UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Utilização de Rede Neural para Determinar a Localização Interna por Meio de Bluetooth Beacon

Trabalho de Conclusão de Curso

Leonardo Vanzin

Cascavel–PR

2021

UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Leonardo Vanzin

Utilização de Rede Neural para Determinar a Localização Interna por Meio de Bluetooth Beacon

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientador(a): Marcio Seiji Oyamada

Cascavel–PR

2021

LEONARDO VANZIN

UTILIZAÇÃO DE REDE NEURAL PARA DETERMINAR A LOCALIZAÇÃO INTERNA POR MEIO DE BLUETOOTH BEACON

Monografia apresentada como requisito parcial para obtendo do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Marcio Seiji Oyamada (Orientador)
Colegiado de Ciência da Computação, UNIOESTE

Prof. Adriana Postal
Colegiado de Ciência da Computação, UNIOESTE

Prof. Luiz Antonio Rodrigues
Colegiado de Ciência da Computação, UNIOESTE

Cascavel, 29 de Julho de 2022

Resumo

A crescente utilização de *smartphones* abriu possibilidades para determinar a localização exata de seus usuários. Apesar do GPS ser amplamente utilizado para localização externa, ele é impreciso e em alguns casos indisponível em ambientes internos. Uma das soluções para determinar a localização interna é utilizando *beacons*, dispositivos que utilizam a tecnologia *Bluetooth Low Energy*. Utilizando *beacons* é possível determinar a distância entre o dispositivo e o *beacon* utilizando o *Received Signal Strength Indicator* (RSSI). A localização do objeto dentro do ambiente pode ser determinada utilizando 3 *beacons* e aplicando métodos como a trilateração. Este trabalho teve como objetivo avaliar o uso de redes neurais para determinar a distância e a posição do objeto em um ambiente interno. Em um primeiro experimento é comparado o *Log Distance Path Loss Model* e rede neural de regressão para determinar distância entre o *beacon* e o objeto. Em um segundo experimento, o objetivo é comparar qual método é mais adequado para determinar a posição de um objeto na sala utilizando três *beacons*. Foram avaliados os métodos: a) trilateração com cálculo de distância utilizando o método *Log Distance Path Loss Model*; b) trilateração com cálculo de distância utilizando rede neural e; c) determinação da posição utilizando rede neural. Nos resultados obtidos foi percebido problemas no método de trilateração e que o método RNA foi o melhor para determinar distância, com erro médio de $-0,228$ metros, comparado ao erro médio de $24,50$ metros do método de *Log Distance Path Loss Model*. Para determinar a localização a RNA também obteve o melhor resultado, com erro médio de $0,084$ metros para coordenadas x e $0,114$ metros para coordenadas y.

Palavras-chave: *Bluetooth Low Energy*, *Beacons*, Rede Neural Artificial, Localização Interna, Trilateração, *Log Distance Path Loss Model*

Lista de figuras

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

BLE	*Bluetooth Low Energy*
CL	*Centroid Localization*
dBm	*Decibel Milliwatts*
FF-MLPs	*Feed Forward Multilayer Perceptron*
GPS	*Global Positioning System*
MAE	*Mean Absolute Error*
MSE	*Mean Square Error*
RFID	*Radio Frenquency Identification*
RNA	Rede Neural Artificial
RSSI	*Received Signal Strength Indicator*
SIG	*Special Interest Group*
UUID	*Universally Unique Identifier*
UWB	*Ultra-Wide Band*

Lista de símbolos

d_i	Distância do nó alvo ao nó ancôra
x	Coordenada x do nó alvo
y	Coordenada y do nó alvo
x_i	Coordenada x do nó i
y_i	Coordenada y do nó i
A_1	Variável intermediária
A_2	Variável intermediária
A_3	Variável intermediária
B_1	Variável intermediária
B_2	Variável intermediária
B_3	Variável intermediária
C_1	Variável intermediária
C_2	Variável intermediária
C_3	Variável intermediária
x_e	Coordenada x calculada do nó alvo
y_e	Coordenada y calculada do nó alvo
$RSSI$	Valor do RSSI (potência do sinal)
RSS_{d_0}	Valor RSSI para a distância d_0
n	Expoente *Path Loss*
d	Distância real entre o *beacon* e o *smartphone*
d_0	Distância utilizada no RSS_{d_0}
X_σ	Variável média randômica com desvio padrão
l	Número de lajotas referente a distância

Sumário

1 Introdução

Com a crescente utilização de smartphones, estimar a sua localização exata abre oportunidades para um grande número de aplicações, desde uma simples busca de usuário até proporcionar segurança. Para estes tipos de tarefas, o modelo mais comumente utilizado é o *Global Positioning System* (GPS), que oferece a posição do alvo em uma região com pouca taxa de erro. No entanto, este tipo de tecnologia possui maior eficácia em ambientes externos, aumentando a taxa de erro ao tentar determinar a localização do alvo em um ambiente interno (RADOI; CIRIMPEI; RADU, 2019).

Para isso tem sido aplicado tecnologias e métodos para obter melhores resultados, sendo principalmente utilizado o Infravermelho, *Radio Frenquency Identification* (RFID), Ultrassom, *Bluetooth*, *Ultra-Wide Band* (UWB), WiFi e outras tecnologias sem fio. Dentre elas, o *Bluetooth* é um dos que consome menos energia enquanto se comunica com outros terminais (KAJIKAWA et al., 2016). Devido ao *Bluetooth* possuir a tecnologia *Frequency Hopping*, ele possui maior resistência a interferências do ambiente durante a transmissão de dados, comparado a outras tecnologias (KAJIKAWA et al., 2016). O *Frequency Hopping* divide a frequência em canais de transmissão, sendo assim, caso dois dispositivos estejam transmitindo na mesma faixa de frequência o canal é trocado e a interferência deixa de existir (HAN et al., 2021) (MARCEL, 2020).

Recentemente a *Special Interest Group* (SIG) desenvolveu uma nova versão do *Bluetooth*, o *Bluetooth Low Energy* (BLE) ou *Bluetooth* 4.0. Seu principal diferencial é o consumo de energia, cerca de 10% menos que o *Bluetooth* clássico.

O protocolo *Bluetooth* 4.0 inclui o padrão *Low Energy*, que possibilita baixo consumo de energia e taxa de transmissão. A inclusão do modo de transmissão *broadcast* permitiu o desenvolvimento de pequenos dispositivos que operam com baterias pequenas, possibilitando a eles vida útil de meses ou até mesmo anos (VANZIN; OYAMADA, 2021).

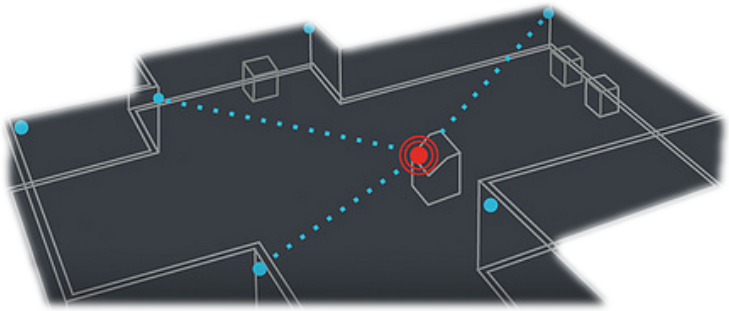
O modo de *broadcast* na tecnologia BLE é utilizado para transmitir dados de *broadcast*,

incluindo identificadores que podem ser captados por alguns aparelhos, como os *smartphones*, *smartwatches*, *tablets* ou outro dispositivo que seja compatível com a tecnologia BLE e que esteja dentro da área de cobertura. Neste modo não há a necessidade de estabelecer conexão o que resulta em rápida interação e baixo custo de energia (VANZIN; OYAMADA, 2021).

Desde o desenvolvimento do *Bluetooth Low Energy* e o crescimento no número de dispositivos que utilizam BLE, sistemas *Bluetooth* de localização interna ganharam bastante atenção. Em um ambiente interno o dispositivo com BLE monitora o posicionamento de outros dispositivos, como celulares ou *tablets*, por meio do *Received Signal Strength Indicator* (RSSI). Um exemplo deste tipo dispositivo são os *Bluetooth* beacons, dispositivos geralmente pequenos que transmitem constantes sinais de rádio com pequenas quantidades de dados por *Bluetooth* (HOU; ARSLAN, 2017) (AKPINAR, 2021).

A Figura 1 apresenta como ocorre a determinação de um objeto no espaço, em que os pontos em azul representam os *beacons*, o ponto vermelho é o alvo que se deseja descobrir a posição e os pontilhados representam a distância entre o *beacon* e o alvo . Um exemplo de uso, é guiar pessoas com deficiência visual em uma galeria de artes, por meio dos *beacons*, e quando ela chegasse perto de uma obra um aplicativo previamente instalado no *smartphone* do usuário detectaria a posição no ambiente e descreveria a obra que usuário está próximo.

Figura 1 – Localização do alvo no ambiente



O RSSI representa a qualidade relativa do sinal recebido em um dispositivo, ele indica o nível de potência que está sendo recebido. Quanto mais positivo o valor RSSI mais forte e melhor é o sinal, como mostra a Figura 2 (NetSpot, 2022).

