BLACK GLASSES – ASSISTENTE PARA DEFICIENTES VISUAIS VIA GEOLOCALIZAÇÃO

William Lopes da Silva, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

wiiu.lopes@gmail.com, dalton@furb.br

Resumo: *Este artigo apresenta o desenvolvimento de um dispositivo móvel off-line para auxílio na locomoção de deficientes visuais que utiliza uma plataforma baseada em Raspberry Pi. O dispositivo recebe e interpreta comandos de voz para iniciar, parar e desligar, e através dos chips de bússola e GPS, verifica e informa para os deficientes visuais via sintetizador de texto para voz, os pontos de interesses que se encontra dentro do raio do seu percurso. Foi desenvolvido na linguagem de programação Python 3 e utilizou os módulos QMC5883L (bússola) e NEO6M (GPS). Os testes comprovaram que o protótipo trabalhou de forma totalmente off-line, não necessitando conexão com a internet para operar e que os módulos de GPS e bússola tiveram uma precisão satisfatória. O protótipo se mostrou capaz de receber comandos de voz, calcular e retornar para o usuário o áudio no headset bluetooth, com a distância e direção dos pontos de interesses dentro do raio do seu percurso.*

Palavras-chave*: Deficiência visual. Raspberry. Text to Speech. Global positioning system. Python.*

1 INTRODUÇÃO

A comunidade de pessoas com problemas visuais busca conquistar seu espaço de forma independente no meio social, porém dada suas limitações existem diversos obstáculos ao longo desta busca.

> Pessoas cegas e com baixa visão dependem de terceiros para identificar ruas, endereços, itinerários de ônibus, avisos, obstáculos e outras referências visuais. Transitam com dificuldade por vias públicas em geral e ficam expostas a constantes situações de risco. (SÁ, 2010, p. 1).

Além disso, a expectativa é de que, em 2020, a cegueira atinja 38,5 milhões de pessoas. Já em 2050, serão quase 115 milhões de cegos no mundo e a quantidade de pacientes com comprometimento da visão, variando de moderado a grave, baterá a casa dos 588 milhões no mesmo ano. Hodiernamente, há 217 milhões de pessoas com comprometimento nesses níveis (O GLOBO, 2017).

Indubitavelmente, o avanço da tecnologia tem proporcionado uma grande evolução na inclusão de pessoas com algum tipo de deficiência, permitindo o contato com um mundo que antes poderia ser inacessível, entre essas tecnologias temos as assistivas. De acordo com Gambarato et al (2012, p. 9), as tecnologias assistivas refere-se a qualquer item, equipamento, produto ou sistema que ajude no desenvolvimento do conhecimento de pessoas com limitações. Radabaugh (1993) afirma que para as pessoas sem deficiência, a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis. Conclui-se então que o objetivo maior da tecnologia assistiva é proporcionar à pessoa com deficiência visual independência, qualidade de vida e inclusão social, através da ampliação de sua comunicação, mobilidade e controle de seu ambiente.

O objetivo deste projeto consiste em construir um dispositivo móvel com geolocalização, reconhecimento de fala e sintetizador de texto para fala trabalhando totalmente off-line. O dispositivo visa auxiliar deficientes visuais informando os pontos de interesses que se encontram dentro do raio do seu percurso. Os objetivos específicos são: interpretar comandos de voz para iniciar, parar e desligar; armazenar nomes e coordenadas de Global Positioning System (GPS) dos pontos de interesses; informar os pontos de interesse que se encontram dentro do raio do percurso através de áudio via sintetizador de texto para voz; e trabalhar de maneira totalmente off-line.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentará os conceitos para o entendimento da pesquisa realizada. Na seção 2.1 é dado um breve resumo sobre como funciona o reconhecimento de fala. Na 2.2 é abordado o funcionamento do processo de sintetização de texto para fala. Na seção 2.3 é abordada a lei dos cossenos e como efetuar o cálculo. Na seção 2.4 é abordada a equação de Haversine e como calcula-la. Por fim, na seção 2.5 são listados e comentados os trabalhos correlatos a este projeto.

2.1 RECONHECIMENTO DE FALA

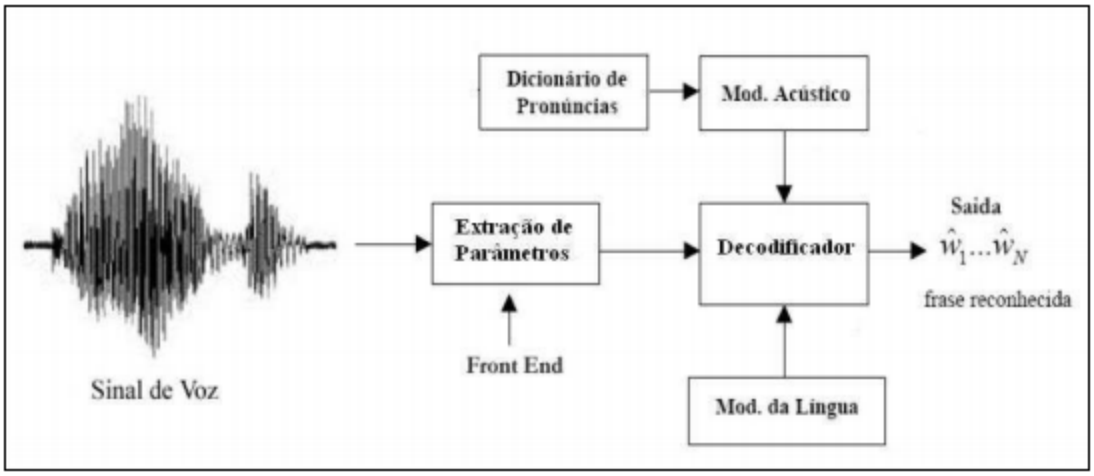
O reconhecimento de fala é o processo pelo qual um computador (ou outro tipo de máquina) identifica as palavras faladas. Basicamente, isso significa falar com uma máquina e ela reconhecer o que foi dito. As seguintes definições são as informações básicas necessárias para se entender tecnologias de reconhecimento de fala (COOK, 2002). Os sistemas de reconhecimentos de fala Automatic Speech Recognition (ASR) são separados em diferentes classes, onde descrevem os tipos de afirmações que têm a habilidade de reconhecer. Essas classes são baseadas no fato de que uma das dificuldades do ASR é a capacidade de determinar quando um orador começa e termina uma dicção. A maioria dos pacotes pode caber em mais de uma classe, dependendo do modo que eles estão sendo usados.

Cook (2002) descreve cinco tipos de reconhecedores de voz:

a) palavras isoladas: os reconhecedores de palavras isoladas normalmente exigem de cada elocução uma parada (falta de um sinal de áudio). Isso não significa que ele aceita uma única palavra, mas exige uma dicção de cada vez. Frequentemente, estes sistemas têm a chamada "Escuta/Não Escuta", em que é requerido que o orador tenha que esperar entre as pausas (normalmente fazendo tratamento durante estas). Expressão isolada pode ser um nome melhor para esta classe;
b) palavras conectadas: sistemas de palavras conectadas são semelhantes às palavras isoladas, mas permitem separar afirmações para serem rodadas em conjunto, com uma pausa mínima entre elas;
c) discurso contínuo: reconhecedores de fala contínua são alguns dos tipos de ASR mais difíceis de serem criados, por que precisam utilizar métodos especiais para determinar os limites da dicção. Esses reconhecedores permitem que os usuários falem quase naturalmente, enquanto o computador determina o conteúdo. Basicamente, é um ditado feito para o computador;
d) fala espontânea: parece haver uma variedade de definições para o que realmente se entende por fala espontânea. Em um nível básico, ela pode ser pensada como um discurso que ocorre naturalmente. Um sistema com ASR de fala espontânea deverá ter a capacidade de lidar com uma variedade de recursos naturais como palavras sendo executadas em conjunto, "ums" e "ahs", e até pequenas gagueiras;
e) verificação / identificação de voz: alguns sistemas de ASR têm a capacidade de identificar usuários específicos, visando assim sistemas mais específicos que tenham em vista a verificação de pessoas como sistema de segurança, por exemplo.

Como Silva (2010) explica em seu trabalho, um sistema de reconhecimento de voz é composto por vários blocos conforme mostrado na Figura 1. Dentre os quais há o *front-end* que é o responsável pela extração de parâmetros (*features*) do sinal de voz; o modelo acústico (MA) que busca modelar, a partir das *features*, o sinal acústico através de modelos matemáticos; o modelo de linguagem (ML) que a partir de textos da língua, tenta obter as possíveis sequências de palavras a serem reconhecidas; e o decodificador que, em conjunto com os blocos já citados, realiza o processo de transcrição do sinal de voz.

Figura 1 – Modelo sistema de reconhecimento de voz



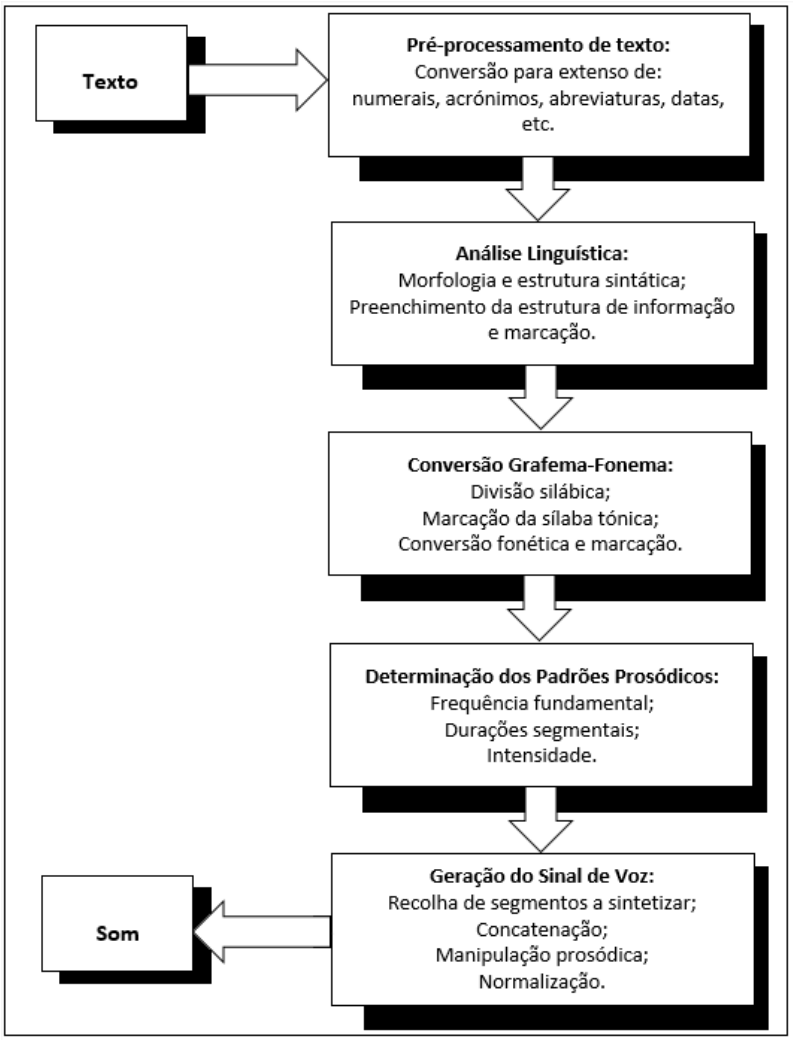
Fonte: Silva (2010).

