APLICATIVO MOBILE DE ACESSIBILIDADE PARA DEFICIENTES VISUAIS EM AMBIENTES INTERNOS

ACCESSIBILITY MOBILE APPLICATION FOR THE VISUALLY IMPAIRED IN INTERNAL ENVIRONMENTS

BERTIVELLO, Felipe[1]; ALVARENGA, Carolina[2]; ANDRIJAUSKAS, Fábio[3];
[1]Engenharia da Computação – Universidade São Francisco;
[2]Engenharia da Computação – Universidade São Francisco;
[3] Engenharia da Computação – Universidade São Francisco
fabio.andrijauskas@usf.edu.br

RESUMO. O desenvolvimento de uma solução para geolocalização em ambientes internos vem sendo cada vez mais necessário para amenizar dificuldades vivenciadas por pessoas com deficiência visual. Em estudos anteriores feitos para tentar encontrar uma resposta para este problema, usou-se a metodologia baseada na identificação de etiquetas (*TAG*) por sensores, contudo, observou-se algumas dificuldades em relação à comunicação entre os componentes pertencentes do sistema e o usuário, sendo a obstrução dos marcadores o maior dos problemas. Baseando-se nisso, a proposta da solução foi utilizar um sistema que emprega o uso de beacons como pontos de acesso para determinar a localização do utilizador, onde, um aplicativo em um aparelho móvel será o responsável por realizar a comunicação e monitoramento destes dispositivos. A utilização deste método apresentou maior usabilidade com relação ao uso de marcadores físicos, pois necessita de menos esforço por parte do usuário no encontro dos marcadores e não possui interferências por obstrução de objetos.

Palavras-chave: geolocalização interna, deficiência visual, beacon, bluetooth low energy.

ABSTRACT. The development of a solution for indoor geolocation has been increasingly necessary to reduce difficulties experienced by people with visual impairments. In previous studies done to try to find an answer to this problem, the methodology based on the identification of tags (*TAG*) by sensors was used, however, some difficulties were observed in relation to the communication between the components belonging to the system and the user, with the obstruction of markers being the biggest problem. Based on this, the solution proposal was to use a system that employs the use of beacons as access points to determine the user's location, where an application on a mobile device will be responsible for carrying out the communication and monitoring of these devices. This method showed greater usability in relation to the use of physical markers, as it requires less effort on the part of the user to find the markers and does not have interference due to obstruction of objects.

Keywords: indoor geolocation, visual impairment, beacon, bluetooth low energy.

INTRODUÇÃO

Com base em dados retirados do Censo da Educação Superior de 2017 realizado pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep), soma-se um total de 3.226.249 estudantes que ingressaram em algum tipo de curso de graduação, onde 14.050 possuem algum tipo de deficiência, sendo que, 32,1% deles possuem algum tipo de

deficiência visual. De inúmeras dificuldades vivenciadas por essas pessoas, uma das mais críticas é a locomoção, pois muitas vezes não há sinalização adequada, sequer pessoas para guiar o aluno pelas dependências de uma universidade. Sendo assim, é importante haver estudos a fim de propor uma solução capaz de promover uma melhora na mobilidade, independência e integração social para estes alunos.

Concomitantemente ao cenário de falta de infraestrutura, surgem novas tecnologias que podem auxiliar os deficientes visuais na autonomia de locomoção, como é o caso do GPS que é amplamente utilizado em aplicativos de navegação em ambientes externos. Contudo, tais tecnologias podem ser onerosas e ineficientes nesses ambientes, pois necessitam de muita precisão e baixa taxa de erros. Considerando essa análise, este projeto abordará uma forma de facilitar a locomoção e localização de deficientes visuais em ambientes internos de uma universidade, onde o aluno poderá saber onde se encontra e, também, conseguirá deslocar-se até o ponto desejado partindo de seu posicionamento atual.

Segundo Lima (2001), existem diversas técnicas para detectar a posição em ambientes fechados, sendo as principais análise do cenário, triangulação e proximidade. A análise do cenário pode ser baseada em uso de imagem visual ou fenômenos físicos para obter a conclusão sobre a posição de um objeto. A triangulação é baseada em dados geométricos para calcular a posição de um objeto, podendo ser utilizado a latência ou angulação, sendo que a latência faz o cálculo a partir da distância de três antenas com relação ao objeto e a angulação através do ângulo de cada antena com relação ao objeto. A técnica de proximidade faz uso de três abordagens, sendo por detecção, pelo monitoramento dos pontos de acesso e pela observação de sistemas de identificação automática.

Tratando-se de uma solução com foco em deficientes visuais, foi preferido a técnica de proximidade com a abordagem de monitoramento dos pontos de acesso. Por esses motivos, a ideia é desenvolver um aplicativo para celular que terá o papel de interpretar o comando de voz enviado pelo usuário, onde o mesmo deve informar o local que deseja ir, efetuar a localização dos pontos de acesso, processar uma resposta contendo qual é o beacon mais próximo para, então, descobrir qual o caminho mais rápido para determinado local escolhido e, assim, devolver a informação de como chegar lá. Portanto, este estudo visa o desenvolvimento de uma aplicação para geolocalização de deficientes visuais em ambientes internos com o uso de beacons e foco no aperfeiçoamento de outros estudos já realizados no desenvolvimento de sistemas de localização interna.

REFERENCIAL TEÓRICO

O presente referencial teórico foi definido com base em dois tópicos, sendo eles: Geolocalização e Beacon BLE.

Segundo Júnior (2015), geolocalização é a arte de descobrir onde um usuário está localizado, sendo esta localização podendo ser obtida através do endereço IP (*Internet Protocol*), uma conexão de rede sem fio (WIFI), através de uma torre de celulares e pelo GPS (*Global Positioning System*). Sendo os sistemas de geolocalização classificados de acordo com o ambiente em que atuam, é possível utilizar de diferentes técnicas para obter os melhores resultados de acordo com o local em que se deseja trabalhar. Esta classificação está separada em ambiente externo (*outdoor*) e interno (*indoor*).