1.1 Objetivos

1.1.1 Objetivo Geral

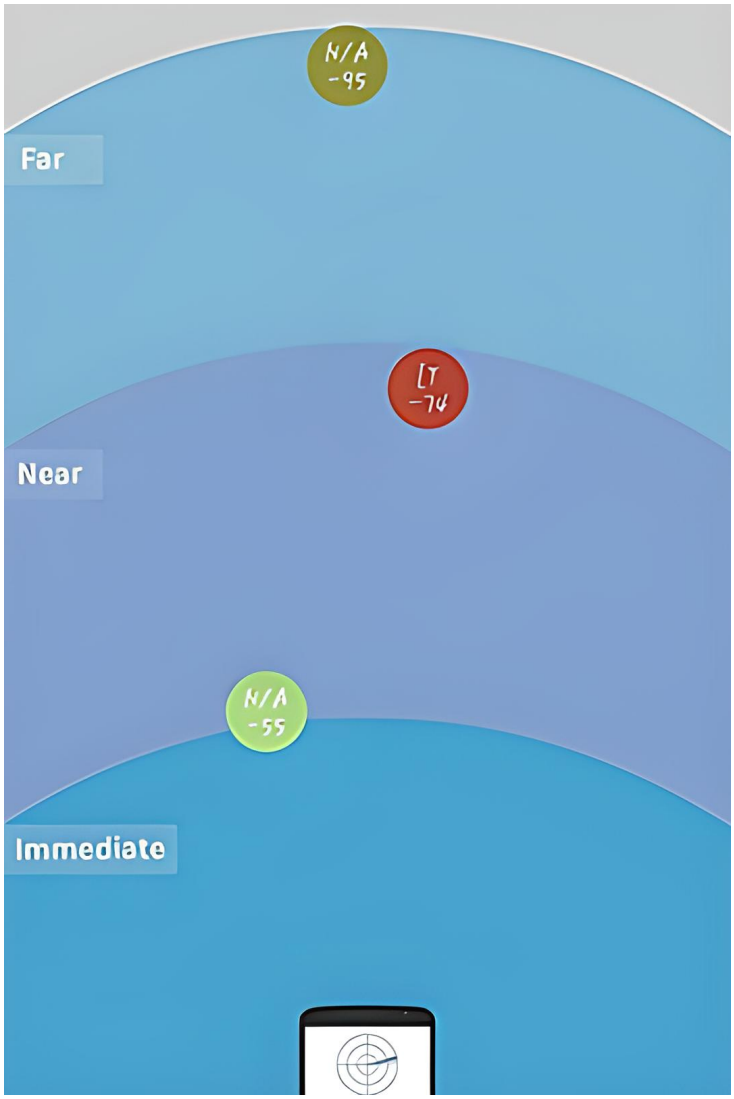
O objetivo deste trabalho é determinar localização interna utilizando *Bluetooth beacons*.

1.1.2 Objetivos Específicos

- Desenvolver uma aplicação para coleta de RSSI;

- Analisar a variação dos valores de RSSI medidos a uma distância fixa;
- Avaliar o modelo *Log Distance Path Loss Model* para determinar distância;
- Avaliar o uso de redes neurais para determinar distância e posicionamento *indoor*;
- Avaliar o uso da técnica de trilateração para determinar a posição de um objeto no espaço.

Figura 2 – Tela de *scanner* do aplicativo BLEScanner



Fonte: (Bluepixel Technologies, 2022)

1.2 Organização do texto

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta o conceito de *beacons*, como determinar distância por meio da equação *Log Distance Path Loss Model*, como determinar a localização interna por meio do método de trilateração, conceito de rede neural e

trabalhos relacionados. O Capítulo 3 abarca o delineamento metodológico, o Capítulo 4 discussão dos resultados obtidos e, por fim, o Capítulo 5 com a conclusão e trabalhos futuros.

2

Referencial Bibliográfico

Este capítulo aborda o conteúdo base necessário para os experimentos, sendo eles *Bluetooth Beacons* e os métodos utilizados, *Log Distance Path Loss Model*, trilateração e rede neural.

2.1 *Bluetooth Beacons*

Os *beacons* (Figura 3) são pequenos dispositivos que utilizam *Bluetooth Low Energy* e transmitem constantes sinais de rádio, com pequenas quantidades de dados. Estes sinais tem alcance de 60 a 100 metros de distância, dependendo do ambiente, e seus dados podem ser recebidos por um outro dispositivo, *smartphone* ou *tablet*, para exercer uma determinada ação. Seu valor está entre 3-60 dólares por *beacon*, podendo variar dependendo da funcionalidade e do tamanho, como mostra a Figura 4 (AKPINAR, 2021).

Figura 3 – *Bluetooth Beacon*

Fonte: (DANOVA, 2014)

Figura 4 – Tipos de *Bluetooth Beacon*

Fonte: (NICK, 2015)

Existem vários protocolos de *Bluetooth beacons*, entre eles os mais populares são o *iBeacon* (APPLE, 2022), desenvolvido pela Apple, e o *Eddystone* (GOOGLE, 2018), desenvolvido pela Google. Neste trabalho será utilizado o *iBeacon*.

O pacote de dados do *iBeacon* é composto por *Universally Unique Identifier* (UUID) com tamanho de 16 *bytes* e é utilizado para diferenciar os *beacons* e representar um determinado grupo ou empresa, os parâmetros *Major* e *Minor*, identificadores exclusivos, também são utilizados para diferenciar cada *beacon*, mesmo que eles possuam o mesmo UUID, e o RSS_{d_0}, que é o valor do RSSI para a distância de 1 metro do *beacon*, como mostra a Figura 5 (RECK, 2016) (KIM; LEE, 2014).

O *Bluetooth beacon* utiliza o modo de *broadcast*, que consiste na transmissão de dados sem a necessidade de realizar o pareamento, e com isso, evita tempo e energia gastos pelo pareamento. Este modo envia pacotes, em intervalos curtos, o tempo todo, independente se há

ou não um dispositivo por perto para utilizá-los. Mas como os pacotes de dados enviados são pequenos o consumo de energia é muito baixo.

Figura 5 – Pacote de dados do BLE *beacon*

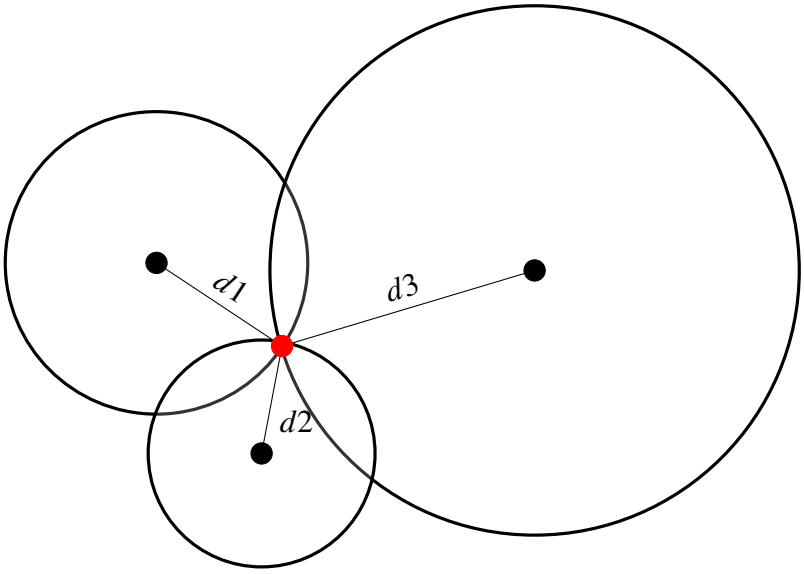
9 bytes	16 bytes	2 bytes	2 bytes	2 bytes
Prefixo	UUID	Major	Minor	RSSd0

Fonte: Adaptado de Reck (2016)

2.2 Trilateração

A trilateração é um método tradicional utilizado para identificar a posição de um nó. São necessários no mínimo outros três nós com posição conhecida e a distância entre o nó alvo e os nós conhecidos. Um exemplo de dispositivo que pode ser usado como nó são os *Bluetooth beacons*, com eles é possível localizar o *smartphone* ou *tablet* no ambiente, por meio do RSSI. Para encontrar a localização do alvo, cada nó conhecido possui um círculo ao seu redor com o raio igual a distância entre ele e o nó alvo. A localização do nó alvo corresponde a linha de interseção dos três círculos, como mostra na Figura 6 (PAKANON; CHAMCHOY; SUPANAKOON, 2020).

Figura 6 – Aproximação da localização do alvo no método da trilateração



As Equações 1 - 9, mostram como calcular as variáveis intermediárias A_1, A_2, A_3, B_1, B_2, B_3, C_1, C_2 e C_3, que são utilizadas na Equação 10. As coordenadas (x_1, y_1), (x_2, y_2), (x_3, y_3) são, respectivamente, a posição dos nós âncoras (*beacons*) 1, 2 e 3 no espaço.

$$A_1 = 2(x_2 - x_1) \tag{1}$$

$$A_2 = 2(x_3 - x_1) \tag{2}$$

$$A_3 = 2(x_3 - x_2) \tag{3}$$

$$B_1 = 2(y_2 - y_1) \tag{4}$$

$$B_2 = 2(y_3 - y_1) \tag{5}$$

$$B_3 = 2(y_3 - y_2) \tag{6}$$

$$C_1 = d_1^2 - d_2^2 + x_2^2 - x_1^2 + y_2^2 - y_1^2 \tag{7}$$

$$C_2 = d_1^2 - d_3^2 + x_3^2 - x_1^2 + y_3^2 - y_1^2 \tag{8}$$

$$C_3 = d_2^2 - d_3^2 + x_3^2 - x_2^2 + y_3^2 - y_2^2 \tag{9}$$

O cálculo das coordenadas do nó alvo é realizado utilizando a fórmula da trilateração, mostrada na Equação 10.

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} A_1^2 + A_2^2 + A_3^2 & A_1B_1 + A_2B_2 + A_3B_3 \\ A_1B_1 + A_2B_2 + A_3B_3 & B_1^2 + B_2^2 + B_3^2 \end{bmatrix}^{-1} \times \begin{bmatrix} A_1C_1 + A_2C_2 + A_3C_3 \\ B_1C_1 + B_2C_2 + B_3C_3 \end{bmatrix} \tag{10}$$

2.3 *Log Distance Path Loss Model*

Primeiramente, antes de determinar a posição do nó utilizando o método de trilateração é necessário determinar a distância dos *beacons* até o *smartphone*. Este cálculo é baseado nos valores de RSSI. Os valores de RSSI sofrem variação conforme a distância, ou seja, quanto maior a distância menor é o valor do RSSI. Devido ao RSSI ser afetado pelos fenômenos do ambiente é necessário a utilização de um modelo que considere esta interferência, como o *Log Distance Path Loss Model*, apresentado na Equação 11, um modelo bem conhecido para transformar valores de RSSI em distância (RÖBESAAT et al., 2017).

$$RSSI = RSS_{d_0} - 10n \times log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{11}$$

O parâmetro RSS_{d_0} representa o valor de RSSI na distância d_0. O parâmetro n representa o expoente *Path Loss* que indica a taxa de crescimento do *path loss* relacionado a distância. A variável d é a distância verdadeira e o X_σ é a variável média randômica com desvio padrão,

usada quando há grandes obstáculos no ambiente de teste para calcular o sombreamento, quando o experimento é realizado em um espaço aberto a variável pode ser considerada 0 (RÖBESAAT et al., 2017).

O expoente *Path Loss n* varia conforme as condições do ambiente. A Tabela 1 apresenta alguns valores de *n* referente ao ambiente.