2.2 TEXTO PARA FALA (TTS)

Os sistemas de conversão de texto-para-fala são sistemas que convertem automaticamente enunciados escritos em enunciados orais. Para que isto se torne possível tem de haver um processamento a vários níveis, desde a linguística à acústica da fala (BARROS, 2002). De acordo com Barros (2002) um sistema TTS é normalmente constituído por cinco módulos conforme mostrado na Figura 2: a) um módulo de pré-processamento de texto necessário para a conversão de qualquer tipo de grafismos, como por exemplo, abreviaturas, data, numerais, telefones, acrônimos, títulos

profissionais ou de gênero, etc; b) um módulo de análise linguística, responsável pelo estudo morfossintático do texto. Este módulo classifica as pontuações, as palavras e grupos de palavras, segundo as suas funções linguísticas, isto é, lexicais e gramaticais; c) um módulo de conversão grafema-fonema. É neste módulo que é realizada a conversão fonética, sendo necessário para isso um conjunto completo de regras, que dependendo da língua que se está a trabalhar é mais ou menos complexo. [...] neste módulo pode ainda ser feita uma divisão silábica, classificação e marcação de sílabas tónicas e átonas; d) um módulo de determinação de padrões prosódicos, essencial para obter naturalidade na produção da fala. Os padrões prosódicos são obtidos através da variação dos parâmetros acústicos da fala: frequência fundamental, durações e intensidade; e e) por fim, o último módulo constituído pela síntese do sinal da fala propriamente dita. Este módulo engloba não só a geração de sinal, como também a manipulação prosódica do sinal de acordo com os padrões obtidos no módulo anterior.

Figura 2 - Conceito de sistemas TSS, divido por módulos

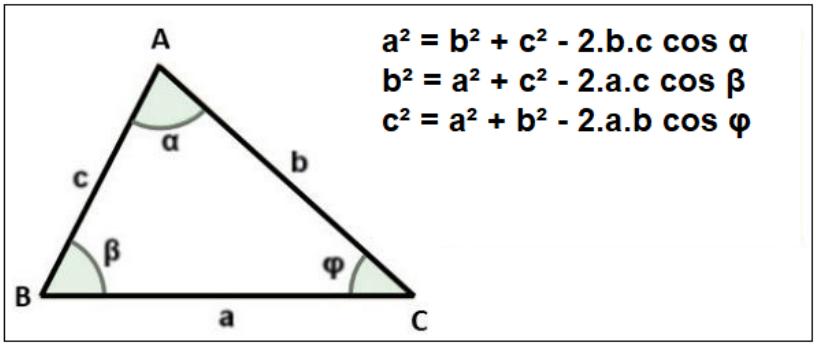


Fonte: Barros (2002).

2.3 LEI DOS COSSENOS

Antes de conhecer a equação de Haversine abordada na seção 2.4, é preciso entender a lei dos cossenos. Ela é utilizada para calcular a medida de um lado ou de um ângulo desconhecido de um triângulo qualquer, desde que conheça suas outras medidas (CAIUSCA, 2018, p. 1). O teorema estabelece que em qualquer triângulo, o quadrado de um dos lados corresponde à soma dos quadrados dos outros dois lados, menos o dobro do produto desses dois lados pelo cosseno do ângulo entre eles. De acordo com Caiusca (2018, p. 1) a lei dos cossenos propõe as seguintes relações entre os lados e os ângulos de um triângulo, conforme Figura 3.

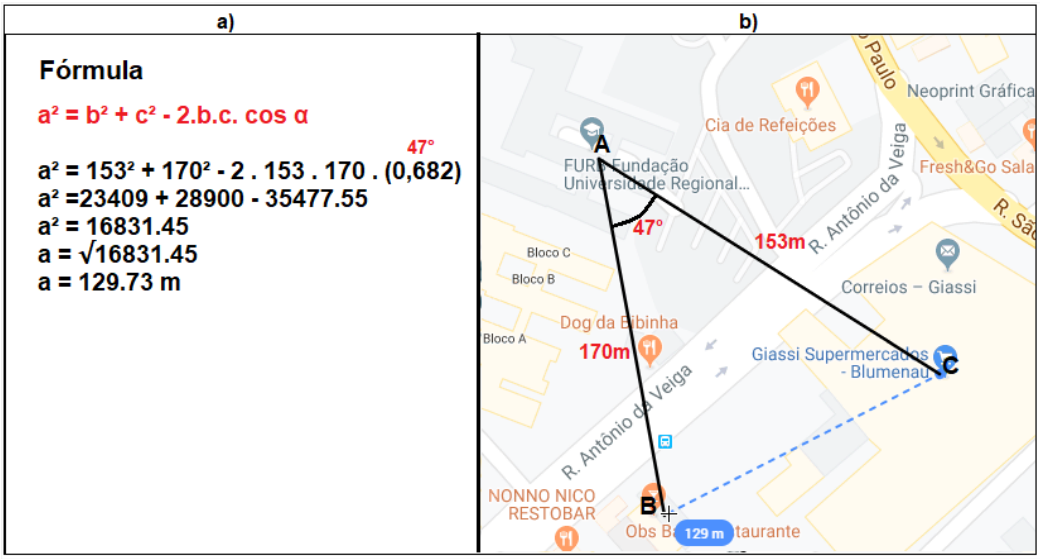
Figura 3 – Relações conforme lei dos cossenos



Fonte: Caiusca (2018, p. 1).

Para entender melhor como funciona a fórmula da lei dos cossenos serão utilizados exemplos do mundo real mostrados na Figura 4 (b). Sendo assim serão dados 3 pontos no mapa: A, B e C que formam um triângulo, porém são conhecidas apenas as distâncias de A para C (b), e de A para B (c), formando entre si um ângulo (cos α). Para calcular a distância desconhecida entre B e C será utilizado a lei dos cossenos. Para isso deve ser considerado que b = 153m, c = 170m e cos α = cos 47º = 0,682 (esse valor convertido de graus para decimais é encontrado em tabelas trigonométricas). Substituindo esses valores na fórmula e calculando conforme Figura 4 (a), será obtido a distância entre B e C conforme Figura 4 (b).

Figura 4 – Formula da lei dos cossenos



Fonte: elaborado pelo autor.

2.4 EQUAÇÃO DE HAVERSINE

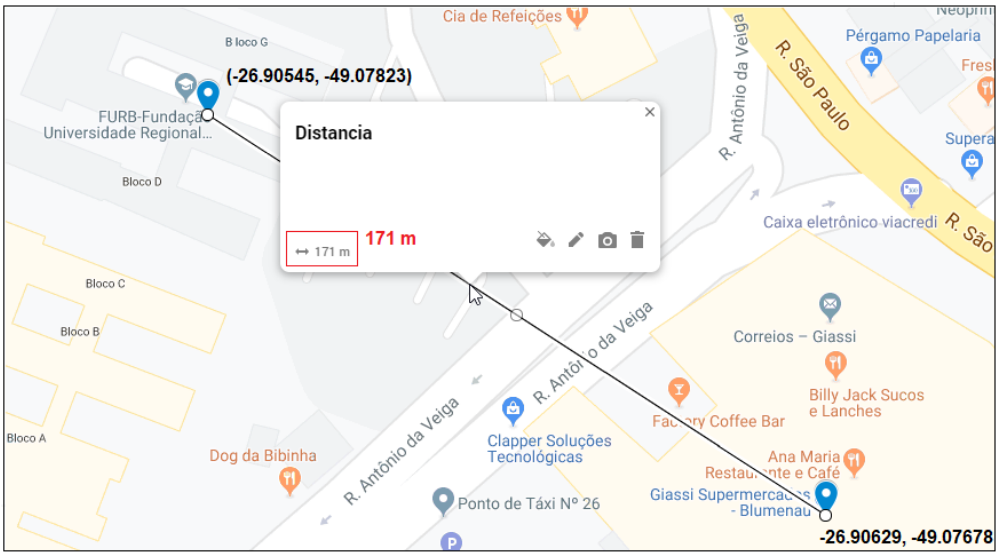
A equação de Haversine utiliza a função *haversin* para calcular a distância entre dois pontos na superfície de uma esfera (SINNOT, 1984). Como Sanches (2012) explica devido o formato da Terra levando em consideração o relevo do terreno, o cálculo da distância exata entre dois pontos em sua superfície não é simples. Segundo Hijmans et al. (2010), para ter uma aproximação simples e com precisão aceitável para se utilizar o cálculo da distância deve considerar o formato da Terra como sendo uma esfera perfeita e utilizar a equação de Haversine, proposta por R. W. Sinnott em 1984.

O raio da Terra é variável, nos polos é da ordem de 6357 km; enquanto no equador é de 6378 km. Sendo assim como o raio é variável, na equação de Haversine é utilizado o valor médio do raio que é 6371 km. Os cálculos começam a ficar imprecisos conforme se afasta da linha do equador. Caso necessitasse de uma precisão ainda maior o ideal seria utilizar a Fórmula de Vincenty, uma vez que ela leva em consideração o achatamento da Terra nos polos, a sua característica elíptica, garante uma precisão de 0.5mm. Contudo, devido seu peso computacional ser consideravelmente superior e mais complexa sua implementação, faz com que a comunidade opte pela fórmula de Haversine, uma vez que conforme explicado por Ribeiro Júnior et al. (2013), o erro médio é de apenas 0,3% nos resultados dos cálculos.

