diagrama de Fluxo do modelo proposto



Fonte: Autor

A figura 13 apresenta dos cenários diferentes, mas que atuam de forma colaborativa para a conclusão do projeto. Os dois cenários são: o *off-line* e o *on-line*. O cenário *off-line* é o responsável por treinar o sistema, para servir de apoio ao cenário *on-line* e o deixar mais robusto a erros promovidos pelo ambiente visando se obter uma melhor precisão. Já o cenário *on-line* é responsável de processar, estimar e fornecer os dados de posicionamento no ambiente em tempo real.

Entrando em mais detalhes do cenário *off-line*, este é composto de seis fluxos macros. A primeira sequencia está relacionada em conhecer o ambiente onde se quer instalar o sistema, saber as suas dimensões, eventuais obstruções que o ambiente pode proporcionar e conhecer quais pontos seriam mais interessantes para a fixação dos *beacons*, já que estes devem ser posicionados em pontos específicos e conhecidos, entrando nos méritos do segundo fluxo. Conhecer os pontos de localização dos *beacons* se torna importante para se possuir pontos de referência para assim quando se estipular as distâncias entres os dispositivos emissores e receptores, poder encontrar um ponto referenciado pelos posicionamentos dos dispositivos. Para finalizar os méritos dimensionais do ambiente, este é mapeado em pontos específicos conhecidas as suas coordenadas, para que nestes pontos possam ser feitas as medidas dos valores de RSSI.

Sobre os pontos mapeados no ambiente são observados os valores de RSSI para cada dispositivos emissor presente. As medições são feitas de diversas formas e

diversas vezes a fim de se encontrar um valor mais fidedigno não se realizando nem uma estimativa otimista e nem pessimista para os valores. O intuito de se considerar valores estatísticos para as medidas é suprimir os valores otimistas e pessimistas trabalhando com um valor médio para os valores obtidos, se analisando juntamente o valor de seu desvio padrão, pois se este apresentar um alto valor certamente haverá imprecisão ao sistema e a analise será deficiente, sendo necessária uma nova análise. No caso onde os valores apresentarem estabilidade estatística, estes são fornecidos a um banco de dados que conterá os valores de todos os pontos medidos para cada dispositivo emissor, onde estes dados serão utilizados no modo *on-line*.

O cenário *on-line*, como descrito anteriormente, visa apresentar os dados de posicionamento no dispositivo que será usado como interface para o sistema, neste caso um dispositivo celular. Para o sistema funcionar duas condições deve, ser satisfeitas: o dispositivo celular deve estar dentro da área de cobertura do sistema e este estar habilitado para a comunicação dos *beacons* presentes no sistema, caso não satisfeitas as duas condições o sistema não funcionará em suas devidas condições. No cenário *on-line*, o dispositivo celular servirá de interface entre o sistema de processamento e o usuário. A fonte receptora possuí a finalidade de receber os valores de RSSI enviados pelos *beacons*, através de um protocolo de comunicação entre os dispositivos, e os enviar para um servidor *on-line* para que estes dados possam ser processados.

O servidor por sua vez recebe os dados enviados pelo dispositivo móvel realiza o processamento dos dados enviados, juntamente com os dados de referencia obtidos no cenário *off-line*, para se estipular uma posição estimada e referenciada ao ambiente que se está sendo utilizado o sistema. Nesta parte do processo se torna útil o cenário *off-line*, onde por meio das medidas obtidas anteriormente, o sistema pode se auto-ajustar e tomar os dados anteriores como referencia para se aumentar a precisão. Com os dados processados e com o valor estimado da posição o resultado é retornado ao ao dispositivo celular para que este possa apresentar a posição ao seu usuário no ambiente referenciado juntamente com algumas informações que o auxiliaram na sua localização.

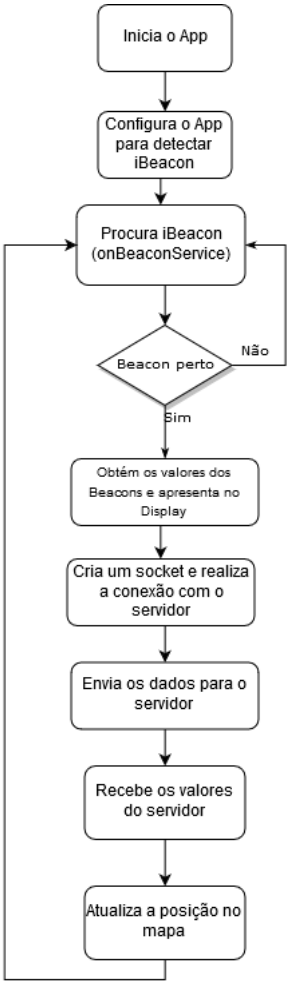
Para o funcionamento correto do sistema os dados devem ser atualizados constantemente, para que assim no movimentar do usuário não se cause erros de posicionamento devido a latência do sistema, já que se é considerado que o usuário esteja em movimento.

4.3 DIAGRAMA DE BLOCOS DO SISTEMA CONSTRUÍDO

Na seguinte seção será abordada um pouco mais das técnicas que foram utilizadas para a elaboração do projeto. Onde nesta é definida os passos lógicos e sequenciais para a obtenção do resultado final, demonstração da posição do usuário

em um ambiente. De mesmo modo que na seção 4.2, será discutido em duas frentes: sobre o lado do aplicativo e sobre o lado do servidor, já que os dois podem operar de maneira isolada um do outro.

FIGURA 14 – Diagrama de blocos do projeto do lado do aplicativo

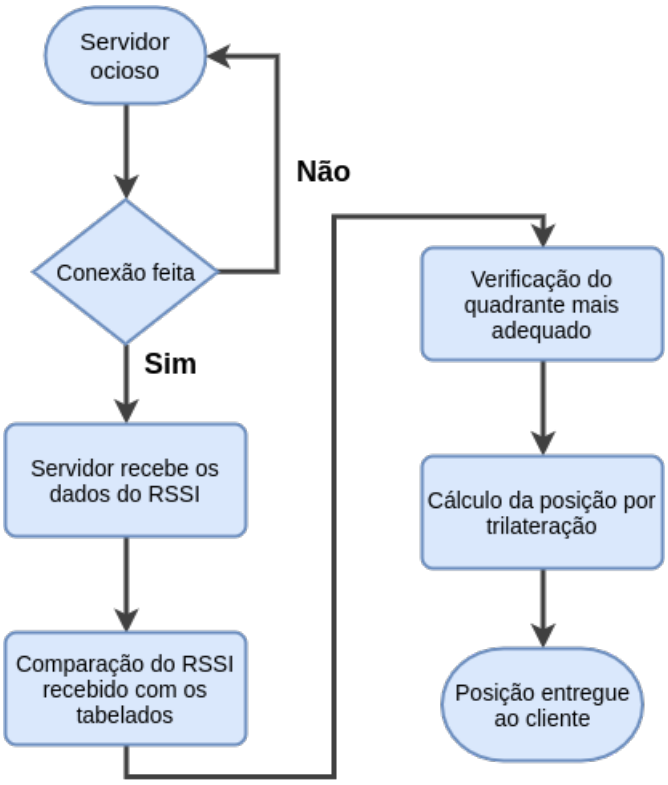


Fonte: Autor

Logo na primeira inicialização do aplicativo, é solicitado que o usuário permite que o app utilize o bluetooth. Além disto *AndroidManifest* também é configurado a permissão de utilizar o bluetooth, internet, wifi e a localização. Na seção de códigos é possível observar os comandos. Logo em seguida é configurado para que o App consiga detectar o Ibeacon (detalhado mais em 4.8.1), e depois ele começa a escanear a área por 3 segundo procurando por um Beacon. Caso ele encontre, é armazenado todos os pacotes que são obtidos neste período e é apresentado para o usuário o nome do Beacon, o Major, Minor e uma média de todos os RSSI obtido neste período dos dispositivos. Entre um escaneio e o outro, tem um delay de 1 segundo até que o comece a procurar novamente por Beacons. Assim que é obtido os dados de potência dos dispostivos, é criado um socket para realizar a conexão com o servidor e então é enviado para ele os valores. Depois que o servidor realizar os cálculos ele retorna com

com uma posiçã, assim no mapa é atualizado para o usuário. Na figura 14 temos um diagrama do processo explicado agora.

FIGURA 15 – Diagrama de blocos do projeto do lado do servidor



A figura 15 mostra o diagrama de blocos pela lado do servidor. A servidor funcionado como um servidor TCP-IP, onde fica esperando uma comunicação que é iniciada por um dispositivo qualquer habilitado para a comunicação. Quando está conexão é estabelecida são enviados dados de RSSI por parte do dispositivo dispositivo móvel que se encontra habilitado para a comunicação e está em um ambiente propicio para este tipo de análise.

Estabelecida a conexão e de posse dos dados de RSSI o servidor será responsável de processar os dados e de realizar uma estimativa da mais provável posição do usuário dentro do ambiente. Como mencionado na seção 4.2 foi realizada uma análise *off-line* onde foi obtido valores de RSSI em algumas posições determinadas para cada *beacons* analisado. Estes valores foram armazenados em um banco de dados para servirem de referencia para a predição da possível posição. Deste modo com os valores de RSSI enviados pelo dispositivo móvel o servidor carrega os dados providos do banco de dados e realiza comparações entre os valores enviados e os valores previamente medidos. A comparação trás como resultado qual ponto, conhecido pelo sistema, possui a menor diferença entre os valores previamente medidos e os valores enviados, sendo que toma as menores diferenças como os possíveis pontos do quadrante, conforme mostra a figura 19, para a posição do usuário no ambiente, onde

está análise é realizada para cada *beacon* previamente de forma isolada. A primeira análise ocorre de forma isolada para se possuírem uma sequencia de possíveis valores prováveis para cada *beacon*, já em uma segunda análise ocorre uma verificação da combinação das possíveis posições, ou seja, é feito de forma comparativa, para os quatro *beacons* neste caso, se os pontos referenciados mais prováveis se comprovam nos outros *beacons* também, se todos os casos possuem o mesmo resultado os resultados se complementam e se alguns resultados possuem divergência de resultados entre outros e se tomado como referencia os possíveis pontos que mais possuem aparições no sistema. Sendo juntamente analisado se estas combinações de resultados são possíveis.

Depois de conhecido o mais provável quadrante que o usuário poderia estar e a diferença do valor do RSSI entre os valor nas posições especificas e conhecidas e o valor enviado pelo dispositivo móvel para cada fonte emissora, por meio de um modelo de perda de canal, o qual foi utilizado o modelo *log-distance* que será discutido na seção 4.7, analisado para o caso específico pode se encontrar as distâncias entre os pontos de referência e o ponto real que se deseja encontrar a posição. Estes valores de distância são importantes pelo fato de usá-los para se determinar por meio da técnica da multilateração um ponto referenciado em coordenadas cartesianas para a posição do usuário, sendo está técnica usada em conjunto com o modelo matemático de mínimos quadrados a fim de se estimar a posição, com base nas distâncias, que minimize o erro de estimação.

Por fim realizado todo o processamento e de posse de um conjunto de pontos referenciados, por exemplo um valor x e y, o servidor retorno os valores obtidos ao cliente que requisitou a comunicação e fica esperando uma nova conexão.

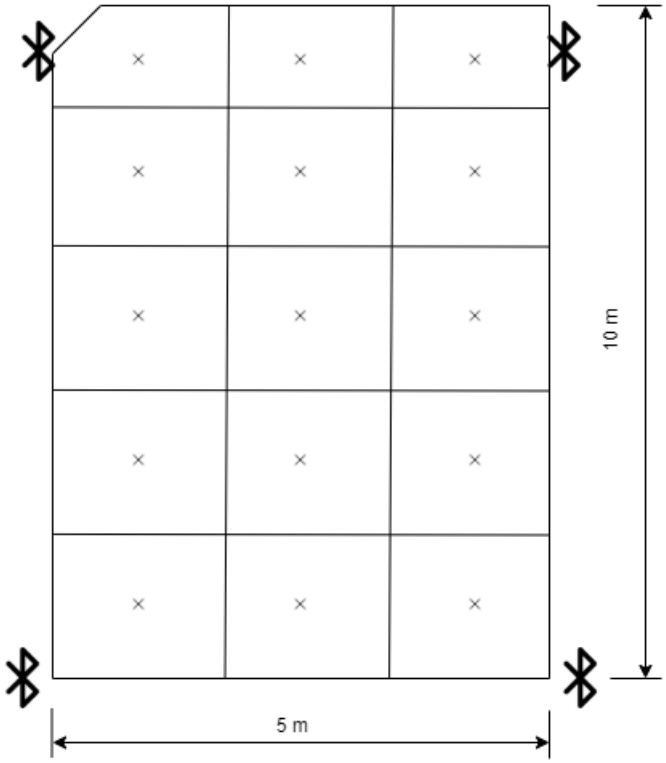
4.4 AMBIENTE DE TESTES

Na elaboração do projeto que foi desenvolvido, foi tomado um ambiente padrão de desenvolvimento para que assim se pudesse realizar os testes e validar os dados obtidos. A intenção de se realizar os testes neste ambiente vem do fato de se conhecer todas as suas medidas dimensionais e a acessibilidade que se tinha este local, já que o ambiente era a garagem de um dos integrantes do grupo. Assim de posse de todas as medidas dimensionais que o ambiente possuía e posteriormente as estimativas de posição que o desenvolvimento do projeto concluído proporcionaria, poderia-se analisar com maior exatidão, de modo comparativo, a diferença da posição real e física e da posição estipulada pelo sistema, conseguindo, também, verificar parâmetros de qualidade do sistema como: precisão e probabilidade de sucesso e assim validar a ideia.

O ambiente físico real se trata de uma garagem localizada na casa de um dos

integrantes do trabalho. Este ambiente possui as seguintes dimensões: largura de 5 metros e comprimento de 10 metros, como demostra a figura 16. O ambiente é um ambiente limpo que não possui muitos objetos.

FIGURA 16 – Parâmetros dimensionais do ambiente de testes



Fonte: Autor

Para o ambiente em questão apresentado pela figura 16, serão utilizados quatro *beacons* que estão posicionados nos vértices do retângulo do ambiente, para se possuir uma melhor cobertura de toda a areá pelos dispositivos. Na figura 17 mostra a garagem com os beacons circulados em vermelho, já nas figuras 18 apresenta os beacons mais de perto.

FIGURA 17 – Local de teste

Fonte: Autor

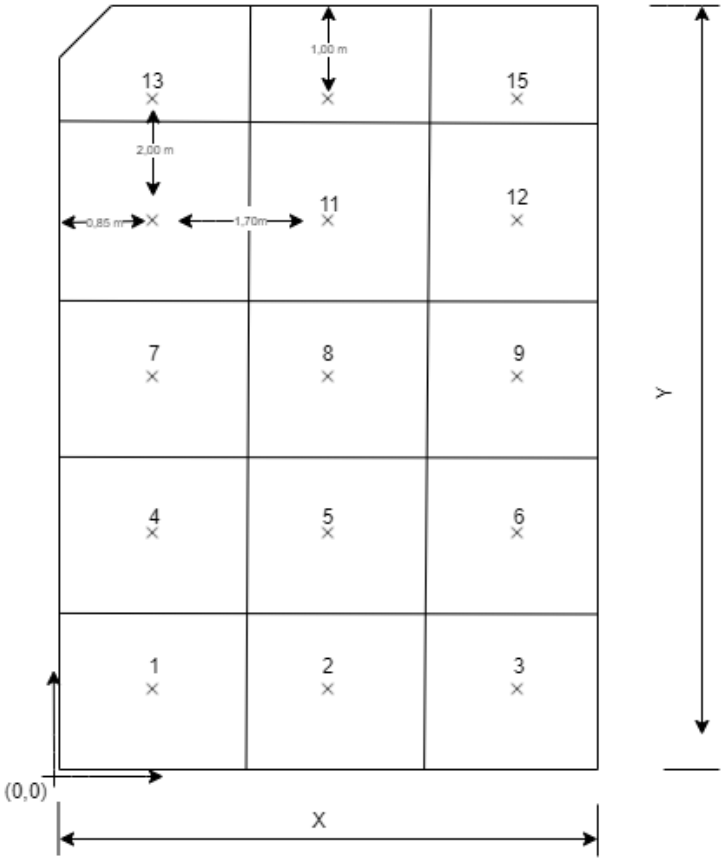
FIGURA 18 – Fotos dos beacons

Fonte: Autor

Com o intuito de se utilizar a técnica de K-NN a fim de se conseguir uma melhor predição da estimativa da posição melhorando a precisão do sistema, o local foi dividido quinze quadrantes, como pode ser visto na figura 19. Onde no centro de cada quadrante deste, foi determinada uma posição sequencial de referencia sendo esta posição conhecida, tomando como base o plano cartesiano, a fim de se treinar o sistema para posteriormente utilizar das posições conhecidas um referencial que ajudaria na descoberta das novas posições em tempo real. A distância entre pontos de referencia e a mesma em todos os pontos como apresentado pela figura 19.