Os sistemas de geolocalização em ambientes externos têm a características de conseguir abranger grandes áreas de localização como é o caso de cidades ou parques. Segundo Lima (2001, p. 10), a tecnologia que dá suporte à localização de objetos e pessoas

utilizando-se da estimativa de sua posição é o GPS. Seu funcionamento consiste na medida de distância entre o usuário e quatro satélites dispostos na órbita da Terra, onde conhecendo as coordenadas dos satélites em um sistema de referência apropriado, é possível calcular as coordenadas da antena do usuário no mesmo sistema de referência dos satélites.

Além disso, para a obtenção de um sinal de alta precisão é necessária a utilização de frequências muito altas, o que requer que exista uma linha reta, sem obstáculos, entre o transmissor e o receptor. Sendo suas principais desvantagens a taxa de atualização muito baixa e a dificuldade de se obter satélites em posições satisfatórias (GARCIA, 2009).

Já a geolocalização interna utiliza o LPS (*Local Positioning System*), que têm a características de abranger pequenas áreas como casas, hospitais, universidades e empresas. Segundo Lima (2001), sistemas *indoor* possuem algumas características:

> "Necessidade de precisão de 10 metros para a maioria das aplicações indoor, embora alguns requeiram precisão de 2 metro ou menos ainda. As tags devem ser tão pequenas e sensíveis à luz quanto possível para terem maior aplicabilidade. As tags não são caras, haja vista a grande simplicidade do projeto em comparação com receptores GPS. Uma infra-estrutura que localiza milhares de tags divergência de efeitos provocados pelos múltiplos caminhos que os sinais transmitidos podem seguir em ambientes indoor. Isto se torna um desafio quando combinado com a exigência de maior precisão. Um sistema de rádio tem que estar em conformidade com a regulamentação do órgão de comunicação responsável pela classificação do espectro de frequências, de modo que um cliente possa instalar e colocar o sistema para funcionar sem a necessidade de uma licença."

Existem, também, diversas técnicas para detectar a posição em ambientes fechados, sendo as principais análise do cenário, triangulação e proximidade onde é possível serem implementadas separadas ou em conjunto.

Conforme Hightower e Borriello (2001), a técnica de análise de cenário consiste em analisar uma cena capturada de um cenário em específico para obter conclusões a respeito da localização de objetos, sendo a cena podendo ser estática ou diferencial. Na análise de cenas estáticas, os recursos obtidos são verificados comparando com dados predefinidos que os mapeiam para encontrar objetos. Já na análise diferencial de cenas, é feita uma comparação de cenas a fim de encontrar diferenças para estimar a localização, e na cena em si pode consistir em imagens visuais, ou qualquer outro fenômeno físico mensurável, como as características eletromagnéticas que ocorrem quando um objeto está em uma determinada posição e orientação.

Para obter a localização geométrica de um objeto esta técnica utiliza de pontos geométricos de referência para efetuar o cálculo, sendo classificada em dois tipos: latência e angulação, onde na latência, o cálculo pode ser feito a partir da distância de três antenas até o objeto, do tempo que demora para o sinal chegar e retornar até o receptor e a intensidade do sinal retornado na antena após seu envio e já na angulação, o cálculo é feito com base no ângulo obtido entre duas antenas e o objeto, porém é necessário também saber a distância entre as antenas para determinar a localização.

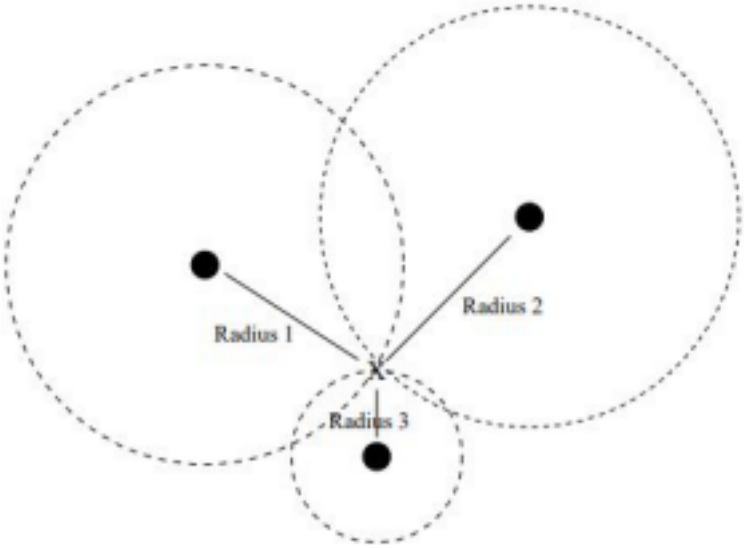

Figura 1 - Determinar a posição 2D usando latência requer medições de distância entre o objeto 'X' e 3 pontos não colineares.
Fonte: Hightower e Borriello (Location Sensing Techniques, 2001)

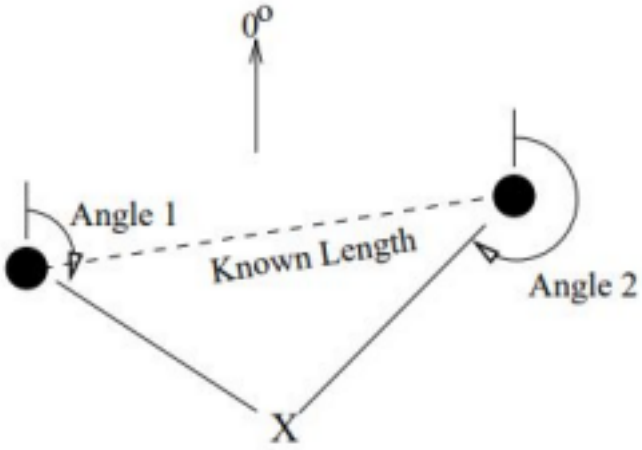

Figura 2 - Localização do objeto 'X' usando ângulos relativos a um vetor de referência 0º e a distância entre dois pontos de referência.
Fonte: Hightower e Borriello (Location Sensing Techniques, 2001)

