CENTRO UNIVERSITÁRIO SENAC
SANTO AMARO

Daniel Gomes dos Santos
Henrique Lima Borges

USO DE TECNOLOGIA ASSISTIVA PARA GUIAR ALUNOS COM DEFICIÊNCIA VISUAL NO CAMPUS SENAC SANTO AMARO

São Paulo
2018

Daniel Gomes dos Santos
Henrique Lima Borges

USO DE TECNOLOGIA ASSISTIVA PARA GUIAR DEFICIENTES VISUAIS NO CAMPUS SENAC SANTO AMARO

Trabalho de conclusão de curso apresentado ao Centro Universitário Senac – Santo Amaro, como exigência parcial para obtenção do grau de Bacharel em Engenharia da Computação.

São Paulo
2018

Borges, Henrique Lima

USO DE TECNOLOGIA ASSISTIVA PARA GUIAR ALUNOS COM DEFICIENCIA VISUAL NO CAMPUS SENAC SANTO AMARO / Henrique Lima Borges, Daniel Gomes dos Santos - São Paulo (SP), 2018.

102 f.: il. color.

Orientador(a): Denis Gabos

Trabalho de Conclusão de Curso (Bacharelado em Henrique Lima Borges) - Centro Universitário Senac, São Paulo, 2018.

Daniel Gomes dos Santos
Henrique Lima Borges

Uso de tecnologia Assistiva para guiar Deficientes visuais no Campus Senac Santo Amaro.

Trabalho de conclusão de curso apresentado ao Centro Universitário Senac – Santo Amaro, como exigência parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador Prof. Denis Gabos

A banca examinadora dos Trabalhos de Conclusão, em sessão pública realizada em _____/_____ /_____, considerou o (a) candidato (a):

1) Examinador (a)

2) Examinador (a)

3) Presidente

AGRADECIMENTOS

Agradecemos aos nossos familiares e amigos, a base sem a qual nada disso seria possível.

Ao professor orientador Denis Gabos que nos acompanhou desde o início, nosso coordenador professor Eduado Heredia e em especial aos professores Stelvio Barboza e Bruno Sanches que também contribuíram para o projeto.

Agradecemos também ao Samir Issa e ao Bruno Ábia Souza que foram essenciais para a implementação de *Machine Learning*.

RESUMO

De acordo com o último censo do IBGE 45,6 milhões de pessoas possuem alguma deficiência, esse número corresponde a 23% dos brasileiros. A deficiência visual é a deficiência que mais ocorre na população atingindo cerca de 8,5 milhões de pessoas, sendo que, 1,72 milhões dessas pessoas possui idades entre 15 e 64 anos. Embora diversas leis que promovem a acessibilidade tenham sido implantadas, as pessoas que possuem alguma forma de deficiência visual frequentam menos ambiente de trabalho, muitas vezes em decorrência de também terem frequentado menos o ambiente acadêmico. Esse projeto visa o desenvolvimento e a implantação de um sistema capaz de guiar o usuário com deficiência visual pelas salas e laboratórios dos prédios acadêmicos I e II do campus Senac Santo Amaro como forma de aprimorar a acessibilidade dos alunos dentro do campus. Para realizar a navegação em ambientes internos as técnicas de *Wi-Fi Fingerprint* e Machine Learning foram escolhidas.

ABSTRACT

According to the last IBGE (IBGE in portuguese) research 45.6 million people have some kind of disability, this number corresponds to 23% Brazilians population. Blindness is the most common deficiency in the population, reaching about 8.5 million people, being that, 1.72 million are aged between 15 and 64 years. Although several laws that promote accessibility have been implanted, people that have some form of visual disability attend less the work environment, many times in consequence of frequenting less the academic environment. This project focuses on a system development and implantation capable of guiding the user with blindness across the academic rooms and labs in 1st and 2nd buildings at Senac's campus in Santo Amaro, in order to improve students accessibilities inside the campus. To realize the navigation in internal places the *Wi-Fi Fingerprint* and Machine Learning technics were chosen.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVEATURAS E SIGLAS

A* – Algoritmo utilizado em busca de caminho

Addf – Comando para Adicionar Fingerprint

API – Interface de programação de aplicação

API REST – Transferência de Estado Representacional

ARM – Manual de referência de arquitetura

Array – Estrutura de dados

Beacons – Dispositivo que emite sinais por meio de tecnologia bluetooth

Bluetooth – Conexão sem fio para troca de informações entre dispositivos

CLI – Interface de linha de comando

Clustering – Conjunto de técnicas de prospecção de dados

Compass – Bússola

CSV – Arquivos de texto de formato regulamentado pelo RFC 4180

Dataset – Conjunto de Dados

Db – Decibel

Dbm – Decibel miliwatt

Decision Tree – Tabela de decisão sobe forma de árvore.