As medidas nos pontos de referencia são medidas empíricas que foram usadas no decorrer do desenvolvimento do projeto. Os pontos, representados, de forma sequencial, de 1 a 15 possuem, tomando como base o plano cartesiano adotado descritos na tabela 2.

FIGURA 19 – Demarcação dos pontos de referencia no ambiente de testes



Fonte: Autor

Pontos	X	Y	Beacon 1	Beacon 2	Beacon 3	Beacon 4
1	0,85	1,00	1,31	4,27	8,34	9,28
2	2,55	1,00	2,74	2,65	8,68	8,65
3	4,25	1,00	4,37	1,25	9,32	8,33
4	0,85	3,00	3,12	5,12	6,36	7,54
5	2,55	3,00	3,94	3,87	6,80	6,76
6	4,25	3,00	5,20	3,09	7,60	6,34
7	0,85	5,00	5,07	6,50	4,38	5,98
8	2,55	5,00	5,61	5,57	5,00	4,95
9	4,25	5,00	6,56	5,06	6,05	4,36
10	0,85	7,00	7,05	8,14	2,45	4,74
11	2,55	7,00	7,45	7,42	3,43	3,36
12	4,25	7,00	8,19	7,04	4,83	2,42
13	0,85	9,00	9,04	9,91	0,90	4,16
14	2,55	9,00	9,35	9,33	2,57	2,47
15	4,25	9,00	9,95	9,03	4,26	0,81

TABELA 2 – Distâncias dos pontos onde foram realizados as medidas em relação a cada dispositivo emissor

Para se estabelecer uma distância de referência entre os dispositivos, *beacons*, é necessário saber as suas posições sendo estas para: *beacon* 1 (0,0), *beacon* 2 (5,

0), *beacon* 3 (0, 9,3) e *beacon* 4 (5, 9,3), tomando como base o plano cartesiano e as distâncias em metros.

4.5 MEDIDAS REALIZADAS NO AMBIENTE DE TESTES

O seguinte trabalho possui a finalidade de desenvolver um sistema de que consiga realizar a medida de pelo menos três distâncias, entre as estações rádio bases e a estação móvel que nada mais se trata do usuário do sistema, para assim com base nestas distâncias, por meio da técnica de trilateração, se poder mitigar a posição aproximada do usuário.

Dentre todas as técnicas que poderiam ser utilizadas para se descobrir as distâncias entre as estações de comunicação, a técnica utilizada foi a de RSSI, apresentada no capítulo 2.5.4. A técnica foi escolhido pelo fato desta ser de fácil implementação, não precisando a sincronização de relógios por parte das estações emissoras e receptora, possuir um protocolo de comunicação otimizado e já difundido no mercado onde o valor do RSSI faz parte deste protocolo sendo que este pode ser utilizado para as mais diversas aplicações e o barateamento de *hardware*, sendo que para a sua implementação não se é necessário a utilização de *hardware* especifico como no caso da técnica de triangulação onde quão mais precisas e direcionais forem as antenas melhor serão os resultados. No entanto está técnica pode trazer bastante erros se não for mensurada corretamente. Pois como é uma técnica que trabalha com a variação da atenuação do sinal enviado em relação ao sinal recebido e sabendo que o ambiente causará a atenuação do sinal por diversos fatores, se este não for modelado da melhor forma causará erros de precisão ao sistema.

A fim de minimizar os erros que o próprio ambiente e dispositivos causavam no sistema, foram desenvolvidos uma série de dez tipos de medidas em uma mesma posição, sendo estas posições apresentadas pela figura 19, para se estabelecer um valor médio, com base nos diferentes tipos de medidas, minimizando o erro que uma determinada posição de leitura poderia causar sobre outra, analisando também a variância dos valores para se constatar se esta não diverge muito de seus valores médios, caso sim este resultado estaria já estaria com um intrínseco a ele. Na realização das medidas foi se analisado os valores dos quatro *becons*, já que o projeto contava com quatro dispositivos, e as medidas foram realizadas em três dias diferentes a fim de se comparar se os resultados se mantinham e possuíam estabilidade. Para a elaboração das medidas de testes para a modelagem do ambiente foram utilizadas as seguintes formas de medidas, todas realizadas com o celular:

- 1 - Virado para o norte
- 2 - Virado para o leste;

- 3 - Virado para o sul;
- 4 - Virado para o oeste;
- 5 - Celular levantado acima da cabeça virado para o norte;
- 6 - Celular no chão virado para o norte;
- 7 - Celular com a tela virada para baixo no sentido norte;
- 8 - Celular virado 90º virado para o norte;
- 9 - Celular virado -90º virado para o norte;
- 10 - Celular encostado no corpo virado para o norte;

A grande quantidade de medidas possui a finalidade de se estipular o ambiente de teste e o modela-lo da melhor forma a fim de ocorrer os menores erros possíveis. Sobre os dados obtidos se obterão os seguintes resultados para cada posição, conforme a figura 16.

4.6 VALORES ESTATÍSTICOS PARA OS VALORES DE RSSI OBTIDOS

Na seção anterior 4.5, foi descrito os modos de medidas que seriam realizados no ambiente de testes a fim de se chegar a uma boa estimativa do valor de RSSI sobre aquele ponto especifico. Como mencionado anteriormente o intuito de de fazerem diversas medidas se vem do fato do sinal sofrer atenuação por diversos fatores e no caso como se estaria realizando diversas medidas e tomando como base o valor médio destas amostras, fazendo com que se obtivesse um resultado que mais se adequasse aos problemas no ambiente como um todo.

A análise estatística realizada cuidou de analisar quatro pontos específicos: o valor médio - o qual seria utilizado no banco de dados para a consulta do servidor, o valor máximo, o valor mínimo - das amostras que foram obtidas, para se constatar se não haveriam *outliers* e o desvio padrão - para a análise de todas as amostras do ponto para ver se estas não possuíam uma variabilidade excessiva, caso onde isso ocorresse se haveria uma sobre ou sub estimação do valor médio no ponto.

Em cada posição, como mostrado na figura 19, foram feitas os dez tipos de medidas apresentadas na 4.5, sendo que para cada *beacon* foi construída uma tabela de referência para a análise dos dados. Para o caso do *beacon* 1 se tem os resultados apresentados pela tabela 3.

Posição	Valor médio	Valor máximo	Valor mínimo	Desvio padrão	Distância (m)
1	73,5	93,0	59,0	9,2	1,31
2	74,0	83,0	64,0	5,5	2,74
4	79,7	92,0	70,0	9,1	3,12
5	80,7	92,0	73,0	7,8	3,94
3	82,3	96,0	79,0	6,7	4,37
7	83,3	93,0	78,0	4,5	5,07
6	84,4	93,0	76,0	6,4	5,20
8	85,2	98,0	74,0	7,9	5,61
9	85,4	92,0	74,0	5,3	6,56
10	85,6	96,0	78,0	5,6	7,05
11	86,1	97,0	78,0	6,1	7,45
12	87,0	96,0	75,0	7,6	8,19
13	87,4	100,0	78,0	7,0	9,04
14	91,5	101,0	83,0	5,1	9,35
15	92,0	96,0	77,0	6,4	9,95

TABELA 3 – Resultado dos valores de RSSI nas diversas posições para o *beacon* 1

As tabelas 3, 4, 5 e 6 foram organizadas pela magnitude da distância, onde assim pode se notar que maiores distâncias apresentam valores maiores de RSSI apresentados pela coluna do valor médio.

Para o caso do *beacon* 2 também foi realizada a mesma análise feita no caso anterior onde foi se encontrado os seguintes resultados apresentados pela tabela 4.

Posição	Valor médio	Valor máximo	Valor mínimo	Desvio padrão	Distância (m)
3	70,2	84,0	56,0	8,2	1,25
2	72,9	80,0	58,0	7,3	2,65
6	73,8	87,0	68,0	6,5	3,09
5	74,7	83,0	68,0	4,7	3,87
1	75,3	93,0	64,0	7,8	4,27
9	76,0	92,0	76,0	5,4	5,06
4	76,9	84,0	66,0	6,0	5,12
8	78,1	86,0	72,0	5,4	5,57
7	79,0	85,0	72,0	4,1	6,50
12	81,2	95,0	73,0	6,9	7,04
11	83,8	92,0	75,0	5,5	7,42
10	83,5	86,0	70,0	4,2	8,14
15	85,0	93,0	76,0	6,5	9,03
14	86,4	91,0	75,0	5,9	9,33
13	87,3	95,0	75,0	7,2	9,91

TABELA 4 – Resultado dos valores de RSSI nas diversas posições para o *beacon* 2

Os resultados dos valores médios para cada *beacon* se diferem de um para o outro, já que também as distâncias são ligeiramente distintas, como pode ser visto nas

tabelas 3, 4, 5 e 6, mas todos possuem uma tendência semelhantes dos resultados. Para o caso do *beacon* 3 se tem os seguintes resultados apresentados pela tabela 5.

Posição	Valor médio	Valor máximo	Valor mínimo	Desvio padrão	Distância (m)
13	72,1	80,0	64,0	5,6	0,90
10	76,5	85,0	70,0	5,3	2,45
14	76,4	89,0	70,0	6,1	2,57
11	76,5	89,0	68,0	6,9	3,43
15	76,8	93,0	75,0	6,5	4,26
7	77,0	94,0	73,0	6,8	4,38
12	77,5	82,0	70,0	3,4	4,83
8	78,7	85,0	73,0	4,1	5,00
9	79,0	86,0	86,0	4,7	6,05
4	82,4	93,0	76,0	6,0	6,36
5	82,8	90,0	76,0	5,3	6,80
6	83,8	93,0	76,0	6,3	7,60
1	87,5	96,0	77,0	5,5	8,34
2	89,6	96,0	81,0	4,5	8,68
3	90,2	95,0	82,0	3,9	9,32

TABELA 5 – Resultado dos valores de RSSI nas diversas posições para o *beacon* 3

Finalmente para a análise de todos os *beacons*, como feito nos casos anteriores, foi realizada uma tabela com os parâmetros de análise para se fechar o ciclo para o *beacon* 4, sendo que os seus resultados podem ser notados na tabela 6.

Posição	Valor médio	Valor máximo	Valor mínimo	Desvio padrão	Distância (m)
15	74,5	84,0	67,0	4,8	0,81
12	79,8	90,0	75,0	4,6	2,42
14	80,8	90,0	73,0	5,3	2,47
11	82,3	90,0	76,0	5,4	3,36
13	83,1	90,0	76,0	4,3	4,16
9	83,5	98,0	78,0	6,0	4,36
10	83,8	89,0	76,0	4,4	4,74
8	84,0	88,0	77,0	3,3	4,95
7	86,1	95,0	82,0	4,5	5,98
6	88,5	100,0	76,0	6,7	6,34
5	89,0	99,0	80,0	5,9	6,76
4	89,1	97,0	83,0	4,5	7,54
3	89,8	103,0	86,0	5,3	8,33
2	90,2	98,0	83,0	5,1	8,65
1	92,8	99,0	87,0	4,1	9,28

TABELA 6 – Resultado dos valores de RSSI nas diversas posições para o *beacon* 4

4.7 CÁLCULO DO MODELO DE PERDA DE CAMINHO

Como descrito na seção 4.4, o ambiente de teste de estudo e de validação para o modelo proposto foi a garagem de uns dos integrantes do grupo. Está garagem apresenta as dimensões mostradas pela figura 16, sendo que os *beacons* se localizariam nos vértices desses retângulo.

A técnica de obtenção da distância entre a fonte emissora e a receptora, como descrito anteriormente, foi a a técnica de RSSI. A técnica trabalha com a atenuação relativa da potencia entre a fonte emissora e a receptora. Portanto a modelagem do canal de comunicação, no caso o ar, para o caso especifico é de suma importância para bons resultados aproximados de distância entre os dispositivos.

Para modelar o canal de comunicação foi utilizado o modelo do log-distance, apresentado na seção 2.3.2. O modelo foi escolhido para o protótipo pelo fato de ser um modelo bastante conhecido e difundido na literatura e principalmente por ser um modelo adaptativo, em outras palavras, o modelo é constituído por meio do modelo de perda caminho no espaço livre juntamente com um termo adicional que distingue o ambiente que se está localizado. Este termo adicional é composto por um termo logarítmico multiplicado por um fator n, como apresentada pela equação 2.5, que possui um resultado diferente para cada ambiente que se é utilizado, conforme mostrado na TABELA. Como cada ambiente possui uma resposta, afetando o valor de n, para o caso proposto foram realizadas medida empíricas para se avaliar o valor do fator n que mais e adequasse ao modelo proposto e juntamente o alcance máximo do enlace de comunicação entre os dispositivos emissores e receptores.

As medições feitas para a análise foram feitas sobre um dispositivo emissor, *beacons*, sendo que foram realizadas medidas a cada um metro começando preliminarmente a um metro de distância e sendo efetuadas medições até vinte e um metros. Do mesmo modo que na seção 4.5, foram feitas três medições em cada ponto e foram considerados os seus valores médios os resultados das medidas podem ser notados na tabela 7.

Para a análise dos resultados estes foram colocados em forma gráfica, o qual pode ser notado na figura 20. Os resultados obtidos, suprimindo as suas variações, se aproximam bastante de uma função logarítmica crescente, senso assim o modelo proposto pela equação 2.5 do *log-distance* se adequando perfeitamente para a modelagem da perda do canal. Na figura 20 também é feita a comparação se só fosse realizada a consideração do espaço livre para o modelo de perdas. Neste caso pode se notar que se apenas fosse realizada está consideração existiria uma alta imprecisão no sistema para previsão da distância entre as fontes emissoras e receptoras.