Tabela 1 – Expoente *Path Loss* para diferentes ambientes

Ambiente	**Expoente *Path Loss n***
Espaço Aberto	2
Área Urbana - Sinal de Celular	2.7 - 3.5
Área Urbana com Sombreamento - Sinal de Celular	3 - 5
Linha Visada em Prédios	1.6 - 1.8
Obstrução em Prédios	4 - 6
Obstrução em Fábricas	2 -3

Fonte: Adaptado de Röbesaat et al. (2017)

No entanto, estes valores padrões podem não ser ideais para todos os casos, com isso, é possível calcular o *n* específico para cada ambiente ou dispositivo utilizando a Equação 12.

$$n = \left(\frac{RSS_{d_0} - RSSI}{10 \times log_{10}(d)} \right) \tag{12}$$

A equação *Log Distance Path Loss Model* também pode ser utilizada para determinar a distância quando se tem o RSSI, conforme apresentado na Equação 13.

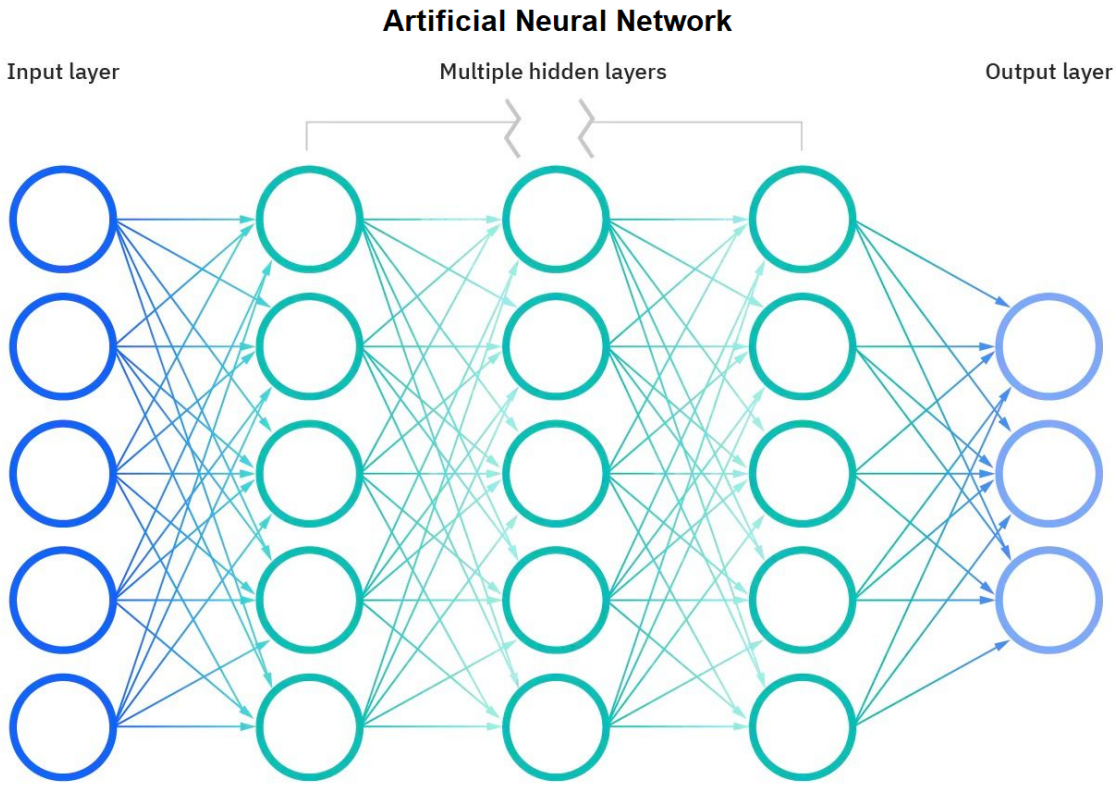
$$d = 10^{\left(\frac{RSS_{d_0} - RSSI}{10 \times n}\right)} \tag{13}$$

2.4 Rede Neural Artificial

As Redes Neurais Artificiais (RNA) são inspiradas no compartamento do cérebro humano, imitando a maneira como os neurônios biológicos enviam sinais uns para os outros, permitindo assim, que programas de computador identifiquem padrões (IBM Cloud Education, 2020). As redes neurais artificiais são compostas por camadas de nós, contendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, como mostra a Figura 7. Cada nó, ou neurônio artificial, conecta-se a outro e possuem um peso e um limite associado.

Se a saída de qualquer nó individual estiver acima do valor do limite especificado, esse nó será ativado, enviando dados para a próxima camada da rede. Caso contrário, nenhum dado será transmitido para a próxima camada da rede. Nas redes neurais existem dois tipos de aprendizagem, o não supervisionado que consiste em conjunto de dados não rotulados, ou seja, não necessitam da interferência humana e o supervisionado que utiliza dados rotulados para treinar a rede neural

Figura 7 – Rede Neural



Fonte: (IBM Cloud Education, 2020)

a fim de que ela aprenda e melhore a sua precisão ao longo do tempo (DELUA, 2021). Portanto, uma vez que esses algoritmos de aprendizagem são refinados para precisão, eles são ferramentas potentes na Ciência da Computação e na inteligência artificial, permitindo classificar e agrupar dados a uma alta velocidade.

As rede neurais são geralmente utilizadas para classificação, que consiste em classificar a entrada dos dados em uma das classes especificadas. No entanto, elas também podem ser treinadas para regressão, que consiste em prever o valor de uma variável (variável dependente) com base no valor de outra variável (variável independente). Essa forma de análise estima os coeficientes da equação não-linear, envolvendo uma ou mais variáveis independentes que melhor predizem o valor da variável dependente. A regressão linear ajusta-se a uma linha reta ou superfície que minimiza as discrepâncias entre os valores de saída previstos e os valores reais. Ao criar uma rede neural de regressão alguns hiperparâmetros são recomendados de serem usados, como o *Mean Square Error* (MSE) (BROWNLEE, 2019) que é geralmente utilizado como função de perda.

2.5 Trabalhos Relacionados

No artigo de Röbesaat et al. (2017), os autores realizaram experimentos para coleta e análise dos valores de RSSI. Esta coleta foi realizada com a equação *Log distance Path Loss Model*, Equação 11, o valor de RSSI para a distância de 1 metro (RSS_{d_0}) e o valor de n foi a média entre os *beacons* utilizados. Além disso, realizaram o cálculo da trilateração utilizando 3 nós (*beacons*) para identificar a localização do nó representado pelo *smartphone*. Após os testes, os autores notaram que houve taxa de erro ao determinar a localização do alvo devido as variações do RSSI que são causadas por interferências, como a direção da antena e objetos que se encontram no meio do caminho da transmissão, como o corpo humano, por este motivo, aplicaram 3 filtros, filtro médio, da mediana e de Kalman, para obter melhores resultados. O método de filtragem de Kalman foi o que mostrou menores erros, obtendo uma variação mais sutil do RSSI. Por fim, concluíram que ao utilizar distâncias menores ou igual a 4 metros os valores do RSSI são mais precisos e que a variação do RSSI aumenta conforme a distância.

No artigo de Puckdeevongs (2021) o autor utilizou uma RNA para calcular a posição do alvo dentro de um laboratório. A tecnologia sem fio utilizada foi o Wifi e uma placa ESP-8266, com *script* em Python para a coleta dos valores de RSSI, como alvo. Na RNA o método utilizado foi *Feed Forward Multilayer Perceptron* (FF-MLPs) em uma rede neural 4-4-2, que possui 4 neurônios de entradas com valores do RSSI referente a cada um dos 4 pontos Wifi utilizados, 1 *hidden layer* com 4 neurônios e 2 de saída com as coordenadas do alvo. O autor realizou dois processos, treinamento e teste, o treinamento consiste na criação do conjunto do dados, que foi coletar os valores de RSSI em uma posição específica em vários locais do ambiente e o treinamento da rede neural, e a fase de teste foi avaliar a eficiência da rede neural em determinar a posição do alvo baseando-se no conjunto de dados criado. A determinação da posição do alvo não foi muito precisa, devido as variações dos valores de RSSI causada pelas interferências do ambiente. Na estimativa da localização do alvo 20.93% dos erros foram de 0.5 metro e 34.88% foram entre 0.5 a 1 metro.

No artigo de Tuncer e Tuncer (2015) os autores compararam a determinação da localização de um *smartphone Android* usando RNA e o método *Centroid Localization* (CL). Foram utilizados 4 placas *Bluetooth* HC-06 conectadas ao sensor MSP430 *controller* e foram posicionadas nos quatro cantos do espaço que foi realizado o experimento. A estrutura da RNA contém 3 camadas, 6 neurônios para entrada com os valores de RSSI e seus respectivos identificadores, uma *hidden layer* e 2 neurônios de saída contendo as coordenadas do alvo. Durante o treinamento foi utilizado as funções *hyperbolic tangent sigmoid* e *linear transfer*, o algoritmo de *back-propagation* para a alteração dos pesos e o *error value* e o *learning ratio* com 0.01 e 0.005, respectivamente. Na entrada foi utilizado o RSSI dos sensores que tiveram maior valor. Durante a etapa de localização, após o treinamento da RNA, os autores obtiveram erro total de 33.26 m para a rede neural e 108.15 m para o algoritmo CL, ou seja, a rede neural foi 30% mais precisa que o método *Centroid Localization*. No final os autores realizaram testes e concluíram que o aumento do número de

camadas e neurônios na rede neural diminui os erros.