Dijkstra – Algoritmo de caminho mais curto

Features – Características

Feedback – Retornos

GPIO – Portas programáveis de entrada e saída de dados.

GPS – Sistema de Posicionamento Global

Grafo – Relação entre objetos

Hard-coded – Desenvolvimento de software de embutir dados diretamente no código fonte

Hash – Algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo

HTTP – protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores

HTTPS – Implementação do protocolo HTTP sobre uma camada adicional de segurança

I2C – Protocolo de comunicação entre dispositivos

IBGE – Instituto Brasileiro de Geografia e Estatística

Idling – Marcha lenta/ ocioso

JSON – Formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas

K-Nearest Neighbors – Método usado para classificação e regressão

Labeled data – Um grupo de exemplos que foram classificados.

Lato sensu – Literalmente/ em sentido amplo.

Mac Address – Endereçamento de Camada 2 do Modelo OSI que permite o Controle de Acesso ao meio

Machine Learning – Método de análise de dados que automatiza a construção de modelos analíticos

Main – Principal

Manhattan distance – Método geométrico de calculo de distâncias

Mongodb – Software de banco de dados orientado a documentos

Numpy – Pacote para a linguagem *Python*

Output – Saída

Overfitting – Termo usado em estatística para descrever quando um modelo estatístico se ajusta muito bem ao conjunto de dados anteriormente observado

Pandas – Biblioteca de software escrita para a linguagem de programação Pyth

on para manipulação e análise de dados

Pathsolver – Módulo de definição de rotas

Penalty – Parâmetro que configura uma técnica de regularização dos coeficientes angulares da equação

Percloreto de ferro – Composto químico

Placa de fenolite – Placa em laminado plástico industrial utilizado como isolante elétrico

POC – Termo utilizado para denominar um modelo prático que possa provar o conceito

POST – Método utilizado para enviar dados do cliente para o servidor

Pre-pruning – Técnicas para contornar o sobreajuste de um modelo

PWM – Modulação de Largura de Pulso

Python – Linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos

Random Forest – Método de aprendizado conjunto para classificação, regressão e outras tarefas

Raspbian Lite – Sistema Operacional baseado em Linux otimizado para Raspberry

RFID – método de identificação automática através de sinais de rádio

Ridge Regression – Técnica para análise de dados de regressão múltipla que sofrem de multicolinearidade

RSSI – É uma medida da potência presente em um sinal de rádio recebido

Scikit-learn – biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python

Scp – Comando de cópia segura

SGD Classifier – Método otimizado de machine learning

SHA-2 - Conjunto de funções hash criptográficas

Shield – Placas que podem ser plugadas

Site-survey – Análise minuciosa do ambiente de rede

SSH – protocolo de rede criptográfico para operação de serviços de rede

SSL Handshake – tipo de segurança digital que permite a comunicação criptografada entre um servidor e um cliente.

SVC – Classificação de vetores de suporte

Underfitting – Quando você treina seu algoritmo e testa ele no próprio conjunto de treino e percebe que ele ainda tem uma tacha de erro considerável e então testa ele no conjunto de teste e percebe que a taxa de erro é semelhante, mas ainda alta

URL – Refere ao endereço de rede no qual se encontra algum recurso

Wi-Fi – Conexão sem fio para troca de informações entre dispositivos

Wi-Fi Fingerprint – Impressão digital de uma determinada posição determinada pela conexão Wi-Fi do local.