Com o intuito de minimizar o erro entre os valores medidos e o resultado

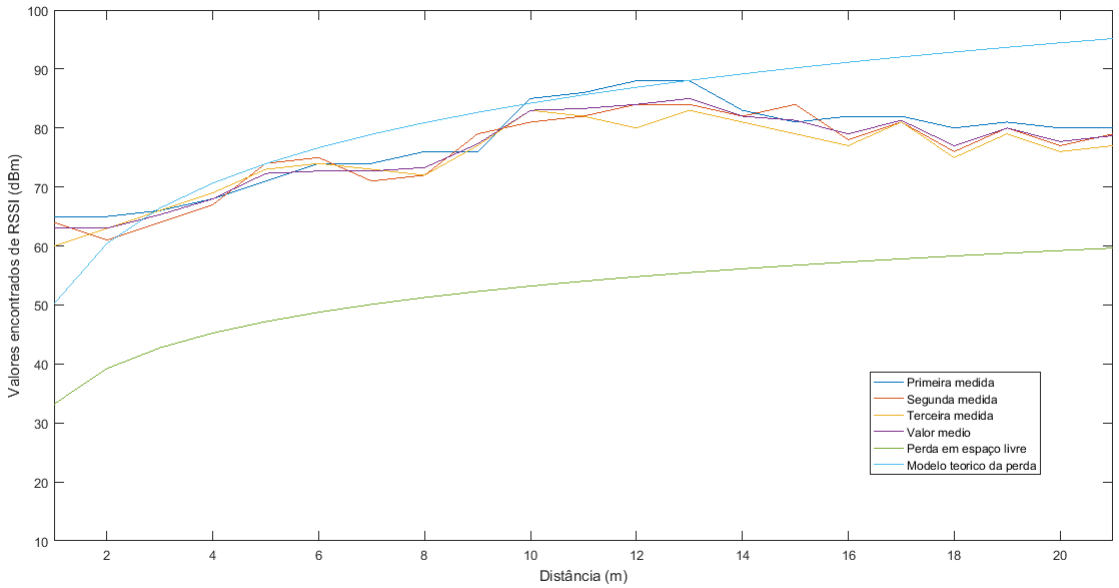
Distância	Valor 01	Valor 02	Valor 03	Valor Médio
1	65,0	64,0	60,0	63,0
2	65,0	61,0	63,0	63,0
3	66,0	64,0	66,0	65,3
4	68,0	67,0	69,0	68,0
5	71,0	74,0	73,0	72,7
6	74,0	75,0	74,0	74,3
7	74,0	71,0	73,0	72,7
8	76,0	72,0	72,0	74,3
9	76,0	79,0	77,0	77,3
10	85,0	81,0	83,0	83,0
11	86,0	82,0	82,0	83,3
12	88,0	84,0	80,0	84,0
13	88,0	84,0	83,0	85,0
14	83,0	82,0	81,0	82,0
15	81,0	84,0	79,0	81,3
16	82,0	78,0	77,0	79,0
17	82,0	81,0	81,0	81,3
18	80,0	76,0	75,0	77,0
19	81,0	80,0	79,0	80,0
20	80,0	77,0	76,0	77,7
21	80,0	79,0	77,0	78,7

TABELA 7 – Valores obtidos de RSSI para a distância analisada

teórico, foram realizadas um conjunto de iterações no modelo teórico variando-se o valor do parâmetro n do modelo *log-distance* a fim de se fazer a melhor aproximação e se possuir os menores erros na aproximação. Desta análise se concluiu que para o modelo proposto o valor de n que minimizava os erros é o de valor de 1,4 como pode ser visto na figura 20. Neste modelo existem um maior erro em pequenas distâncias que são tratadas vai *software* para minimizar os erros.

Sobre a análise dos valores mensurados, conclui-se que a a maior distância onde ainda assim se teriam dados satisfatórios para a análise, considerando os dispositivos utilizados, é de treze metros de distância. Este resultado fica evidenciado pala análise da figura 20, onde pode-se notar que o sistema começa apresentar uma resposta com um aumento de erro progressivo na estipulação do valor de RSSI em relação a distância se comparado com o valor teórico. A distância máxima analisada não compromete a utilização e no posicionamentos dos *beacons* do modo que foram projetados, já que as distâncias máximas para o ambiente se situam na ordem de no máximo doze metros.

FIGURA 20 – Valores encontrados para o modelo de perda de caminho proposto



Fonte: Autor

4.8 TIPOS DE COMUNICAÇÃO UTILIZADAS ENTRE OS DISPOSITIVOS

Para a conclusão e elaboração do projeto são necessários de três dispositivos básicos: *beacons*, dispositivo móvel e servidor e, por escolha, um banco de dados. Estes dispositivos estão fisicamente separados e por esse motivo deve se estabelecer uma forma de comunicação entre eles. Os modelos que foram adotados para as comunicações serão discutidos na sequência.

4.8.1 Entre *beacon* e dispositivo celular

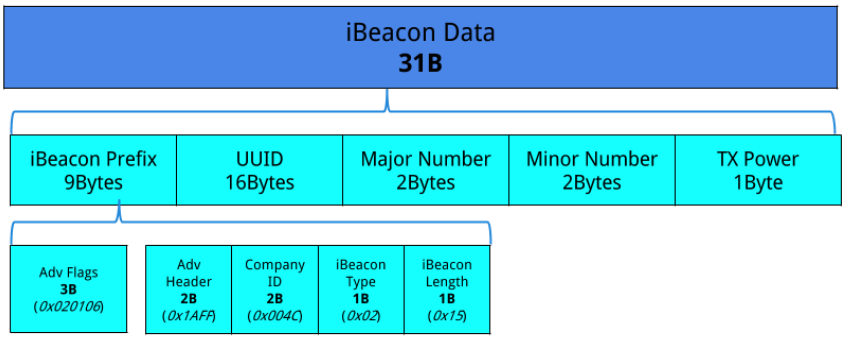
Como explicado na seção 2.13 que atualmente possuimos alguns modelos de Beacons no mercado. No projeto será utilizado o módulo bluetooth Hm-10 (4.12, ou seja, o protocolo que será utilizado será o da Apple. Antes de realizar qualquer desenvolvimento, é necessário que o Aplicativo seja possível detectar o Ibeacon e com o auxilio da biblioteca AltBeacon (ALTBEACON, 2019) é feito esta configuração. Na figura 21 é apresentado os dados que um pacote de iBeacon envia.

Verificando a especificação do iBeacon (INC, 2015), os ultimos 2 bytes do *iBeacon Prefix* é o tipo do Beacon, no qual deve ser obrigatoriamente para 0x0215 (22). Logo depois vem o UUID, major e o minor do dispositivo.

No programa, o layout do Beacon foi configurado com a seguinte especificação: "m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24". No qual significa que:

- No byte 2 e 3 está configurado o tipo de Beacon

FIGURA 21 – Pacote do iBeacon



Fonte: Ferreira, Resende e Martinho (2018)

FIGURA 22 – Tipo de Beacon

Byte(s)	Name	Value	Notes
5	Company ID[0]	0x4C	Must not be used for any purposes not specified by Apple.
6	Company ID[1]	0x00	Must not be used for any purposes not specified by Apple.
7	Beacon Type[0]	0x02	Must be set to 0x02 for all Proximity Beacons
8	Beacon Type[1]	0x15	Must be set to 0x15 for all Proximity Beacons

Fonte: Apple Inc (2015)

- O primeiro identificador é o UUID (byte 4 a 19)
- O segundo identificar é o Major (byte 20 e 21)
- O terceiro identificar é o Minor (byte 22 e 23)
- No byte 24 é a potencia.

Sendo assim possível se comunicar com o Beacon. Para cada fabricante, o protocolo pode mudar, sendo assim necessário configurar o programa para o layout do dispositivo.

4.8.2 Entre dispositivo celular e servidor

Uma das premissas do projeto era de desenvolver um servidor de dados que estivesse acessível ao usuário seja onde ele estivesse, levando em consideração que se existisse alguma forma de comunicação com a internet. Portanto a comunicação seria dada por meio da rede de dados, onde o usuário apto para para a transferência de dados para encontrar a sua localização, poderia se conectar ao servidor desenvolvido.

A comunicação entre estas duas partes se da através do protocolo TCP/IP entre cliente (celular do usuário) que inicia a comunicação pois sabe em qual endereço

IP (*Internet protocol*) o servidor está alocado e qual porta este está usando para a comunicação de dados e o servidor propriamente dito que possui um endereço IP e uma porta de comunicação, configurados previamente, que fica esperado requisições de comunicação, sendo que quando estas são aceitas o servidor recebe os dados necessários para o seu processamento e retorna os dados de posição para o cliente.

O modo que se é realizada a comunicação se da por meio do mecanismo de *socket* para a internet entre um aplicativo desenvolvido em linguagem de programação JAVA e um servidor desenvolvido em linguagem de programação Python, não havendo qualquer empecilho na comunicação das duas partes. Neste caso algumas configurações são necessárias: o endereço IP do servidor, a porta de comunicação, a família do protocolo que se está realizando a comunicação, nestes caso TCP/IP, o limite de conexões permitidas ao mesmo tempo e um objeto que cria a interface de comunicação entre as duas partes permitindo a recepção e envio de dados. As configurações comentadas são os elementos fundamentais para o inicio da comunicação sendo possível mais alguns ajustes para o protocolo.

4.8.3 Entre servidor e banco de dados

A escolha por um banco de dados foi adotada pelo fato de se possuir um local onde apenas seriam colocados os dados dos diversos ambientes que se poderiam existir. O banco de dados seria comum par os diversos ambientes apenas diferenciando-se as tabelas destes banco de dados juntamente com seus valores internos. Cada tabela possuiria uma identificação que estaria correlacionada aos *beacons* de um determinado ambiente que estariam configurados com a mesma identificação. Fazendo-se assim com que o banco de dados retorna-se apenas os dados relacionados ao ambiente requisitante.

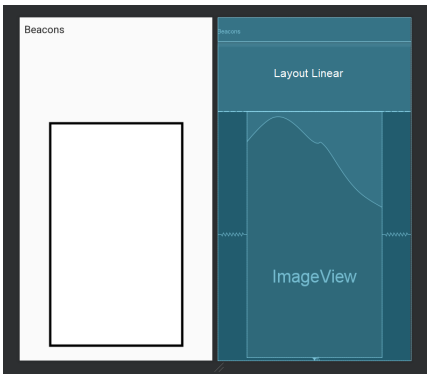
O banco de dados possui apenas duas colunas de dados: uma que trás o endereço da posição previamente determinada constituída de um numero sequencial e o valor de RSSI naquele ponto já tratado. O banco de dados foi desenvolvido sobre a plataforma MySQL Workbench 6.1 CE e a comunicação com o servidor foi realizada em linguagem de programação Python. O Python possui uma biblioteca de comunicação que facilita a manipulação e comunicação entre as duas partes.

Possuindo o endereço do banco de dados a tabela que se gostaria acessar, devida a identificação dos *beacons* em questão do ambiente, é feita por meio do servidor a requisição dos dados contidos na tabela do banco de dados para o servidor. São carregados todos os dados contidos no banco de dados em um espaço de memória do servidor e assim posteriormente processados pelo servidor.

4.9 APLICATIVO DESENVOLVIDO PARA A DEMONSTRAÇÃO DOS DADOS PARA O USUÁRIO

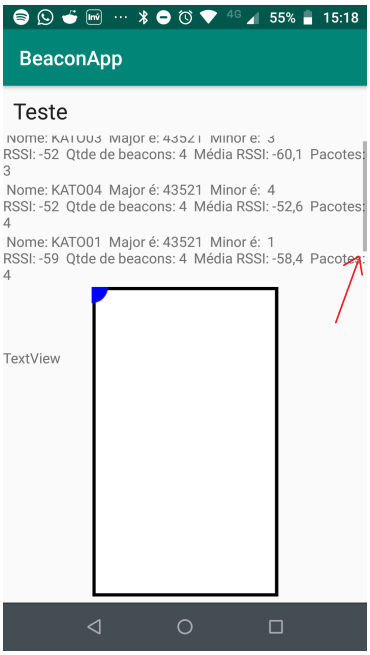
O aplicativo foi montado tendo em mente que seria necessário apresentar para o usuário os beacons que estivessem sendo captados e o mapa mostrando a sua localização. Após realizar alguns testes, o app foi configurado de maneira que fosse observado diversos beacons e não ficasse limitado pelo tamanho que teria disponível para apresentar os dados. Na figura 23 é mostrado o layout final, no qual como é possível observar com a utilização de um um layout linear, no qual ele permite olhar mais beacons deslizando a tela para cima. Na figura 24 é mostrado o aplicativo e no lado esquerdo é verificado que é o usuário consegue desligar a tela.

FIGURA 23 – Layout do aplicativo



Fonte: Autor

FIGURA 24 – Aplicativo



Fonte: Autor

A versão do layout em linhas de programa está na seção de códigos.

4.10 ERRO DE ESTIMATIVA DEVIDO A RESOLUÇÃO DE MEDIDA

Na elaboração do projeto foi analisado quanto o próprio sistema trazia de erro já em sua concepção, já que este, o erro na estimativa da posição, é um parâmetro de qualidade na validação do sistema. Sendo que este erro somado juntamente ao erro de medição em tempo real dirá o quão preciso o sistema é quanto a sua estimativa da posição.

O erro analisado nesta seção diz a respeito ao erro causado pelo nível de resolução de valores de RSSI que o sistema é capaz de captar, juntamente com o modelo de canal de comunicação que foi proposto, apresentado na seção 4.7. A resolução de valores de RSSI está associada ao quanto o sistema de captação consegue perceber como mínimo necessário para alterar a ordem de grandeza do valor de RSSI. A tabela 8 mostra os valores do erro conforme a variação mínima dos valores de RSSI, que no caso apresenta uma resolução de apenas valores absolutos. Já o erro associado com o modelo de canal está relacionado ao próprio modelo do canal que não é perfeito e ao número de casas decimais utilizadas no calculo. Pode-se notar pela tabela 8 que é erro é progressivo a medida que se aumentam o valor de RSSI, fato que também pode-se notar analisando a curva do modelo proposto com os valores empíricos apresentados pela figura 20.

Valor de RSSI	Distância	Erro
60	1,94	-
61	2,08	0,136
62	2,22	0,146
63	2,38	0,156
64	2,55	0,167
65	2,72	0,178
66	2,92	0,191
67	3,12	0,204
68	3,34	0,219
69	3,57	0,234
70	3,82	0,250
71	4,09	0,268
72	4,38	0,287
73	4,68	0,307
74	5,01	0,328
75	5,36	0,351
76	5,74	0,376
77	6,14	0,402
78	6,57	0,430
79	7,03	0,460
80	7,52	0,493

TABELA 8 – Valores do erro no sistema devido a resolução de valores absolutos

A tabela 9 apresenta um segundo caso onde se tem uma melhor resolução dos valores de captação de RSSI, na ordem de uma casa decimal, e deixando o modelo e os cálculos matemáticos idênticos ao caso anterior. Pode-se constatar que o erro é diminuído neste caso. A resolução de uma casa decimal exige um *hardware* mais complexo e robusto para se realizar está captação.

Valor de RSSI	Distância	Erro
60,1	1,96	-
60,2	1,97	0,013
60,3	1,98	0,013
60,4	2,00	0,013
60,5	2,01	0,014
60,6	2,02	0,014
60,7	2,04	0,014
60,8	2,05	0,014
60,9	2,06	0,014
61,0	2,08	0,014
61,1	2,09	0,014
61,2	2,11	0,014
61,3	2,12	0,014
61,4	2,14	0,014
61,5	2,15	0,015
61,6	2,16	0,015
61,7	2,18	0,015
61,8	2,19	0,015
61,9	2,21	0,015
62,0	2,22	0,015
62,1	2,24	0,015

TABELA 9 – Valores do erro no sistema devido a resolução de uma casa decimal

O sistema adotado e utilizado para validação apresenta uma resolução igual ao apresentado pela tabela 8 pelo fato de se utilizar de um *hardware* mais simples. Possuindo-se o inconveniente de se introduzir mais erro ao sistema.

4.11 TÉCNICAS UTILIZADAS PARA ESTIPULAR A POSIÇÃO NO AMBIENTE

A determinação da possível posição que um dado usuário poderia estar em um certo ambiente é obtida através dos dados de RSSI fornecidos pelo dispositivo móvel ao servidor. Os valores de RSSI são usados pelo servidor para se encontrar as prováveis distâncias entre os dispositivos emissores, *beacons*, e a fonte receptora, no caso o celular. O valor do RSSI nada mais é do que um valor de potência relativa entre a potência emitida e a recebida, portanto através deste dado pode-se calcular uma estimativa da distância por meio de uma modelo de perda de canal, onde no caso foi utilizado o modelo *log-distance*. Mesmo utilizando um bom modelo de canal

a estimativa da distância para os cálculos da posição pode sofrer erros. Por isso no projeto proposto são utilizadas duas técnicas para tentar minimizar este erro devido a deficiência na estipulação da distância, sendo estes utilizando: a técnica dos mínimos quadrados e a técnica do K-NN modificada para se adequar ao projeto, sendo que o código do projeto por parte do servidor pode ser visto na seção de códigos.

A técnica dos mínimos quadrados se resume em resolver a equação matemática, apresentada pela equação 2.35, com que faz que o erro entre os dois pontos seja reduzido já a técnica utilizando do K-NN se baseia em se conhecer alguns pontos já do ambiente, treinamento do método, e utilizar estes pontos conhecidos para estipular uma posição desconhecida atribuindo pesos as diferenças dos valores analisados e os já sabidos.