No artigo Bouchard et al. (2016) os autores utilizaram 4 *smartwatches* de 2 tipos e marcas diferentes, um tipo contendo *chip Bluetooth Broadcom* BCM4334 e outro *Broadcom* BCM43341 e 10 *Bluetooth beacons* omnidirecionais de mesmo modelo, porém de marcas diferentes, para analisar o comportamento destes dispositivos. O primeiro experimento consistiu em posicionar um *beacon* na parede e com um *smartwatch* coletar os valores de RSSI da distância de 1 a 7 metros, neste experimento os autores notaram que houve variação nos valores de RSSI coletados pelos *smartwatches* que possuíam a mesma placa *Bluetooth*, mesmo com eles estando posicionados nos mesmos lugares. Outro experimento foi o teste do ângulo de chegada do sinal, primeiro foi posicionado um dos *smartwatch* no meio de um círculo com diâmetro de 1m e 2m, e posicionou 8 *beacons* nos limites deste círculo. Verificou-se com base no valor médio do RSSI de cada *beacons*, que os *beacons* que estavam na parte de trás do *smartwatch* tinham sinal mais forte do que os que estavam na parte da frente (tela). Os 2 *beacons* que obtiveram melhor sinal estavam posicionados perpendicularmente a antena *Bluetooth* do relógio, localizada no canto inferior esquerdo do aparelho. Na segunda parte do experimento o *smartwatch* foi colocado em um braço humano e foi posicionado dentro de uma caixa contendo 8 *beacons* espalhados, neste experimento verificou-se que o *beacon* que estava embaixo do braço se tornou o segundo *beacon* com pior sinal, resultado diferente do obtido na primeira parte do experimento, com isso, foi possível verificar a significante interferência que o corpo humano exerce. Por fim, concluíram que o ângulo de chegada do sinal possui maior interferência, cerca de 13dBm, em comparação a distância, cerca de 8dBm para a distância de 7m, e que as diferentes marcas dos dispositivos apresentaram variação nos valores de RSSI, com variação média de 5.02dBm para os *smartwatches* e 2.8dBm para os *beacons*.

No artigo de Fachri e Khumaidi (2019) é analisado a eficiência dos *Bluetooth beacons* na microlocalização. Para o experimento utilizou-se 5 *beacons* comerciais, cada um de uma marca diferente, que foram configurados para funcionarem com os padrões de fábrica, ou seja, sem nenhuma calibração. Cada um dos 23 *smartphones* utilizados foi posicionado a distância de 1, 5 e 10 metros de cada *beacon* e coletaram valores de RSSI durante 2 minutos em cada distância. Para converter RSSI em distância os autores utilizaram a Equação 13. A imprecisão da distância variou entre os *beacons* e a distância utilizada. Nem todos os *beacons* foram precisos em todas as distâncias, onde alguns apresentaram maior precisão em distâncias menores e maior imprecisão em distâncias maiores. Erros de até 5.50 metros foram encontrados na distância de 10 metros.

3
Metodologia

Para determinar a localização interna com *Bluetooth beacons*, foram utilizados 3 placas HM-10 para serem as âncoras, uma trena de 10 metros, uma aplicação python desenvolvida para coletar os valores de RSSI, uma placa Arduíno e 3 baterias de lítio para alimentar os *beacons*.

Primeiramente as placas HM-10 (Jinan Huamao technology Co., LTD., 2022) foram configuradas para funcionarem como *beacons* utilizando os comandos apresentados na Tabela 2. Estas configurações ficam salvas no dispositivo, logo não é necessário refazer os comandos toda vez que for utilizá-los.

Tabela 2 – Comandos para programar o HM-10

Comandos	**Descrição**
AT+RENEW	Restaura para configuração de fábrica
AT+RESET	Redefine o dispositivo
AT	Testa a conexão
AT+MARJ0x2000	Define o número *Major* do *Beacon*
AT+MINO0x0010	Define o número *Minor* do *Beacon*
AT+ADVI5	Define o intervalo de envio do sinal do *beacon* 5 = 546,25 ms
AT+NAMEiBeacon1	Define o nome do módulo
AT+ADTY3	3 = Apenas modo *beacon*
AT+IBEA1	Modo *Beacons*, 1 = ativa o *beacon*
AT+DELO2	Modo de implantação do *Beacon*, 2 = somente *broadcasting*
AT+RESET	Redefine o dispositivo

Fonte: Adaptado de Vanzin e Oyamada (2021)

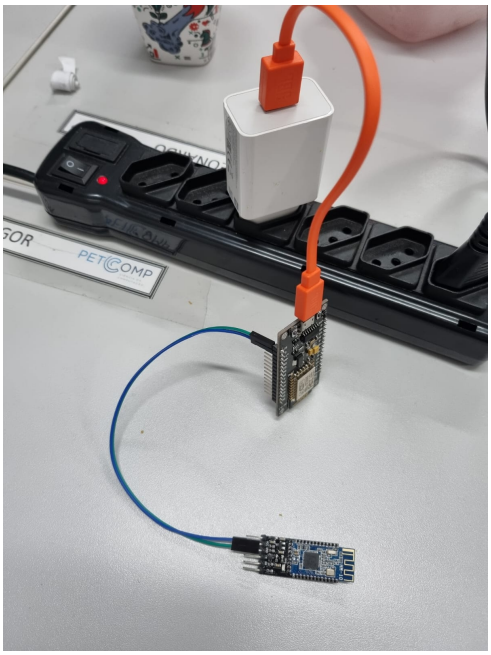
O trabalho foi dividido em 3 etapas e 2 experimentos. A primeira etapa foi a calibração da equação *Log Distance Path Loss Model*, a segunda etapa a criação de conjuntos de dados de distância e desenvolvimento de redes neurais para calcular distância e a terceira etapa a criação do conjunto de dados de coordenadas e o desenvolvimento de uma rede neural para determinar posição no espaço. O primeiro experimento foi a comparação da equação *Log Distance*

Path Loss Model com rede neural para calcular a distância entre o *beacon* e o *notebook* e o segundo experimento foi comparar o método de trilateração utilizando distâncias determinadas pela equação *Log Distance Path Loss Model*, o método de trilateração utilizando as distâncias determinadas pela rede neural e a rede neural, para determinar o posicionamento do alvo. Todos os códigos utilizados se encontram no GitHub (VANZIN, 2022).

3.1 Primeira Etapa

A primeira etapa foi a calibração do *Log Distance Path Loss Model*. Para isso foi conectado uma das placas HM-10 a uma placa Arduíno, que estava conectado a tomada (Figura 8), e foi posicionada no espaço para ser feita a coleta de valores de RSSI em uma linha reta com a aplicação em Python desenvolvida, utilizando a biblioteca *Bleak* (BLIDH et al., 2020). Foram coletados 24 valores de RSSI a cada 0.5 metro até a distância de 6 metros. Isto foi feito também com as outras duas placas. Os valores coletados foram planilhados e ordenados para calcular a média, descartando os dois maiores e dois menores valores em cada distância. Esta média foi utilizada para calcular os valores de n específicos para cada placa, utilizando a fórmula Equação 12, onde o valor do RSS_{d_0} é o valor do RSSI da distância de 1 metro, de cada placa.

Figura 8 – HM-10 conectado na tomada



3.2 Segunda Etapa

A segunda etapa consistiu em criar um conjunto de dados de valores de RSSI, com base nas distâncias, para cada *beacon* e desenvolver uma rede neural de regressão, que determina distância, também para cada *beacon*. Para a criação dos conjuntos de dados, foram coletados

1000 valores de RSSI por distância, de 0.5 a 6 metros, com cada placa. Os conjuntos de dados foram divididos aleatoriamente em 80% para treinamento e 20% para teste, utilizando a função *random_split* do *pytorch.utils.data* (PYTORCH, 2022b). As redes neurais de regressão, foram desenvolvidas utilizando a biblioteca Pytorch (PYTORCH, 2022a).

Durante o desenvolvimento das redes neurais foram pré-definidos alguns hiperparâmetros como otimizador Adam, MSE (*Mean Square Error*) como função de perda, *batch size* igual a 32 e número de épocas igual a 1000. O restante dos hiperparâmetros foram determinados por meio de duas buscas, para obter os hiperparâmetros que apresentam melhor desempenho. A primeira busca foi feita com os valores presentes na Tabela 3. Os resultados dessa busca foram salvos em um csv juntamente com os erros MSE (FROST, 2022) e MAE (*Mean Absolute Error*) (STEPHANIE, 2016), estes resultados foram ordenados pelo erro MSE e analisado os 10 resultados que apresentaram menor MSE. Os dados utilizados na segunda busca foram baseados nos hiperparametros que mais apareceram nestes 10 resultados.

Tabela 3 – Hiperparâmetros utilizados na primeira busca - Distância

Placa	**1º *Hidden Layer***	**2º *Hidden Layer***	***Activation Function***	***Learning Rate***
90:E2:02:91:62:09	10,30,50,70,90	10,30,50,70,90	Tahn, ReLu, Sigmoid	0.1, 0.01, 0.001, 0.0001
3C:A3:08:96:68:60	10,30,50,70,90	10,30,50,70,90	Tahn, ReLu, Sigmoid	0.1, 0.01, 0.001, 0.0001
4C:24:98:5D:05:A3	10,30,50,70,90	10,30,50,70,90	Tahn, ReLu, Sigmoid	0.1, 0.01, 0.001, 0.0001

3.3 Terceira Etapa

A terceira etapa foi criar um conjunto de dados com valores de RSSI com base nas coordenadas e desenvolver uma rede neural de regressão para determinar coordenadas x e y. Esta etapa foi realizada na sala 15 do Bloco F, do Curso de Ciência da Computação. Para a criação do conjunto de dados foi coletado 1000 valores de RSSI em pontos específicos da sala, sendo estes pontos considerados como pontos x e y em um espaço 2D. Os pontos de coordenada foram determinados conforme as lajotas do chão da sala, ou seja cada lajota foi considerada como um ponto de coordenada x e y. Nesta etapa não foi considerado o tamanho total da sala, devido ao seu formato e objetos grandes presentes nela, com isso, foram utilizados 13 coordenadas (lajotas) em y e 12 em x.

Na fase da coleta cada uma das placas foi conectada a uma bateria de lítio, como mostra a Figura 9, em seguida elas foram posicionadas em lugares estratégicos na sala. A Figura 10 mostra o espaço de coleta, na qual os pontos em azul são as coordenadas onde os *beacons* estavam posicionados e os pontos pretos são onde o *notebook* foi posicionado para coletar os valores de RSSI dos 3 *beacons* ao mesmo tempo.