Wipy – Nome dado ao módulo de *Wi-Fi Fingerprints*

Wireframe – Protótipo usado em design de interface para sugerir a estrutura dos relacionamentos

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização

Em 2004 a Organização Mundial de Saúde – OMS divulgou a Classificação Internacional de Funcionalidade, Incapacidade e Saúde – CIF onde incapacidade é descrita como resultado da limitação das estruturas do corpo e das funções do mesmo somadas a fatores sociais e ambientais. De acordo com o censo de 2010 do IBGE cerca de 23% dos brasileiros possuem alguma deficiência, esse número corresponde a aproximadamente 45,6 milhões de pessoas. Quando as deficiências são agrupadas nas categorias visual, auditiva, motora, mental ou intelectual do censo do IBGE, a categoria com maior ocorrência na população brasileira é a visual que chega a 18,8%, cerca de 8,5 milhões de indivíduos. Desses 8,5 milhões, 20,1% possuem idades entre 15 e 64 anos, totalizando 1,72 milhões de pessoas com alguma forma de deficiência visual em idade ativa. É importante atentar que o percentual de pessoas com idade acima de 14 anos que declararam possuir alguma deficiência e não tinham instrução ou possuíam o ensino fundamental incompleto era de 61,1% que sozinho já é um dado preocupante, porém ao compararmos com o percentual de pessoas na mesma faixa etária que declararam não possuir nenhuma deficiência esse número cai para 38,2%, uma diferença de 22,9 pontos percentuais. Ao verificar as características do trabalho é possível notar que, embora 40,2% das pessoas com mais de 10 anos de idade que declararam possuir alguma deficiência tivessem carteira assinada, esse número ainda era menor que a porcentagem de pessoas da mesma faixa etária que declararam não possuir nenhuma deficiência (49,9%). No âmbito salarial, 46,7% das pessoas com mais de 10 anos de idade que possuíam deficiência ganhavam até um salário mínimo ou não tinham renda, a diferença em comparação com pessoas da mesma faixa etária que declararam não possuírem nenhuma deficiência permaneceu com mais de 9 pontos percentuais (37,1%).

1.2 Problemática

Com o passar dos anos a visão da sociedade para com a parcela da população que possui alguma forma de deficiência mudou bastante, entretanto os dados do censo do IBGE citados anteriormente revelam que as pessoas com deficiência ainda estão frequentando menos as instituições de ensino e possuindo menos acesso ao mercado de trabalho. O Brasil toma uma série de medidas para assegurar a acessi-

bilidade a essas pessoas, essas medidas estão presentes na área da educação, do transporte público, dos edifícios públicos, desporto, lazer, cultura e trabalho. Alguns exemplos dessas medidas são, as seguintes:

- Atendimento educacional especializado gratuito aos educandos com deficiência, transtornos globais do desenvolvimento e altas habilidades ou superdotação, transversal a todos os níveis, etapas e modalidades, preferencialmente na rede regular de ensino; (Redação dada pela Lei nº 12.796, de 2013). (Artigo 4º lei Nº 9.394, item III, 1996).

- A empresa com 100 (cem) ou mais empregados está obrigada a preencher de 2% (dois por cento) a 5% (cinco por cento) dos seus cargos com beneficiários reabilitados ou pessoas portadoras de deficiência, habilitadas, na seguinte proporção: 1 – até 200 empregados, 2%; 2 – de 201 a 500, 3%; 3 – de 501 a 1.000, 4%; 4 – de 1.001 em diante, 5%. [...] . (Artigo 93º lei Nº 8.213/91, 1991).

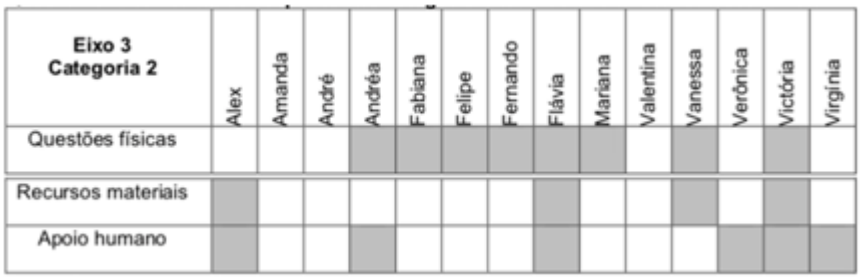
Embora existam diversas leis quase 20 anos mais novas que o censo de 2010 do IBGE, os resultados divulgados pelo mesmo ainda são alarmantes. O Brasil tem uma quantidade massiva de pessoas em idade ativa que possuem alguma forma de deficiência e são capazes de trabalhar, também existem leis para garantir a contratação de funcionários que possuem alguma forma de deficiência, porém muitos continuam desempregados. A educação tem um papel importante nas relações de trabalho das pessoas que possuem alguma forma de deficiência como demonstrado no artigo Direito dos Portadores de Deficiência e o nível básico de escolaridade e capacitação limita a área de atuação.

