BLACK GLASSES – ASSISTENTE PARA DEFICIENTES VISUAIS VIA GEOLOCALIZAÇÃO

Aluno: William Lopes da Silva

Orientador: Dalton Solano Reis

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Trabalhos Correlatos
- Requisitos
- Implementação
- Resultados, Testes e Desafios
- Conclusão e Sugestões

Introdução

- Deficientes visuais e suas limitações
- 38,5 milhões de pessoas cegas até 2020
- Avanço da tecnologia e inclusão social
- Tecnologia Assistivas

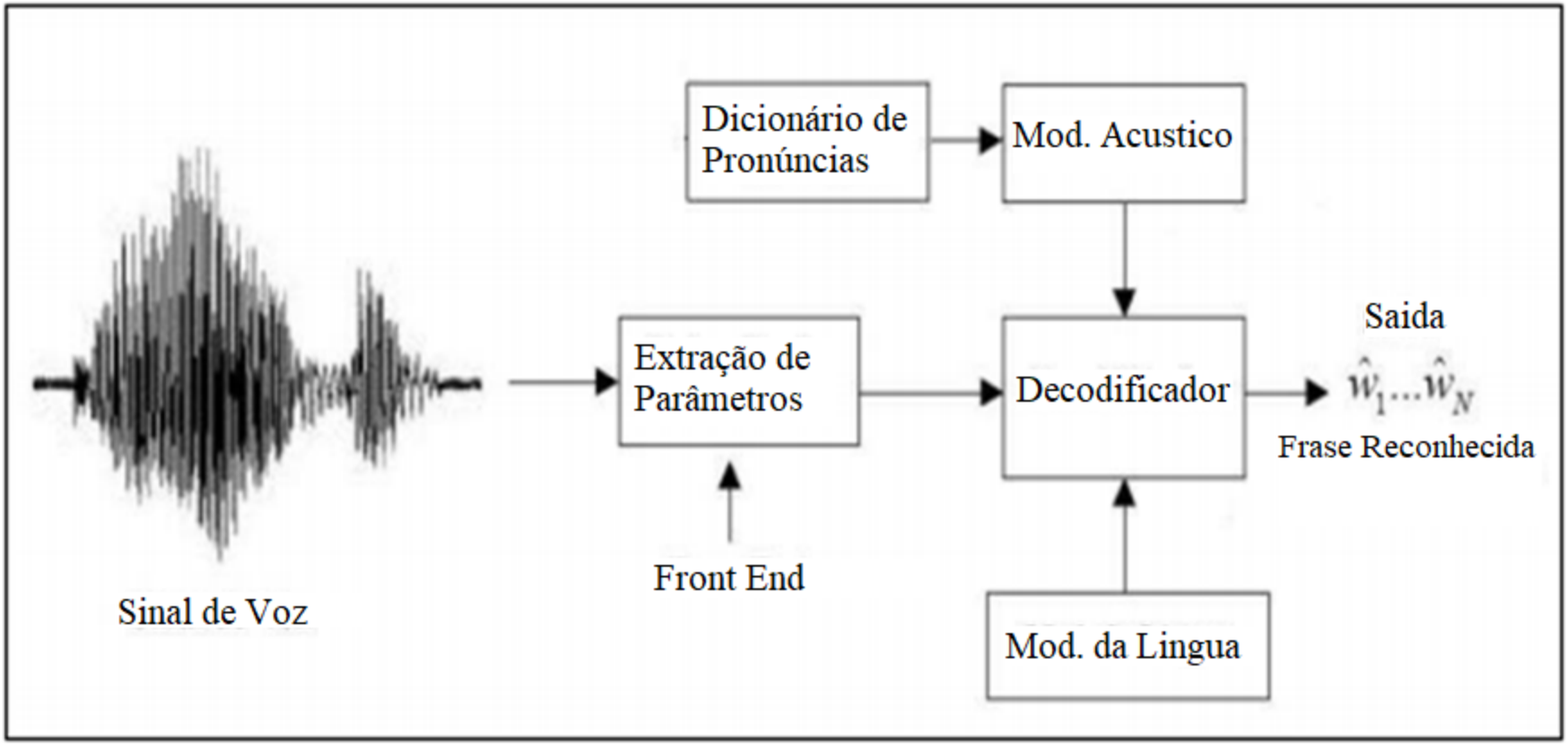
Objetivos

- 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

Fundamentação Teórica

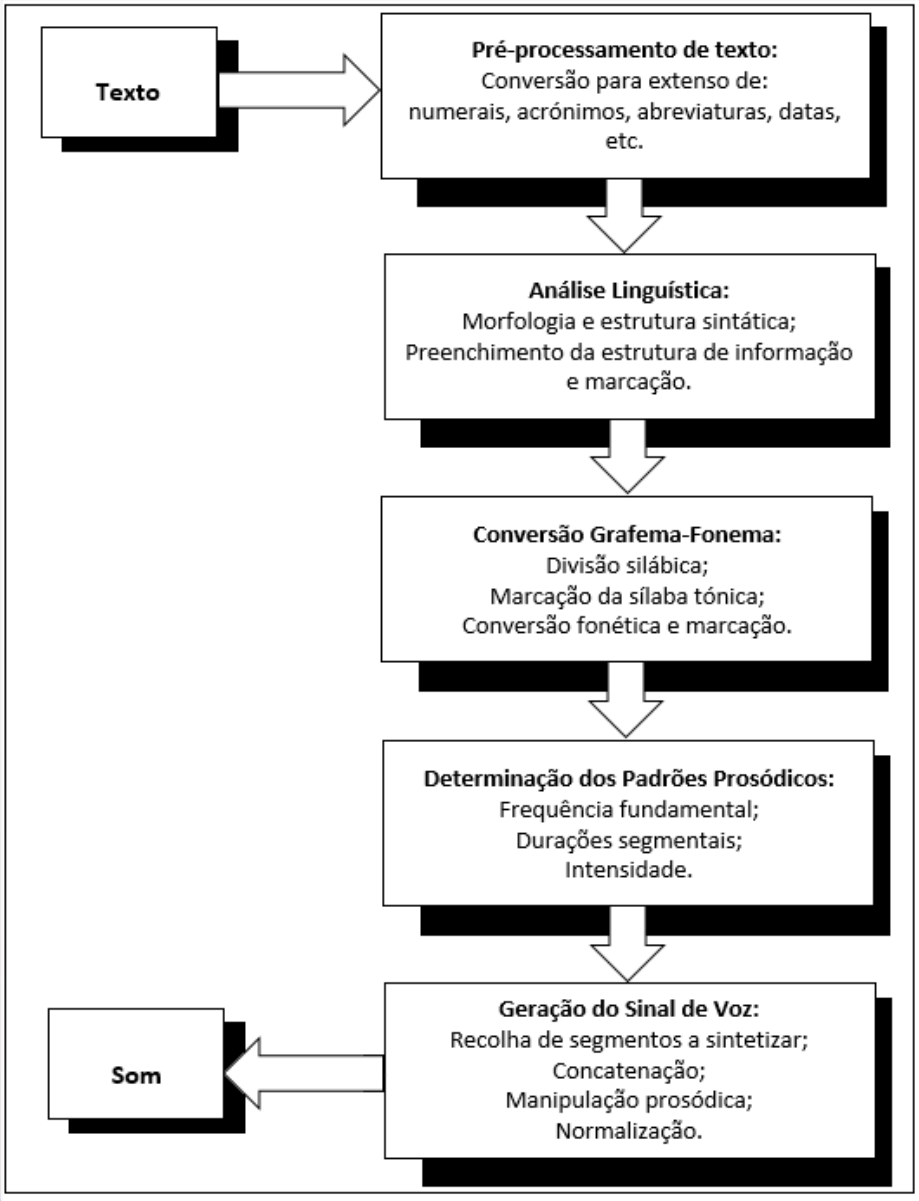
- Reconhecimento de Fala
- Texto para fala
- Lei dos Cossenos
- Equação de Haversine

Reconhecimento de Fala



Palavras isoladas
Palavras conectadas
Discurso contínuo
Fala espontânea
Verificação / identificação

Texto para Fala



- Processa o texto (2 para dois, etc.)
- Analisa a linguística (classifica pontuações, etc.)
- Conversão Grafema-Fonema (trabalha como funciona sons da fala)
- Determinação dos Padrões Prosódicos (trabalha a naturalidade da fala: duração, pausas, etc.)
- Geração de Sinal de Voz (gera o sinal de voz com os tratamentos feitos pelos módulos anteriores)

Lei dos Cossenos

a)

Fórmula