O desenvolvimento da rede neural de regressão também foi feito com a biblioteca Pytorch. Nela foram utilizados os mesmos hiperparâmetros pré-definidos da segunda etapa, otimizador, função de perda, *batch size* e número de épocas. Os demais hiperparâmetros foram definidos por

Figura 9 – HM-10 conectado a bateria de lítio

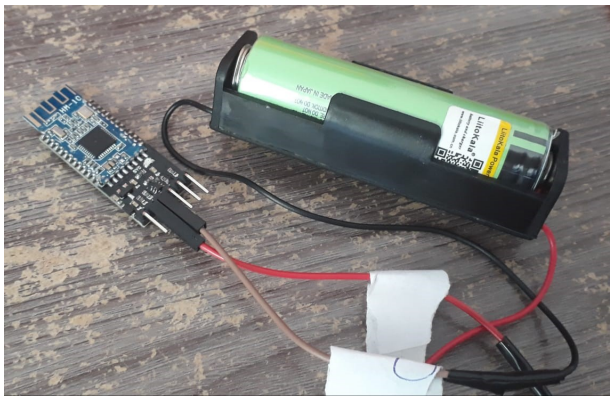
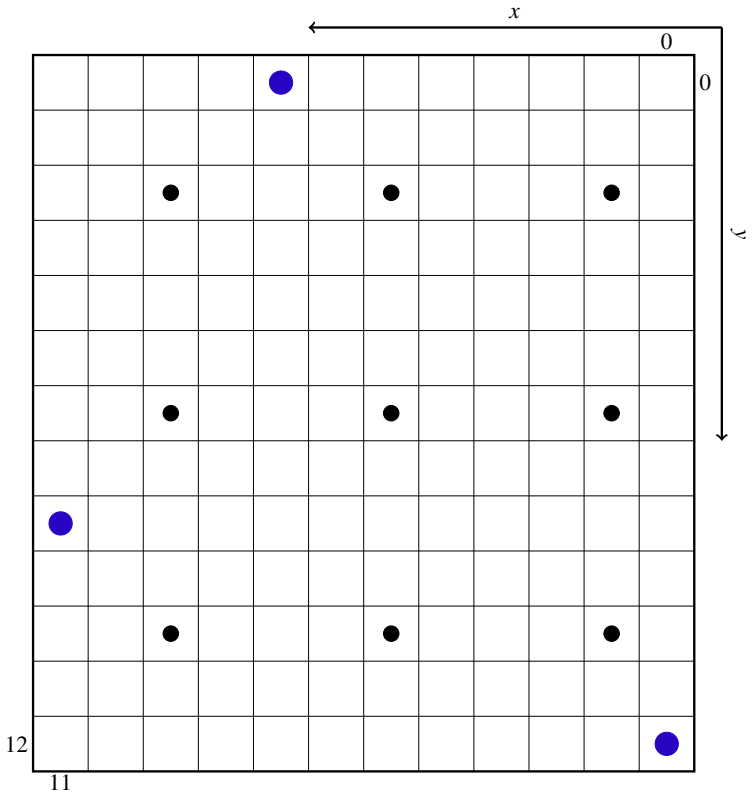


Figura 10 – Pontos de coleta



meio de duas buscas. A primeira busca foi feita com os valores apresentados na Tabela 4, em que os resultados foram salvos em um csv e foram ordenados pelo MSE. Os 10 melhores resultados foram utilizados para identificar padrões que foram usados para fazer a segunda busca.

Tabela 4 – Hiperparâmetros utilizados na primeira busca - Coordenadas

1º *Hidden Layer*	**2º *Hidden Layer***	***Activation Function***	***Learning Rate***
10,30,50,70,90,110	10,30,50,70,90,110	Tahn, ReLu, Sigmoid	0.1, 0.03, 0.01, 0.003, 0.001, 0.0001

3.4 Primeiro Experimento

O primeiro experimento foi analisar o desempenho da equação *Log Distance Path Loss Model* e da rede neural para determinar distância. Foi utilizado os conjuntos de dados criados na segunda etapa para a realização dos cálculos. Cada rede neural desenvolvida na etapa 2 foi treinada com 80% do seu conjunto de dados correspondente, por exemplo, a rede neural do *beacon* 1 foi treinada com o conjunto de dados que contém os valores de RSSI coletados dele. As distâncias foram calculadas com 100% dos conjuntos de dados.

Como os conjuntos de dados utilizados possuem 10 valores de entrada por linha, como mostra a Figura 11, e o método *Log Distance Path Loss Model* utiliza apenas um dado por vez, foi feita a média desses 10 valores de cada linha para serem aplicados na Equação 13 para calcular as distâncias.

Ao utilizar rede neural ela é geralmente treinada com uma parte do conjunto de dados e a outra parte é utilizada para testar a sua precisão. No entanto, como é feita a comparação da rede neural com o *Log Distance Path Loss Model* e não é possível saber os valores que estarão no conjunto de testes, foi utilizado 100% dos conjuntos de dados para poder, posteriormente, compará-los estatisticamente.

Figura 11 – Conjunto de dados - Distância

```
1   -43,-48,-48,-48,-49,-45,-48,-49,-51,-49
2   -47,-48,-47,-48,-49,-50,-46,-48,-46,-49
3   -47,-49,-48,-48,-48,-52,-48,-48,-46,-50
4   -46,-48,-48,-47,-48,-48,-48,-49,-49,-48
5   -45,-49,-47,-45,-48,-45,-47,-49,-50,-49
6   -46,-49,-49,-48,-48,-51,-48,-48,-48,-46
7   -47,-49,-49,-45,-48,-46,-50,-48,-47,-46
8   -47,-49,-49,-48,-48,-49,-48,-48,-50,-45
9   -47,-50,-48,-45,-48,-48,-50,-49,-49,-48
10  -46,-48,-46,-48,-48,-48,-47,-50,-48,-48
11  -45,-50,-46,-48,-48,-48,-48,-48,-52,-48
12  -45,-50,-49,-48,-45,-49,-44,-49,-48,-47
```

3.5 Segundo Experimento

O segundo experimento consistiu em analisar a rede neural e o método de trilateração, utilizando as distâncias calculadas pela rede neural e a equação *Log Distance Path Loss Model*, para determinar coordenadas no espaço. Foi utilizado o conjunto de dados criado na terceira etapa. A rede neural foi treinada com 80% do conjunto de dados e as coordenadas foram calculadas com 100% dele. No método de trilateração as distâncias foram calculadas utilizando os métodos

do primeiro experimento, rede neural e *Log Path*. Estas distâncias foram utilizadas nas Equações 1 - 9, que posteriormente foram utilizadas na Equação 10.

Durante o cálculo das coordenadas com o método de trilateração foi preciso transformar as distâncias calculadas em lajotas, para que estejam na mesma unidade de medida. Para isso foi feita uma regra de 3 para transformar distância em lajotas, utilizando a Equação 14, onde l é o número de lajotas referente a distância e cada lajota mede 0.45 metros.

$$l = \frac{distancia}{0.45} \tag{14}$$

Para analisar o desempenho de cada método, foram calculados o erro máximo, mínimo, MSE e MAE em cada experimento.

4
Resultados

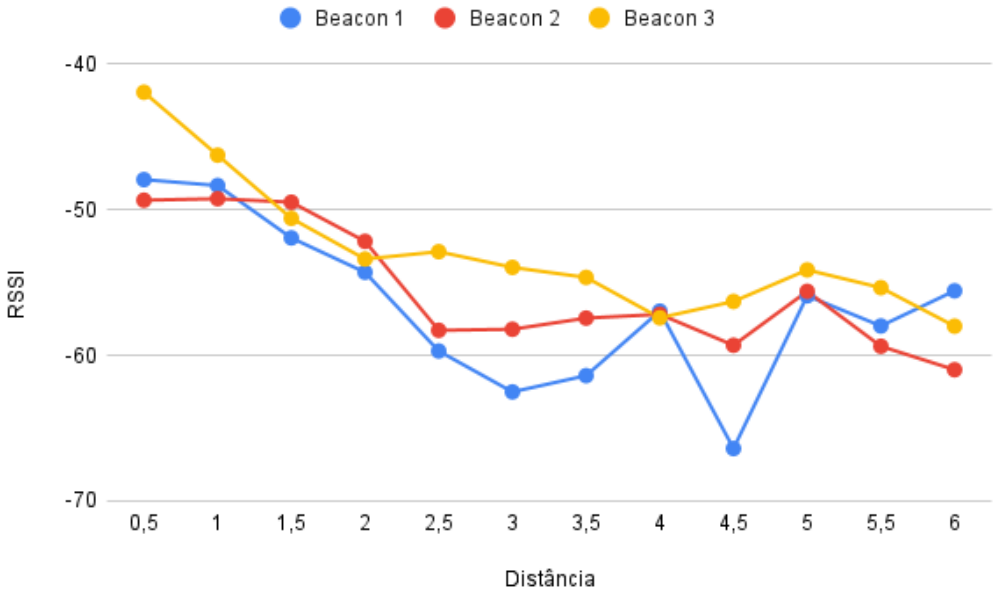
O capítulo de resultados foi divido em 4 seções. A primeira seção aborda os problemas identificados durante as coletas dos valores de RSSI; a segunda seção apresenta as configurações das redes neurais obtidas com as buscas; a terceira seção apresenta os resultados obtidos utilizando *Log Distance Path Loss Model* e rede neural para determinar distância; e a quarta seção apresenta os resultados obtidos ao determinar as posições do alvo com os métodos trilateração e rede neural, trilateração e *Log Distance* e rede neural.

4.1 Coleta dos valores de RSSI

Durante a coleta dos valores de RSSI para a criação dos conjuntos de dados foi notado que eles são muito instáveis, pois os valores variavam mesmo estando com o *notebook* completamente parado. Além disso, foi observado que o sinal emitido pelo *beacon* sofre muita interferência do ambiente, tanto de objetos presentes no local quanto sinais de outros dispositivos *bluetooth*, isso foi observado quando foram feitas coletas em dias diferentes. Notou-se que quando as coletas foram realizadas com várias pessoas e dispositivos no local, os dados obtidos apresentaram maior variação e quando a ocupação do local era baixa e o número de dispositivos sem fio era praticamente nulo, os valores coletados eram bem mais estáveis.

Um detalhe percebido durante as coletas da segunda etapa foi que os valores de RSSI variam conforme a distância, o que foi observado por Röbesaat et al. (2017) e Vanzin e Oyamada (2021). A Figura 12 mostra esta variação, em que quando o *notebook* estava mais perto do *beacon* os valores tinham pouca variação e quando a distância entres eles aumentava os valores de RSSI se tornavam mais inconsistentes. No entanto, como já mencionado essa instabilidade também é causada pelo ambiente, como por exemplo a distância 4 foi um dos poucos pontos durante a coleta que não havia objetos muito próximos do *notebook* e, consequentemente, foi uma distância onde os valores de RSSI sofreram pouca variação.