Para se testar a performance das duas técnicas foram adotados cinco modelos diferentes de analise, sendo eles: quando todas as distâncias estavam corretas, quando umas das distâncias apresentava deficiência, quando duas distâncias apresentavam deficiência, quando três medidas apresentavam deficiência e quando quatro medidas apresentavam deficiência, já que se estava trabalhando com uma quantidade de quatro *beacons*. Em todos os casos foram analisadas as combinações dos erros onde em todos os casos se havia um erro na estimação da distância de um metro tanto para mais quanto para menos. A variação de um metro para o sistema de captura gira em torno de se variar o valor de RSSI em aproximadamente três pontos, sendo este um valor comum no sistema de medição para um mesmo ponto.

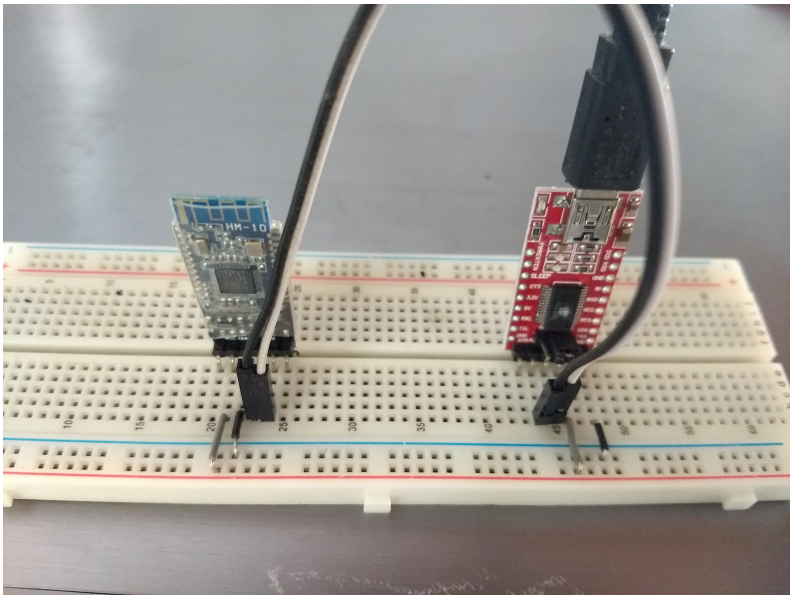
Definidas as analises que seriam realizadas foi definido um certo ponto para cada caso, sabido as distâncias e o seu erro associado pode-se encontrar a possível posição que as duas técnicas e o erro de estimativa para a posição.

4.12 HM-10

Dentre os módulos disponíveis no mercado, foi selecionado o Hm-10 para utilizar, pois possuía um datasheet detalhado, é fácil de encontrar no mercado e possuí um preço acessível, assim o projeto fica focado em apenas um modelo e não é necessário se preocupar com a compatibilidade com outros módulos Bluetooth.

De acordo com Karydis (2015) e Pandit (2019) o módulo Bluetooth 4.0 Hm-10 é um produto baseado no sistema Texas Instruments CC2540 ou CC2541 BLE, já o firmware e o esquema são feitos e gerenciados pela *Jinan Huamao Technology*. O módulo é controlado por comandos AT que são enviados através de uma conexão serial UART. Para se comunicar com o Hm-10 foi utilizado a placa FTDI232, no qual é um dispositivo que possibilita a comunicação USB para UART, no qual possui a tensão de saída que o módulo (3,3V) FTDI (2019). Na figura 25 pode ser observado como foi

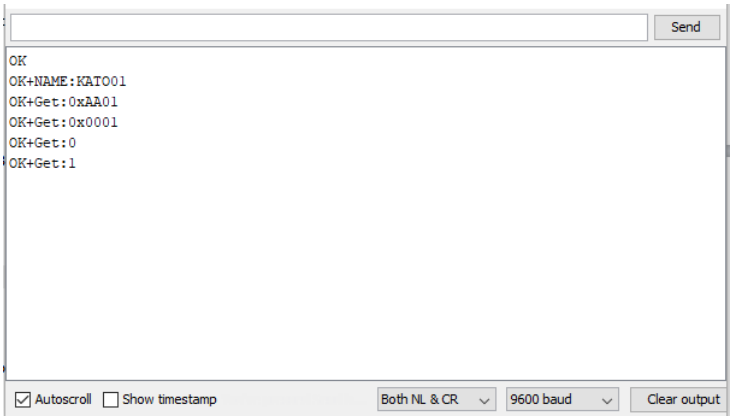
FIGURA 25 – Comunicação Hm-10 e FTDI232



Fonte: Autor

feito a conexão, sendo que a placa faz a comunicação pelo USB.

FIGURA 26 – Comunicação com HM-10



Fonte: Autor

Como foi comentado anteriormente, a comunicação com o módulo é através de comandos AT. Na figura 26 apresenta um exemplo da resposta que recebemos do dispositivo.

Utilizando o Datasheet, o Hm-10 foi configurado com os comandos abaixo.

- AT+ADVI6 - Configuração do intevalo de anúncio (figura 27 apresenta todos os intervalos
- AT+NAMEKATO0x (x sendo o número do Beacon)
- AT+MINO000x (x sendo o número do Beacon)

FIGURA 27 – Configuração do intervalo de anúncio

Send	Receive	Parameter
AT+ADVI?	OK+ Get:[Para]	None
AT+ADVI[Para]	OK+ Set:[Para]	Para: 0 ~ F 0: 100ms 1: 152.5 ms 2: 211.25 ms 3: 318.75 ms 4: 417.5 ms 5: 546.25 ms 6: 760 ms 7: 852.5 ms 8: 1022.5 ms 9: 1285 ms A: 2000ms B: 3000ms C: 4000ms D: 5000ms E: 6000ms F: 7000ms Default: 9

Fonte: JNHuaMao Technology Company (2014)

- AT+IBEA1 - Comando para que o HM-10 se torne um iBeacon

Como geralmente os Beacons são alimentados por bateria é essencial que ele consuma pouca energia, sendo assim é possível realizar algumas configurações que fazem este consumo diminuir como:

- Não ser conectável
- iBeacon só envia informação
- Possibilita o Beacon dormir, reduzindo o consumo de 8 para 0,18mA
- Diminuir a potência (sendo o padrão 0dbm)

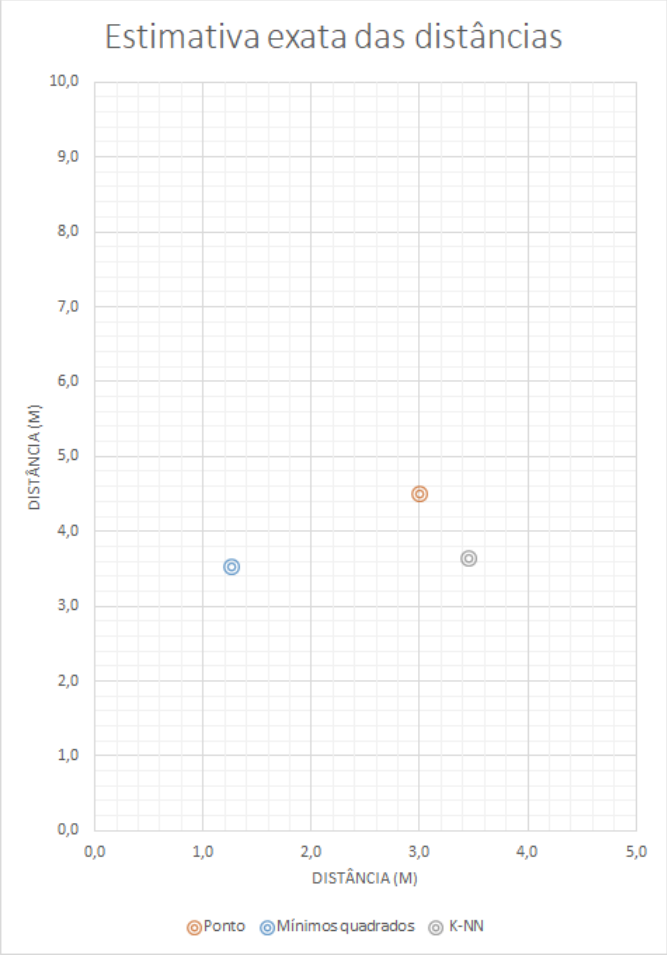
Como o objetivo do projeto apresentar o projeto funcionando, não foi realizado nenhuma configuração com o foco da redução de consumo. Por causa disto, nas configurações e nos testes inciais estava sendo utilizado uma bateria de lítio de 3V para realizar a alimentação, porém não durou mais de 4 dias mexendo constantemente no hm-10. Sendo assim foi realizado a troca da alimentação para uma bateria de Li-ion recarregável de 3,7V, eliminando o nosso problema de energia, mas aumentando o custo do projeto.

5 RESULTADOS

5.1 PRECISÃO DO SISTEMA CONSTRUÍDO

Para o sistema construído dois requisitos principais deferem se o sistema apresentou exito na sua construção o requisito de precisão, que será apresentado nesta seção, e o requisito de probabilidade de sucesso que será apresentado na seção 5.2. Estes dois parâmetros bem desenvolvidos e com bons resultados resultaram em uma melhor exatidão ao sistema, sendo a esta à conformidade ao valor real e ao medido dentro de um raio aceitável de erro. Já a precisão está relacionada ao grau de variação dos resultados do sistema de medição.

FIGURA 28 – Caso onde todas as medidas de distâncias são exatas

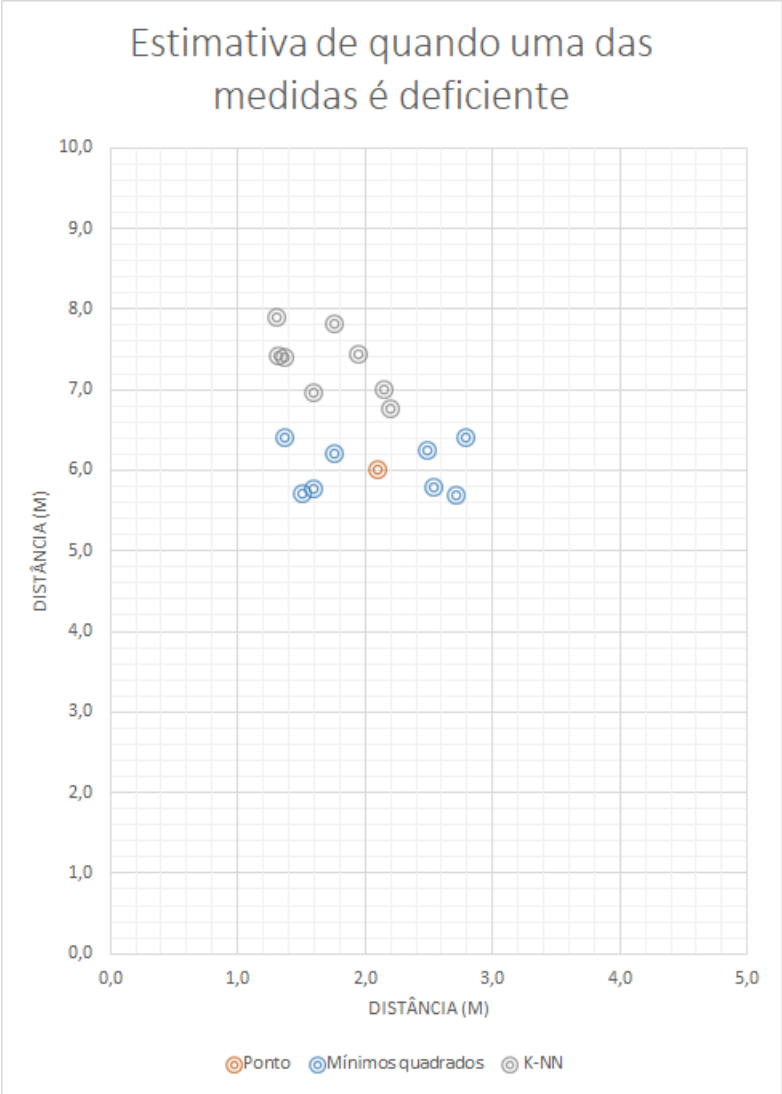


Fonte: Autor

Para o caso das quatro análises apresentadas na seção 4.11 foram feitas os acompanhamentos dos resultados obtidos nestes casos simulados. Para o caso onde se possui todas as medidas de distância de forma exata convergindo a um ponto específico obteve-se a figura 28, onde mostra o ambiente de teste onde foi desenvolvido, o ponto real e o ponto estimado pela técnica utilizada.

Neste caso pode-se notar que a a técnica do K-NN apresentou uma melhor estimativa se comparada ao ponto real de origem levando em consideração uma menor distância de erro. Avaliando o erro na estimativa pode-se notar que a melhor resposta para a precisão no caso do K-NN vem do fato que a técnica dos mínimos quadrados já supor que os dados não são exatos buscando assim ajusta-los. Para o segundo caso foi analisado se o sistema possui-se uma das medidas apresentando um erro na estimativa. Neste segundo caso foram analisadas todas as combinações possíveis onde existia a variação de uma medida errada da distância tanto para mais quanto para menos originando oito combinações. Os resultados são apresentados na figura 29.

FIGURA 29 – Caso onde uma das medidas estava equivocada

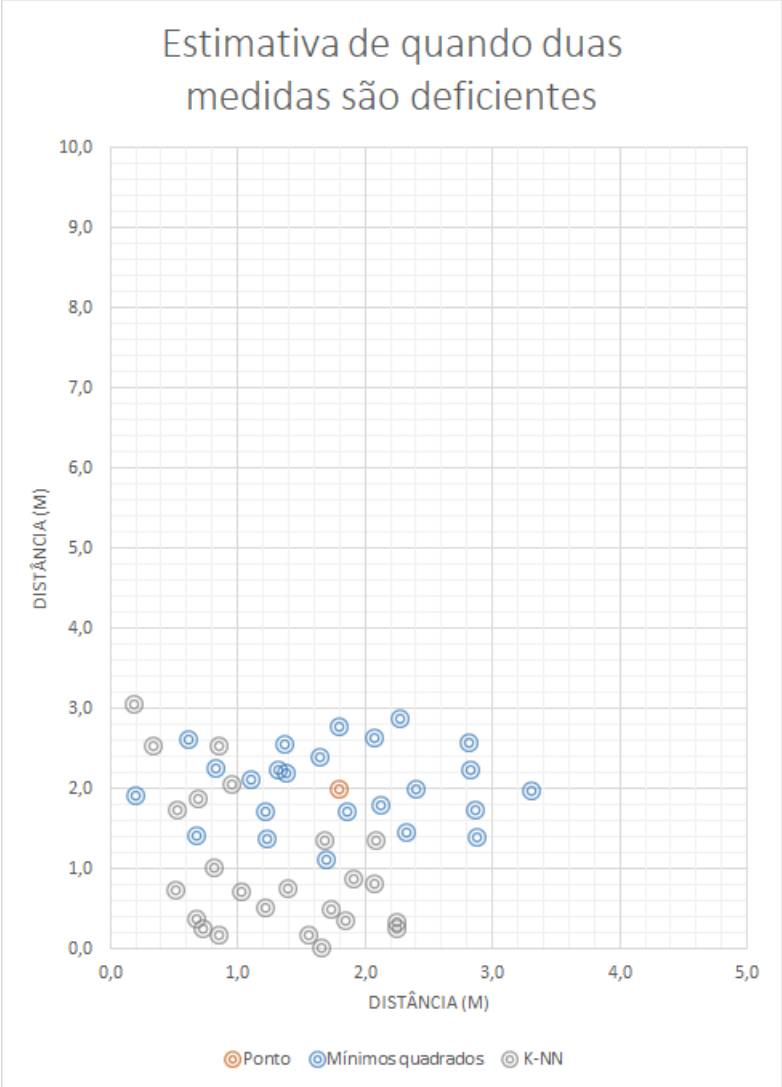


Fonte: Autor

Neste segundo caso analisado notou-se um deslocamento em y das amostras utilizando a técnica do K-NN e um espalhamento maior na técnica utilizando os mínimos quadrados que também apresentou uma melhor precisão se comparado ao ponto real de medida.