A técnica de localização por proximidade consiste em saber previamente a localização de um dispositivo, assim quando o objeto fizer algum tipo de interação é possível saber onde se encontra. Essa técnica, conforme Hightower e Borriello (2001), pode ser classificada em de três formas:

- **Contato Físico:** Onde por meio do contato físico pode-se obter a localização através do uso de sensores *touch* e botões de pressão.
- **Monitoramento de Pontos de Acesso:** Consiste em monitorar quando um dispositivo móvel entrou no alcance do sinal de um ponto de acesso *wireless*.
- **Observação de Sistema de Identificação Automática:** Consiste na identificação de

etiquetas (*TAG*) por sensores, histórico de login em computadores, registros de bloqueio de cartões eletrônicos, etc.

Em conjunto com as técnicas de geolocalização interna uma nova tecnologia vem facilitando o acesso à comunicação e localização, os Beacon BLE (*bluetooth low energy*). Esses pequenos aparelhos foram desenvolvidos para proporcionar experiências melhores de localização e proximidade para diversas aplicações, sendo uma delas a de promover uma maior precisão em ambientes internos. Além disso, eles podem ser instalados em qualquer tipo de superfície, emitindo um sinal *broadcast* que qualquer dispositivo próximo que possua *bluetooth* possa identificar.

O *beacon* consegue transmitir apenas um pequeno pacote de dados, onde o envio de informações complexas acaba não sendo possível. Entretanto, é possível enviar diversos tipos de informações como de um único ID, um link ou status de seu funcionamento. Abaixo estão alguns tipos de beacons e suas funcionalidades exemplificadas conforme as fabricantes Google (*Eddystone*) e Apple (*iBeacon*):

- **Eddystone-UID:** Transmitem um ID único e estático.
- **Eddystone-URL:** Transmitem um URL compactado que, uma vez analisado e descompactado, pode ser usado diretamente pelo cliente.
- **Eddystone-TLM:** Transmitem dados de status de beacon que são úteis para a manutenção da frota de beacon e capacitam o *endpoint* de diagnóstico da API de beacon de proximidade do Google.
- **Eddystone-EID:** Transmitem um identificador que muda a cada poucos minutos. O identificador pode ser convertido em informações úteis por um serviço que compartilha uma chave com o beacon individual.
- **iBeacon:** Esta tecnologia pode ser usada para estabelecer contato com um dispositivo iOS para determinar quando ele se aproximou e afastou de uma região.

Figura 3 - Logo de iBeacon e Eddystone.
Fonte: Beacon Simulator.

Levando em conta tais vantagens a integração com um aplicativo *mobile* que suporte satisfatoriamente a troca de informações com tais dispositivos é altamente necessária, podendo-se considerar a utilização de uma linguagem que está ganhando cada vez mais mercado pela sua fácil adaptabilidade e eficácia no desenvolvimento, o Flutter.

METODOLOGIA

A partir do estudo de inúmeras tecnologias de localização interna, o artigo *Identification of Markers in Challenging Conditions for People with Visual Impairment Using Convolutional Neural Network* feito por Elgendy, Guszsvinecz e Silk-Lanyi

serviu como base de desenvolvimento para esse projeto através da ideia de utilização de um aplicativo para realizar a interface com o usuário utilizando comandos de voz onde é traçado as rotas de destinos através da identificação de marcadores.
Pelo meio de busca por uma tecnologia que auxiliasse na resolução do problema por obstrução de pessoas e objetos, foi descoberto a tecnologia *wireless* dos dispositivos Beacons que utilizam *bluetooth low energy* para transmitir algumas informações, sendo uma delas um identificador assim como os arucos porém através de ondas de rádio.
Assim, com a descoberta deste dispositivo juntamente com a necessidade de implementação de suas funcionalidades foi visto que a linguagem de programação Flutter já possuía bibliotecas desenvolvidas para tratar dos beacons, sendo então, a tecnologia escolhida para o desenvolvimento do aplicativo.

RESULTADOS E DISCUSSÃO

O seguinte tópico contém a explicação do mapa de marcadores de beacons, do simulador de beacon e sua configuração utilizada, o fluxo do sistema do momento em que é aberto o aplicativo ao final quando se chegou ao destino desejado e o desenvolvimento do software.
A ideia inicial é dispor todos os beacons nos corredores e também nas salas para que se possa fazer a geolocalização e assim saber se o usuário está perto ou longe do alvo, após a disposição dos dispositivos será criado em um banco de dados local um mapa baseado em grafos.
A teoria dos grafos possui aplicação direta na engenharia da computação por trazer os conceitos dos modelos matemáticos mais difíceis e importantes da atualidade, incluindo o conceito do caminho mínimo, onde consiste em uma soma de custos das arestas até um determinado nó que possui seu próprio peso, fazendo-se então a soma e determinando qual é o caminho mais curto até o destino.

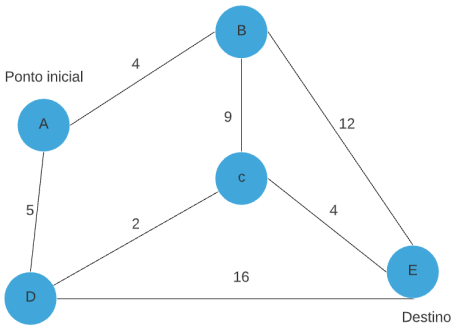

Figura 4 - Grafo caminho mínimo
Fonte: Própria autoria.

Conforme mostrado na Figura 4, para determinar qual é a melhor rota até o destino é necessário definir-se o ponto inicial e traçar todas as possibilidades até o destino sem repetir as arestas (grafo hamiltoniano).