“A baixa escolaridade é um dos principais motivos ressaltados pelos empregadores na hora de fechar as portas do mercado de trabalho às pessoas com deficiência.” (CONSOLO; GIANULLO. 2011, p.10).

Muitas instituições de ensino não possuem um ambiente que favoreça a integração de alunos com deficiência; é preciso compreender a necessidade da remoção de barreiras que impedem que as pessoas, independente das limitações das estruturas do corpo e funções do mesmo, participem de atividades do cotidiano. Na pesquisa realizada no artigo Condições de Acessibilidade na Universidade (SILVIA, 2016) os alunos categorizaram os aspectos de suas respectivas faculdades que

consideravam promover a acessibilidade (Figura 1). A categoria “Questões físicas” relaciona aspectos do campus como piso tátil, placas em braile, banheiros adaptados, rampas e elevadores. Já a categoria “Recursos materiais” diz respeito a itens como gravador de voz que auxiliam os estudantes dentro do ambiente escolar. Por último a categoria de “Apoio humano” relaciona atividades como apoio na execução de atividades por parte de profissionais da universidade dentro do campus. É importante notar que a solução para a acessibilidade não está presente no uso exclusivo de uma das categorias citadas, porém no conjunto delas, um aluno com deficiência visual pode caminhar pelo campus de sua universidade utilizando o piso tátil e identificar as salas através das placas em braile, porém seu problema de locomoção no campus ainda está longe de ser resolvido. Técnicas como piso tátil auxiliam na locomoção de deficientes visuais, enquanto placas em braile possibilitam a identificação de salas, porém existe uma lacuna que é o auxílio de navegação durante o deslocamento do deficiente visual pelos edifícios das instituições de ensino.

Figura 1 – Ocorrência de respostas na categoria “Recursos Considerados acessíveis”.



Eixo 3 Categoria 2	Alex	Amanda	André	Andréa	Fabiana	Felipe	Fernando	Flávia	Mariana	Valentina	Vanessa	Verônica	Victória	Virgínia
Questões físicas				■	■	■	■	■	■		■		■	
Recursos materiais	■							■			■		■	
Apoio humano	■			■				■				■	■	■

Fonte: SILVA, 2016, p.110 adaptado

1.3 Justificativa

O Senac São Paulo foi fundado em 1981 com o foco em ensino superior e técnico. Atualmente o Senac São Paulo possui quatro campi: Santo Amaro, Campos do Jordão, Tiradentes e Águas de São Pedro. O campus de Santo Amaro possui prédios modernos que são distribuídos horizontalmente cujo projeto arquitetônico estimula a integração entre o corpo docente, alunos, e os demais funcionários. A área é constituída pelos prédios Acadêmicos um e dois, Centro Gastronômico, Biblioteca, Reitoria, Prédio de Design, Centro Esportivo, Centro de Convenções além de três praças de alimentação, totalizando cerca de 154 mil metros quadrados onde são oferecidos mais de 40 cursos de graduação, além de diversos títulos de pós-graduação *lato sensu* e extensão universitária. O campus de Santo Amaro imple-

menta uma série de técnicas para tornar o ambiente mais acessível para todos, são elas: piso tátil, elevadores, rampas de acesso e até placas identificadoras em braile em frente às salas e laboratórios. Os prédios Acadêmicos um e dois possuem dois andares cada um com as salas de aula e laboratórios distribuídos ao longo do térreo e do primeiro andar.