Harversine significa a metade do seno (*half versine*). Sendo assim, temos a seguinte relação: (1-cos (α)) /2 = sen (α /2) * sen (α /2). Para entender melhor como funciona a equação de Haversine serão utilizados exemplos do mundo real mostrados na Figura 5, com isso dadas duas coordenadas no mapa (-26.90545, -49.07823) e (-26.90629, -

49.07678), na Figura 6 mostra os passos para a criação e execução da equação de Haversine feita no Google Sheets, que retorna a distância de 171 metros entre as duas coordenadas.

Figura 5 – Distância entre duas coordenadas



Fonte: elaborado pelo autor.

Figura 6 – Equação de Haversine

	A	B	C	D
3	**Latitude 1**	-26,9054500	-0,4695886892	**"=(B3*pi())/180"**
4	**longitude 1**	-49,0782300	-0,8565767046	**"=(B4*pi())/180"**
5	**Latitude 2**	-26,9062900	-0,46960335	**"=(B5*pi())/180"**
6	**longitude 2**	-49,0767810	-0,8565514147	**"=(B6*pi())/180"**
7				
8		**Resultado**	**Formula**	
9	**dLat**	-0,0000073	**"=(RADIANS(B5-B3)/2)"**	diferença das latitudes dos pontos em radianos
10	**dLon**	0,0000126	**"=(RADIANS(B6-B4)/2)"**	diferença das longitudes dos pontos em radianos
11				
12	**R**	6371		O valor de R é o raio da Terra em KM.
13	**A**	0,0000000002	**"=SIN(B9)*SIN(B9)+COS(C3)*COS(C5)*SIN(B10)*SIN(B10)"**	O valor A é o quadrado da metade do arco entre os pontos.
14	**C**	0,000026898717	**"=2*ASIN(SQRT(B13))"**	O valor C é a distância em ângulos radianos encontrada.
15	**D**	171,3717303	**"=B12*B14*1000"**	O valor de D é a distância em metros entre as duas coordenadas.

Fonte: elaborado pelo autor.

2.5 TRABALHOS CORRELATOS

Esta seção apresenta trabalhos correlatos que possuem características e funcionalidades semelhantes ao que está sendo apresentado neste artigo. O primeiro (Quadro1) aborda um aplicativo móvel na plataforma Android, que visa ajudar na difusão de sinais regionais de libras (LINGNER, 2016). O segundo (Quadro 2) apresenta um aplicativo móvel como um módulo do Tagarela (REIS et al., 2014), que permita o desenvolvimento e aquisição de fala por crianças autistas (SAUTNER, 2017).

Quadro 1 – BLULIBRAS

Referência	Lingner (2016)
Objetivos	A autor descreve o BLULIBRAS como um aplicativo que visa ajudar na difusão de sinais regionais de LIBRAS.
Principais funcionalidades	O aplicativo permite o cadastro de sinais através do envio de imagem pela aplicação web, suportando o formato de imagem WebP e permite consultar e selecionar os novos sinais cadastrados.
Ferramentas de desenvolvimento	A aplicação web foi desenvolvida utilizando a linguagem PHP 5.6, juntamente com a ferramenta phpDesigner 8, no front-end tem como base o template AdminLTE que se baseia no framework Bootstrap. O aplicativo móvel foi desenvolvido utilizando a linguagem Java, através da Integrated Development Environment (IDE) Android Studio 2.1.2.
Resultados e conclusões	"O aplicativo foi bem aceito pelos professores de LIBRAS que consideram a possibilidade de usá-lo como ferramenta para auxiliar os alunos, bem como os próprios estudantes se mostraram interessados devido a facilidade de acesso a esse conteúdo". (LINGNER, 2016, p. 7).

Fonte: elaborado pelo autor.

A Figura 7 apresenta algumas telas do aplicativo BLULIBRAS, como a tela de boas-vindas e a tela para selecionar os sinais desejados.

Figura 7 – Aplicativo BLULIBRAS

Fonte: Adaptado de Lingner (2016, p. 50-51).

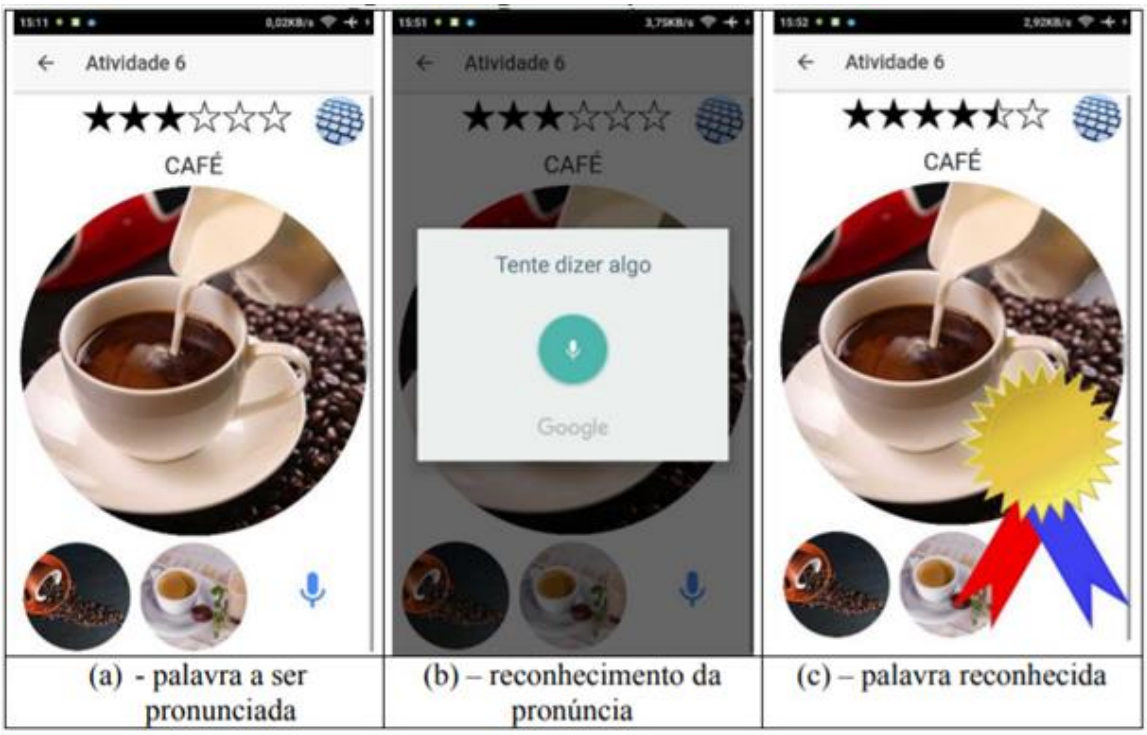
Quadro 2 – Tagarela

Referência	Sautner (2017)
Objetivos	A autor descreve o módulo do Tagarela como um aplicativo que permita o desenvolvimento e aquisição de fala por crianças autistas.
Principais funcionalidades	O aplicativo possui reconhecimento de voz e verifica se a palavra que foi dita condiz com a imagem apresentada ao usuário autista.
Ferramentas de desenvolvimento	A aplicação possui arquitetura cliente-servidor, foi implementada utilizando o *flamework* Ionic versão 3.3.0, que por sua vez faz uso do Angular na versão 4.1.3. Foi utilizado a plataforma para desenvolvimento Firebase, que fornece um conjunto de APIs e produtos voltados ao desenvolvimento de aplicativos (web, Android, Ios, c++, Unity).
Resultados e conclusões	Conforme explicado por Sautner (2017) o trabalho "apresentou um aplicativo que permite auxiliar o desenvolvimento e aquisição de linguagem para crianças autistas. Visto que a criança com autismo possui características próprias, a capacidade de personalização e individualização das atividades é um recurso fundamental no processo de desenvolvimento". (SAUTNER, 2017, p. 46). Conclui que o trabalho atendeu o objetivo de fornecer um aplicativo para o desenvolvimento e aquisição de fala por crianças autistas.

Fonte: elaborado pelo autor.

A Figura 8 apresenta algumas telas do aplicativo Tagarela desenvolvido por Sautner (2017). O processo de reconhecimento é iniciado após ser pressionado o botão com o microfone, conforme apresentado na Figura 8 (b). Se o aplicativo reconhecer a palavra pronunciada, um indicativo visual é apresentado (Figura 8 (c)) e, após alguns segundos, a próxima palavra da atividade é apresentada.

Figura 8 – Jogo: realização de atividade



Fonte: Sautner (2017).

3 DESCRIÇÃO DO PROTÓTIPO

Esta seção apresenta os detalhes de hardware, implementação e descrição do funcionamento do protótipo. Na seção 3.1 é explicado o hardware e a arquitetura do dispositivo. Na seção 3.2 os detalhes da criação e impressão do case para proteger o dispositivo. Por fim, na seção 3.3 são apresentados todos os detalhes da implementação e comportamentos do dispositivo.

3.1 HARDWARE E ARQUITETURA

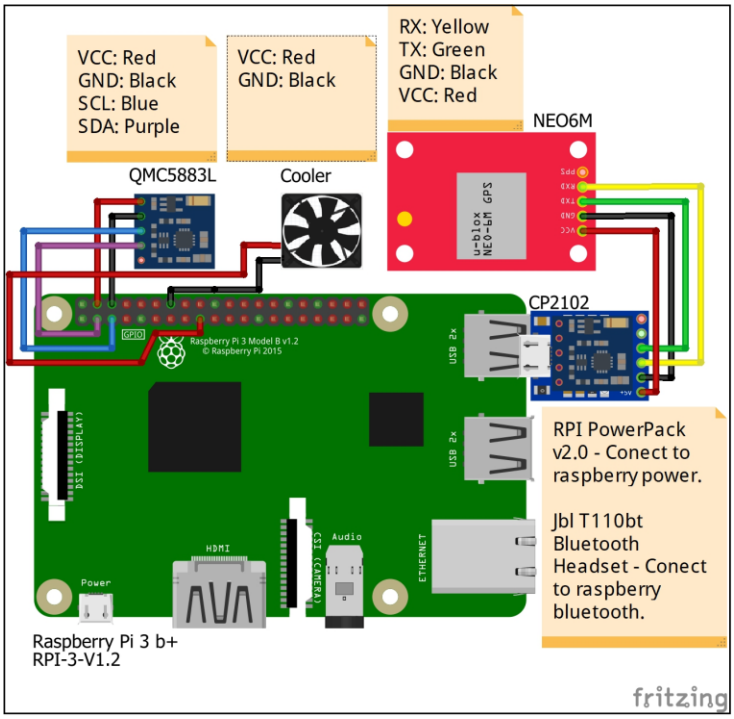
O Quadro 3 explica os detalhes de cada componente do protótipo apresentado na Figura 9. A Figura 9 apresenta o esquema de montagem.