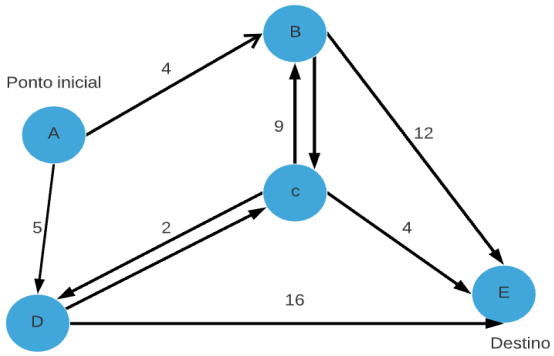

Figura 5 - Grafo percorrendo o caminho mínimo.
Fonte: Própria autoria.

Na Figura 5, do ponto inicial A é possível definirmos os seguintes caminhos:

- Do Ponto A ao ponto D ao ponto E com custo 21
- Do ponto A ao ponto B ao ponto E com custo de 16.
- Do ponto A ao ponto B ao ponto C ao ponto E com custo 17.
- Do Ponto A ao ponto D ao ponto C ao ponto B ao ponto E com custo 28.
- Do Ponto A ao ponto B ao ponto C ao ponto D ao ponto E com custo 31.
- Do Ponto A ao ponto D ao ponto C ao ponto E com custo 11.

A partir da análise das arestas é possível definir que o melhor caminho a se tomar é do ponto A ao ponto D ao ponto E com o custo de 11. Partindo desse princípio foi calculado qual seria o melhor caminho a partir do ponto de partida do usuário.

Com o intuito de poder executar diversas simulações com parâmetros diferentes para determinar qual é a melhor configuração a ser utilizada pelo beacon, foi utilizado o software Beacon Simulator disponibilizado pelo Vincent Hiribarren na *Play Store.* De acordo com ele, este aplicativo transforma seu Android em um transmissor beacon BLE, podendo criar sua própria coleção para emular um beacon físico. Este app dispõe de vários tipos de beacons para realizar simulações, possuindo tanto o *iBeacon* como todos os tipos de *Eddystone*. Todos esses possuindo as configurações que poderiam ser utilizadas em um ambiente real.

Partindo do princípio que pode-se criar qualquer configuração, foi dado início a busca para descobrir qual o melhor tipo de beacon e seus respectivos parâmetros a serem utilizados para este sistema. Como este projeto está sendo desenvolvido para um sistema operacional Android e envolve somente a necessidade de saber qual é o dispositivo presente em determinado local, foi determinado que o tipo de beacon *Eddystone-UID* seria o melhor a ser utilizado dentre os disponíveis, pois sua função é apenas transmitir um ID numérico.

Definido qual dispositivo seria empregado na proposta, foi visto quais os parâmetros disponíveis para edição. Sendo eles o próprio ID do beacon (*Instance ID*), seu nome para identificação, o *TX Power* e a configuração do *Broadcast* que compõem o poder de transmissão (*Transmission Power)* e a frequência (*Frequency Mode*). Nos parágrafos abaixo está a explicação de como foi obtida a configuração final do projeto, sendo que todos os testes foram realizados de forma manual, juntamente do aplicativo desenvolvido para coleta dos dados.

Para as características do *Broadcast*, existem disponíveis os valores de alto (~-56 dBm), médio (~-66 dBm), baixo (~-75 dBm) e muito baixo para o poder de transmissão e de baixa

latência(~10 Hz), balanceado (~3 Hz) e baixo poder (~1 Hz) para a frequência. Dado a disponibilidade dos valores, as características do projeto em obter dados a longa distância, para efeito de verificação se o caminho está sendo percorrido corretamente, e a necessidade de ter uma baixa latência em conjunto com alto poder, a configuração definida é a de alto para o poder de transmissão e balanceado para a frequência.

Definido os parâmetros do *Broadcast*, logo de início já foi visto que o valor definido para *Tx Power* interferia diretamente na informação da distância recebida pelo software final de geolocalização. Portanto foi realizado um teste onde foi comparado os dados recebidos com o beacon emitindo o *Tx Power* com valor padrão do Beacon Simulator (-65) e uma de maior valor (-10) a uma distância de 5 metros do aplicativo final do projeto. Abaixo estão as Figuras 6 e 7 mostrando, respectivamente, estas diferenças. Em conclusão do teste, foi analisado que o *Tx Power* de maior valor apresentou uma distância passível de melhor aplicabilidade no desenvolvimento do software por apresentar uma menor variação de valores e, portanto, maior precisão.

```
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -86, distance: 0.1235736457741454
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -94, distance: 0.3007598664781325
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -83, distance: 0.08664076357335604
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -84, distance: 0.09766393261896002
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -90, distance: 0.19470021977475097
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -92, distance: 0.24256024399070816
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -93, distance: 0.2702531924482457
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -93, distance: 0.2702531924482457
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -94, distance: 0.3007598664781325
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -91, distance: 0.217447926403712
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -92, distance: 0.24256024399070816
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -94, distance: 0.3007598664781325
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -92, distance: 0.24256024399070816
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -91, distance: 0.217447926403712
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -93, distance: 0.2702531924482457
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -93, distance: 0.2702531924482457
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -95, distance: 0.33433157961685533
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -93, distance: 0.2702531924482457
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -94, distance: 0.3007598664781325
I/flutter (30363): beaconId: 000000000002, tx: -65, rssi: -96, distance: 0.3712390714603643
```

Figura 6 - Valor de -65 para *Tx Power*
Fonte: Própria autoria.

```
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -88, distance: 60.450758888736054
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -86, distance: 50.65041602033556
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -82, distance: 35.118597291628056
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -94, distance: 100.44395351773085
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -99, distance: 149.72214347538312
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -87, distance: 55.361738531414716
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -91, distance: 78.24571342102239
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -93, distance: 92.50285811165236
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -94, distance: 100.44395351773085
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -98, distance: 138.45866705174376
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -85, distance: 46.29266826467707
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -93, distance: 92.50285811165236
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -86, distance: 50.65041602033556
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -89, distance: 65.94294185535738
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -95, distance: 108.97255773392655
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -86, distance: 50.65041602033556
I/flutter (30363): beaconId: 000000000002, tx: -10, rssi: -86, distance: 50.65041602033556
```

Figura 7 - Valor de -10 para *Tx Power*
Fonte: Própria autoria.