Para o terceiro caso analisado resultados apresentados pela figura 30, onde se foi analisado as combinações de duas amostras se equivocando quanto a medida das suas distâncias tanto para mais quanto para menos se originando vinte e quatro combinações.

FIGURA 30 – Caso onde duas das medidas estavam equivocadas



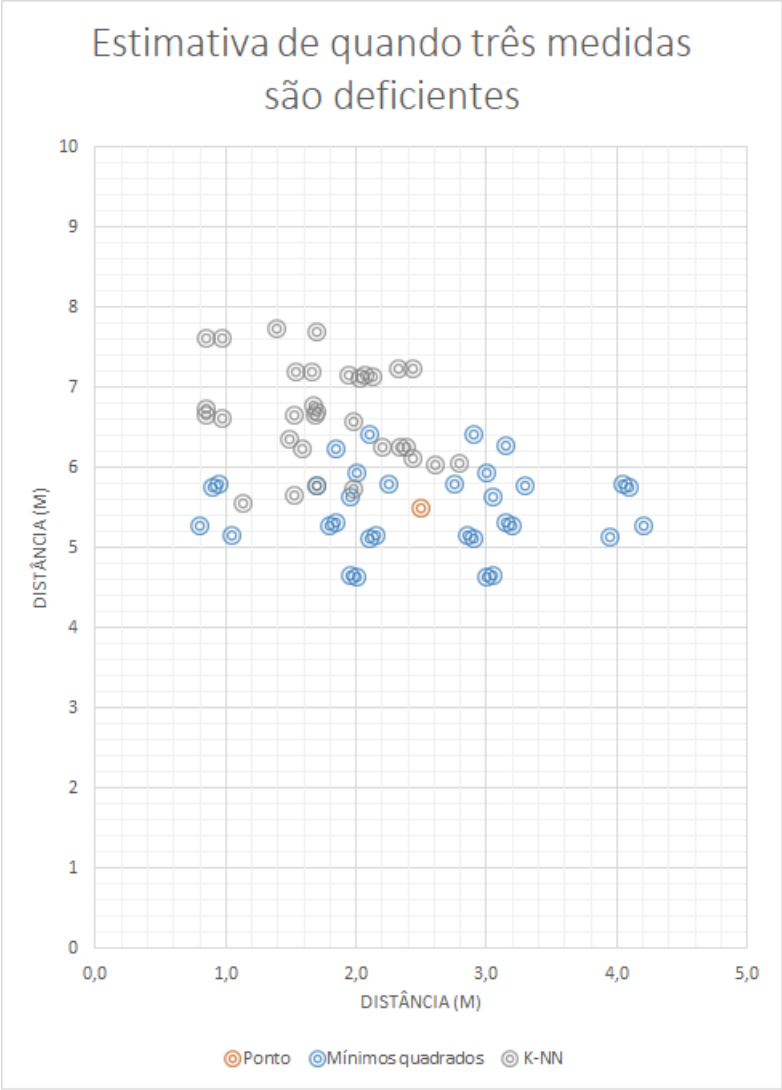
Fonte: Autor

Como no caso anterior pode-se notar um deslocamento dos resultados da técnica utilizando o K-NN em relação ao ponto real, sendo que as duas técnicas apresentaram erros grandes, na ordem de mais de 1,5 metros de diferença se comparado com a posição real em algumas combinações de análise. Apresentando também níveis de espalhamento maiores em comparação ao caso analisado pela figura 29.

Os resultados apresentados pela figura 31, são os casos onde se tem as combinações quando se há a estimativa de três medidas equivocadas para a distância tanto para mais quanto para menos, originando trinta e duas combinações.

Para o caso apresentado pela figura 31 já se pode notar uma maior dispersão das possíveis posições apresentadas pelo sistema em comparação a posição real. Verificando-se que os dados para o caso dos mínimos quadrados sofrem uma dispersão maior em x já os resultados apresentados pela técnica do K-NN se concentram mais, mas em um ponto deslocado da posição real.

FIGURA 31 – Caso onde três das medidas estavam equivocadas



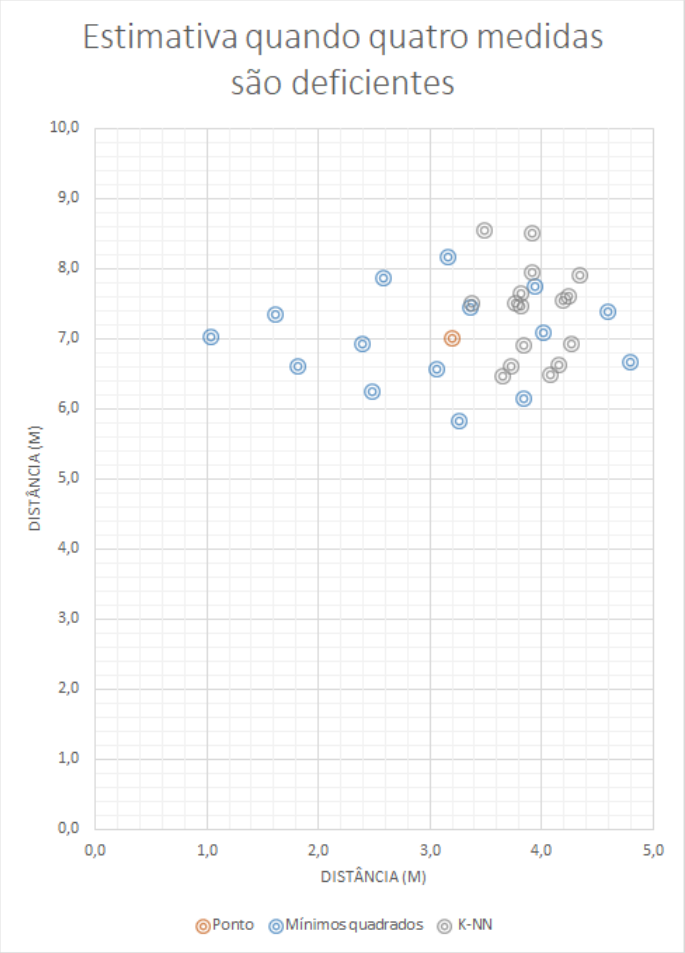
Fonte: Autor

Finalmente os dados apresentados pela figura 32, tratam do caso onde as quatro medidas apresentam deficiência quanto a estipulação da distância originando dezesseis combinações.

Neste ultimo caso mais radical, pode-se notar que os níveis de dispersão da estimativa da posição são altos podendo apresentar erros de até um pouco mais de 2,0 metros para o caso da técnica dos mínimos quadrados e de aproximadamente 1,5 metros para a técnica utilizando o K-NN, onde novamente apresenta uma maior

concentração de resultados em uma dada região, levando sempre em consideração o ponto real de medição..

FIGURA 32 – Caso onde todas as medidas estavam equivocadas



Fonte: Autor

Com os resultados apresentados nas figuras 29, 30, 31 e 32 foram realizadas as análises de valor médio das amostras e o desvio padrão apresentadas pela tabela 10.

Caso	Valor médio X	Valor médio Y	Desvio padrão X	Desvio padrão Y
Mínimos quadrados	2,10	6,03	0,59	0,32
K-NN	1,71	7,34	0,37	0,41
Mínimos quadrados	1,79	2,04	0,82	0,49
K-NN	1,24	1,01	0,64	0,86
Mínimos quadrados	2,50	5,50	0,93	0,50
K-NN	1,76	6,68	0,54	0,62
Mínimos quadrados	3,21	7,00	1,20	0,65
K-NN	3,91	7,39	0,28	0,66

TABELA 10 – Resultados dos pontos para as duas técnicas

Com a a ajuda da análise dos resultados apresentados na tabela 10 pode-se constatar que a técnica do K-NN possui uma precisão melhor se comparada a técnica

dos mínimos quadrados, onde através da análise do desvio padrão para o eixo x, dos valores das amostras, uma menor variabilidade dos resultados em todos os casos e em alguns casos bastante expressiva, como pode ser notado nas figuras 29, 30, 31 e 32. Para o caso do eixo y se tem uma melhor precisão par o caso dos mínimos quadrados, mas a técnica do K-NN apresenta resultados bem próximos ao apresentado pela outra técnica na grande maioria dos casos. Somente existindo o empecilho , no caso da utilização da técnica do K-NN, que está apresenta os seus resultados deslocados ao ponto ao ponto real, mas podendo ser corrigido. Diferentemente do fato onde existe uma grande variação dos resultados que não se é capaz de ajustar facilmente o sistema.

5.2 PROBABILIDADE DE SUCESSO DO SISTEMA CONSTRUÍDO

A probabilidade de sucesso do sistema está relacionada a mitigação da ocorrência ou não ocorrência de sucesso ou fracasso dos eventos partindo de um referencial para o caso do projeto analisado.

Na seguinte análise foram considerados os resultados obtidos como demostrados na seção 5.1 e considerado um raio de distância centrado no ponto de real de posição para todos os casos descritos. Onde para está analise foram analisados três casos: quando se é considerado um erro de 1,0 metros, um erro de 1,5 metros e um erro de 2,0 metros.

No primeiro caso analisado, caso menos seletivo, foram analisadas as probabilidades de sucesso de um raio de 2,0 metros onde se obtiveram os seguintes resultados apresentados pela tabela 11.

Caso	Mínimos Quadrado	K-NN
Caso de uma medida equivocada	100%	88%
Caso de duas medidas equivocadas	100%	92%
Caso de três medidas equivocadas	100%	81%
Caso de quatro medidas equivocadas	88%	100%

TABELA 11 – Probabiliza de sucesso para um raio de 2,0 metros

Nas duas técnicas analisadas na tabela 11, a probabilidade de sucesso da resposta do sistema estar dentro do raio de 2,0 metros de distância é satisfatória tento como pior resultado uma probabilidade de 81% para o caso da técnica do K-NN. Já o caso apresentado pela tabela 12 refere-se ao caso onde se tem um raio de 1,5 metros.

Caso	Mínimos Quadrado	K-NN
Caso de uma medida equivocada	100%	50%
Caso de duas medidas equivocadas	92%	42%
Caso de três medidas equivocadas	81%	47%
Caso de quatro medidas equivocadas	75%	88%

TABELA 12 – Probabiliza de sucesso para um raio de 1,5 metros

Nesta segunda análise apresentada pela tabela 12, pode-se notar uma piora significativa dos resultados para a técnica utilizando o K-NN apresentando ainda bons resultados no caso da técnica dos mínimos quadrados. Obtendo-se um percentual de 75% das amostras estando dentro do raio de 1,5 metros no pior caso. Sobre o terceiro caso onde se tem um raio de 1,0 metros são apresentados os dados da tabela 13.

Caso	Mínimos Quadrado	K-NN
Caso de uma medida equivocada	100%	13%
Caso de duas medidas equivocadas	63%	13%
Caso de três medidas equivocadas	66%	28%
Caso de quatro medidas equivocadas	25%	44%

TABELA 13 – Probabiliza de sucesso para um raio de 1,0 metros

Neste ultimo caso apresentado pela tabela 13, demostra uma grande piora na estimativa dos resultados para as duas técnicas, levando em considerações as premissas de análise mencionadas e considerando-se um raio de 1,0 metros.

Portanto para a analise de probabilidade de sucesso pode-se concluir que a técnica dos mínimos quadrados possui melhores resultados se comparada a técnica utilizando o K-NN, realizando uma melhor estimativa da posição considerando os três casos de raios analisados. A técnica do K-NN apresentou resultados menos satisfatórios pelo fato de apresentar um resultado deslocado, como questionado na seção 5.1, sendo que corrigido este deslocamento pode apresentar resultados melhores já que possui uma melhor precisão se comparada a técnica dos mínimos quadrados.

5.3 TEMPO DE RESPOSTA

O tempo de resposta do sistema foi também um requisito importante ao desenvolvimento do sistema, não tanto quanto a precisão e a probabilidade de sucesso, mas sem dúvidas um ponto importante a ser analisado.

O tempo de resposta do sistema deve-ser bom o suficiente a não proporcionar saltos de demostração da posição ao usuário, supondo-se o pior caso onde o usuário esteja andando pelo ambiente. A demostração da posição para o usuário seria atualizada a cada 3,0 segundo, um bom tempo de demostração já que se é suposto que tal não esteja se movendo em alta velocidade. Portanto o tempo de resposta do sistema deveria ser no mínimo menor que o tempo de uma nova atualização.

Para se realizar o testes foi realizada uma série de 30,0 requisições de atualização de posição e se foi anotado o tempo gasto do sistema de enviar as informações ao servidor e o servidor enviar novamente as informações ao sistema requisitante. Certamente para cada usuário pode se distinguir o resultado muito em fator ao poder de processamento do seu celular e da latência da rede internet, mas nas análises reali-

zadas o valor médio do tempo de resposta se obteve na ordem de 30,8 ms, possuindo um valor de pior caso de 48,5 ms.

5.4 DISPONIBILIDADE DE UTILIZAÇÃO

A disponibilidade do sistema está diretamente relacionada ao quantas conexões o servidor é possível de estabelecer com seus clientes, já que os dispositivos *beacons* enviam os seus sinais em *broadcast* sendo que um usuário dentro dos limites de alcance do sistema não teria problema em captar o sinal enviado pelos *beacons*.

No sistema desenvolvido se está configurado para se aceitar por parte do servidor um total de 100,0 conexões permitidas. Tomando-se como base o resultado médio de tempo de resposta apresentado pela seção 5.3 de 30,8 ms e considerando o pior caso onde todas as requisições chegassem ao servidor ao mesmo tempo, poderiam ser feitas 90,0 conexões de forma satisfatória e mesmo assim seria possível atender um tempo de resposta de atualização de 3,0 segundos.

Este problema poderia ser melhorado ao uso de *thread* no servidor, caso que não foi utilizado no sistema desenvolvido.

6 CONCLUSÃO

O presente trabalho atingiu seu objetivo que era de criar um sistema de localização *indoor* onde um usuário apto para o seu uso poderia se localizar fisicamente neste ambiente configurado para tal utilização.

O sistema foi construído em forma de aplicativo para *smartfones* para a usabilidade e apresentação dos dados de sua posição ao usuário. Assim um usuário dentro do campo de resposta do sistema e habilitado para tal fato conseguiria identificar sua posição de uma maneira simples e poder se orientar para uma eventual rota ou somente se identificar no ambiente.

Todo o processamento foi realizado em um servidor externo ao aplicativo, neste caso o dispositivo móvel do usuário enviava solicitava uma conexão com o servidor informado-o os dados e este retornava por meio de seu processamento uma dada posição que se referia a posição do usuário e está provável posição era demostrada ao usuário em seu aplicativo.

O servidor utilizou de duas técnicas de análise de ajustamento para melhorara a predição da posição: a técnica dos mínimos quadrados e a técnica utilizando o K-NN. As duas objetivaram seus pontos fortes e fracos como no caso onde a técnica que utiliza o K-NN possui uma melhor precisão mas menor probabilidade de sucesso se comparada a técnica dos mínimos quadrados. Mesmo assim as duas técnicas obtiveram resultados muito satisfatórios para uma precisão de 2,0 metros de distância possuindo resultados piores, principalmente no caso da técnica utilizando o K-NN, para um raio de precisão, centrado no ponto de analise, de 1,5 metros. Existindo um empecilho da técnica utilizando o K-NN se comparado a do mínimos quadrados, do fato desta necessitar a realização e treinamento do sistema prévio a aplicação do mesmo. A respeito do tempo de resposta as duas técnicas apresentaram resultados semelhantes.

Portanto pode-se concluir que o projeto atendeu as expectativas preliminares esperadas, possuindo ainda pontos que podem e ser melhorados e melhor desenvolvido, mas atingindo os seus objetivos estipulados no inicio do projeto.