Quadro 3 – Detalhes dos componentes do protótipo

NOME	COMPONENTE	ESPECIFICAÇÃO
Raspberry Pi 3 B	Módulo principal	Um computador de baixo custo com tamanho equivalente a um cartão de crédito
QMC5883L	Bússola/Magnetômetro	Funciona como bússola eletrônica para detectar o norte magnético da Terra, possui um sensor magnético de 3 eixos X, Y e Z.
NEO6MV2	GPS	Obtém dados de satélite como data, hora, latitude, longitude, altitude e velocidade.
CP2102	Conversor USB TO UART	Um circuito integrado compacto que fornece uma solução simples para conversão USB-UART.
RPI PowerPack v2.0	Módulo de bateria	Placa de expansão responsável para fornecer alimentação de energia para Raspberry Pi 3 B, através de bateria de lítio.
Jbl T110bt Bluetooth Headset	Microfone/fone bluetooth	Headset bluetooth com microfone/fone responsável por receber e enviar audios para Raspberry Pi 3 B.
Cooler 30x30 3.3v	Cooler	Responsável por manter a temperatura do processador em um nível aceitável.

Fonte: elaborado pelo autor.

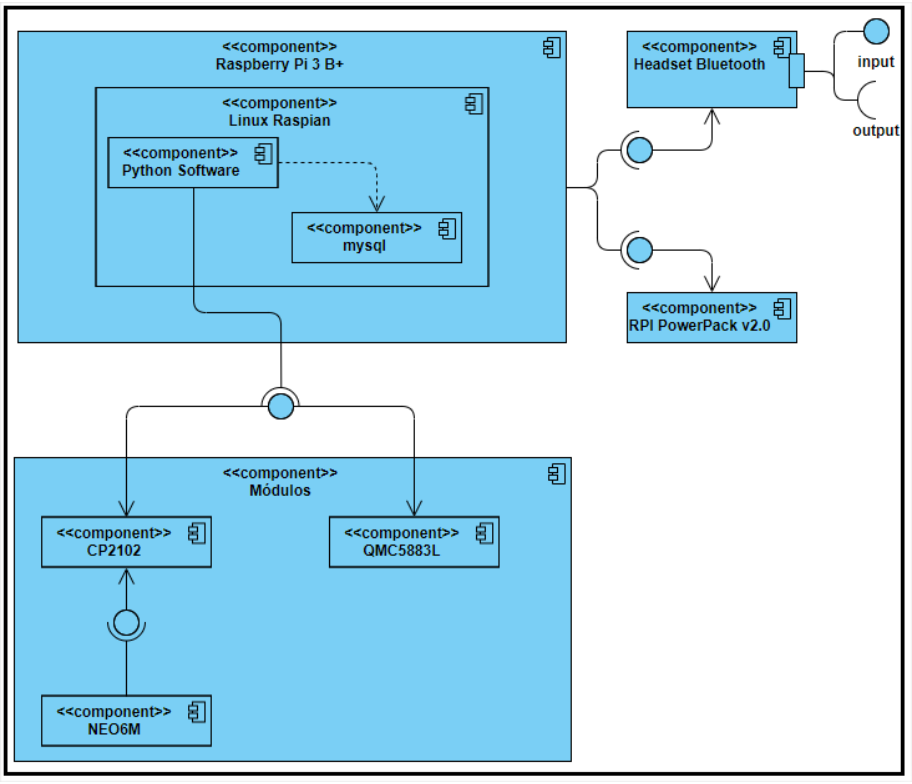
Figura 9 – Esquema de montagem



Fonte: elaborado pelo autor.

A arquitetura foi montada conforme a representação no diagrama de componentes na Figura 10. O Raspberry Pi 3 B é ligado através da energia fornecida pela bateria de lítio da placa de expansão RPI PowerBack v2.0 e recebe (*input)*, os comandos via *Headset* para iniciar, parar e desligar a aplicação. Ao receber o comando de ligar, a aplicação verifica a posição atual do usuário (latitude e longitude), através do módulo de GPS NEO6M, que envia os dados via porta Universal Serial Bus (USB) graças ao conversor CP2102 e verifica também a direção através do módulo de bússola QMC5883L. A aplicação por sua vez, com os dados de posição e direção do usuário, processa os cálculos consultando o banco de dados MySql e retorna (*output*) via *Headset* os pontos de interesse num raio de 50 metros.

Figura 10 – Arquitetura

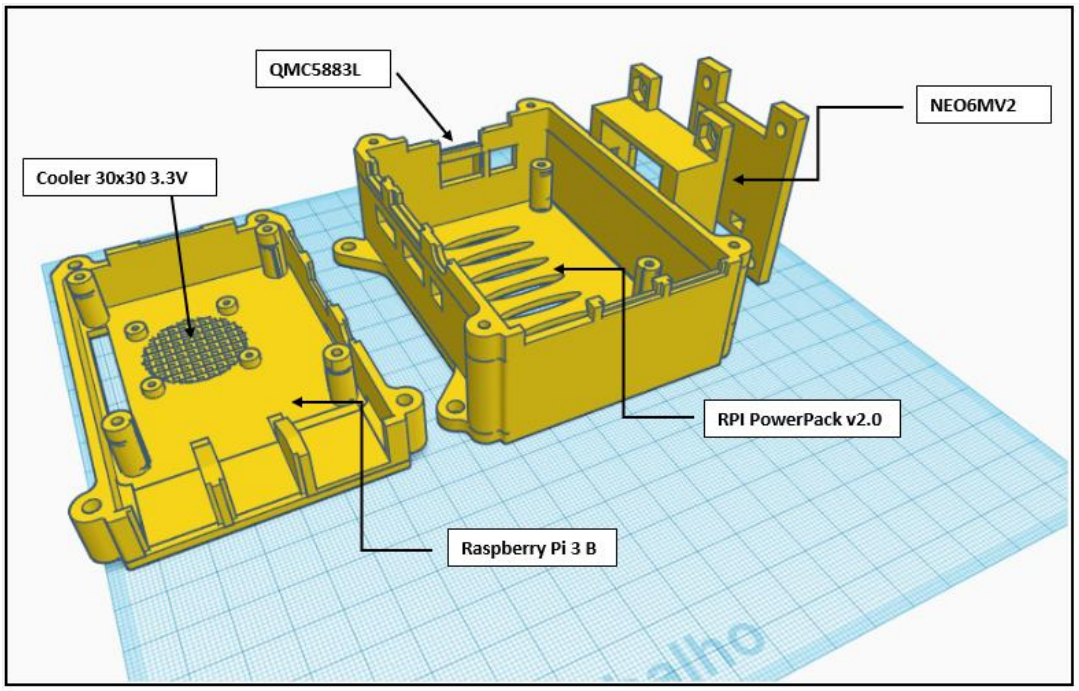


Fonte: elaborado pelo autor.

3.2 CRIAÇÃO DO CASE

Na criação do case para o protótipo foi utilizado a plataforma TinkerCad (TINKERCAD, 2019). O case foi criado para comportar todos os componentes do protótipo conforme Figura 11. Foi necessário modelar o case em 3 partes para facilitar a sua impressão 3D e sua exportação foi em formato Standard Triangle Language (STL), esse formato é aceito pelas maiorias das impressoras 3D do mercado atualmente. Os arquivos com extensão STL estão disponíveis em: https://github.com/wiiulopes/TCC-Furb.

Figura 11 – Modelagem do Case



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

Esta seção apresenta o desenvolvimento do dispositivo móvel off-line para auxílio na locomoção de deficientes visuais que utiliza uma plataforma baseada em Raspberry Pi, sendo este o foco desse artigo. Os Requisitos Funcionais (RF) do dispositivo estão apresentados no Quadro 4 e os Requisitos Não Funcionais (RNF) estão apresentados no Quadro 5.

Quadro 4 – Requisitos Funcionais do dispositivo

Requisitos Funcionais
RF01: permitir cadastros de novos pontos de interesses via arquivo Comma Separated Values (CSV)
RF02: permitir iniciar, parar e desligar o dispositivo via comandos de voz
RF03: permitir que o usuário escute via fone Bluetooth a distância dos pontos de interesses no raio do seu percurso
RF04: permitir que o usuário escute via fone Bluetooth a direção dos pontos de interesses no raio do seu percurso

Fonte: elaborado pelo autor.

Quadro 5 – Requisitos Não Funcionais do dispositivo

Requisitos Não Funcionais
RNF01: utilizar a placa Raspberry Pi 3 B como plataforma de desenvolvimento
RNF02: utilizar a linguagem de programação Python para o desenvolvimento do *Backend*
RNF03: utilizar o banco de dados MySQL
RNF04: funcionar no sistema operacional Linux Raspian

Fonte: elaborado pelo autor.

Para a criação do *backend* do protótipo, foi utilizada a linguagem Python na sua versão 3.6 no ambiente de desenvolvimento Pycharm Edu. Para persistência de dados foi utilizado o banco de dados relacional MySQL na versão 5.1. O desenvolvimento foi divido em 4 partes: a) o reconhecimento de fala ASR; b) o sintetizador de texto para fala TTS; c) os cálculos que buscam a distância e direção dos pontos de interesses; d) a classe principal responsável pelas validações ao iniciar o dispositivo e permite com que os módulos trabalhem de formar paralela compartilhando memória usando *Multiprocessing*.