Contudo, mesmo com essa distância sendo mais precisa quando comparada ao *Tx Power* de -65, ainda existe uma variação muito grande para se utilizar no programa, sendo que esta oscilação ocorre entre 35 e 149 para os mesmos 5 metros. Portanto, a solução encontrada para esta situação, foi utilizar da média dos dez primeiros resultados encontrados, assim, esta variação seria de aproximadamente 86.

Por consequência da necessidade de se obter uma média a fim de aumentar a precisão dos valores de distância encontrados no software, um novo teste foi realizado para determinar qual é a distância média apresentada pelo Beacon Simulator com relação ao aplicativo nas distâncias de 1 metro, 3 metros, 5 metros, 8 metros e 10 metros. A Figura 8 abaixo representa os resultados obtidos e na Figura 9 representa a configuração final do beacon *Eddystone-UID* no Beacon Simulator.

TX: -10	Média
	Distance
1m	20
3m	27
5m	81
8m	112
10m	122

Figura 8 - Valor da distância média com relação ao aplicativo.
Fonte: Própria autoria.

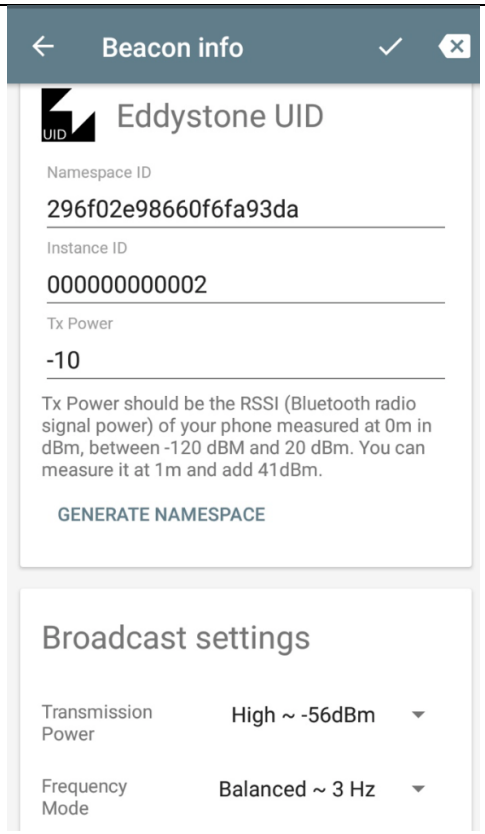

Figura 9 - Configuração Final do Beacon Simulator.
Fonte: Própria autoria.

Seguindo o fluxograma abaixo, Figuras 10 e 11, a interface do sistema será através de um aplicativo instalado no dispositivo móvel com sistema Android. Para iniciar o software é necessário clicar em qualquer lugar da tela e informar o que deseja ou onde deseja ir, caso a palavra informada seja 'tutorial' o software explicará como deve ser utilizado, porém caso seja diferente, será feita uma busca no banco de dados para verificar sua existência, onde, se não existir o usuário será informado de que a palavra informada não existe, caso o contrário, o fluxo do sistema iniciará.

Quando a palavra é reconhecida e encontrada no banco de dados, o destino é confirmado e se inicia a busca pelos beacons. Se o beacon encontrado não for do tipo Eddystone-UID a busca inicia novamente. Do contrário, verifica se seu ID existe na base de dados e atualiza a tabela que possui a informação dos beacons lidos em caso positivo, caso não seja, a busca por beacons se inicia novamente.

Após a atualização da tabela com a informação da distância do beacon lido, é verificado se os beacons que já foram lidos, foram atualizados pelo menos 10 vezes e, em caso positivo, será feita uma busca no banco de dados para saber qual o beacon que está mais próximo com base na distância média apresentada.

Se a distância do beacon for menor que 40 e sua identificação for igual a identificação do beacon de destino, é informado ao usuário que chegou ao destino desejado, porém se não for o beacon final, o próximo local que é necessário chegar será identificado e registrado, assim como é feita uma atualização do último local em que se chegou, para então, limpar a tabela de

registro dos beacons lidos para o uso em um novo ciclo e, por fim, o usuário é informado como chegar no próximo destino.
Se a distância do beacon for maior que 85 ele limpa a tabela de registro dos beacons lidos e inicia a busca pelos beacons novamente. Caso seja menor que 85 e o ID for o mesmo do próximo destino significa que o usuário está indo para o caminho correto, porém se o ID não for o mesmo o usuário será informado que está indo na direção errada.