É louvável o foco que o campus de Santo Amaro do Senac coloca em acessibilidade, e é muito importante que a instituição de ensino proporcione um ambiente com o qual todos possam interagir, pois quando tornamos um ambiente mais acessível possibilitamos que cada vez mais pessoas possam aproveitar o que ele tem a oferecer. Fernando Morais (2011), autor do artigo “A Importância da Acessibilidade na Cidade” salienta que, no momento em que acessibilidade começar a ser vista como um direito e se tornar um interesse da coletividade assegurando respeito à diversidade, será possível compreender o quanto a acessibilidade é imperiosa na sociedade.

Ao mesclar a tecnologia com toda a gama de implementações já existentes no campus do Senac Santo Amaro podemos desenvolver uma solução de navegação para usuários com deficiência visual, tornando-o mais acessível do que é hoje. Embora essa solução não afete de forma alguma a locomoção de pessoas que não possuem deficiência visual pelo campus, ela proporcionaria maior autonomia na locomoção de pessoas com alguma forma de deficiência visual. É importante atentar ao fato de que muitas das implementações em prol da acessibilidade em ambientes não fazem uso de tecnologia ainda, porém o impacto que a tecnologia é capaz de gerar nessa área é imenso, Mary Pat Radabaugh da *IBM National Support Center for Persons with Disabilities* denota a real importância da tecnologia no campo da acessibilidade (*National Council on Disability*, 1993) afirmando que para a maioria das pessoas a tecnologia torna as coisas mais fáceis, mas para pessoas com alguma forma de deficiência a tecnologia torna as coisas possíveis,
“*For most people, technology makes things easier. For people with disabilities, technology makes things possible*. (MARY,1993, apud RAJA, 2016, p. 5) ”.

Uma solução capaz de guiar o usuário com deficiência visual faria do campus Senac Santo Amaro uma das instituições mais bem preparadas para receber essas pessoas. De acordo com o artigo Condições de Acessibilidade na Universidade

(SILVA, 2016): existe uma demanda cada vez maior de alunos que possuem alguma forma de deficiência e desejam ingressar nas instituições de ensino superior.

> Ao se comparar a evolução das matrículas na universidade por um período de cinco anos (2010-2014), de acordo com o Censo da Educação Superior dos referidos anos, podemos notar que, embora ainda incipiente há significativo avanço no acesso de pessoas deficiência nesse nível de ensino. (SILVA, 2016, p. 59).

Por fim, o potencial impacto do uso de um sistema capaz de guiar o usuário entre as salas dos prédios Acadêmicos um e dois não se limita somente a alunos ou funcionários deficientes visuais, em casos de eventos dentro do campus do Senac Santo Amaro seria possível guiar os visitantes também.

2. OBJETIVOS

2.1 Objetivo Geral

- Desenvolver um sistema capaz de guiar um deficiente visual pelas salas de aula e laboratórios dos prédios Acadêmicos um e dois do Centro Universitário Senac Santo Amaro utilizando a técnica de navegação chamada *Wi-Fi Fingerprint* para aumentar o nível de acessibilidade dentro do campus.

2.2 Objetivos específicos

- Mapear todas as salas e laboratórios dos prédios Acadêmicos um e dois do Senac campus Santo Amaro gerando identificadores para cada sala ou laboratório baseado na intensidade dos sinais *Wi-Fi* emitidos pelos roteadores próximos.
- Implementar uma técnica para reconhecimento da posição atual baseado nos identificadores *Wi-Fi Fingerprints* gerados.
- Desenvolver um dispositivo para deficientes visuais capaz de identificar a posição atual do indivíduo, receber uma sala ou laboratório de um dos prédios Acadêmicos do Senac campus Santo Amaro como destino e guiar o usuário por voz até o destino.