$a^2 = b^2 + c^2 - 2.b.c.\cos\alpha$

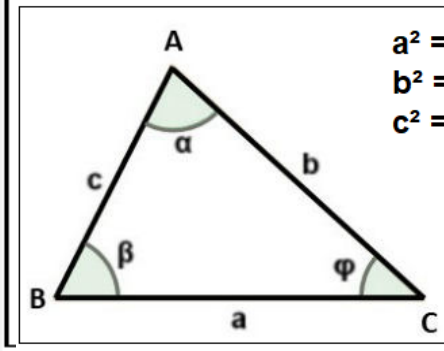
$a^2 = 153^2 + 170^2 - 2 \cdot 153 \cdot 170 \cdot (0{,}682)$ (47°)

$a^2 = 23409 + 28900 - 35477.55$

$a^2 = 16831.45$

$a = \sqrt{16831.45}$

$a = 129.73\ m$



$a^2 = b^2 + c^2 - 2.b.c\cos\alpha$

$b^2 = a^2 + c^2 - 2.a.c\cos\beta$

$c^2 = a^2 + b^2 - 2.a.b\cos\varphi$

b)

Descobrindo distâncias e ângulos desconhecidos...

FURB
UNIVERSIDADE DE BLUMENAU

Equação de Haversine

	Valor	Resultado	Formula
Latitude 1	-26,9054500	-0,4695886892	"=(D4*pi())/180"
longitude 1	-49,0782300	-0,8565767046	"=(D5*pi())/180"
Latitude 2	-26,9062900	-0,46960335	"=(D6*pi())/180"
longitude 2	-49,0767810	-0,8565514147	"=(D7*pi())/180"

	Resultado	Formula	
dLat	-0,0000073	"=(RADIANS(D6-D4)/2)"	diferença das latitudes dos pontos em radianos
dLon	0,0000126	"=(RADIANS(D7-D5)/2)"	diferença das longitudes dos pontos em radianos
R	6371		O valor de R é o raio da Terra em KM.
A	0,0000000002	"=SIN(E9)*SIN(E9)+COS(E4)*COS(E6)*SIN(E10)*SIN(E10)"	O valor A é o quadrado da metade do arco entre os pontos.
C	0,000026898717	"=2*ASIN(SQRT(E12))"	O valor C é a distância em ângulos radianos encontrada.
D	171,3717303	"=D14*E13*1000"	O valor de D é a distância entre as duas coordenadas.