No desenvolvimento do reconhecimento de fala foi utilizado o pacote de ferramentas CMUSphinx (MELLON, 2019), no idioma inglês-americano, devido não ter suporte para o idioma português-brasil. Como Naslausky (2010) explica em seu trabalho, este pacote fornece um sistema de reconhecimento de voz fazendo uso das cadeias ocultas de Markov e com processamento local, isto é, sem a necessidade de uma conexão à Internet. O pacote de ferramentas CMUSphinx fornece o PocketSphinx que é uns dos seus pacotes que foca no reconhecimento de voz e o mesmo foi utilizado nesse trabalho. Foi necessário instalar também o pacote `SpeechRecognition`, que facilita o reconhecimento de áudio enviado via microfone e esse pacote aceita como Application Programming Interface (API) o pacote PocketSphinx para o Python. Ambos os pacotes mencionados necessitam de instalação no Linux, o passo a passo de como instalar e configurar está disponível em: https://github.com/wiiulopes/TCC-Furb.

A classe principal do dispositivo `run.py` possui a função `task_speech` conforme Quadro 6, responsável por reconhecer a fala e gerenciar os módulos do dispositivo conforme os comandos do usuário. A classe `Recognizer` importada do pacote `SpeechRecognition` na linha 14 do Quadro 6 é fundamental, pois é por meio de instancias desta classe que as configurações e funcionalidades são ajustadas conforme linha 52 do Quadro 6 ao utilizar a função `sr.Recognizer`. O primeiro passo ao utilizar essa classe é informar onde ela irá obter o áudio. Como o dispositivo irá obter através do microfone, foi utilizado o método `listen` da classe `Recognizer` na linha 58 do Quadro 6. Na linha 63 do Quadro 6 é chamado o método da API de reconhecimento de fala que o pacote `SpeechRecognition` suporta, como por exemplo o `recognize_google` (necessita conexão), ou, no caso desse trabalho, `recognize_sphinx`.

O método `recognize_sphinx` recebe como parâmetro o idioma (por padrão inglês), o áudio obtido, as palavras chaves e o arquivo de gramática. Ao adicionar as palavras chaves recebidas na variável `keywords`, faz com que o mesmo arbitre a probabilidade de certas expressões e combinações ditas, privilegiando palavras mais comuns. No caso desse trabalho é útil, uma vez que ele receberá apenas 3 comandos do usuário, sendo eles: `lets go`, `stop` e `offline`. Ao adicionar um arquivo de gramática com formato Java Speech Grammar Format (JSGF), possibilita a criação de regras através de Backus-Naur Form (BNF), para dado idioma. No caso desse trabalho foi criado o arquivo `Grammar.jsfg` conforme o Quadro 7, contendo regras que o usuário irá falar `lets go`, `stop` ou `off-line`. A adição do arquivo JSGF em conjunto com as `keywords`, foi de grande importância para esse trabalho melhorando muito o reconhecimento dos comandos ditos pelo usuário. Após o processamento o método `recognize_sphinx` irá retornar a palavra dita na variável `word` na linha 63 do Quadro 6.

Quadro 6 - Método de reconhecimento de fala

```
14  import speech_recognition as sr
27  keywords = [("lets go", 1), ("stop", 1), ("offline", 1), ]
47  def task_speech():
48      global task_main, task_black, keywords, tts_on
49      print("Iniciando reconhecimento de voz")
50      play_audio_tts("start_speech")
51      print("Thread: Microfone")
52      microfone = sr.Recognizer()
53
54      while task_main:
55          if not tts_on.value:
56              with sr.Microphone() as source:
57                  microfone.adjust_for_ambient_noise(source)
58                  print("Diga alguma coisa: ")
59                  audio = microfone.listen(source)
60              try:
61                  # Passa o audio para o reconhecedor de padroes do speech_recognition
62                  # utilizando a API PocketSphinx(recognize_sphinx)
63                  word = microfone.recognize_sphinx(audio,
64                                                    keyword_entries=keywords,
65                                                    grammar='Grammar.jsgf')
66                  print("Voce disse:", frase)
```

Fonte: elaborado pelo autor.

Quadro 7 - Arquivo de gramática

```
#JSGF V1.0;

/**
 * JSGF Grammar for English
 */

grammar counting;

public <counting> = ( <say> ) +;

<say> = start | stop | offline ;
```

Fonte: elaborado pelo autor.

Para o sintetizador de texto para fala TTS, foi necessário utilizar o pacote Google Text-to-Speech (gTTS), que funciona apenas com conexão com a Internet. Sua utilização foi necessária pois ao testar em campo, os pacotes de sintetizador de texto para fala que funcionavam de forma off-line o áudio sintetizado era de difícil compreensão devido ao baixo poder de processamento local da Raspberry Pi 3 B. A solução para esse problema foi definida em duas etapas: a) a função `new_audio_point_interest` da classe `new_points.py` de cadastros pontos de interesses, ao inserir um novo ponto de interesse no banco de dados, envia o `id` e `nome` do novo ponto de interesse para a função TTS `save_audio_tts` da classe `tts.py`; b) a função `save_audio_tts` da classe `tts.py` ao receber o `id` e `nome` do ponto de interesse envia o `nome` para o sintetizador de texto para fala na nuvem da Google e retorna o áudio sintetizado no idioma português-brasileiro e salva em um arquivo MPEG Layer 3 (MP3), com a nomenclatura `id_nome.mp3` conforme Quadro 8. Ambos os módulos só funcionam se o dispositivo estiver conectado com a internet. Com essa solução os áudios dos nomes dos pontos de interesse cadastrado no banco relacional, ficam disponíveis de forma off-line para serem reproduzidos ao usuário com uma ótima qualidade. Utilizando esse método também foram criados os áudios que definem a direção do ponto de interesse e das distâncias que vão de 1 a 50 metros, essa distância foi definida através dos testes em campo, pois foi percebido que distâncias maiores de 50 metros ficavam fora do campo de visão humano.

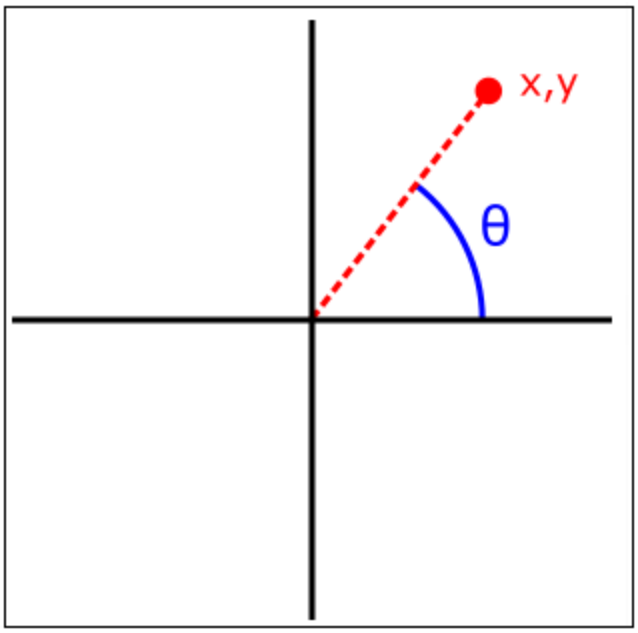
Quadro 8 - Função do sintetizador de texto para fala gTTS

```
def save_audio_tts(id, ponto_interesse):
    tts = gTTS(text=ponto_interesse, lang='pt-br')
    # Replace all runs of whitespace with a single dash
    ponto_interesse = re.sub(r"\s+", '_', ponto_interesse)
    # Save the audio file
    tts.save('/Users/william.silva/gttsAudios/' + str(id) + '_' + ponto_interesse + '.mp3')
```

Fonte: elaborado pelo autor.

Para determinar a direção em graus entre duas coordenadas (usuário e ponto de interesse), conforme linha 115 no Quadro 9, na função `calculate_initial_compass_bearing` da classe `actions.py` foi utilizada a fórmula `atan2` muito conhecida na área da trigonometria que é herdada do módulo matemático `Math` do Python, conforme explicado em OFFICE (2019). Essa formula retorna o arco tangente, ou a tangente inversa, das coordenadas x e y informadas, sendo o arco tangente o ângulo entre o eixo x e uma linha que contém a origem (0,0) e um ponto com coordenadas (x, y) conforme Figura 12. O ângulo nessa fórmula é retornado em radiano. Desta forma foi utilizado a formula `degress` para converter em graus. Na fórmula `atan2` um resultado positivo representa um ângulo no sentido anti-horário do eixo x, um resultado negativo representa um ângulo no sentido horário, como só será trabalhado ângulo positivos de 0º a 360º. Desta forma quando a formula `atan2` retorna um angulo negativo, é aplicado a fórmula apresentada na linha 119 do Quadro 9, que o converte esse ângulo para positivo, uma vez que utilizando como exemplo -90º equivale a 270º.

Figura 12 – Ângulo retornado por atan2



Fonte: elaborado pelo autor.

Quadro 9 - Função que calcula o ângulo entre dois pontos

```
113  def calculate_initial_compass_bearing(pointA, pointB):
114      startx, starty, endx, endy = pointA[0], pointA[1], pointB[0], pointB[1]
115      angle = math.atan2(endy - starty, endx - startx)
116      if angle >= 0:
117          return math.degrees(angle)
118      else:
119          return math.degrees((angle + 2 * math.pi))
```

Fonte: elaborado pelo autor.

Como a fórmula `atan2` leva em consideração que o ponto origem (x), está sempre fixo em direção ao norte ou 0º, foi necessário elaborar um método para levar em consideração o grau atual da bússola QMC5883, conforme demonstrado na Figura 13, uma vez que esse grau é oscilante de 0º a 360º, pois determina a direção do usuário.