Figura 12 – Variações dos valores de RSSI conforme a distância



A coleta realizada para a criação do conjunto de dados na terceira etapa reforçou que os *beacons* sofrem interferência de outros dispositivos *bluetooth*, pois nesta coleta os 3 *beacons* estavam ativos, ao mesmo tempo, e foi observado que os seus sinais *bluetooth* estavam interferindo uns nos outros. Outro detalhe notado foi que a direção da antena interfere bastante no sinal, o mesmo concluído no artigo de Bouchard et al. (2016), devido a isso, foi tentado utilizar a antena *bluetooth* do *notebook* apontada para os *beacons*, apesar de que nem sempre foi possível posicioná-la de forma que os *beacons* estivessem posicionados na parte da frente.

4.2 Configurações Redes Neurais

Os 10 melhores resultados de cada primeira busca dos hiperparâmetros das redes neurais de distância foram analisados para selecionar os valores que foram utilizados na segunda busca, apresentados na Tabela 5. O resultado final das buscas dos hiperparâmetros é apresentado no Quadro 1, que mostrou que cada rede neural ficou com configurações diferentes. Isso ocorreu devido as características únicas que cada *beacon* tem, e é por este motivo que na equação *Log Distance Path Loss Model* são calculados *n* específicos para cada um. A Figura 13 mostra um exemplo de como mais ou menos ficou a estrutura dessas redes neurais.

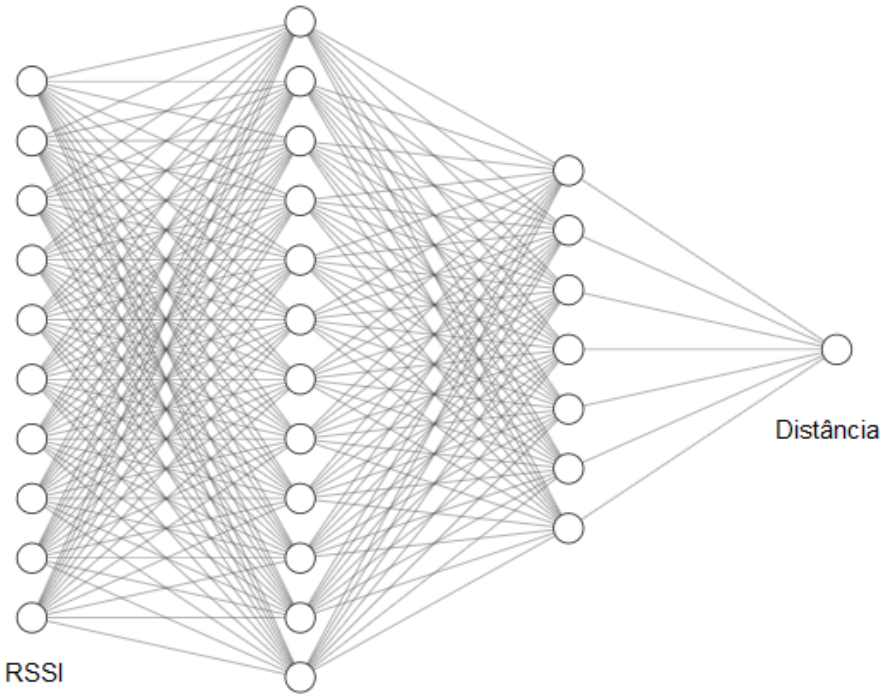
Tabela 5 – Hiperparâmetros utilizados na segunda busca - Distância

Placa	**1º *Hidden Layer***	**2º *Hidden Layer***	***Activation Function***	***Learning Rate***
90:E2:02:91:62:09	30,40,...,90,110,130	10,20,...,90,110,130	Tahn, Sigmoid	0.001
3C:A3:08:96:68:60	30,40,...,90,110,130	10,20,...,70	ReLu	0.1, 0.03, 0.01, 0.003, 0.001
4C:24:98:5D:05:A3	70,90,110,130,150	10,30,50,70,90,110	Sigmoid	0.001

Quadro 1 – Configuração final das redes neurais de distância

Beacon	1	2	3
Entrada	10	10	10
1º Camada	130	40	130
2º Camada	30	50	10
Saída	1	1	1
Função de Ativação	ReLu	ReLu	Sigmoid
Taxa de Aprendizagem	0,001	0,003	0,001
Otimizador	Adam	Adam	Adam
Função de Perda	MSE	MSE	MSE
Batch Size	32	32	32
Número de Épocas	1000	1000	1000

Figura 13 – Estrutura RNA - Distância

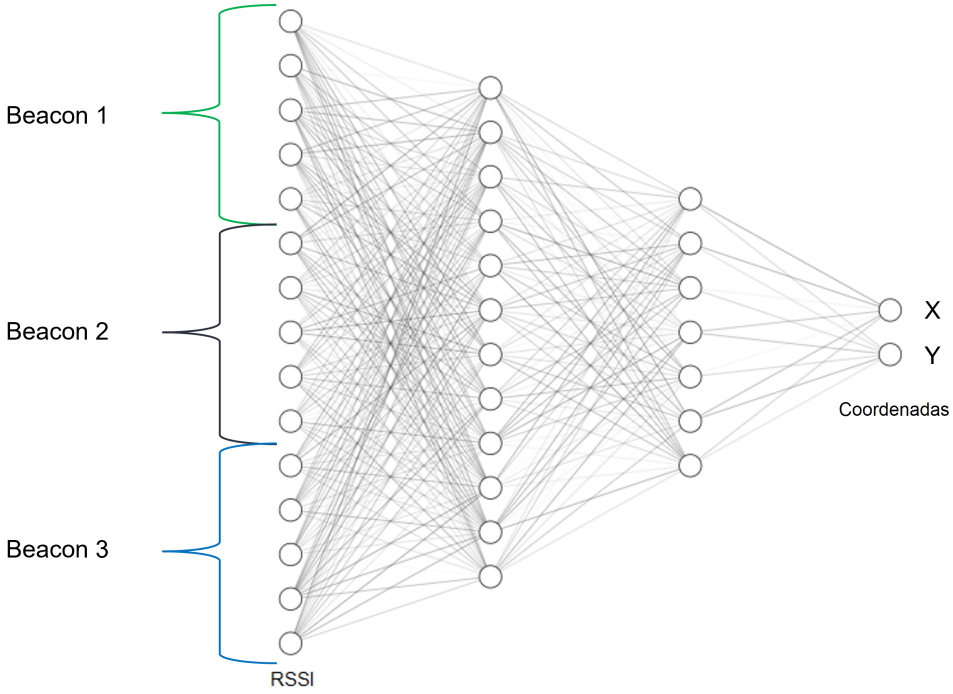


Para obter a rede neural para estimar a posição do alvo na sala, os valores utilizados na segunda busca dos hiperparâmetros são apresentados na Tabela 6. Para definir esses parâmetros foram analisados os 10 melhores resultados da primeira busca. O resultado final das buscas dos hiperparâmetros é apresentado no Quadro 2 e a Figura 14 mostra um exemplo de estrutura semelhante a da rede neural obtida.

Tabela 6 – Hiperparâmetros utilizados na segunda busca - Coordenadas

1º Hidden Layer	***2º Hidden Layer***	***Activation Function***	***Learning Rate***
50,60,70,80,90,100	50,60,70,80,90,100,110,120	ReLu, Sigmoid	0.003, 0.001

Figura 14 – Estrutura RNA - Coordenadas



Quadro 2 – Configuração final da rede neural de coordenada

Entrada	15
1º Camada	100
2º Camada	110
Saída	2
Função de Ativação	Sigmoid
Taxa de Aprendizagem	0,001
Otimizador	Adam
Função de Perda	MSE
Batch Size	32
Número de Épocas	1000

4.3 Primeiro Experimento

Os quadros 3, 4 e 5, apresentam os erros obtidos ao determinar distância com *Log Distance Path Loss Model* e rede neural. Neles é claramente perceptível que o método que apresentou melhor taxa de acerto foi a rede neural. Isso se deve por a rede neural conseguir identificar padrões, como por exemplo, em certos momentos são coletados valores nada relacionados com a distância atual e a rede neural atribui um peso menor a eles, o que faz com que o erro não propague para as demais camadas. Isto já não ocorre com a equação *Log Distance Path Loss Model*, mesmo também sendo uma equação não linear, pois ela não é tão flexível quanto a rede neural, no entanto, podem ser utilizados filtros para tornar os dados mais consistentes.

Para realização dos cálculos das distâncias utilizando a equação *Log Distance Path Loss Model* foi necessário calcular o valor de n (coeficiente *Path Loss*) para cada *beacon*. Ele foi

calculado na etapa de calibração (primeira etapa), obtendo os valores 1, 88, 2, 80 e 2, 48 para os *beacons* 1, 2 e 3, respectivamente.

Quadro 3 – Erros na determinação da distância em metros do *Beacon* 1

Método	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
Log Distance Path Loss Model	-69,627	357,738	4765,239	53,286
Rede Neural	-3,932	2,659	1,011	0,762

Quadro 4 – Erros na determinação da distância em metros do *Beacon* 2

Método	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
Log Distance Path Loss Model	-63,245	184,840	2914,055	41,461
Rede Neural	-2,573	2,374	0,866	0,701

Quadro 5 – Erros na determinação da distância em metros do *Beacon* 3

Método	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
Log Distance Path Loss Model	-66,672	43,175	1562,070	34,603
Rede Neural	-2,977	1,859	0,734	0,641

Nestes teste de verificação das redes neurais foi utilizado 100% do conjunto de dados, no entanto, como foi feito o treinamento delas com 80% do conjunto de dado geralmente é utilizado o restante do conjunto de dados, que não foi usado para o treinamento, para realizar a verificação. Além disso, a utilização de 100% do conjunto de dados pode acarretar em *overfit*. Devido a isso, também foi feita a verificação da eficiência das redes neurais com 20% do conjunto de dados restante. Os quadros 6, 7 e 8 mostram os erros obtidos com cada porcentagem e com eles é possível observar que não ocorreu *overfit* e também que a diferença de erro entre elas é muito pequena.