Significa metade do seno
Distância entres duas coordenadas (origem e destino)
Erro médio de apenas 0,3%
Formula de Vincenty (considera o achatamento da terra e tem precisão de 0.5mm), peso computacional alto

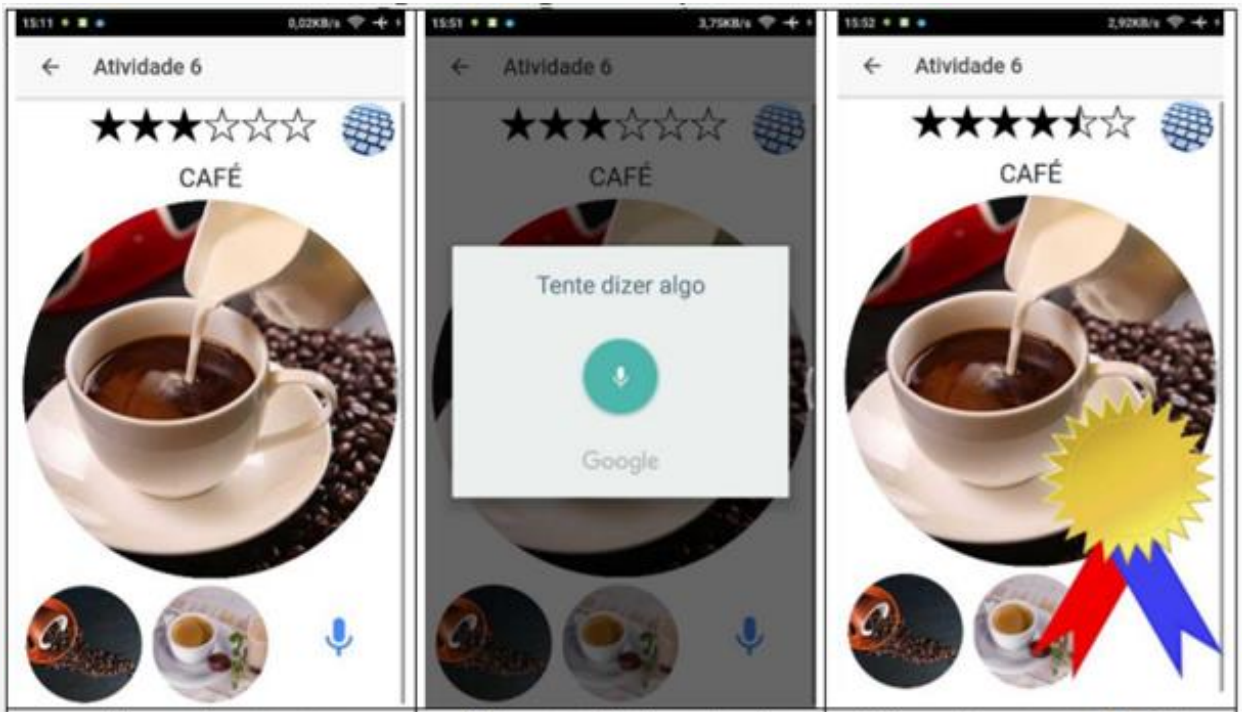
Trabalhos Correlatos

BLULIBRAS - visa ajudar na difusão de sinais regionais de libras (LINGNER, 2016)

Características \ Correlatos	Lingner (2016)
plataforma	Android
utiliza localização	sim
trabalha off-line	não
converte texto-fala	não
Converte fala-texto	não
móvel	sim
armazena localização	não

Trabalhos Correlatos

Tagarela – um módulo do Tagarela (REIS et al., 2014), que permite o desenvolvimento e aquisição de fala por crianças autistas (SAUTNER, 2017)



(a) - palavra a ser pronunciada | (b) – reconhecimento da pronúncia | (c) – palavra reconhecida

Características \ Correlatos	Sautner (2017)
plataforma	Android
utiliza localização	não
trabalha off-line	não
converte texto-fala	Não
Converte fala-texto	Sim
móvel	sim
armazena localização	não