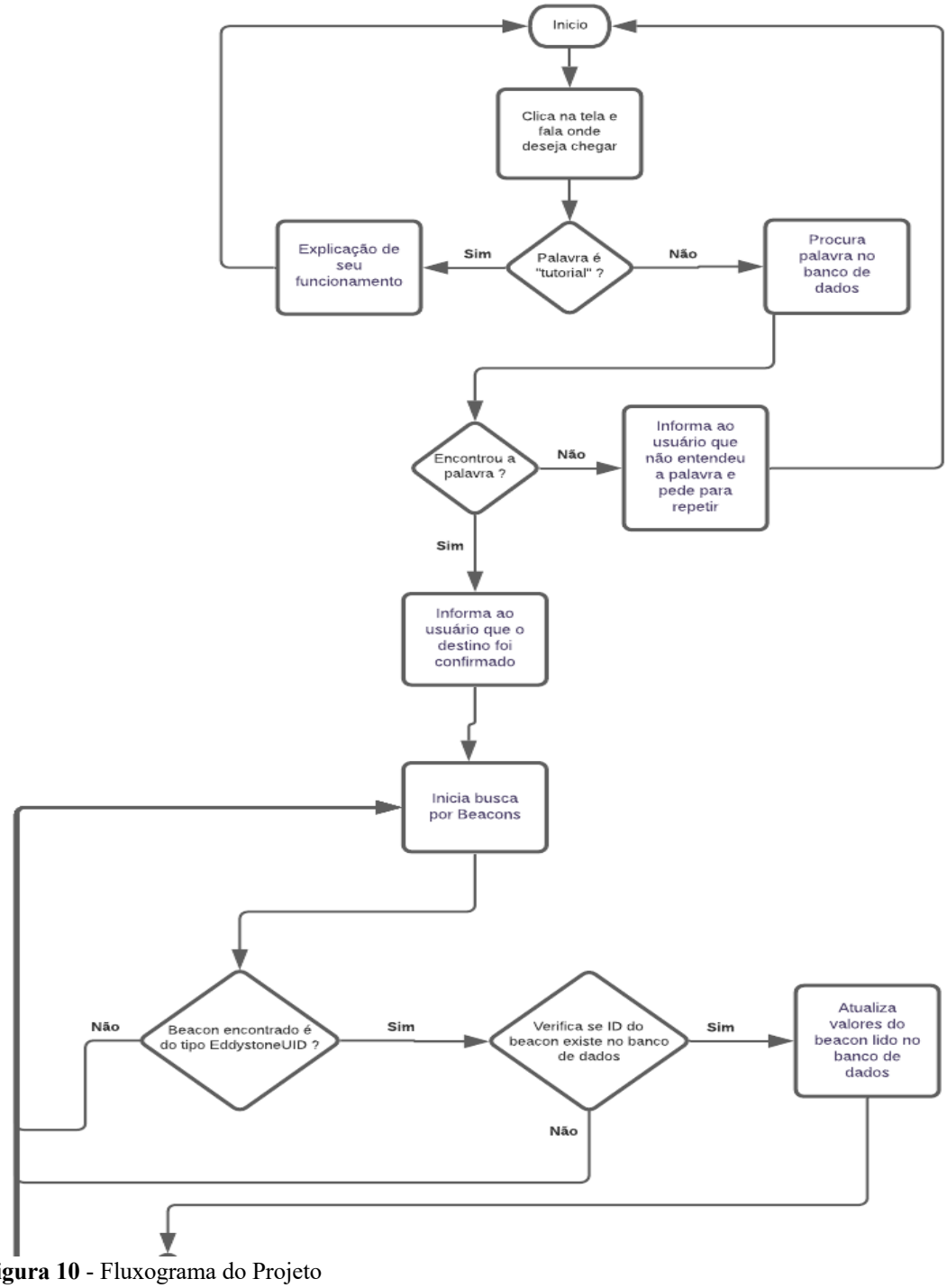

Figura 10 - Fluxograma do Projeto
Fonte: Própria autoria.