Figura 13 – Método para aceitar grau oscilante da bússola

Ajustar grau quando direção do usuário é oscilante:
Quando os **grau da direção do usuário (A)** for menor que o **grau da direção até o segundo ponto(B)** faça:
D = B - A

Quando os **grau da direção do usuário (A)** for maior que o **grau da direção até o segundo ponto(B)** faça:
C = A - B
D = 360 - C

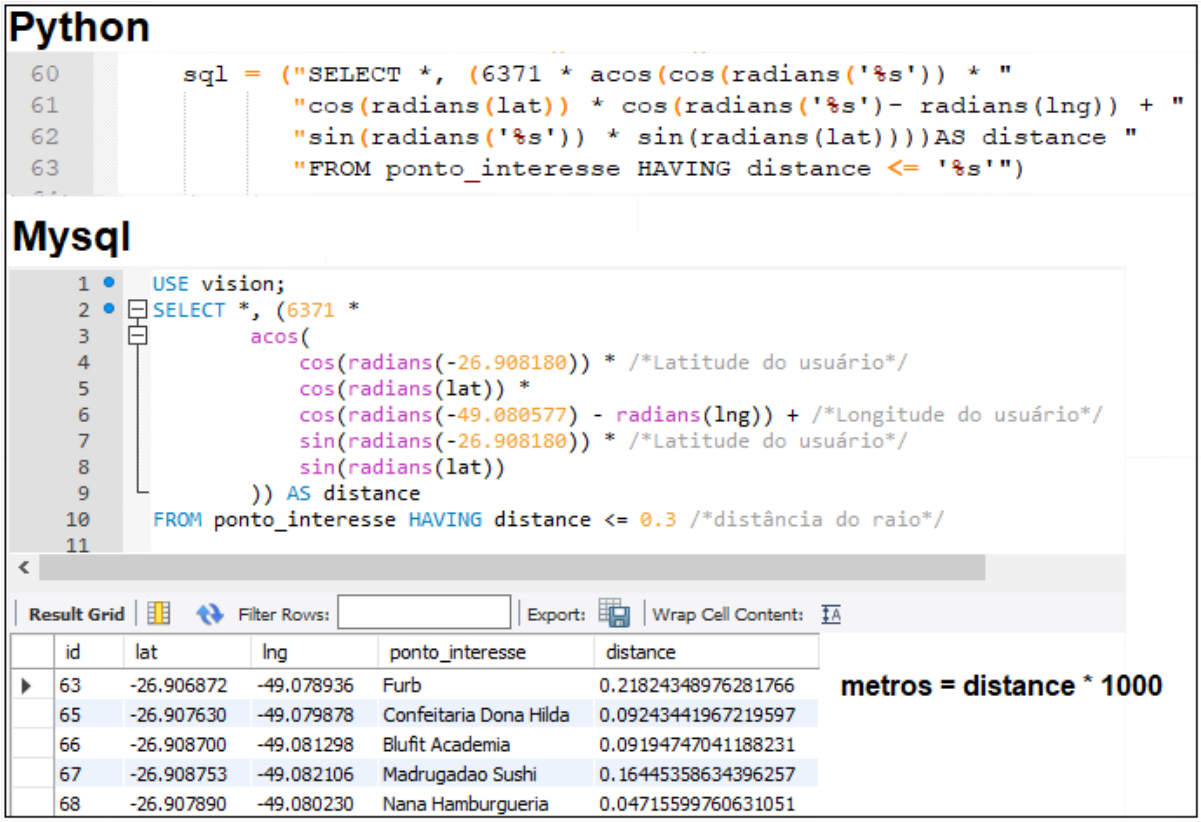
```
75          if currentBearing < finishBearingPosition:
76              resultBearing = finishBearingPosition - currentBearing
77          else:
78              auxBearing = currentBearing - finishBearingPosition
79              resultBearing = 360 - auxBearing
```

Fonte: elaborado pelo autor.

Para determinar a distância entre o usuário e os pontos de interesses, foi utilizado a fórmula Haversine que consta todos os detalhes da sua implementação na seção 2.4. Nesse trabalho optou-se em utilizar um `SELECT` do Mysql constando a fórmula do Haversine ao invés do módulo `Math` conforme Figura 14, pois o `SELECT` percorre e aplica a

função de Haversine em uma requisição, retornando as distancias de todos pontos de interesses cadastrados no banco de dados que estão dentro de um raio pré-definido, enquanto que na função do Math seria necessário percorrer o banco e aplicar separadamente a fórmula de Harvesine para cada ponto de interesse encontrado dentro do raio, exigindo assim várias requisições.

Figura 14 - Equação de Haversine utilizando select do MySql



Fonte: elaborado pelo autor.

Por fim a classe principal foi desenvolvida de forma que todos as funções mencionadas, funcionem ao mesmo tempo de forma individual compartilhando memória, usando *multiprocessing*. Não foi possível utilizar *threading* pois esse módulo usa *threads*, e o modulo *multiprocessing* usa processos. A diferença é que as *threads* são executadas no mesmo espaço de memória, enquanto os processos possuem memória separadas. Com isso usando *threads* torna um pouco mais difícil compartilhar objetos entre processos com multiprocessamento, impossibilitando assim o paralelismo. Os módulos precisam enviar comandos entre si, porém ao utilizar *multiprocessing* eles são criados em memórias separadas, impossibilidade a troca de informação. Para isso o módulo *multiprocessing* oferece o método `multiprocessing.Value(tipo, valor)` demonstrando na linha 23, 24 e 25 do Quadro 10, que possibilita iniciar variáveis globais sincronizadas em memórias separadas. E ao utilizar o método multiprocessing.Lock() demonstrado na linha 26 do Quadro 10, garante que apenas uma função possa atualizar as variáveis compartilhadas na memória.

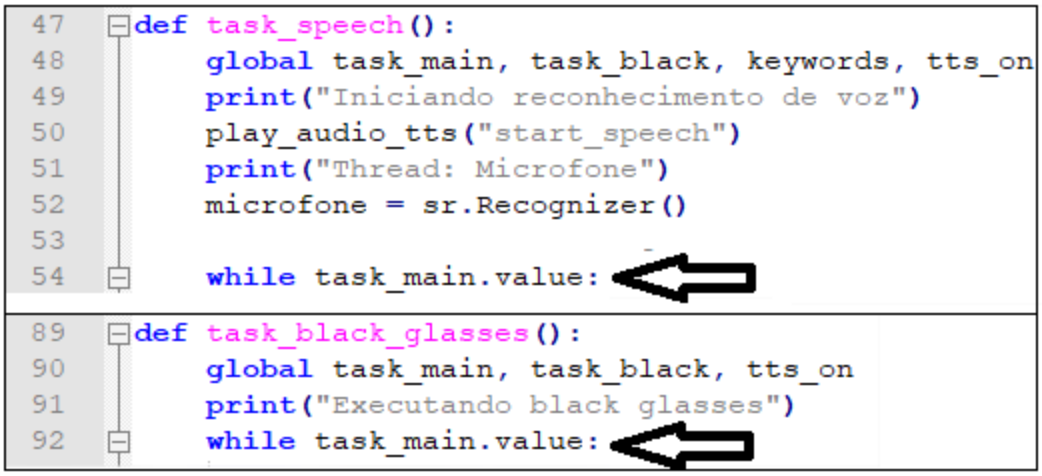
Quadro 10 - Variáveis globais utilizando Multiprocessing

```
23  task_black = multiprocessing.Value(ctypes.c_bool, False)  # (type, init value)
24  task_main = multiprocessing.Value(ctypes.c_bool, True)  # (type, init value)
25  tts_on = multiprocessing.Value(ctypes.c_bool, False)  # (type, init value)
26  lock = multiprocessing.Manager().Lock()
```

Fonte: elaborado pelo autor.

As duas principais funções do dispositivo são a de reconhecimento de voz (`task_speech`), e de busca de distância e direção dos pontos de interesses (`task_black_glasses`), são iniciadas pela função `validation_task` somente após efetuar todas as validações que garantem a conexão com o banco, sinal do GPS e bússola. Essas duas funções conforme Figura 15, são iniciadas em paralelo e ficam executando dentro de um *while* que valida o estado da variável compartilhada em memória `task_main` inicializada com valor `True` conforme Quadro 10.

Figura 15 - Inicialização das funções com variável compartilhada



```
47  def task_speech():
48      global task_main, task_black, keywords, tts_on
49      print("Iniciando reconhecimento de voz")
50      play_audio_tts("start_speech")
51      print("Thread: Microfone")
52      microfone = sr.Recognizer()
53
54      while task_main.value:
```

```
89  def task_black_glasses():
90      global task_main, task_black, tts_on
91      print("Executando black glasses")
92      while task_main.value:
```

Fonte: elaborado pelo autor.

A função `task_speech` aguarda o comando de voz enviado pelo usuário e é responsável por gerenciar todo comportamento do dispositivo, como é visto no Quadro 11. Após receber o comando `lets go` na linha 65, é chamado a função `play_audio_tts` na linha 66, que irá reproduzir o áudio informando ao usuário que a aplicação está iniciando. Com isso, na linha 67 chama o método `lock` que garante que apenas a função `task_speech` possa atualizar a variável compartilhada na memória `task_black` que recebera o valor `True` conforme linha 68. A variável `task_black`, conforme linha 93 do Quadro 11, é responsável pela validação. Que caso seja `True`, chama a função `distanciaDirecao` que irá buscar a distância e direção dos pontos de interesse no raio do usuário. Essa variável por padrão é iniciada com valor `False`, e seu comportamento e determinado pelos comandos do usuário sendo eles : a) `lets go`: responsável por iniciar a função que busca a distância e direção dos pontos de interesse dentro do raio do usuário e é executada a cada 30 segundos caso não receba o comando para parar; b) `pause`: responsável por parar a aplicação e voltar ao seu estado inicial aguardando novos comandos; e c) `offline`: responsável por parar a aplicação e desligar o dispositivo.

Quadro 11 - Gerenciamento do dispositivo

```
61      # Passa o audio para o reconhecedor de padroes do speech_recognition
62      frase = microfone.recognize_sphinx(audio, keyword_entries=keywords, grammar='Grammar.jsgf')
63      print("Voce disse:", frase)
64      # Após alguns segundos, retorna a frase falada
65      if "lets go" in frase:
66          play_audio_tts("start_black")
67          with lock:
68              task_black.value = True
69              tts_on.value = True
70          print("Recebido o comando para iniciar a Aplicação: ", frase)

89  def task_black_glasses():
90      global task_main, task_black, tts_on
91      print("Executando black glasses")
92      while task_main.value:
93          if task_black.value:
94              with lock:
95                  tts_on.value = True
96              pontosInteresses = distanciaDirecao()
97              play_audio_tts(pontosInteresses)
98              with lock:
99                  tts_on.value = False
100             time.sleep(30)
101         else:
102             print("Black glasses aguardando comando para iniciar!")
103             time.sleep(5)
```

Fonte: elaborado pelo autor.