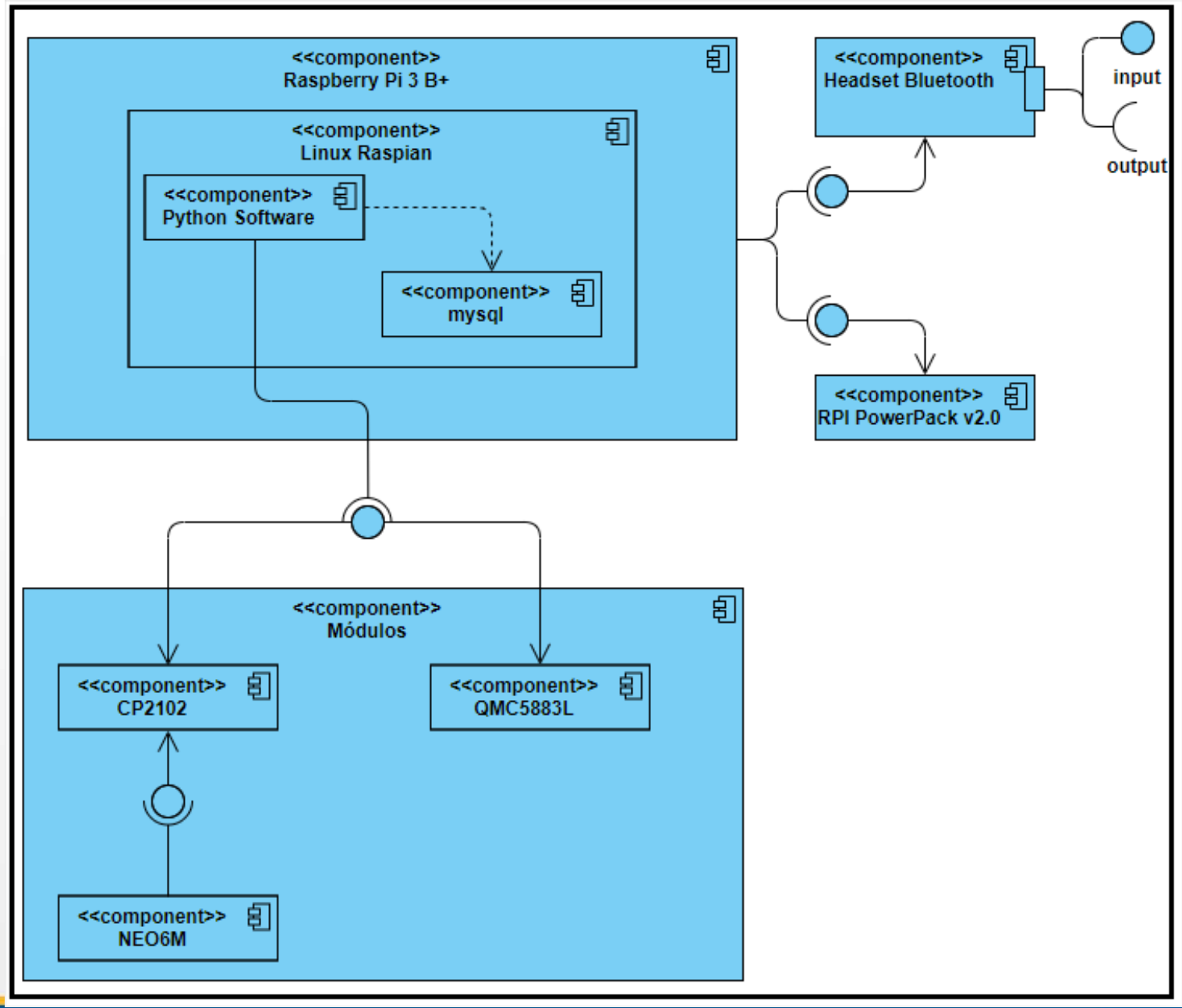
Requisitos Funcionais

- permitir cadastros de novos pontos de interesses via arquivo csv
- permitir iniciar, parar e desligar o dispositivo via comandos de voz
- permitir que o usuário escute via fone Bluetooth a distância e direção dos pontos de interesses no raio do seu percurso