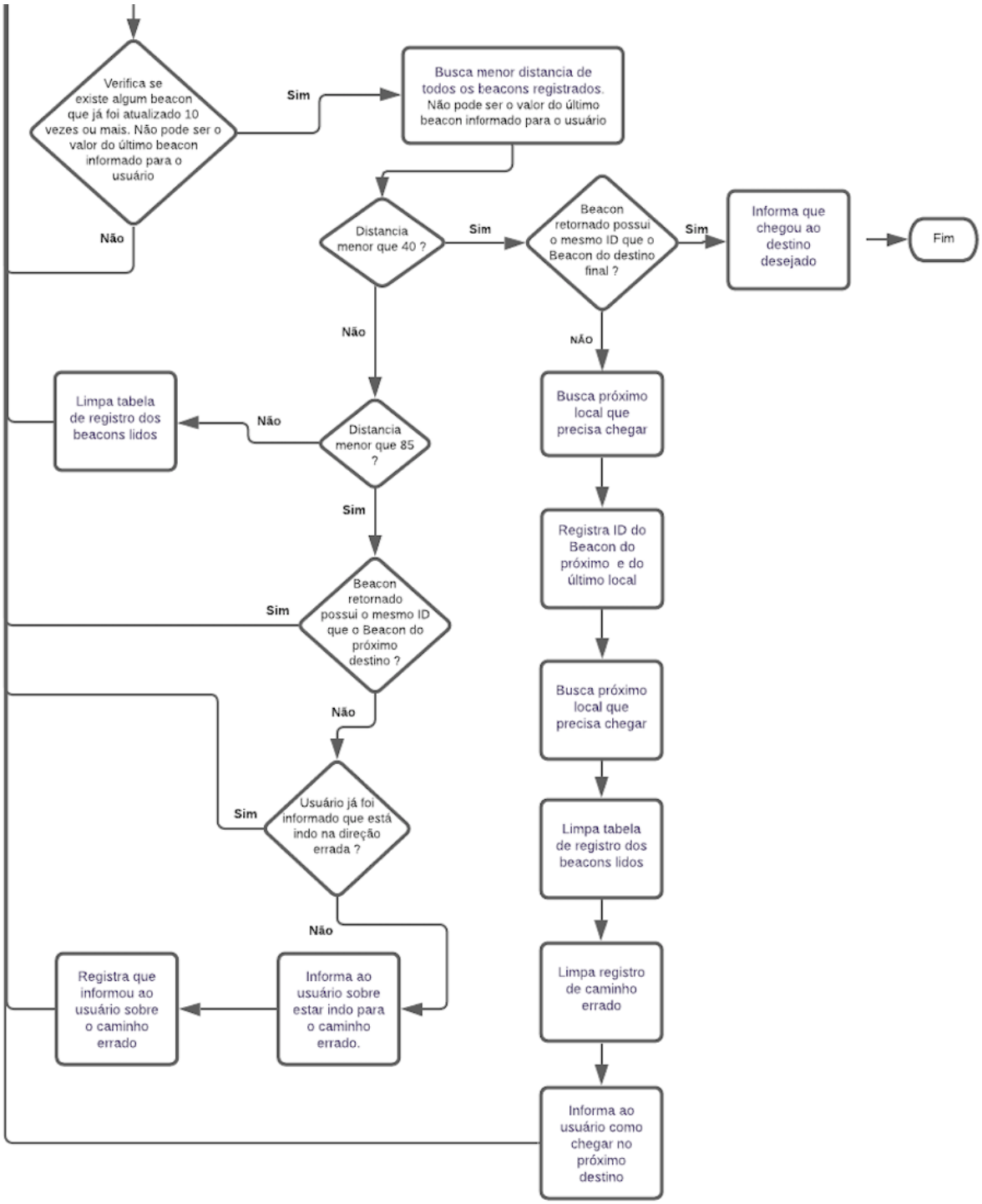

Figura 11 - Fluxograma do Projeto
Fonte: Própria autoria.

Como a ideia do sistema é funcionar para os usuários que possuem algum tipo de deficiência visual, o aplicativo deverá funcionar por comandos por voz, sendo necessário tanto o reconhecimento da voz como a fala sendo dita pelo software. Pensando na melhor maneira para realizar este desenvolvimento, foi visto que a linguagem de programação Flutter juntamente com o banco de dados SQFlite possui algumas bibliotecas já prontas para sua utilização, sendo necessário apenas importá-las e implementá-las. Logo, o desenvolvimento da interação com o usuário foi o primeiro passo dado, sendo, em seguida, o processo de busca por beacons e a implementação da lógica para chegar ao destino desejado.

Para o reconhecimento da voz do utilizador, foi utilizado a biblioteca *speech_to_text,* que segundo o site *pub.dev*, o plugin contém um conjunto de classes que facilitam o uso dos recursos de reconhecimento de voz do dispositivo móvel no Flutter, onde os casos de uso alvo para esta biblioteca são comandos e frases curtas suportando os sistemas Android e iOS. Já para a interação do aplicativo com o usuário, foi utilizado a biblioteca *flutter_tts*, que possui a função de converter um texto em fala.

Seguindo com a implementação para o reconhecimento de beacons, foi preciso a utilização das bibliotecas *flutter_blue* e *flutter_blue_beacon*. O plugin responsável pela conexão e comunicação com dispositivos Android, iOS e bluetooth low energy. Já o segundo funciona em conjunto com o primeiro plugin, sendo este responsável por implementar o uso do Eddystone e do iBeacon.

Após as implementações feitas, foi dado continuidade ao desenvolvimento da interface, onde após um clique em qualquer parte da tela pelo usuário pode-se inserir a devida instrução por voz, no qual o sistema dará a tratativa verificando pela palavra tutorial ou encontrando a palavra na lista de beacon disponíveis. A Figura 12 abaixo mostra a parte do código onde a palavra é verificada.

```
_speech.listen(
  cancelOnError: false,
  onResult: (val) => setState(() {
    if (val.hasConfidenceRating && val.confidence > 0) {
      if(val.recognizedWords == 'tutorial'){
        _fala = 'Olá, você está usando o BlindGui. Para utilizar, basta clicar em qualquer lugar da tela e falar onde deseja ir.';

        _confidence = val.confidence;
        _speak(_fala);
      } else {
        banco.getPalavra(val.recognizedWords).then((list){
          if(list == null){
            _fala       = 'Não entendi o que você disse. Poderia repetir ?';
            _confidence = val.confidence;
            _speak(_fala);
          } else {
            _destino    = list.palavra;
            _destinoId  = list.beacon;
            _fala       = 'Destino confirmado, você está indo para a $_destino';
            _confidence = val.confidence;
            _speak(_fala);
            _startScan();
          }
        });
      }
    }
  }),
);
```

Figura 12 - Parte do código que mostra o reconhecimento da fala e sua tratativa.
Fonte: Própria autoria.

Seguindo para a busca dos beacons, caso a palavra seja reconhecida, será verificado se o tipo do beacon é o *Eddyston-UID*, conforme Figura 13. Caso seja, será dado continuidade

ao fluxo, onde será inserido a somatória para efeito de média, conforme a Figura 14, e dado a tratativa da distância média, que caso seja menor que 40 trata se usuário chegou no destino e se for entre 40 e 85 verifica se está indo para o caminho certo. Isso é retratado pelas Figuras 15 e 16 respectivamente.

```
_startScan() {
  print("Scanning now");
  _scanSubscription = flutterBlueBeacon.scan(timeout: const Duration(seconds: 3600)).listen((beacon) {
    if (beacon is EddystoneUID) {
      print("beaconId: ${beacon.beaconId}, tx: ${beacon.tx}, rssi: ${beacon.rssi}, distance: ${beacon.distance}");
```

Figura 13 - Parte do código que mostra a verificação do tipo de beacon.
Fonte: Própria autoria.

```
//busca últimos valores de beacon lido
banco3.getWalk(beacon.beaconId).then((list2){
  int dis = beacon.distance.round()+list2.dist;
  int rss = (beacon.rssi*-1)+list2.rssi;
  int qtd = list2.qtd+1;

  //atualiza valores de beacon lido
  banco3.atualizaWalk(beacon.beaconId, dis, rss, qtd).then((list3){
```