4 RESULTADOS

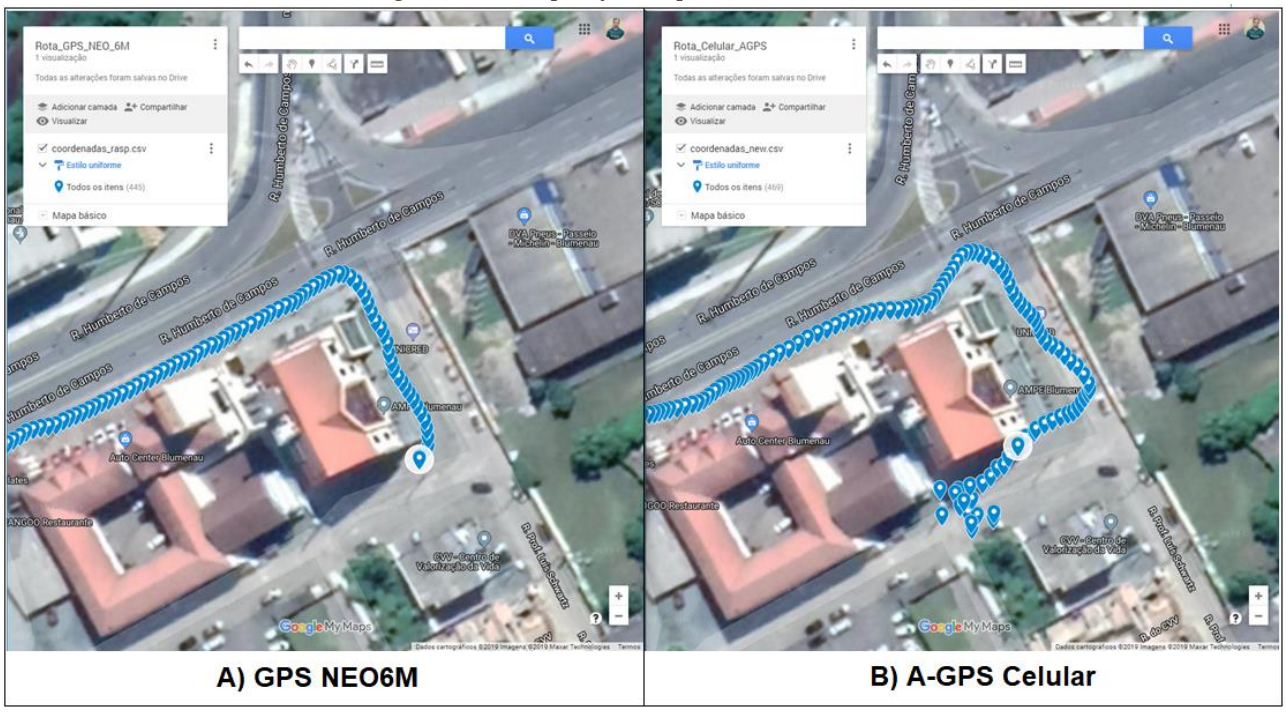
Esta seção apresenta os testes do dispositivo, desafios e resultados. Na seção 4.1 detalha os testes e desafios encontrados durante a implementação do projeto e na seção 4.2 os resultados obtidos após finalizar o protótipo.

4.1 TESTES E DESAFIOS

Para garantir segurança na utilização do dispositivo, foram feitas várias baterias de testes em campo, que proporcionaram novas ideias e consequentemente desafios foram encontrados no decorrer do desenvolvimento do projeto. O primeiro deles foi que inicialmente o dispositivo iria apenas informar ao usuário as distâncias e direções dos pontos de interesses via fone de ouvido conectado ao Tip-Ring-Sleeve (TRS) de tamanho P2 (3,5 mm) da Raspberry PI 3 B, porém foi visto que seria necessário que o usuário com dificuldade visual deveria ter um controle maior do dispositivo, sendo assim foi desenvolvido o módulo de reconhecimento de fala off-line onde o usuário envia comandos ao dispositivo para iniciar, parar e desligar. Como a entrada P2 da Raspberry PI 3 B não possuí entrada de áudio via microfone foi utilizado um *headset bluetooth*. Dessa forma, foi necessário utilizar o *bluetooth* nativo da Raspberry PI 3 B. Porém, ao utilizar o *bluetooth* nativo foi encontrado um desafio a nível de hardware na Raspberry PI 3 B. O módulo de GPS NEO6M e o *bluetooth* utilizam ambos os mesmo pinos 8 e 10 da Raspberry PI 3 B do Universal Asynchronous Receiver / Transmitter (UART), que estão disponíveis nos General Purpose Input/Output (GPIO) 14 e 15. Conforme Apêndice A. Como só é possível usar apenas um UART por vez nesses pinos, foi necessário comprar o conversor CP2102 para utilizar o GPS NEO6M na porta USB e o *bluetooth* se manter utilizando o UART.

O segundo desafio estava em testar a eficácia da precisão do módulo de GPS NEO6M, visto que não foram encontradas nenhuma pesquisa na comunidade comparando o GPS assistido (A-GPS), que consta em aparelho celulares e o chip GPS NEO6M. Foi desenvolvido um aplicativo Android para o celular e em Python para a Raspberry/NEO6M, que foram iniciados ao mesmo tempo para coletar as coordenadas em céu aberto em uma determinada rota. Com os dados das coordenadas coletadas de ambos dispositivos foram importadas para o aplicativo MyMaps do Google (GOOGLE, 2019), para comparar a diferença da precisão. E chegou-se à conclusão que o GPS NEO6M se manteve mais preciso como é possível ver na Figura 16, podendo assim ser utilizado com segurança nesse projeto.

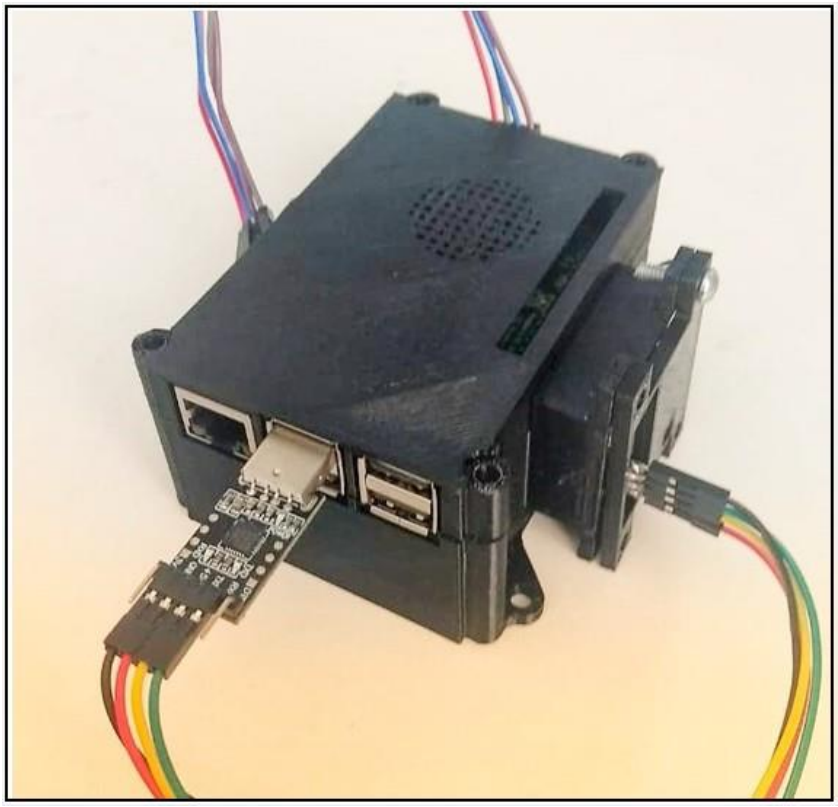
Figura 16 – Comparação de precisão GPS e A-GPS



Fonte: elaborado pelo autor.

O terceiro e último desafio foi a construção e impressão do case para proteger o protótipo, para isso foi dedicado um período de estudo da plataforma gratuita TinkerCad (TINKERCAD, 2019). Isso se fez necessário, pois os cases disponíveis na comunidade não atendiam a proteção do protótipo desenvolvido nesse projeto, uma vez que além da Raspberry PI 3 B, ele deveria comportar todos demais componentes. Após 4 dias de modelagem foi concluído o case para atender a proteção total dos componentes (como pode ser visto na seção 3.2) e enviado para uma empresa terceira fazer sua impressão 3D. O resultado do protótipo e case finalizados podem ser vistos na Figura 17.

Figura 17 – Case impresso em impressora 3D



Fonte: elaborado pelo autor.

4.2 RESULTADOS

Ao efetuar os testes do dispositivo em campo foi preciso dizer várias vezes os comandos (`lets go`, `pause` ou `offline`) para o sintetizador de fala conseguir compreender o que foi dito. Esse problema ocorre por causa dos ruídos externo que dificultam o reconhecimento de fala, porém o comportamento do dispositivo em trabalhar de forma paralela sempre aguardando comandos e retornando em áudio as distâncias e direções do ponto de interesses via *headset bluetooth* após receber o comando de iniciar, foram satisfatórios. O case protegeu bem os componentes e não impactou no funcionamento dos mesmos. A rotina de cadastro de novos pontos de interesses validou duplicidade e não teve problema de integração com o módulo conversor de texto para fala. O protótipo trabalhou de forma totalmente off-line, não necessitando conexão com a internet para operar.

5 CONCLUSÕES

Diante dos resultados apresentados conclui-se que o projeto atingiu os objetivos pretendidos, mesmo limitado a sua capacidade de hardware e de operar sem conexão com a internet. A escolha da plataforma Raspberry PI 3 B ao invés de versões mais fracas ou até Arduino foi fundamental, pois foi visto que para esse protótipo foi necessário trabalhar com multiprocessos compartilhando memória, utilizando módulos externos e cálculos, exigindo um hardware com processadores melhores como no caso da Raspberry PI 3 B. O retorno para o usuário da distância e direção dos pontos de interesses via *headset bluetooth* se mostrou satisfatórios. A criação do módulo de reconhecimento de fala foi de grande importância para o protótipo, tornando a acessibilidade do deficiente visual com o dispositivo muito mais prática. A resposta dos módulos de GPS NEO6M e bússola QMC5883L trabalhando de forma off-line foram muito precisas, sendo assim provou-se que é viável a criação de projetos de tecnologias assistivas para áreas sem cobertura de internet. Outro ponto a ser mencionado é que se somando a todas componentes utilizados no projeto o custo para criação desse protótipo se mostra muito acessível conforme Apêndice B.