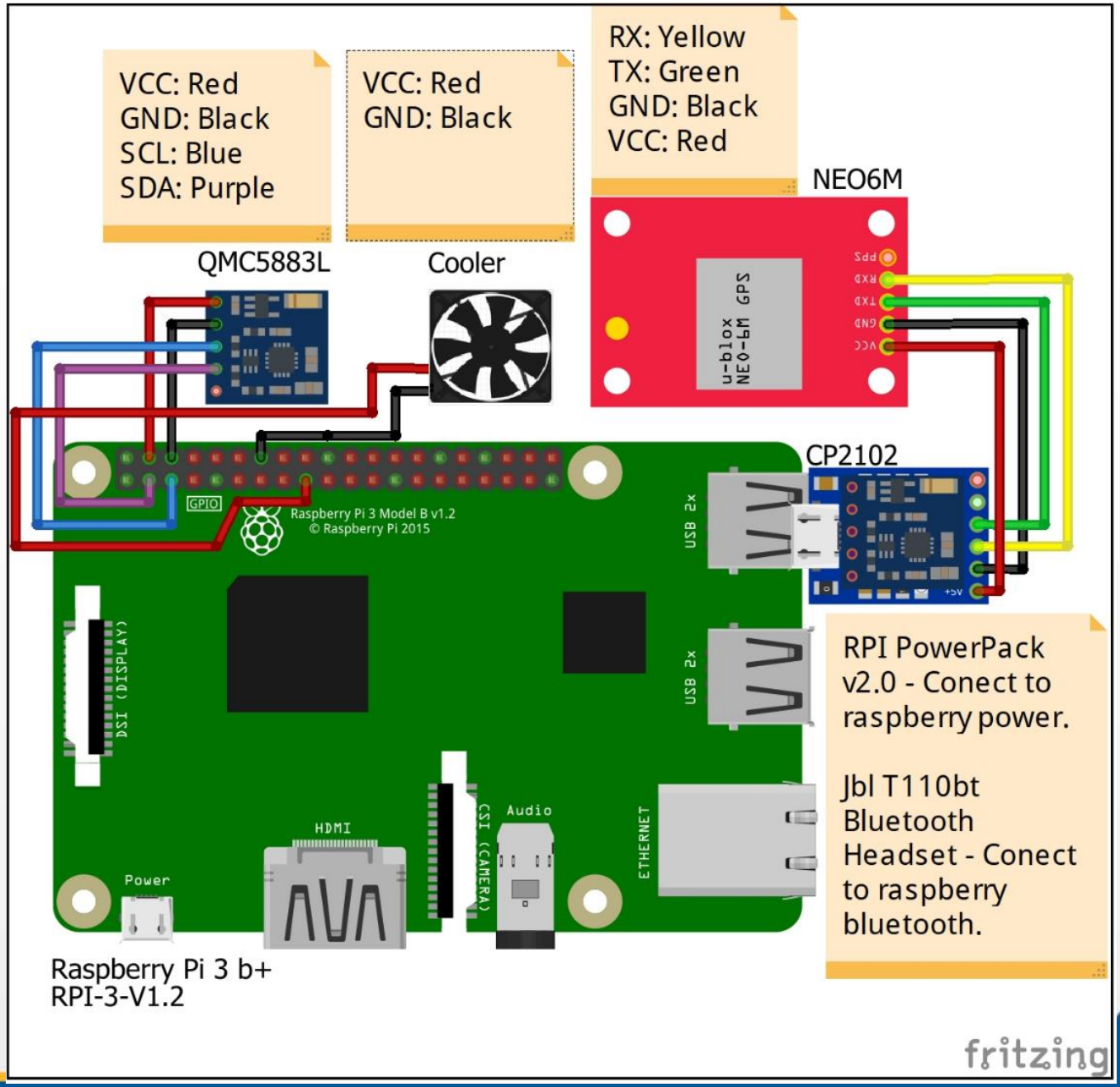
Requisitos Não Funcionais

- utilizar a placa Raspberry Pi 3 B como plataforma de desenvolvimento
- utilizar a linguagem de programação Python para o desenvolvimento do *Backend*
- utilizar o banco de dados MySQL
- funcionar no sistema operacional Linux Raspian