6.1 TRABALHOS FUTUROS

O sistema que foi desenvolvido cuidou apresentar a usuário os dados de dua posição em um ambiente referenciado por meio de um dispositivo celular sendo o processamento realizado em um servidor localizado na rede. Para a estipulação da posição foram utilizadas duas técnicas principais: a de mínimos quadrados e a técnica do K-NN, mas que atuam de forma individual e sem conferencia se a posição encontrada é de fato a mais adequada. Um dos próximos e principais passos seria

de desenvolver no *software* do servidor um mecanismo de conferencia utilizando as duas técnicas, qual apresentar a melhor resposta a aquele ponto, com base nos seus valores anteriores e em conjunto com a outra técnica, para aumentar a probabilidade de sucesso e a precisão do sistema.

Um outro desenvolvimento poderia estar relacionado em expandir a utilização do sistema para um sistema para a localização de objetos e pessoas em um ambiente apto, como em um ambiente hospitalar que existe o compartilhamento de materiais de uso, na localização de pessoas em uma aglomeração de pessoas ou até mesmo a disponibilização do recurso compartilhado mais próximo possível.

Uma outra possibilidade de trabalho futuro poderia de desenvolver o sistema para a utilização de pessoas com deficiência, principalmente pessoas com problema de visão, onde algumas modificações no sistema poderiam orientar uma pessoa com deficiência visual a prosseguir e chegar a um determinado ponto em um ambiente *indoor*.

7 CÓDIGOS

7.1 MANIFEST

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="henrique.beaconapp">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />

    <!-- If your app targets Android 9 or lower, you can declare
         ACCESS_COARSE_LOCATION instead. -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

7.2 MAIN

```
package henrique.beaconapp;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.RemoteException;
import android.util.Log;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import org.altbeacon.beacon.Beacon;
import org.altbeacon.beacon.BeaconConsumer;
import org.altbeacon.beacon.BeaconManager;
import org.altbeacon.beacon.BeaconParser;
import org.altbeacon.beacon.RangeNotifier;
import org.altbeacon.beacon.Region;
import org.w3c.dom.Text;

import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.lang.reflect.Array;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Vector;

public class MainActivity extends AppCompatActivity implements BeaconConsumer {

    protected static final String TAG = "RangingActivity";
    private BeaconManager beaconManager;

    private static final int PERMISSION_REQUEST_COARSE_LOCATION = 1;

    int n = 4; // tamanho do vetor
    double v[] = new double[n]; // declarao do vetor "v"
    int i; // ndice ou  posio
    String responce;
    String[] array;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



// beacon
        beaconManager = BeaconManager.getInstanceForApplication(this);

        beaconManager.getBeaconParsers().add(new BeaconParser()
                .setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));


        beaconManager.setForegroundScanPeriod(3000l);
        beaconManager.bind(this);

        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(this);
            builder.setTitle("This app needs location access");
            builder.setMessage("Please grant location access so this app can
                detect beacons.");
            builder.setPositiveButton(android.R.string.ok, null);
            builder.setOnDismissListener(new
                DialogInterface.OnDismissListener() {
                @Override

                public void onDismiss(DialogInterface dialog) {
                    requestPermissions(new
                        String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                        PERMISSION_REQUEST_COARSE_LOCATION);
                }
            });
            builder.show();
        }
    }



}


public class Socket_AsyncTask extends AsyncTask<Void,Void,String> {


    @Override
    protected String doInBackground(Void... params) {

        String responce = null;
        try {

            System.out.println("Iniciando Cliente");

            System.out.println("Iniciando Conexo com o servidor");

            Socket socket = new Socket("192.168.100.46", 7000);
```

```
System.out.println("Conexo estabelecida.");

DataOutputStream dataOutputStream = new
    DataOutputStream(socket.getOutputStream());
PrintWriter out = new PrintWriter(new BufferedWriter(
        new OutputStreamWriter(socket.getOutputStream())), false);


//  configurao  para enviar os valores

String enviar = Arrays.toString(v);
System.out.println("Antes " + enviar);
enviar = enviar.replace(", ","#");
enviar = enviar.replace("[","");
enviar = enviar.replace("]","");


out.println(enviar);

out.flush();


// input
DataInputStream dataInputStream = new
    DataInputStream((socket.getInputStream()));

InputStream input = socket.getInputStream();
int lockSeconds = 10*1000;

long lockThreadCheckpoint = System.currentTimeMillis();
int availableBytes = input.available();
while(availableBytes <=0 && (System.currentTimeMillis() <
    lockThreadCheckpoint + lockSeconds)){
    try{Thread.sleep(10);}catch(InterruptedException
        ie){ie.printStackTrace();}
    availableBytes = input.available();
}


byte[] buffer = new byte[availableBytes];
input.read(buffer, 0, availableBytes);
responce = new String(buffer);
```

```
            System.out.println("Resposta: " + responce);

            //  configurao  para receber os valores
            array = responce.split("#");
            System.out.println("Resposta: " + array[0]);
            System.out.println("Resposta: " + array[1]);

            dataOutputStream.close();
            input.close();
            socket.close();

            circulo(array);



        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}


@Override
protected void onDestroy() {
    super.onDestroy();
    beaconManager.unbind(this);
}

@Override
protected void onPause() {
    super.onPause();
    beaconManager.unbind(this);
}

@Override
protected void onResume() {
```

```
    super.onResume();
    beaconManager.bind(this);
}



@Override
public void onBeaconServiceConnect() {

    RangeNotifier rangeNotifier = new RangeNotifier() {
        @Override
        public void didRangeBeaconsInRegion(Collection<Beacon> beacons,
            Region region) {

            if (beacons.size() > 0) {
                LinearLayout lyt = findViewById(R.id.linear_layout);
                lyt.removeAllViews();

                //  Configurao  da arraylist

                ArrayList<Beacon> mylist = new ArrayList<Beacon>(beacons);
                for (int j = 0; j < mylist.size(); j++) {
                    String rangedMinor = mylist.get(j).getId3().toString();
                    switch (rangedMinor) {
                        case "1":
                            v[0] =
                                Math.round((mylist.get(j).getRunningAverageRssi())*
                            10)/10.0;
                            v[0] = Math.abs(v[0]);
                            break;
                        case "2":
                            v[1] =
                                Math.round((mylist.get(j).getRunningAverageRssi())*
                            10)/10.0;
                            v[1] = Math.abs(v[1]);
                            break;
                        case "3":
                            v[2] =
                                Math.round((mylist.get(j).getRunningAverageRssi())*
                            10)/10.0;
                            v[2] = Math.abs(v[2]);
                            break;
```

```
                case "4":
                    v[3] =
                        Math.round((mylist.get(j).getRunningAverageRssi())*
                    10)/10.0;
                    v[3] = Math.abs(v[3]);
                    break;
                default:
                    System.out.println ("Algo de errado");


            }


        }

        for (Beacon beacon : beacons) {

            Log.d(TAG, "I see a beacon that is about
                "+beacon.getId1() + " Quantidade de beacons: " +
                beacons.size() );
            Beacon firstBeacon = beacons.iterator().next();
            logToDisplay( " Nome: " + beacon.getBluetoothName() + "
                Major : " + beacon.getId2() + " Minor : " +
                beacon.getId3() + "\n" + " Qtde de beacons: " +
                beacons.size() + " Mdia RSSI: " +
                String.format("%.1f",beacon.getRunningAverageRssi() )
                + " Pacotes: " +beacon.getPacketCount() );
            new Teste();
            // " Distncia: " + String.format("%.2f",
                beacon.getDistance()) +
        }
        beacons.clear();
        beaconManager.setDebug(true);



        Socket_AsyncTask task = new Socket_AsyncTask( );
        task.execute();


    }
```

```
        }

    };
    try {
        beaconManager.startRangingBeaconsInRegion(new
            Region("myRangingUniqueId", null, null, null));
        beaconManager.addRangeNotifier(rangeNotifier);
        beaconManager.startRangingBeaconsInRegion(new
            Region("myRangingUniqueId", null, null, null));
        beaconManager.addRangeNotifier(rangeNotifier);
    } catch (RemoteException e) { }
}

private void logToDisplay(final String line) {
    runOnUiThread(new Runnable() {
        public void run() {
            LinearLayout linearLayout = (LinearLayout)
                MainActivity.this.findViewById(R.id.linear_layout);
            TextView dist = new TextView(MainActivity.this);
            dist.setText(line);
            dist.setLayoutParams(new
                LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT));
            ImageView imageView2 = (ImageView)findViewById(R.id.mapa);
            Log.d(TAG, "Tamanho"+imageView2.getHeight() + " e " +
                imageView2.getWidth() );
            linearLayout.addView(dist);
        }
    });
}

private void circulo(final String[] array) {
    runOnUiThread(new Runnable() {
        public void run() {


            BitmapFactory.Options myOptions = new BitmapFactory.Options();
            myOptions.inScaled = false;
            myOptions.inPreferredConfig = Bitmap.Config.ARGB_8888;// important

            Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.mapa,myOptions);
```

```
            Paint paint = new Paint();
            paint.setAntiAlias(true);
            paint.setColor(Color.BLUE);

            Bitmap workingBitmap = Bitmap.createBitmap(bitmap);
            Bitmap mutableBitmap =
                workingBitmap.copy(Bitmap.Config.ARGB_8888, true);


            Canvas canvas = new Canvas(mutableBitmap);

            System.out.println ("Posicao"+ array[1] );
            float[] numbers = new float[array.length];
            for (int i = 0; i < array.length; ++i) {
                float number = Float.parseFloat(array[i]);
                float rounded = (int) Math.round(number * 100) / 100;
                numbers[i] = rounded;
            }

            ImageView imageView = (ImageView)findViewById(R.id.mapa);
            int width = imageView.getDrawable().getIntrinsicWidth();
            int height = imageView.getDrawable().getIntrinsicHeight();

            int x = 10;
            int y = 6;
            System.out.println ("Posicao "+ numbers[1] );
            int posicaox = (int)numbers[0]*width/x;
            int posicaoy = (int)numbers[1]*height/y;

            System.out.println ("Posicaoo : "+ posicaox + " tamanho " +
                posicaoy +" "+ height );
            canvas.drawCircle(posicaox, posicaoy, 15, paint);



            imageView.setAdjustViewBounds(true);
            imageView.setImageBitmap(mutableBitmap);


        }
    });
}
```

```
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[]
                                               grantResults) {
        switch (requestCode) {
            case PERMISSION_REQUEST_COARSE_LOCATION: {
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    Log.d(TAG, "coarse location permission granted");
                } else {
                    final AlertDialog.Builder builder = new
                        AlertDialog.Builder(this);
                    builder.setTitle("Functionality limited");
                    builder.setMessage("Since location access has not been
                        granted, this app will not be able to discover beacons
                        when in the background.");
                    builder.setPositiveButton(android.R.string.ok, null);
                    builder.setOnDismissListener(new
                        DialogInterface.OnDismissListener() {

                        @Override
                        public void onDismiss(DialogInterface dialog) {
                        }

                    });
                    builder.show();
                }
                return;
            }
        }
    }

}
```

7.3 LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <TextView
        android:id="@+id/Descricao"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="Beacons"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
            />


    <ScrollView
        android:id="@+id/beacons_app"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_below="@id/Descricao">

        <LinearLayout
            android:id="@+id/linear_layout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" />

    </ScrollView>

    <ImageView
        android:id="@+id/mapa"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/beacons_app"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="0dp"
        android:layout_marginBottom="6dp"
        app:srcCompat="@drawable/mapa" />
```

```
</RelativeLayout>
```

7.4 SERVIDOR

```
import socket
import math
import timeit

while 1:

    host = ''
    port = 7000
    addr = (host, port)
    serv_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    serv_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    serv_socket.bind(addr)
    serv_socket.listen(10)

    con, cliente = serv_socket.accept()
    recebidos = con.recv(1024)
    dados_recebidos = recebidos.decode()
    entrada = dados_recebidos.replace('\n', '')
    entrada = entrada.split("#")
    entrada = [float(i) for i in entrada]
    #entrada = list(map(float,entrada))
    print("Valor recebido: ", entrada)


    ########################## Variavies para se encontar a estimativa do
        quadrante mais adequado ###############

    inicio = timeit.default_timer()

    indice = 0
    i = 0
    j = 0

    auxiliar_posicao_1 = 0
```

```
auxiliar_posicao_2 = 0
auxiliar_posicao_3 = 0
auxiliar_posicao_4 = 0

 indice_da_posio_1  = []
 indice_da_posio_2  = []
 indice_da_posio_3  = []
 indice_da_posio_4  = []

 indice_da_posio_1_ordenado  = []
 indice_da_posio_2_ordenado  = []
 indice_da_posio_3_ordenado  = []
 indice_da_posio_4_ordenado  = []

distancia_1 = []
distancia_2 = []
distancia_3 = []
distancia_4 = []

distancia_1_auxiliar = []
distancia_2_auxiliar = []
distancia_3_auxiliar = []
distancia_4_auxiliar = []

resultado_1_ordenado = []
resultado_2_ordenado = []
resultado_3_ordenado = []
resultado_4_ordenado = []

############################ Dados e vetor de entrada com dados ####

A1 = [73.5, 74.0, 82.3, 79.7, 80.7, 84.4, 83.3, 85.2, 85.4, 85.6, 86.1,
   87.0, 87.4, 91.5,
     92.0] # Vetor que comtem os valores dos beacons mdios RSSI
A2 = [75.3, 72.9, 70.2, 76.9, 74.7, 73.8, 79.0, 78.1, 76.0, 83.5, 83.8,
   81.2, 87.3, 86.4, 85.0]
A3 = [87.5, 89.6, 90.2, 82.4, 82.8, 83.8, 77.0, 78.7, 79.0, 76.5, 76.9,
   77.5, 72.1, 76.4, 76.8]
A4 = [92.8, 90.2, 89.8, 89.1, 89.0, 88.5, 86.1, 84.0, 83.5, 83.8, 82.3,
   79.8, 83.1, 80.8, 74.5]

#   entrada = [70, 81, 79, 85] # Vetor de entrada de dados RSSI
```

```
##########################################################

while indice < len(
       A1): # Realiza a medida das distncias entre o valor d entrada e os
           valores previamente obtidos para o caso do beacon 1
    distancia_1.insert(indice, A1[indice] - entrada[0])
    if A1[indice] - entrada[0] <= 0:
        del (distancia_1[indice])
        distancia_1.insert(indice, (A1[indice] - entrada[0]) * -1)
    indice = indice + 1

indice = 0
while indice < len(A2):
    distancia_2.insert(indice, A2[indice] - entrada[1])
    if A2[indice] - entrada[1] <= 0:
        del (distancia_2[indice])
        distancia_2.insert(indice, (A2[indice] - entrada[1]) * -1)
    indice = indice + 1

indice = 0
while indice < len(A3):
    distancia_3.insert(indice, A3[indice] - entrada[2])
    if A3[indice] - entrada[2] <= 0:
        del (distancia_3[indice])
        distancia_3.insert(indice, (A3[indice] - entrada[2]) * -1)
    indice = indice + 1

indice = 0
while indice < len(A4):
    distancia_4.insert(indice, A4[indice] - entrada[3])
    if A4[indice] - entrada[3] <= 0:
        del (distancia_4[indice])
        distancia_4.insert(indice, (A4[indice] - entrada[3]) * -1)
    indice = indice + 1

resultado_1_ordenado = sorted(distancia_1) # Organizao dos vetores conforme
    as menores distancias para as maiores
resultado_2_ordenado = sorted(distancia_2)
resultado_3_ordenado = sorted(distancia_3)
resultado_4_ordenado = sorted(distancia_4)
```