Apesar dos resultados satisfatórios obtidos e apresentados, os testes encontraram alguns pontos que podem prejudicar a experiência de acessibilidade do usuário com deficiência visual. O primeiro se trata da precisão do reconhecimento de fala que se mostrou não satisfatória por consequência do ruído externo que atrapalha o reconhecimento da fala. O segundo foi no processo para cadastrar novos pontos de interesses que atualmente no dispositivo é feito via arquivo CSV, pois no projeto atual optou-se em focar na implementação do reconhecimento de fala ao invés de um mapa colaborativo para cadastros. Contudo como projeto numa linha de pesquisa relativamente

recente, este pode engajar trabalhos futuros a continuar e evoluir na mesma linha de pesquisa. As possíveis extensões propostas para continuar a linha de pesquisa deste projeto são:

a) pesquisar uma maneira que possa melhorar o reconhecimento de voz de forma off-line, reduzindo os ruídos externos;
b) encontrar outras alternativas para que os usuários com deficiência visual interajam com o dispositivo, como via comando táteis;
c) criar um mapa colaborativo para que o usuário possa cadastrar novos pontos de interesses e o dispositivo os receba ao conectar com a internet.

REFERÊNCIAS

BARROS, Maria J. A. S. **Estudo Comparativo e Técnicas de Geração de Sinal para a Síntese da Fala**. 2001. 165 f. Dissertação (Mestrado em Engenharia Electrotécnica e de Computadores) – Departamento de Engenharia Electrotécnica e de Computadores, Universidade do Porto, Porto.

CAIUSCA, Alana. **Lei dos Senos e dos Cossenos**, [2018]. Disponível em < https://www.guiaestudo.com.br/lei-dos-senos-e-dos-cossenos>. Acesso em 26 de novembro de 2019.

COOK, Stephen. Speech Recognition HOWTO. **The Linux Documentation Project**. 2002. Disponível em: <http://www.faqs.org/docs/Linux-HOWTO/Speech-Recognition-HOWTO.html>. Acesso em: 17 de nov. 2019.

GAMBARATO, Vivian Toledo Santos; BATISTA, Ana Paula; DE SOUZA GIANDONI, Larissa. Uso de tecnologias assistivas na educação superior tecnológica. **Tekhne e Logos**, v. 3, n. 1, p. 109-127, 2012. Disponível em: <http://www.fatecbt.edu.br/seer/index.php/tl/article/view/126/113>. Acesso em: 19 set. 2018.

GOOGLE. **MyMaps**. 2019. Disponível em: <https://www.google.com›mymaps>. Acesso em: 22 set. 2019.

HIJMANS, Robert J.; WILLIAMS, Ed; VENNES, Chris. **The geosphere package**, v.1, p. 2-4, 2010. Disponível: <http://cran.r-project.org/web/packages/geosphere/index.htm>. Acesso em: 17 nov 2019.

LINGNER, Rejane L. **Blulibras:** dicionário regional de libras. 2016. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MELLON, Carnegie. **Open Source Speech Recognition Toolkit**. 2019. Disponível em: <https://cmusphinx.github.io> Acesso em 20 setembro de 2019.

NASLAUSKY, Eduardo. **Transcrição de áudio utilizando bibliotecas offline**. 2018. 63 f. Trabalho de Conclusão de Curso (Engenharia Eletrônica e de Computação) - Universidade Federal do Rio de Janeiro, Rio de Janeiro.

O GLOBO. **Cegueira já afeta 36 milhões de pessoas no mundo, diz pesquisa**, [S.l.], [2017]. Disponível em: <https://oglobo.globo.com/sociedade/saude/cegueira-ja-afeta-36-milhoes-de-pessoas-nomundo-diz-pesquisa-21661673#ixzz5R93n1slJ>. Acesso em: 15 set. 2018.

OFFICE. **Atan2 (Função ATAN2)**, [S.l.], [2019]. Disponível em: <https://support.office.com/pt-br/article/atan2-fun%C3%A7%C3%A3o-atan2-51123ced-348c-416a-b2e2-833f7868569f>. Acesso em: 30 nov. 2019.

RADABAUGH, M. P. **NIDRR's Long Range Plan-Technology for access and function research section two: NIDDR Research Agenda Chapter 5:** Technology for access and function. [S.l.], 1993. Disponível em: <http://www.ncddr.org/new/announcements/>. Acesso em: 15 de set. 2018.

RIBEIRO JÚNIOR, José Geraldo et al. Sistema para monitoramento descentralizado de trânsito baseado em redes veiculares infraestruturadas. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 31., 2013, Brasília. **Anais** [...]. Brasília: [s. n.], 2013. Disponível: <https://www.gta.ufrj.br/ftp/gta/TechReports/JCC13.pdf >. Acesso em: 17 nov 2019.

REIS, Dalton S. et al. **Tagarela:** rede de comunicação alternativa. Blumenau, 2014.Disponível em: <http://gcg.inf.furb.br/?page_id=992>. Acesso em: 22 set. 2018.

SÁ, Elizabet D. **Acessibilidade: as pessoas cegas no itinerário da cidadania**, [2010]. Disponível em: <http://www.bancodeescola.com/acessibilidade.htm />. Acesso em: 15 de set. 2018

SANCHES, Rodrigo M. **Desenvolvimento de um sistema de planejamento de trajetória para veículos autônomos agrícolas**. 2012. 131 f. Dissertação (Mestrado em Engenharia Mecânica) – Departamento de Engenharia Mecânica, Universidade de São Paulo. São Paulo.

SAUTNER, Guilherme. **Tagarela:** módulo de desenvolvimento e aquisição de linguagem para crianças autistas. 2017. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SILVA, Carlos P. A. **Um software de reconhecimento de voz para português brasileiro**. 2010. 85 f. Dissertação (Mestrado em Engenharia Elétrica) – Setor de Tecnologia, Universidade Federal do Pará, Belém.

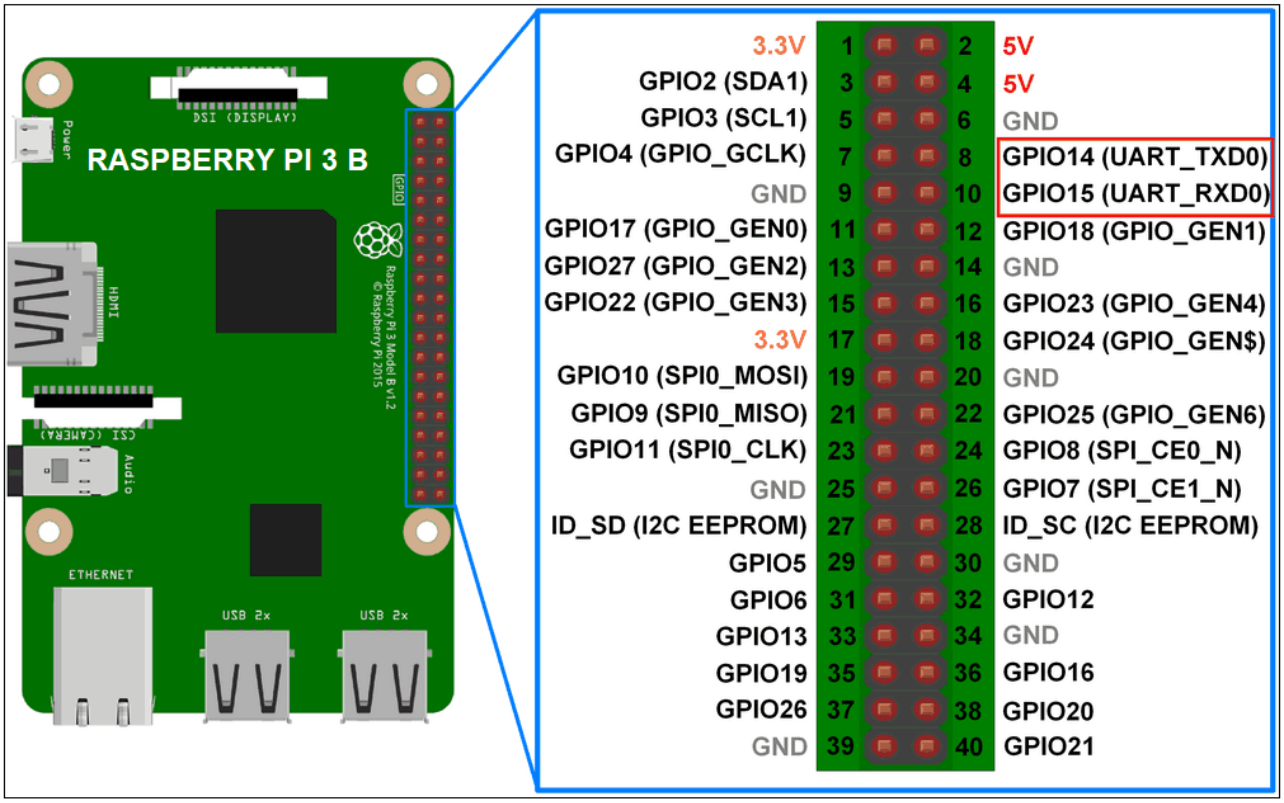
SINNOTT, Roger W. Virtues of the Haversine. **Sky Telesc**., v. 68, p. 159, 1984. Disponível em: <http://inspirehep.net/record/889399/ >. Acesso em: 17 nov. 2019.

TINKERCAD. **Circuits has arrived on Tinkercad**. 2019. Disponível em: <https://www.tinkercad.com/circuits>. Acesso em: 22 nov 2019.

APÊNDICE A – CONECTORES UART DA RASPBERRY PI 3 B

Neste apêndice é apresentado o GPIO da Raspberry PI 3 B, onde é possível verificar que só existe uma entrada UART nos pinos 8 e 10 (Figura 18).

Figura 18 – Pinos do UART da Raspberry PI 3 B



Fonte: elaborador pelo autor.

APÊNDICE B – CUSTO DO PROTOTIPO

Neste apêndice é apresentado o valor gasto com cada peça utilizada no protótipo (Quadro 12).

Quadro 12 – Custo de peças

NOME	Valor	Origem
Raspberry Pi 3 B	R$ 130,00	China
QMC5883L	R$ 5,94	China
NEO6MV2	R$ 12,34	China
CP2102	R$ 11,00	Brasil
RPI PowerPack v2.0	R$ 70,00	China
Jbl T110bt Bluetooth Headset	R$ 99,00	Brasil
Cooler 30x30 3.3v	R$ 2,91	China

Fonte: elaborador pelo autor.