Quadro 6 – Erros com diferentes porcentagens do conjunto de dados de distância - *Beacon* 1

Conjunto de dados	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
100%	-3,932	2,659	1,011	0,762
20% (teste)	-3,886	2,776	1,910	1,071

Quadro 7 – Erros com diferentes porcentagens do conjunto de dados de distância - *Beacon* 2

Conjunto de dados	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
100%	-2,573	2,374	0,866	0,701
20% (teste)	-2,181	2,120	0,769	0,673

Quadro 8 – Erros com diferentes porcentagens do conjunto de dados de distância - *Beacon* 3

Conjunto de dados	**Erro Mínimo**	**Erro Máximo**	**MSE**	**MAE**
100%	-2,977	1,859	0,734	0,641
20% (teste)	-2,522	1,850	0,596	0,578

4.4 Segundo Experimento

O Quadro 9 apresenta os erros obtidos com cada método ao determinar a localização do *notebook* no local. Dentre os métodos utilizados, o que apresentou menores erros foi a rede neural. Isso ocorreu devido a capacidade de adaptação que a rede neural possui, o que é muito vantajoso ao utilizar com dados que possuem certa variância, como é o caso dos valores de RSSI.

Para realização dos testes também foi utilizado 100% do conjunto de dados, por este motivo foi realizada a mesma comparação do primeiro experimento, para verificar se a utilização dos 20% do conjunto de dados, não utilizados para treinamento, apresentaria melhores resultados. O Quadro 10 mostra a comparação entre as duas porcentagens utilizadas, nele é possível observar que os valores foram muito semelhantes e que não ocorreu *overfit*.

Quadro 9 – Erros em metros de cada método ao determinar a localização do *notebook*

Método	Erro Mínimo X	Erro Máximo X	Erro Mínimo Y	Erro Máximo Y	**MSE**	**MAE**
***Log Distance* / Trilateração**	-3,980	4,760	-4,300	4,200	8,379	2,591
Rede Neural / Trilateração	-2,830	4,820	-4,340	4,060	9,113	2,669
Rede Neural	-4,020	4,136	-6,574	4,280	0,503	0,347

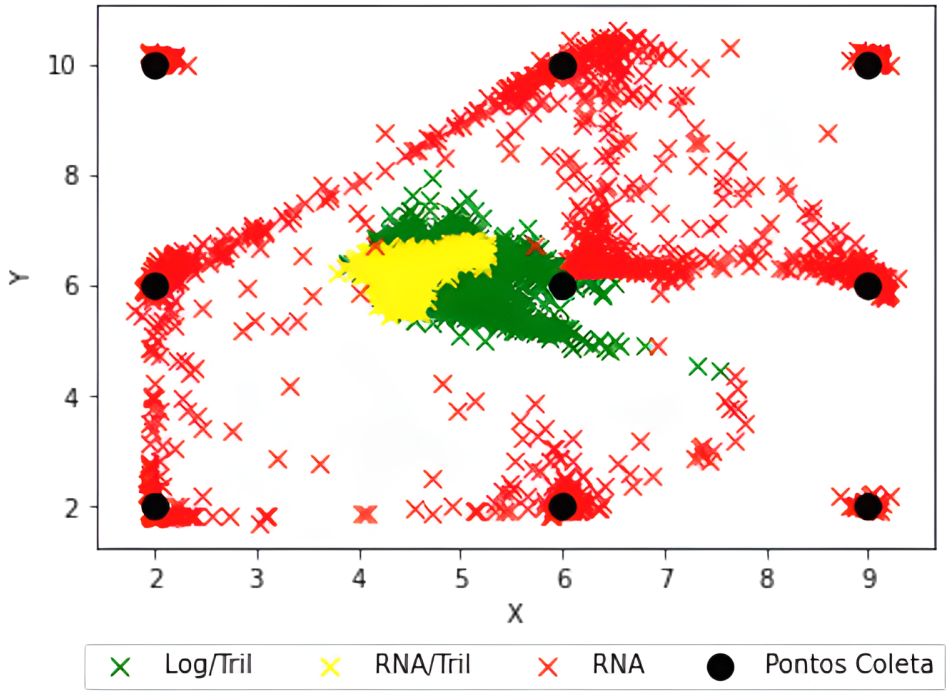
Quadro 10 – Erros com diferentes porcentagens do conjunto de dados de coordenadas

Conjunto de dados	**Erro Mínimo X**	**Erro Máximo X**	**Erro Mínimo Y**	**Erro Máximo Y**	**MSE**	**MAE**
100%	-4,020	4,136	-6,574	4,280	0,503	0,347
20%	-4,027	6,407	-4,164	3,310	0,611	0,358

A Figura 15 mostra um gráfico de dispersão com as coordenadas que cada um dos 3 métodos utilizados determinou. Nela é possível observar que ao utilizar as distâncias calculadas o método de trilateração praticamente obteve nenhum ponto de interseção entre os círculos, o que fez com que a maioria das coordenadas determinada por ele se concentrassem no centro do espaço. Isso ocorreu devido a variação que os valores de RSSI sofrem, fazendo com que a precisão da distância seja baixa e, consequentemente, o tamanho dos círculos seja diferente do real.

Como dentre os métodos utilizados a rede neural foi a que mostrou melhores resultados na determinação da localização, foram plotados outros gráficos de dispersão, Figuras 16, 17 e 18, para poder analisar melhor a sua eficiência. Algo intrigante que aconteceu com as determinações foi que a rede neural determinou coordenadas de um ponto, por exemplo (6,10), mais próximo a outro ponto (6,2), isso provavelmente ocorreu devido a variação dos valores de RSSI no momento da coleta. Outro detalhe percebido foi que os pontos mais ao meio ((2,6), (6,2), (6,6), (6,10)

Figura 15 – Coordenadas determinadas com cada método



e (9,6)) apresentaram menor precisão do que os dos cantos ((2,10), (2,2), (9,2) e (9,10)), isso provavelmente aconteceu por causa da direção dos *beacons* referente a antena.

Figura 16 – Coordenadas determinadas com rede neural nos pontos A (2,10), B (2,6) e C (2,2)

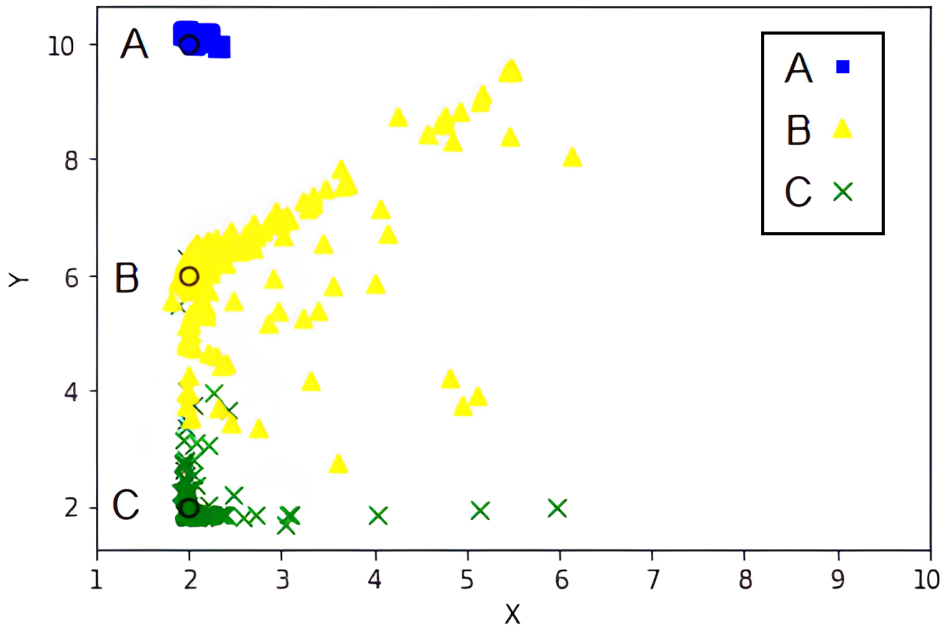


Figura 17 – Coordenadas determinadas com rede neural nos pontos (6,2), (6,6) e (6,10)

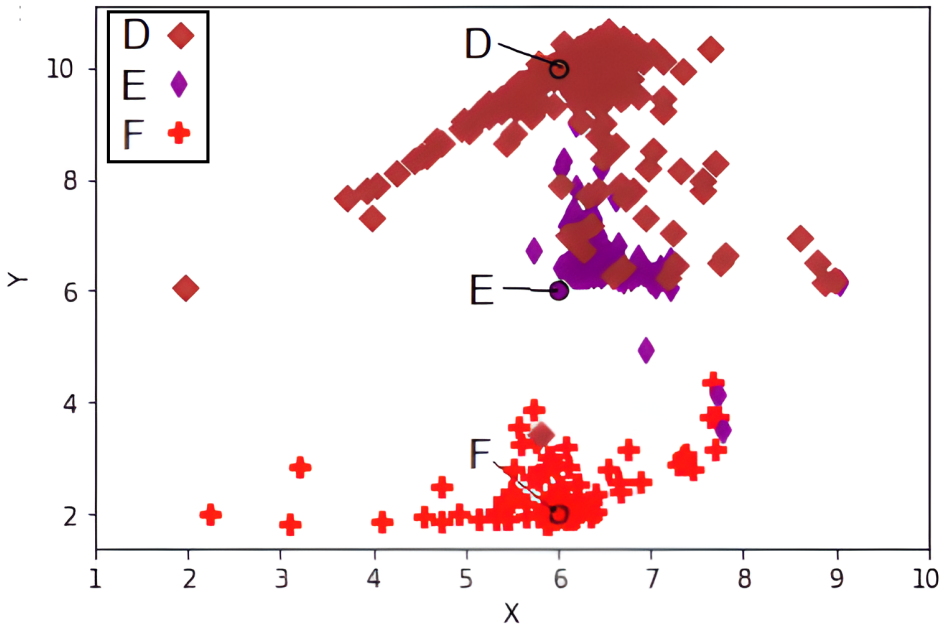
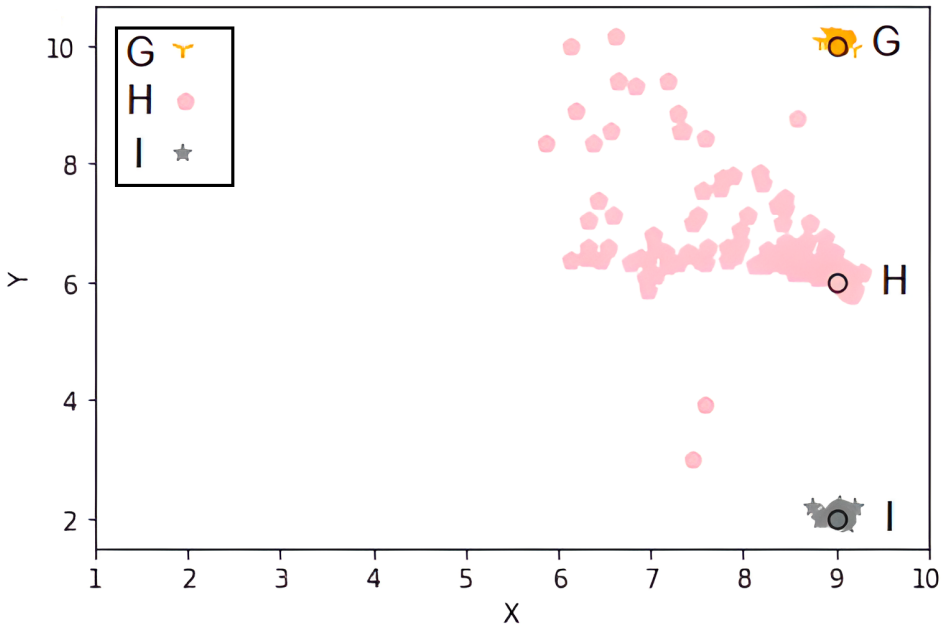