```
distancia_1_auxiliar = distancia_1 # Criando vetor de backup para ser usado
    o vetor original posteriormente
distancia_2_auxiliar = distancia_2
distancia_3_auxiliar = distancia_3
distancia_4_auxiliar = distancia_4

while j < 4:

    i = 0
    while i < len(
           distancia_1_auxiliar): # Esta parte do programa cuidara de
               indicar as posies dos indices dos pontos correspondentes a
               microregio que se est localizado
        if resultado_1_ordenado[j] == distancia_1_auxiliar[i]:
             indice_da_posio_1 .insert(j, i + auxiliar_posicao_1)

            if ( indice_da_posio_1 [j - 1] ==  indice_da_posio_1 [j]) & j != 0:
                del ( indice_da_posio_1 [j])
                 indice_da_posio_1 .insert(j, i + auxiliar_posicao_1 - 1)

            auxiliar_posicao_1 = auxiliar_posicao_1 + 1
            del (distancia_1_auxiliar[i])
            break
        i = i + 1

    i = 0
    while i < len(distancia_2_auxiliar):
        if resultado_2_ordenado[j] == distancia_2_auxiliar[i]:
             indice_da_posio_2 .insert(j, i + auxiliar_posicao_2)

            if  indice_da_posio_2 [j - 1] ==  indice_da_posio_2 [j]:
                del ( indice_da_posio_2 [j])
                 indice_da_posio_2 .insert(j, i + auxiliar_posicao_2 - 1)

            auxiliar_posicao_2 = auxiliar_posicao_2 + 1
            del (distancia_2_auxiliar[i])
            break
        i = i + 1

    i = 0
    while i < len(distancia_3_auxiliar):
        if resultado_3_ordenado[j] == distancia_3_auxiliar[i]:
```

```
            indice_da_posio_3 .insert(j, i + auxiliar_posicao_3)

            if ( indice_da_posio_3 [j - 1] == indice_da_posio_3 [j]) & j != 0:
                del ( indice_da_posio_3 [j])
                 indice_da_posio_3 .insert(j, i + auxiliar_posicao_3 - 1)

            auxiliar_posicao_3 = auxiliar_posicao_3 + 1
            del (distancia_3_auxiliar[i])
            break
        i = i + 1

    i = 0
    while i < len(distancia_4_auxiliar):
        if resultado_4_ordenado[j] == distancia_4_auxiliar[i]:
             indice_da_posio_4 .insert(j, i + auxiliar_posicao_4)

            if ( indice_da_posio_4 [j - 1] == indice_da_posio_4 [j]) & j != 0:
                del ( indice_da_posio_4 [j])
                 indice_da_posio_4 .insert(j, i + auxiliar_posicao_4 - 1)

            auxiliar_posicao_4 = auxiliar_posicao_4 + 1
            del (distancia_4_auxiliar[i])
            break
        i = i + 1
    j = j + 1

indice_da_posio_1_ordenado  = sorted( indice_da_posio_1 )
indice_da_posio_2_ordenado  = sorted( indice_da_posio_2 )
indice_da_posio_3_ordenado  = sorted( indice_da_posio_3 )
indice_da_posio_4_ordenado  = sorted( indice_da_posio_4 )

print(indice_da_posio_1_ordenado)
print(indice_da_posio_2_ordenado)
print(indice_da_posio_3_ordenado)
print(indice_da_posio_4_ordenado)


vetor_concatenado = []
vetor_auxiliar = []
vetor_indice = []
posicoes = []
```

```
para_4 = 0
para_3 = 0
para_2 = 0
i = 0
ERRO = 0

vetor_concatenado = indice_da_posio_1 + indice_da_posio_2_ordenado +
    indice_da_posio_3_ordenado + indice_da_posio_4_ordenado  # Coloca as
     posies  estipuladas para cada beacons isolado em um unico vetor

posicao_1 = vetor_concatenado.count(
    0)  # Conta as possibilidades que existem no vetor_concatenado para cada
       possivel valor estipulado pelos beacons
posicao_2 = vetor_concatenado.count(1)
posicao_3 = vetor_concatenado.count(2)
posicao_4 = vetor_concatenado.count(3)
posicao_5 = vetor_concatenado.count(4)
posicao_6 = vetor_concatenado.count(5)
posicao_7 = vetor_concatenado.count(6)
posicao_8 = vetor_concatenado.count(7)
posicao_9 = vetor_concatenado.count(8)
posicao_10 = vetor_concatenado.count(9)
posicao_11 = vetor_concatenado.count(10)
posicao_12 = vetor_concatenado.count(11)
posicao_13 = vetor_concatenado.count(12)
posicao_14 = vetor_concatenado.count(13)
posicao_15 = vetor_concatenado.count(14)

vetor_auxiliar = [posicao_1, posicao_2, posicao_3, posicao_4, posicao_5,
    posicao_6, posicao_7, posicao_8, posicao_9,
                posicao_10, posicao_11, posicao_12, posicao_13, posicao_14,
                posicao_15] # Vetor que comtem o numero de aparies em cada
                    posio  no caso de 0 a 14
print(vetor_concatenado)
print(vetor_auxiliar)

quantas_vezes_numero_4 = vetor_auxiliar.count(
    4)  # Variaveis que vo conter o quanto de vezes existe um determinado
       valor no vetor concatenado
quantas_vezes_numero_3 = vetor_auxiliar.count(3)
quantas_vezes_numero_2 = vetor_auxiliar.count(2)
```

```
print(quantas_vezes_numero_4)
print(quantas_vezes_numero_3)
print(quantas_vezes_numero_2)

################ Verificando as mais provveis posies ######################

while (para_4 < quantas_vezes_numero_4) & (
       i < 4): # Aqui  ser  feita a  verificao  e incluido no vetor indice o
           valor da posio que mais aparecem no vetor concatenado
    vetor_indice.insert(i, vetor_auxiliar.index(4))
    del (vetor_auxiliar[vetor_indice[i]])
    vetor_auxiliar.insert(vetor_indice[i], -1)
    i = i + 1
    para_4 = para_4 + 1;

while (para_3 < quantas_vezes_numero_3) & (i < 4):
    vetor_indice.insert(i, vetor_auxiliar.index(3))
    del (vetor_auxiliar[vetor_indice[i]])
    vetor_auxiliar.insert(vetor_indice[i], -1)
    i = i + 1
    para_3 = para_3 + 1;

while (para_2 < quantas_vezes_numero_2) & (i < 4):
    vetor_indice.insert(i, vetor_auxiliar.index(2))
    del (vetor_auxiliar[vetor_indice[i]])
    vetor_auxiliar.insert(vetor_indice[i], -1)
    i = i + 1
    para_2 = para_2 + 1;

posicoes = vetor_indice # Vetor que contm os valores das posiveis posies
    com base no banco de dados e na entrada.

if len(vetor_indice) != 4: # Log de erro que ser usado posteriormente
    ERRO = -1
    posicoes = indice_da_posio_1_ordenado

print(" Posies  mais provaveis: ", posicoes)

vetor_posicao = []
vetor_posicao = [A1[0], A1[1], A1[3], A1[4], A1[5], A1[6], A1[7], A1[8],
    A1[9], A1[10], A1[11], A1[12], A1[13],
                A1[14],
```

```
                A2[0], A2[1], A2[3], A2[4], A2[5], A1[6], A2[7], A2[8],
                    A2[9], A2[10], A2[11], A2[12], A2[13],
                A2[14],
                A3[0], A3[1], A3[3], A3[4], A3[5], A3[6], A3[7], A3[8],
                    A3[9], A3[10], A3[11], A3[12], A3[13],
                A3[14],
                A4[0], A4[1], A4[3], A4[4], A4[5], A4[6], A4[7], A4[8],
                    A4[9], A4[10], A4[11], A4[12], A4[13],
                A4[14]]
# print(vetor_posicao)


i = 0
perda_de_caminho = []

while i < 4:
    perda_de_caminho.insert(i, entrada[i] - vetor_posicao[posicoes[i]])
    if perda_de_caminho[i] < 0:
        perda_de_caminho[i] = perda_de_caminho[i] * -1
    i = i + 1

# print("Perda de caminho : " , perda_de_caminho)

distancia = []

i = 0

while i < 4:
    distancia.insert(i, 10 ** ((perda_de_caminho[i] - 152.2004225) / 34) *
        1000)
    i = i + 1

print("Distancias: ", distancia)




distancia_virtual = 0
distancia_meio = 0
auxiliar_posicoes = 0

################################### Trocar estes pontos
```

```
    ############################

vetor_posicao_referenciada_x = [0.85, 2.55, 4.25, 0.85, 2.55, 4.25, 0.85,
    2.55, 4.25, 0.85, 2.55, 4.25, 0.85, 2.55,
                                4.25]
vetor_posicao_referenciada_y = [1.00, 1.00, 1.00, 3.00, 3.00, 3.00, 5.00,
    5.00, 5.00, 7.00, 7.00, 7.00, 9.00, 9.00,
                                9.00]

auxiliar_posicoes = sorted(posicoes)
print("Auxiliar  posies ; ", auxiliar_posicoes)

x_ponto_1 = vetor_posicao_referenciada_x[auxiliar_posicoes[2]]
y_ponto_1 = vetor_posicao_referenciada_y[auxiliar_posicoes[2]]
x_ponto_2 = vetor_posicao_referenciada_x[auxiliar_posicoes[3]]
y_ponto_2 = vetor_posicao_referenciada_y[auxiliar_posicoes[3]]
x_ponto_3 = vetor_posicao_referenciada_x[auxiliar_posicoes[0]]
y_ponto_3 = vetor_posicao_referenciada_y[auxiliar_posicoes[0]]
x_ponto_4 = vetor_posicao_referenciada_x[auxiliar_posicoes[1]]
y_ponto_4 = vetor_posicao_referenciada_y[auxiliar_posicoes[1]]

print("Valor das  posies :", x_ponto_1, y_ponto_1, x_ponto_2, y_ponto_2,
    x_ponto_3, y_ponto_3, x_ponto_4, y_ponto_4)

################################################################################

xp = 0
yp = 0

i = 0 # Indice
j = 0
indice_x_1 = 0
indice_y_1 = 0
indice_x_2 = 0
indice_y_2 = 0

ponto_1_x = [
              0] * 90 #  Criao  dos vetores que receberam os valores dos
                  pontos das coordenadas x e y para as distancias obtidas
ponto_1_y = [0] * 90
ponto_2_x = [0] * 90
ponto_2_y = [0] * 90
```

```
ponto_3_x = [0] * 90
ponto_3_y = [0] * 90
ponto_4_x = [0] * 90
ponto_4_y = [0] * 90

x_virtual_1 = 0
y_virtual_1 = 0
x_virtual_2 = 0
y_virtual_2 = 0
distancia_virtual = 0

angulo = 0.017453292 #  Criao  da variavel angulo em radianos

while i < 90: # Calculo de todos os pontos de x e y para cada ponto
    ponto_1_x[i] = distancia[0] * math.cos(angulo) # Ponto 1 x = mximo
    ponto_1_y[i] = distancia[0] * math.sin(angulo) # Ponto 1 y = mnimo
    ponto_2_x[i] = distancia[1] * math.cos(angulo) # Ponto 2 x = mximo
    ponto_2_y[i] = distancia[1] * math.sin(angulo) # Ponto 2 y = mnimo
    ponto_3_x[i] = distancia[2] * math.cos(angulo) # Ponto 3 x = mximo
    ponto_3_y[i] = distancia[2] * math.sin(angulo) # Ponto 3 x = mnimo
    ponto_4_x[i] = distancia[3] * math.cos(angulo) # Ponto 4 x = mximo
    ponto_4_y[i] = distancia[3] * math.sin(angulo) # Ponto 4 x = mnimo

    angulo = angulo + 0.017453292
    i = i + 1

i = 89 #  Criao  das vaiaveis que auxiliaram no desenvolvimento da lgica
    para descobir qual indice de x[i] e y[i] se est trabalhando
j = 89
aux_x_1 = 100
aux_y_1 = 100
aux_x_2 = 100
aux_y_2 = 100

while i > 0: # O primeiro while refere-se a variao do valor de x no ponto 1
    while j > 0: # O segundo while refere-se a vaiao do valor de x no ponto 2
        x_virtual_1 = (x_ponto_1 + ponto_1_x[j]) - (x_ponto_2 - ponto_2_x[
            i]) # Consideando os pontos de referencia sabidos pega-se o
                valor de x calculado e soma-se ou subtrai do ponto de
                referencia
        y_virtual_1 = (y_ponto_1 - ponto_1_y[90 - j]) - (y_ponto_3 +
            ponto_3_y[90 - i])
```

```
        if x_virtual_1 < 0: # Se o resultado da conta possuir um valor
            negativo este e colocado em valor positivo
            x_virtual_1 = x_virtual_1 * -1

        if y_virtual_1 < 0:
            y_virtual_1 = y_virtual_1 * -1

        if aux_x_1 >= x_virtual_1: # A variavel aux auxilia na comparao
        do item anterior se este o valor atual possuir menor valor do que o
            anterior este substitui a varivel aux e  gaurdado o valo do indice
            aux_x_1 = x_virtual_1
            indice_x_1 = i

        if aux_y_1 >= y_virtual_1:
            aux_y_1 = y_virtual_1
            indice_y_1 = i

        x_virtual_2 = (x_ponto_4 - ponto_4_x[j]) - (x_ponto_3 + ponto_3_x[i])
        y_virtual_2 = (y_ponto_4 + ponto_4_y[90 - j]) - (y_ponto_2 -
            ponto_2_y[90 - i])

        if x_virtual_2 < 0:
            x_virtual_2 = x_virtual_2 * -1

        if y_virtual_2 < 0:
            y_virtual_2 = y_virtual_2 * -1

        if aux_x_2 >= x_virtual_2:
            aux_x_2 = x_virtual_2
            indice_x_2 = i

        if aux_y_2 >= y_virtual_2:
            aux_y_2 = y_virtual_2
            indice_y_2 = i

        j = j - 1
    j = 89
    i = i - 1

x_virtual_1 = x_ponto_1 + ponto_1_x[
    indice_x_1] # Encontrando as coordenadas dos pontos virtuais obtidos po
```

```
        meio das aproximaes
y_virtual_1 = y_ponto_1 - ponto_1_y[indice_y_1]
x_virtual_2 = x_ponto_4 - ponto_4_x[indice_x_2]
y_virtual_2 = y_ponto_4 + ponto_4_y[indice_y_2]

# print(x_virtual_1, y_virtual_1)
# print(y_virtual_1, y_virtual_2)

distancia_virtual = math.sqrt(math.pow((x_virtual_1 - x_virtual_2), 2) +
    math.pow((y_virtual_1 - y_virtual_2),
  2))  # Calculando a distancia entre os dois pontos virtuais eo ponto mdio
      considerando se um triangulo retangulo
distancia_meio = distancia_virtual / 2

if x_virtual_1 > x_virtual_2: # Achamdo o ponto estimado, considerando trs
    casos:
    xp_distancia = ((
                               x_virtual_1 - x_virtual_2) * distancia_meio) /
                                   distancia_virtual
                               # 1   -> Onde x_virtual 1 est mais a direita de
                                   x_virtual 2
    xp = x_virtual_1 - xp_distancia
elif x_virtual_1 < x_virtual_2: # 2 -> Onde x_virtual 1 est mais a esquerda
    de x_virtual 2
    xp_distancia = ((x_virtual_2 - x_virtual_1) * distancia_meio) /
        distancia_virtual
    xp = x_virtual_1 + xp_distancia
else: # 3   -> Onde x_virtual 1 e x_virtual 2 so iguais
    xp = x_virtual_1

if y_virtual_1 > y_virtual_2:
    yp_distancia = ((y_virtual_1 - y_virtual_2) * distancia_meio) /
        distancia_virtual
    yp = y_virtual_1 - yp_distancia
elif y_virtual_1 < y_virtual_2:
    yp_distancia = ((y_virtual_2 - y_virtual_1) * distancia_meio) /
        distancia_virtual
    yp = y_virtual_1 + yp_distancia
else:
    yp = y_virtual_1
```