Arquitetura - Diagrama de componentes

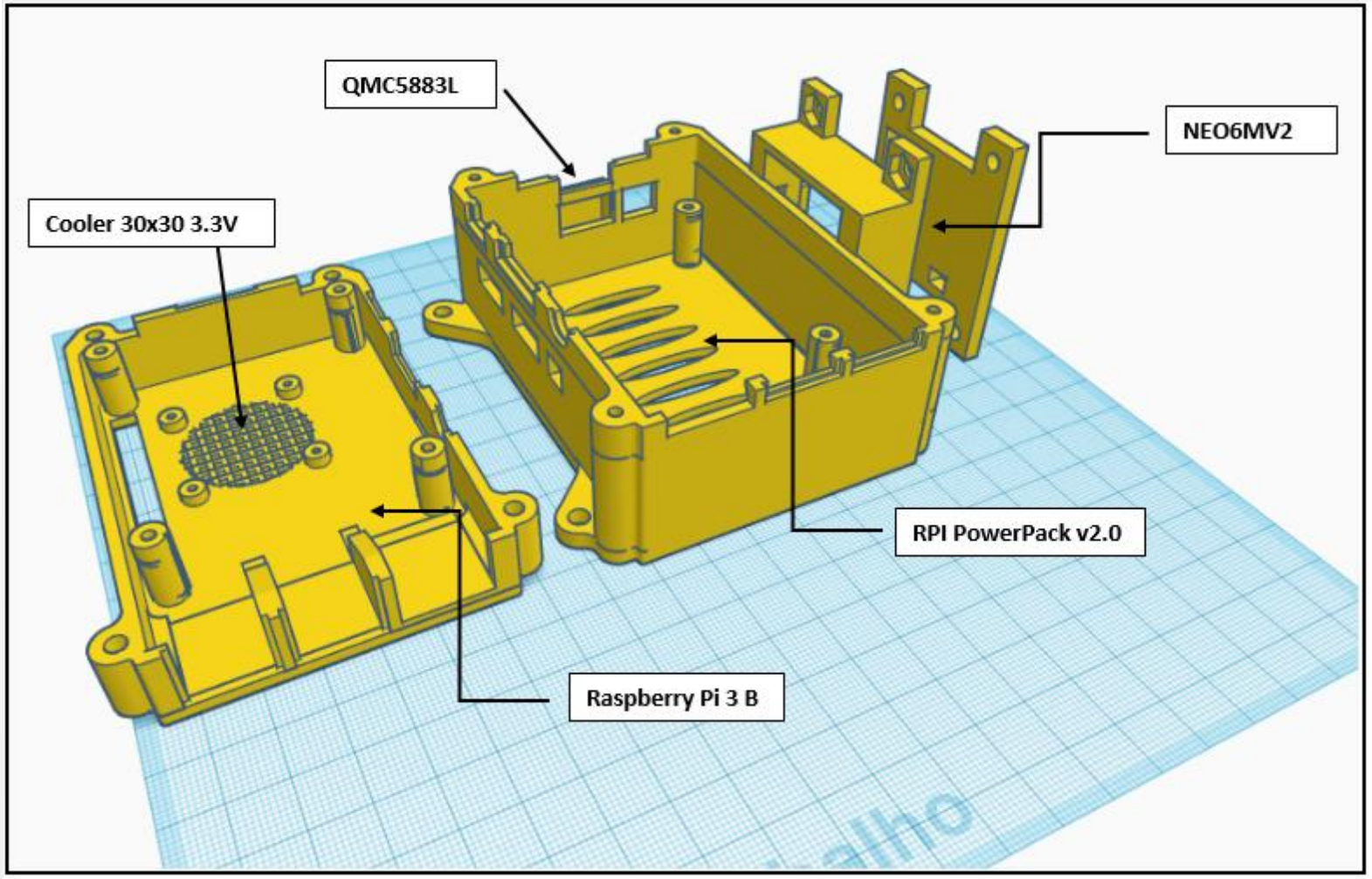


Esquema de montagem



Implementação

Criação do Case para impressão 3D



Implementação

Multiprocessing e não Thread, por quê?

```
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()
```

```
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:
```

- Multiprocessing - Processos em memórias separadas
- Threads – Processos no mesmo espaço de memória
- Lock – garante que somente um processo possa manipular a variável compartilhada em memória

Implementação

Gerência todo comportamento 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)
```

Implementação

Reconhecimento de voz - CMUSphinx /PocketSphinx

```
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)
```

```
1   #JSGF V1.0;
2
3   /**
4    * JSGF Grammar for English
5    */
6
7   grammar counting;
8
9   public <counting> = ( <say> ) +;
10
11  <say> = start | stop | offline ;
```

Implementação

Sintetizador de texto para fala - gTTS

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

- Por que utilizou sintetizador online gTTS e não offline pyttsx?
- Ainda funcionará off-line o dispositivo?

Implementação

Calculando distância - Haversine

Python

```
60      sql = ("SELECT *, (6371 * acos(cos(radians('%s')) * "
61             "cos(radians(lat)) * cos(radians('%s')- radians(lng)) + "
62             "sin(radians('%s')) * sin(radians(lat))))AS distance "
63             "FROM ponto_interesse HAVING distance <= '%s'")
```

Mysql

```
1  USE vision;
2  SELECT *, (6371 *
3          acos(
4              cos(radians(-26.908180)) * /*Latitude do usuário*/
5              cos(radians(lat)) *
6              cos(radians(-49.080577) - radians(lng)) + /*Longitude do usuário*/
7              sin(radians(-26.908180)) * /*Latitude do usuário*/
8              sin(radians(lat))
9          )) AS distance
10 FROM ponto_interesse HAVING distance <= 0.3 /*distância do raio*/
11
```

Por que usou um SELECT do Mysql ao invés do módulo Math?

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	lat	lng	ponto_interesse	distance
63	-26.906872	-49.078936	Furb	0.21824348976281766
65	-26.907630	-49.079878	Confeitaria Dona Hilda	0.09243441967219597
66	-26.908700	-49.081298	Blufit Academia	0.09194747041188231
67	-26.908753	-49.082106	Madrugadao Sushi	0.16445358634396257
68	-26.907890	-49.080230	Nana Hamburgueria	0.04715599760631051

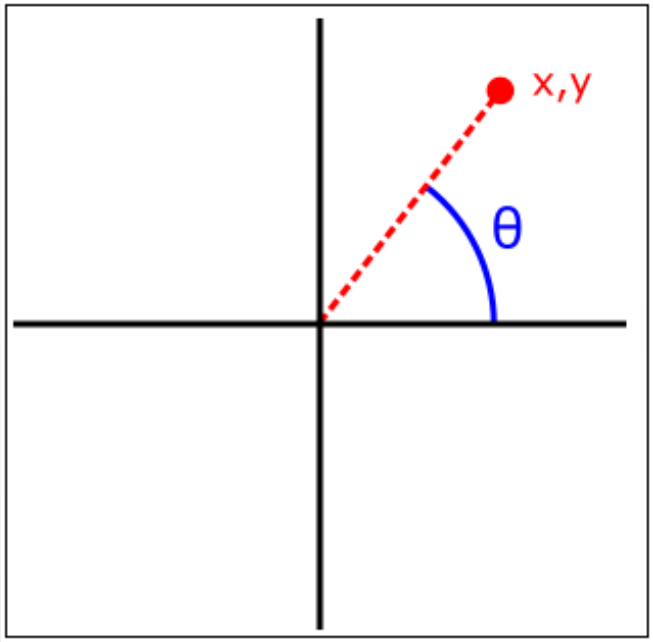
metros = distance * 1000

Implementação

Calculando direção – ATAN2

```
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))
```

- Origem do Fortran melhorada da formula ATAN que retorna negativos
- Otimizada para trabalhar com valores positivos das direções da bussola de 0° a 360



Implementação

Calculando direção – ATAN2

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
```