Figura 18 – Coordenadas determinadas com rede neural nos pontos (9,2), (9,6) e (9,10)



5
Conclusão

Os resultados obtidos mostraram que os valores de RSSI sofrem bastante variação, pois são muito sensíveis a interferência de objetos e dispositivos, incluindo outros *beacons*, no local, além de que, o aumento da distância implica no aumento da imprecisão dos valores. Outro fator muito importante percebido foi que a posição da antena do receptor também interfere na obtenção de valores mais precisos, o mesmo observado por Bouchard et al. (2016). Uma maneira de melhorar a precisão dos valores de RSSI seria aplicar filtros, como fez Röbesaat et al. (2017).

Durante o segundo experimento foi observado que o método de trilateração apresenta problemas ao determinar as coordenadas quando as distâncias utilizadas não formam círculos que intercedem em um ponto, e como padrão ela sempre coloca que o alvo se encontra no meio do espaço. Isto é problemático, especialmente com dispositivos que utilizam tecnologias que sofrem variações do ambiente, como os *beacons*. Sendo assim este método não é vantajoso ao utilizar dados da vida real coletados com este tipo de tecnologia, pois elas tendem a sofrem muita interferência.

Dentre os métodos utilizados o que obteve os melhores resultados foi a rede neural, em ambos os testes, graças a sua habilidade de adaptação, no entanto, ela ainda não é 100% precisa. Porém, como o objetivo não era determinar precisamente a posição do *notebook* na sala e sim a região onde ele se encontra, os resultados com a rede neural se mostraram muito satisfatórios. Com este nível de precisão ela já poderia ser utilizada por grandes lojas para *marketing* ativo.

Entre os trabalhos futuros podemos citar o desenvolvimento de um aplicativo *mobile* para determinar a posição em um ambiente interno ou em *stands* de grandes feiras; avaliar o posicionamento com o alvo em movimento e verificar o tempo gasto para estabilização; realizar os testes com mais *beacons* para verificar se a precisão aumenta; realizar os testes em um espaço maior e com poucos objetos; avaliar outros hiperparâmetros visando aumentar a precisão da rede neural; usar filtros nos dados antes de aplicar os métodos e utilizar outros métodos de regressão, como *random florest* e regressão múltipla, e comparar com a rede neural.

Referências

AKPINAR, E. *Bluetooth beacons: Everything you need to know*. 2021. Disponível em: <https://www.pointr.tech/blog/beacons-everything-you-need-to-know>. Acesso em: 09 mar 2022. Citado 2 vezes nas páginas 11 e 14.

APPLE. *iBeacon*. 2022. Disponível em: <https://developer.apple.com/ibeacon/>. Acesso em: 09 mar 2022. Citado na página 15.

BLIDH, H. et al. *Bleak*. 2020. Disponível em: <https://bleak.readthedocs.io/en/latest/index.html>. Acesso em: 27 jun 2022. Citado na página 23.

Bluepixel Technologies. *BLE Scanner*. 2022. Disponível em: <https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner>. Acesso em: 09 mar 2022. Citado na página 12.

BOUCHARD, K. et al. Evaluation of bluetooth beacons behavior. In: *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. [S.l.: s.n.], 2016. p. 1–3. Citado 3 vezes nas páginas 21, 29 e 36.

BROWNLEE, J. *How to Choose Loss Functions When Training Deep Learning Neural Networks*. 2019. Disponível em: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/#:~:text=The%20Mean%20Squared%20Error%2C%20or,to%20use%20for%20regression%20problems.> Acesso em: 08 jul 2022. Citado na página 19.

DANOVA, T. *BEACONS: What They Are, How They Work, And Why Apple's iBeacon Technology Is Ahead Of The Pack*. 2014. Disponível em: <https://www.businessinsider.com/beacons-and-ibeacons-create-a-new-market-2013-12>. Acesso em: 06 abr 2022. Citado na página 14.

DELUA, J. Supervised vs. unsupervised learning: What's the difference? March 2021. Disponível em: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>. Citado na página 19.

FACHRI, M.; KHUMAIDI, A. Positioning accuracy of commercial bluetooth low energy beacon. In: *2019 Fourth International Conference on Informatics and Computing (ICIC)*. [S.l.: s.n.], 2019. p. 1–4. Citado na página 21.

FROST, J. *Mean Squared Error (MSE)*. 2022. Disponível em: <https://statisticsbyjim.com/regression/mean-squared-error-mse/>. Acesso em: 07 jul 2022. Citado na página 24.

GOOGLE. *Eddystone*. 2018. Disponível em: <https://github.com/google/eddystone>. Acesso em: 09 mar 2022. Citado na página 15.

HAN, K. et al. Dff-edr: An indoor fingerprint location technology using dynamic fusion features of channel state information and improved edit distance on real sequence. *China Communications*, v. 18, n. 4, p. 40–63, 2021. Citado na página 10.

HOU, X.; ARSLAN, T. Monte carlo localization algorithm for indoor positioning using bluetooth low energy devices. In: *2017 International Conference on Localization and GNSS (ICL-GNSS)*. [S.l.: s.n.], 2017. p. 1–6. Citado na página 11.

IBM Cloud Education. *What are Neural Networks?* 2020. Disponível em: <https://www.ibm.com/cloud/learn/neural-networks>. Acesso em: 15 jun 2022. Citado 2 vezes nas páginas 18 e 19.

Jinan Huamao technology Co., LTD. *HM-10 Bluetooth*. 2022. Disponível em: <http://jnhuamao.cn/bluetooth.asp?id=1>. Acesso em: 09 mar 2022. Citado na página 22.

KAJIKAWA, N. et al. On availability and energy consumption of the fast connection establishment method by using bluetooth classic and bluetooth low energy. In: *2016 Fourth International Symposium on Computing and Networking (CANDAR)*. [S.l.: s.n.], 2016. p. 286–290. Citado na página 10.

KIM, C.; LEE, S. A research on beacon code architecture extension using category and code beacon structure. In: *2014 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2014. p. 187–188. Citado na página 15.

MARCEL, J. *How Bluetooth Technology Uses Adaptive Frequency Hopping to Overcome Packet Interference*. 2020. Disponível em: <https://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/>. Acesso em: 09 mar 2022. Citado na página 10.

NetSpot. *What is RSSI and its relation to a Wi-Fi network*. 2022. Disponível em: <https://www.netspotapp.com/wifi-signal-strength/what-is-rssi-level.html>. Acesso em: 09 mar 2022. Citado na página 11.

NICK. *The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs (2015)*. 2015. Disponível em: <https://www.aislelabs.com/reports/beacon-guide/>. Acesso em: 06 abr 2022. Citado na página 15.

PAKANON, N.; CHAMCHOY, M.; SUPANAKOON, P. Study on accuracy of trilateration method for indoor positioning with ble beacons. In: *2020 6th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*. [S.l.: s.n.], 2020. p. 1–4. Citado na página 16.

PUCKDEEVONGS, A. Indoor localization using rssi and artificial neural network. In: *2021 9th International Electrical Engineering Congress (iEECON)*. [S.l.: s.n.], 2021. p. 479–482. Citado na página 20.

PYTORCH. *Pytorch*. 2022. Disponível em: <https://pytorch.org/>. Acesso em: 09 mar 2022. Citado na página 24.

PYTORCH. *TORCH.UTILS.DATA*. 2022. Disponível em: <https://pytorch.org/docs/stable/data.html>. Acesso em: 16 jul 2022. Citado na página 24.

RADOI, I. E.; CIRIMPEI, D.; RADU, V. Localization systems repository: A platform for open-source localization systems and datasets. In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. [S.l.: s.n.], 2019. p. 1–8. Citado na página 10.

RECK, M. S. *Beacons BLE – BLUETOOTH LOW ENERGY – Design e análise de um sistema de localização indoor*. Dissertação (Monografia de Conclusão de Curso) — UCS – Universidade de Caxias do Sul, Caxias do Sul - RS, 2016. Citado 2 vezes nas páginas 15 e 16.

RÖBESAAT, J. et al. An improved ble indoor localization with kalman-based fusion: An experimental study. *Sensors*, v. 17, n. 5, April 2017. Disponível em: <https://www.mdpi.com/1424-8220/17/5/951>. Citado 5 vezes nas páginas 17, 18, 20, 28 e 36.

STEPHANIE. *Absolute Error and Mean Absolute Error (MAE)*. 2016. Disponível em: <https://www.statisticshowto.com/absolute-error/>. Acesso em: 07 jul 2022. Citado na página 24.

TUNCER, S.; TUNCER, T. Indoor localization with bluetooth technology using artificial neural networks. In: *2015 IEEE 19th International Conference on Intelligent Engineering Systems (INES)*. [S.l.: s.n.], 2015. p. 213–217. Citado na página 20.

VANZIN, L. *Localizador-Bluetooth*. 2022. Disponível em: <https://github.com/EnergyFall266/Localizador-Bluetooth>. Acesso em: 14 jul 2022. Citado na página 23.

VANZIN, L.; OYAMADA, M. S. Calibration of ble beacons and its impact on distance estimation using the log-distance path loss model. In: *2021 10th Latin-American Symposium on Dependable Computing (LADC)*. [s.n.], 2021. p. 1–4. Disponível em: <https://ieeexplore.ieee.org/document/9672575>. Citado 4 vezes nas páginas 10, 11, 22 e 28.