```
x_beacon = [0.0, 5.0, 0.0, 5.0]
y_beacon = [0.0, 0.0, 9.3, 9.3]

distancia_minimos = []
xp_minimos = []
yp_minimos = []
xp_minimos_resultado = 0
yp_minimos_resultado = 0
auxiliar_negativo_x = 0
auxiliar_negativo_y = 0

i = 0

while i < 4:
    distancia_minimos.insert(i, 10 ** ((entrada[i] - 152.2004225) / 34) *
        1000)
    i = i + 1

xp_minimos.insert(0, (((distancia_minimos[1] ** 2 - distancia_minimos[2] **
    2 + y_beacon[2] ** 2 - y_beacon[
    1] ** 2 + x_beacon[2] ** 2 - x_beacon[1] ** 2) * (2 * (y_beacon[1] -
        y_beacon[0]))) - ((distancia_minimos[
                0] ** 2 -
            distancia_minimos[
                1] ** 2 + y_beacon[
                1] ** 2 - y_beacon[
                0] ** 2 + x_beacon[
                1] ** 2 - x_beacon[
                0] ** 2) * (2 * (
            y_beacon[2] - y_beacon[1]))))) / (
                        2 * (x_beacon[2] - x_beacon[1]) * 2 *
                            (y_beacon[1] - y_beacon[0]) - 2 * (
                            x_beacon[1] - x_beacon[0]) * 2 * (y_beacon[2]
                                - y_beacon[1])))
xp_minimos.insert(1, (((distancia_minimos[1] ** 2 - distancia_minimos[3] **
    2 + y_beacon[1] ** 2 - y_beacon[
    1] ** 2 + x_beacon[3] ** 2 - x_beacon[1] ** 2) * (2 * (y_beacon[1] -
        y_beacon[0]))) - ((distancia_minimos[
            0] ** 2 -
        distancia_minimos[
            1] ** 2 + y_beacon[
            1] ** 2 - y_beacon[
```

```
            0] ** 2 + x_beacon[
            1] ** 2 - x_beacon[
            0] ** 2) * (2 * (
            y_beacon[3] - y_beacon[1]))))) / (
                            2 * (x_beacon[3] - x_beacon[1]) * 2 *
                                (y_beacon[1] - y_beacon[0]) - 2 * (
                                x_beacon[1] - x_beacon[0]) * 2 * (y_beacon[3]
                                    - y_beacon[1])))
xp_minimos.insert(2, (((distancia_minimos[2] ** 2 - distancia_minimos[3] **
    2 + y_beacon[3] ** 2 - y_beacon[
    2] ** 2 + x_beacon[3] ** 2 - x_beacon[2] ** 2) * (2 * (y_beacon[2] -
        y_beacon[1]))) - ((distancia_minimos[
                1] ** 2 -
            distancia_minimos[
                2] ** 2 + y_beacon[
                2] ** 2 - y_beacon[
                1] ** 2 + x_beacon[
                2] ** 2 - x_beacon[
                1] ** 2) * (2 * (
            y_beacon[3] - y_beacon[2]))))) / (
                            2 * (x_beacon[3] - x_beacon[2]) * 2 *
                                (y_beacon[2] - y_beacon[1]) - 2 * (
                                x_beacon[2] - x_beacon[1]) * 2 * (y_beacon[3]
                                    - y_beacon[2])))
xp_minimos.insert(3, (((distancia_minimos[2] ** 2 - distancia_minimos[3] **
    2 + y_beacon[3] ** 2 - y_beacon[
    2] ** 2 + x_beacon[3] ** 2 - x_beacon[2] ** 2) * (2 * (y_beacon[2] -
        y_beacon[0]))) - ((distancia_minimos[
                    0] ** 2 -
                distancia_minimos[
                    2] ** 2 + y_beacon[
                    2] ** 2 - y_beacon[
                    0] ** 2 + x_beacon[
                    2] ** 2 - x_beacon[
                    0] ** 2) * (2 * (
            y_beacon[3] - y_beacon[2]))))) / (
                            2 * (x_beacon[3] - x_beacon[2]) * 2 *
                                (y_beacon[2] - y_beacon[0]) - 2 * (
                                x_beacon[2] - x_beacon[0]) * 2 * (y_beacon[3]
                                    - y_beacon[2])))

yp_minimos.insert(0, (((distancia_minimos[1] ** 2 - distancia_minimos[2] **
```

```
    2 + y_beacon[2] ** 2 - y_beacon[
    1] ** 2 + x_beacon[2] ** 2 - x_beacon[1] ** 2) * (2 * (x_beacon[1] -
        x_beacon[0]))) - ((distancia_minimos[
                    0] ** 2 -
                distancia_minimos[
                    1] ** 2 + y_beacon[
                    1] ** 2 - y_beacon[
                    0] ** 2 + x_beacon[
                    1] ** 2 - x_beacon[
                    0] ** 2) * (2 * (
        x_beacon[2] - x_beacon[1])))) / (
                            2 * (x_beacon[1] - x_beacon[0]) * 2 *
                                (y_beacon[2] - y_beacon[1]) - 2 * (
                                x_beacon[2] - x_beacon[1]) * 2 * (y_beacon[1]
                                    - y_beacon[0])))
yp_minimos.insert(1, (((distancia_minimos[1] ** 2 - distancia_minimos[3] **
    2 + y_beacon[3] ** 2 - y_beacon[
    1] ** 2 + x_beacon[3] ** 2 - x_beacon[1] ** 2) * (2 * (x_beacon[1] -
        x_beacon[0]))) - ((distancia_minimos[
                        0] ** 2 -
                    distancia_minimos[
                        1] ** 2 + y_beacon[
                        1] ** 2 - y_beacon[
                        0] ** 2 + x_beacon[
                        1] ** 2 - x_beacon[
                        0] ** 2) * (2 * (
            x_beacon[3] - x_beacon[1])))) / (
                            2 * (x_beacon[1] - x_beacon[0]) * 2 *
                                (y_beacon[3] - y_beacon[1]) - 2 * (
                                x_beacon[3] - x_beacon[1]) * 2 * (y_beacon[1]
                                    - y_beacon[0])))
yp_minimos.insert(2, (((distancia_minimos[2] ** 2 - distancia_minimos[3] **
    2 + y_beacon[3] ** 2 - y_beacon[
    2] ** 2 + x_beacon[3] ** 2 - x_beacon[2] ** 2) * (2 * (x_beacon[2] -
        x_beacon[1]))) - ((distancia_minimos[
                        1] ** 2 -
                    distancia_minimos[
                        2] ** 2 + y_beacon[
                        2] ** 2 - y_beacon[
                        1] ** 2 + x_beacon[
                        2] ** 2 - x_beacon[
                        1] ** 2) * (2 * (
```

```
            x_beacon[3] - x_beacon[2]))))) / (
                                2 * (x_beacon[2] - x_beacon[1]) * 2 *
                                    (y_beacon[3] - y_beacon[2]) - 2 * (
                                    x_beacon[3] - x_beacon[2]) * 2 * (y_beacon[2]
                                        - y_beacon[1])))
yp_minimos.insert(3, (((distancia_minimos[2] ** 2 - distancia_minimos[3] **
    2 + y_beacon[3] ** 2 - y_beacon[
    2] ** 2 + x_beacon[3] ** 2 - x_beacon[2] ** 2) * (2 * (x_beacon[2] -
        x_beacon[0]))) - ((distancia_minimos[
                                0] ** 2 -
                            distancia_minimos[
                                2] ** 2 + y_beacon[
                                2] ** 2 - y_beacon[
                                0] ** 2 + x_beacon[
                                2] ** 2 - x_beacon[
                                0] ** 2) * (2 * (
            x_beacon[3] - x_beacon[2]))))) / (
                                2 * (x_beacon[2] - x_beacon[0]) * 2 *
                                    (y_beacon[3] - y_beacon[2]) - 2 * (
                                    x_beacon[3] - x_beacon[2]) * 2 * (y_beacon[2]
                                        - y_beacon[0])))

i = 0

while i < 4:
    if xp_minimos[i] < 0:
        auxiliar_negativo_x = xp_minimos[i] * -1
        del (xp_minimos[i])
        xp_minimos.insert(i, auxiliar_negativo_x)

    if yp_minimos[i] < 0:
        auxiliar_negativo_y = yp_minimos[i] * -1
        del (yp_minimos[i])
        yp_minimos.insert(i, auxiliar_negativo_y)

    i = i + 1

xp_minimos_resultado = sum(xp_minimos) / len(xp_minimos)
yp_minimos_resultado = sum(yp_minimos) / len(yp_minimos)

print("Minimos quadrados: ", xp_minimos_resultado, yp_minimos_resultado)
```

```
print("Pontos XP e YP: ", xp, yp) # Ponto localizado pela lgica

fim = timeit.default_timer()

print("Tempo gasto: ", fim-inicio)

envio_1 = str(xp_minimos_resultado)
envio_2 = str(yp_minimos_resultado)
pontos_envio = envio_1 + "#" + envio_2

con.send(pontos_envio.encode())
serv_socket.close()
```

REFERÊNCIAS

ALBACORE, Conectware Tecnology. Sistemas de localização em tempo real (RTLS). Av. Fagundes Filho, 141 - 04304-010 - SP, 2011. Albacore comércio e serviços de informática Ltda. Citado nas pp. 24–27.

ALBERT HUANG, Larry Rudolph. **Bluetooth for Programmers**. [S.l.: s.n.], 2005. Acesso em 25 de março de 2018. Citado na p. 34.

ALTBEACON. **Android Beacon Library**. [S.l.: s.n.], 2019. Disponível em: <https://altbeacon.github.io/android-beacon-library/>. Acesso em: 20 nov. 2019. Citado na p. 64.

ARAUJO, Grabriel L. S. et al. Protocolo de Roteamento AODV Ad-hoc On-demand Distance Vector, 2007. Citado na p. 34.

BENSKY, Alan. **Wireless Position Technologies and Applications (GNSS Technology and Applications)**. 2. ed. [S.l.]: Artech House Publishers, 2008. Citado nas pp. 23–26, 28–31.

COMPANY, JNHuaMao Technology. **iBeacons Datasheet**. [S.l.], 2014. Disponível em: <http://jnhuamao.cn/iBeacon_en.zip>. Citado na p. 72.

DEVELOPERS, Android. **Activity**. [S.l.: s.n.], 2018. Disponível em: <https://developer.android.com/reference/android/app/Activity>. Acesso em: 18 out. 2019. Citado nas pp. 41–43.

ERGAWY, Amr. Routing Protocols Wireless for Ad Hoc Wireless Networks: Classifications of Protocols and A review of Table Driven Protocols. v. 1, 2006. Citado na p. 33.

FERREIRA, João; RESENDE, Ricardo; MARTINHO, Stuart. Beacons and BIM Models for Indoor Guidance and Location. **Sensors**, v. 18, p. 4374, dez. 2018. DOI: 10.3390/s18124374. Citado nas pp. 43, 65.

FTDI. **FT232RUSB UART ICDatasheet**. [S.l.: s.n.], 2019. Disponível em: <https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>. Acesso em: 14 nov. 2019. Citado na p. 70.

GEMAEL, Camil; MACHADO, Alvaro Muriel Lima. **Introdução ao ajustamento de observações**. 2. ed. [S.l.]: Editora UFPR, 2015. ISBN 978-85-8480-008-7. Citado nas pp. 39–41.

GRUS, Joel. **Data Science do zero - Primeiras regras com o python**. 1. ed. [S.l.]: Alta Books, 2016. Capítulo 12. ISBN 978-85-7608-998-8. Citado nas pp. 38, 39.

HEKMAT, Ramin. **Ad-hoc Networks: Fundamental properties and network topologies**. 2. ed. [S.l.]: Springer, 2006. ISBN 978-1-4020-5166-1. Citado na p. 33.

HUANG, Xu; BARRALET, Mark. **Accuracy of location Identification with Antenna Polarization on RSSI)**. Hong Kong: [s.n.], mar. 2009. v. 1. Proceedings of the International MultiConference of Engineers and Computer Scientist. Citado nas pp. 18, 19.

HUH, Jun-Ho; SEO, Kyungryong. An Indoor Location-Based Control System Using Bluetooth Beacons for IoT Systems. **Sensors**, v. 17, p. 2917, dez. 2017. DOI: 10.3390/s17122917. Citado nas pp. 45, 46.

INC, Apple. **Proximity Beacon Specification**. [S.l.: s.n.], 2015. Disponível em: <https://developer.apple.com/ibeacon/>. Acesso em: 20 nov. 2019. Citado nas pp. 64, 65.

KARYDIS, Thrasyvoulos. **Bluetooth Interfacing with HM-10**. [S.l.: s.n.], 2015. Disponível em: <http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth.html>. Acesso em: 14 nov. 2019. Citado na p. 70.

KE, Chih-Kun et al. Developing a BLE Beacon-Based Location System Using Location Fingerprint Positioning for Smart Home Power Management. **Energies**, v. 11, p. 3464, dez. 2018. DOI: 10.3390/en11123464. Citado nas pp. 46, 47.

KROGH, Anders; VEDELBY, Jesper. **Neural Network Ensembles, Cross Validation and Active Learning in NIPS - Advances in neural information processing systems**. [S.l.: s.n.], 1995. v. 7. The MIT Press. Citado na p. 37.

KUPPER, Axel. **Location-Based Services: Fundaments and Operation**. Wiley: [s.n.], 2005. Citado nas pp. 17, 18, 20–22.

LABIOD, Houda; AFIFI, Hossam; SANTIS, Constantino De. **Wi-Fi Bluetooth ZigBee and WiMax**. [S.l.: s.n.], 2007. Springer. Citado na p. 34.

LAU, Erin-Ee-Lin. **Enhanced RSSI-based high acuracy real-time user location tracking system for inddor and outdoor environments**. 2. ed. [S.l.: s.n.], jun. 2008. v. 1. International Journal on Smart Sensing and Intelligent Systems. Citado nas pp. 17, 28, 31.

LEE, Kung-Chung. **Localization system using signal strength fingerprinting**. [S.l.: s.n.], 2010. Electrical and Computer Engineering. Citado nas pp. 19, 20.

MENDONÇA, F et al. Análise do ajustamento por mínimos quadrados de uma trilateração topográfica com injunções nos planos UTM e topocêntrico. **III Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação**, jul. 2010. Citado na p. 40.

MITTAL, Shubhi. **iBeacon vs Eddystone: Which one works better for your Pilot Project?** [S.l.: s.n.], 2016. Disponível em: <https://blog.beaconstac.com/2016/01/ibeacon-vs-eddystone/>. Acesso em: 14 nov. 2019. Citado na p. 44.

MOECKE, Marcos. Multiplexação por divisão de tempo e transmissão digital, 2017. UNISINOS - Universidade do Vale do Rio dos Sinos. Citado nas pp. 35, 36.

OSÓRIO, Fernado. Redes neurrais - Aprendizado artificial. v. 1, 2017. Forum de I.A. '99. Citado nas pp. 36, 37.

PANDIT, Abhiemanyu. **How to Use HM-10 BLE Module with Arduino to Control an LED using Android App**. [S.l.: s.n.], 2019. Disponível em: <https://circuitdigest.com/microcontroller-projects/how-to-use-arduino-and-hm-10-ble-module-to-control-led-with-android-app>. Acesso em: 14 nov. 2019. Citado na p. 70.

PARK, ChulYoung. **Localization Algorithm Design and Implementation to Utilization RSSI and AOA of Zigbee**. [S.l.: s.n.], mai. 2010. Future Information Technology (FutureTech), 5th Internation Conference on. Citado nas pp. 25, 29, 32.

PERKIN CHARLES; ROYER, Elizabeth M. Ad-hoc On-Demand Distance Vector Routing, 2002. Citado na p. 34.

PUC, Rio. **Modelos de propagação em ambientes fechados**. R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900: PUC-RIO, 2016. Citado nas pp. 16–18, 20, 21, 23.

SILVA, César D.; KOMATI, Karin S. Sistema mobile para localização indoor usando tratamento de posição e correção de rota. Coordenação de Informática - Instituto Federal do Espírito Santo (IFES), 2015. XXXVII Congresso da sociedade brasileira de computação. Citado na p. 20.

YADAV, Narendra Singh; YADAV, R. P. Y. Performance Comparison and Analysis of Table-Driven and On-Demand Routing Protocols for Mobile Ad-hoc Networks, 2007. Citado na p. 33.