- Ajuste para levar em consideração o grau oscilante da bússola QMC5883

Implementação

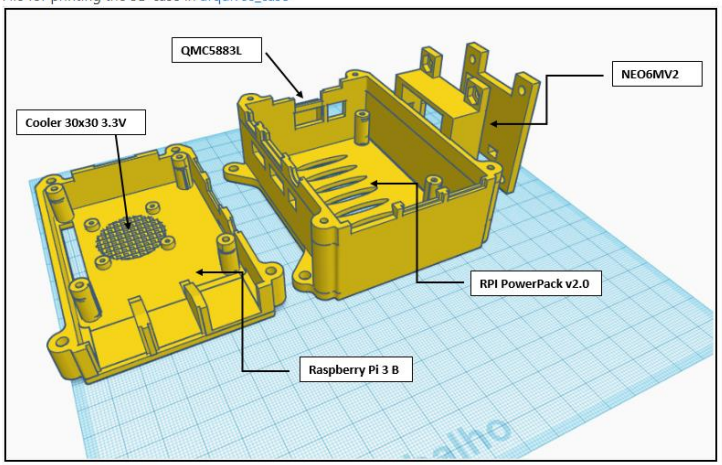
- Tutorial no github para instalações e criação do Case

Black Glasses - Assistente para deficientes visuais via geolocalização

William Lopes da Silva
Orientador - Dalton Solano dos Reis
Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

Installation guide for the tools used in this project.

File for printing the 3D case in arquivos_case



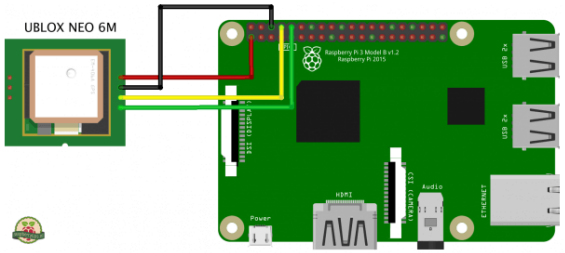
Install CMUSphinx/Pocketsphinx

The following steps show you how to test Google/Sphinx Speech Recognition using PyAudio and SpeechRecognition module on Raspberry Pi using a C-Media USB Microphone.

Install NEO-6M-GPS-Raspberry-Pi

Adapted tutorial from FranzTscharf
Python script for the NEO-6M GPS module on the Raspberry Pi

1. Connecting Schema



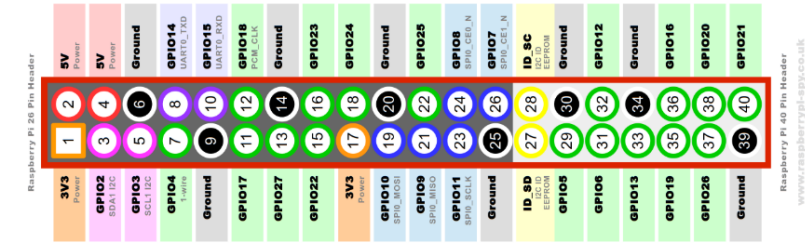


Image of Yaktoc2at2

2. Install the Dependencies

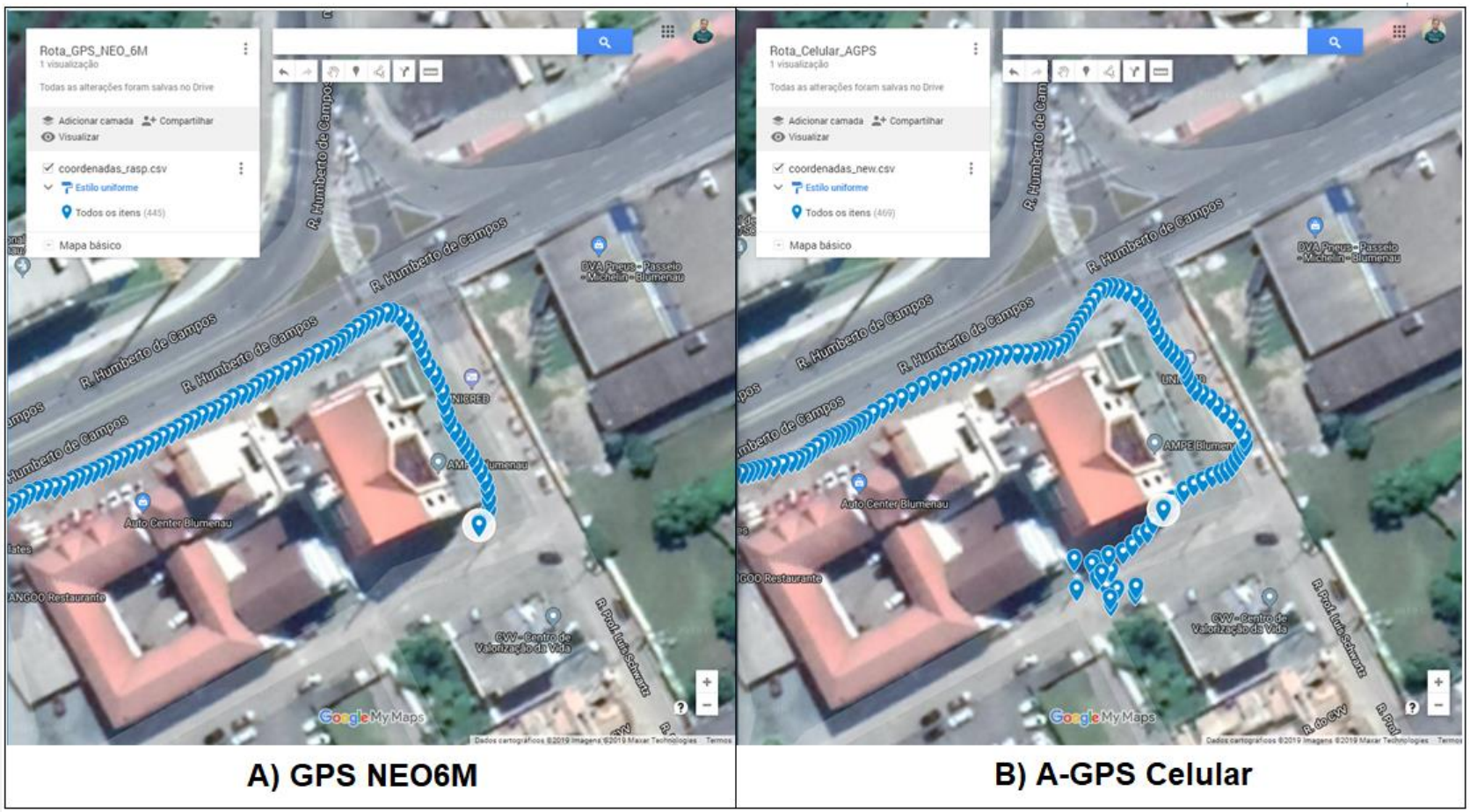
- pip installed.