Figura 14 - Parte do código onde é inserido a somatória no banco para efeito de média.
Fonte: Própria autoria.

```
//verifica se melhor distancia estiver menor que 40
if(_ultimoDistance <= 40){
  print("----------_ultimoDistance <= 40----------");
  print(_ultimoDistance);
  //verifica se é o destino desejado
  if(list6.beacon == _destinoId){
    _fala = 'Voce chegou ao seu destino!';
    _speak(_fala);
    _stopScan();
  } else {
    _ultimoLocalId = list6.beacon;
    _ultimoLocal   = list6.palavra;

    print("----------_ultimoLocalId----------");
    print(_ultimoLocalId);
    print("----------_ultimoLocal----------");
    print(_ultimoLocal);

    //busca proximo caminho
    banco2.getCaminho(_ultimoLocalId, _destinoId).then((list7){
      _dirErrada      = 0;
      _proximoLocalId = list7.proximo;
      _fraseLocal     = list7.frase;

      banco3.limpaWalk();
      _fala = 'Voce está na $_ultimoLocal. Para chegar na $_destino, é preciso $_fraseLocal .';
      _speak(_fala);
    });
  }
```

Figura 15 - Parte do código que mostra a verificação e tratativa caso a distância seja menor que 40.
Fonte: Própria autoria.

```
} else if(_ultimoDistance > 40 && _ultimoDistance <= 85) {
  print("----------_ultimoDistance > 40 && _ultimoDistance <= 85----------");
  print(_ultimoDistance);
  //verifica se esta indo na direção errada.
  if(list6.beacon != _proximoLocalId && _dirErrada == 0){
    print("----------list6.beacon != _proximoLocalId && _dirErrada == 0----------");
    _dirErrada = 1;
    banco3.limpaWalk();
    _speak('Você está indo na direção errada. É preciso retornar onde estava, $_fraseLocal para chegar na $_destino.');
  }
```

Figura 16 - Parte do código que mostra a verificação e tratativa caso a distância esteja entre 40 e 85.
Fonte: Própria autoria.

Com a implementação finalizada do software, foi observado que as bibliotecas utilizadas tanto para interface com o usuário como para a busca de beacons cumpriram com exatidão seu propósito. Tendo como destaque a biblioteca *flutter_tts* por apresentar diversos tipos de vozes e tonalidades para configurar o melhor ambiente para quem usa o software.
A respeito do aplicativo Beacon Simulator, foi explorado seu funcionamento em ambiente de testes onde apresentou as configurações padrões de beacons reais, assim, podendo ser feito a implementação do projeto a fim de verificar sua eficácia. Porém, também apresentou uma variação muito grande do valor da distância, tendo esta que ser tratada para apresentar maior precisão.

CONCLUSÃO

Por este estudo ter como foco o aperfeiçoamento de sistemas de geolocalização interna com base em outros estudos já realizados, pode-se observar ao fim deste, que o uso dos dispositivos beacon apresentou uma melhor solução ao uso de marcadores físicos identificados por câmera, pois não ocorre interferência na leitura dos dados por obstrução direta de objetos, não é mais necessário o uso de uma câmera, dados mais precisos em relação ao caminho e apresenta como vantagem a identificação prévia do caminho para verificar se este está sendo percorrido corretamente. Contudo este projeto ainda apresenta a construção do mapa de marcadores de forma manual, podendo este ser implementada de forma automática em trabalhos futuros.

REFERÊNCIAS

BIANCHETTI, L; FREIRE, I. **Um olhar sobre a diferença: Interação trabalho e cidadania,** 6.a ed. São Paulo: Papirus, 2004

BRASIL. MEC. Instituto Nacional de Estudos e Pesquisas Educacionais (Inep). Censo da Educação Superior: **Resumo Técnico do Censo da Educação Superior 2017**.

ELGENDY, M; GUSZSVINECZ T; SILK-LANYI, C. **Identification of Markers in Challengin Conditions for People with Visual Impairment Using Convolutional Neural NetworkEm**, p. 3-5, Veszprém, Hungary 26 nov. 2019.

GARCIA, C. **Sistemas de Localização Indoor e Outdoor.** 2009. Universidade de São Paulo, São Paulo, 2009.

HIGHTOWER, J.; BORRIELLO, G. **Location Sensing Techniques.** 2001. Dissertação (Computer Science and Engineering) - University of Washington, Seattle, 2001.

JÚNIOR, G. **Desenvolvimento de Sistema de Geolocalização em Realidade Aumentada para Multiplataforma Móvel.** Dissertação (Pós-Graduação em Engenharia Elétrica) - Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, 2015.

LIMA, E. **Sistema para Localização de Pessoas e Objetos em Ambientes Indoor.** 2001. Dissertação (Mestrado em Computação Móvel) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2001.

MACHADO, V. **Geolocalização Indoor: Um Estudo de Caso Usando RFID na Plataforma Arduino.** 2013. Trabalho de Conclusão de Curso (Curso de Tecnologia em Sistemas para Internet) - Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio Grandense, Passo Fundo, 2013.

MENDES, G. **Sistema Mobile Web Para Busca Georreferenciada de Imóveis.** 2011. Monografia de Especialização (Curso de Especialização em Tecnologia Java) - Universidade Tecnológica Federal do Paraná, Curitiba, 2011.

MONICO, J. **Posicionamento pelo NAVSTAR-GPS.** São Paulo: UNESP, 2000.

MOTA, M. **Orientação e Mobilidade**: Conhecimentos Básicos Para a Inclusão Da Pessoa Com Deficiência Visual. Brasília: [s. n.], 2003. Disponível em: http://portal.mec.gov.br/seesp/arquivos/pdf/ori_mobi.pdf. Acesso em: 28 jun. 2020.

NICOLAU, H; GUERREIRO, T; JORGE, J. **Blobby: How to Guide a Blind Person**, p. 2-3, Boston-MA, 2009.

ROMERO, T; KIRNER, C; SISCOUTTO R. **Fundamentos e Tecnologia de Realidade Virtual e Aumentada**, Livro do Pré-Simpósio VIII Symposium onVitual Reality, p. 23, Belém - PA 2006.

Publicado em 14/03/2022