Testes/Desafios/Resultados

Desafios

- Reconhecimento de fala adicionado ao dispositivo após testes
- Bluetooth e GPS utilizavam mesmo pino 8 e 10 do UART disponível no GPIO 14 e 15
- Utilizado conversor de USB para UART CP2102, para utilizar o GPS na porta USB

Testes



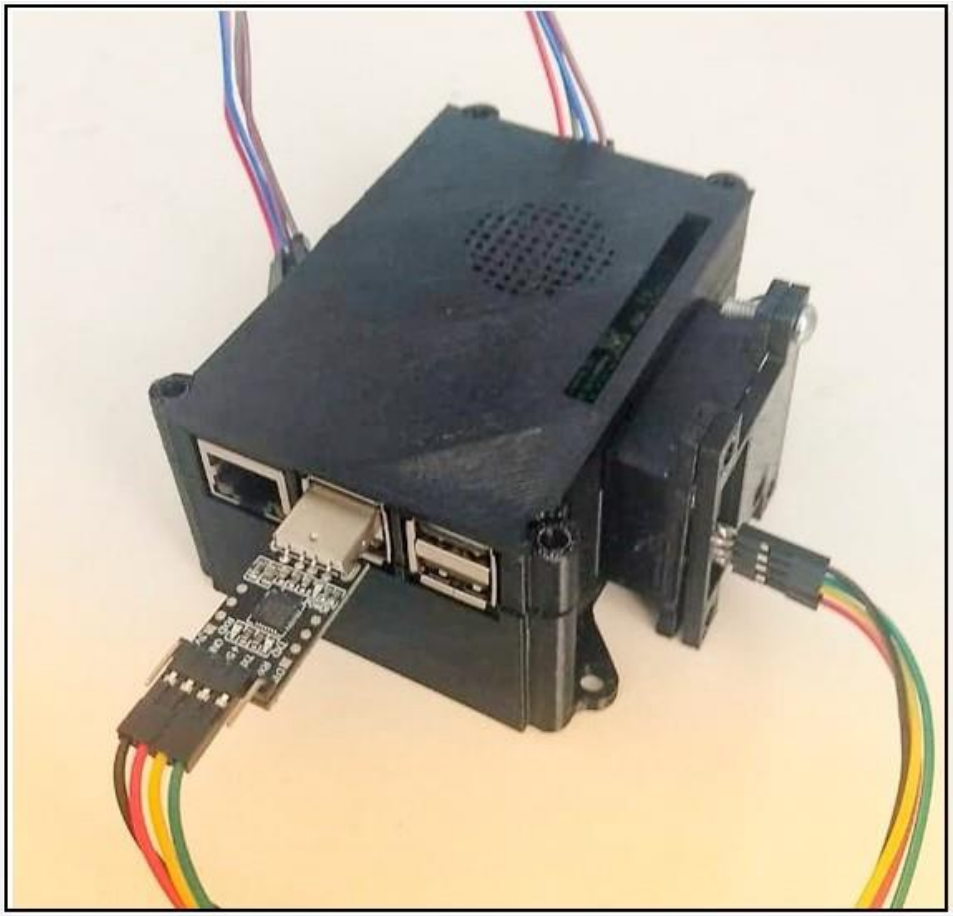
A) GPS NEO6M

B) A-GPS Celular

Testes da precisão do módulo de GPS NEO6M

FURB
UNIVERSIDADE DE BLUMENAU

Desafios



Criação do Case para impressão 3D

Resultados

- Dificuldades no reconhecimento de voz por causa dos ruídos externo
- Trabalhou de forma paralela com multi-processos
- Retornou com precisão a distância e direção dos pontos de interesses
- Trabalhou de forma totalmente off-line

Conclusões

- Atingiu os objetivos pretendidos
- Escolha da Raspberry Pi 3B fundamental
- Reconhecimento de voz ajudou na acessibilidade
- GPS e Bússola trabalharam bem off-line
- Custo baixo para criação

Sugestões de Extensão

- Melhorar reconhecimento de voz eliminando ruídos
- Outra maneira de interagir com dispositivo. Ex. Comandos táteis
- Criar um mapa colaborativo para cadastrar pontos de interesses