3. FUNDAMENTAÇÃO TEÓRICA E LITERATURA SOBRE O TEMA

3.1. Sistemas de posicionamento

Hoje em dia sistemas de navegação estão presentes em nosso cotidiano de forma quase orgânica. Conectados aos nossos smartphones os sistemas de navegação são capazes de nos guiar através de cidades e estradas, conectados a outras tecnologias são capazes de nos alertar sobre trânsito e demais problemas. Tudo isso é possível graças ao sistema de *GPS (Global Positioning System)*, constituído de 24 satélites ao redor do planeta terra e que é capaz de nos fornecer informações sobre a nossa posição atual (latitude, longitude e altitude). Embora essencial para a navegação em ambientes externos, seu sinal é fraco e facilmente bloqueável, tornando a navegação utilizando o GPS muito complicada em ambientes internos. Porém, existem inúmeras tecnologias que podem ser usadas (até mesmo em conjunto) em ambientes fechados, tais como: *beacons Bluetooth, RFID*, comunicação utilizando luz visível, tecnologias utilizando ondas de som, *WiFi,* entre outras. É importante notar que o ambiente interno é passível de mudança dos obstáculos, flutuação de sinal, ruído em caso de grande fluxo de pessoas como um shopping, até o corpo humano se torna uma barreira para os sinais como demonstrado em *DorFin: WiFi Fingerprint-based Localization Revisited* o corpo humano pode enfraquecer o sinal recebido em até 10dB (dB é uma medida que indica a diferença de potência de sinal entre dois pontos).

"*Human body blockage to smartphones can remove line-of-sight and weaken the received signal by up to 10dB*" (CHENSHU; ZHENG; ZIMU, YUNHAO; MINGYAN, 2013).

Embora existam uma série de fatores que dificultam a implementação de um sistema de navegação em ambientes externos esse é um mercado com estimativa de crescimento de até U$ 1,246 milhões até o ano de 2025 (MARKET RESEARCH FUTURE, 2018).

3.2 Introdução ao *Wi-Fi Fingerprint*

Atualmente a maioria dos espaços públicos possui roteadores *Wi-Fi* (pontos de acesso sem fio à internet), então é natural que soluções que utilizem as redes *Wi-*

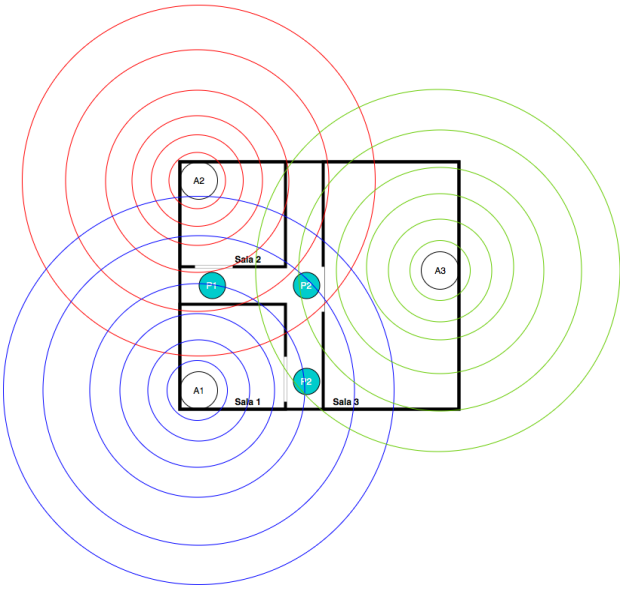
Fi sejam mais atrativas e mais facilmente implementadas devido ao fato dos roteadores estarem presentes nesses espaços. O *Wi-Fi Fingerprint* se destaca por ser um método que requer menos processamento que técnicas mais famosas que utilizam a propagação do sinal. *Wi-Fi Fingerprint* é um processo de coleta de sinais dos pontos de acesso e associação dos mesmos a pontos de interesse dentro do seu ambiente fechado, essa técnica se torna mais viável para ambientes internos pois não necessita da posição exata dos seus pontos de acesso ou cálculos envolvendo ângulos.

O processo se divide em duas fases: *site-survey* e consulta. Na fase de *site-survey* são eleitos diversos pontos de interesse dentro do ambiente interno e nesses pontos são coletados os sinais dos pontos de acesso que incidem sobre o ponto de interesse. A partir desses sinais é gerado e armazenado um identificador único para cada ponto de interesse, chamado de *Wi-Fi fingerprint*. Na fase de consulta é realizada a leitura de todos os sinais dos pontos de acesso que incidem sobre a localização atual e um novo *Wi-Fi Fingerprint* é gerado, esse *Wi-Fi Fingerprint* é comparado através de algoritmos determinísticos ou probabilísticos com aqueles gerados na primeira fase e uma localização baseada na relação do *Fingerprint* atual com os dos *Wi-Fi Fingerprints* armazenados é obtida.

3.2.1 Site-survey

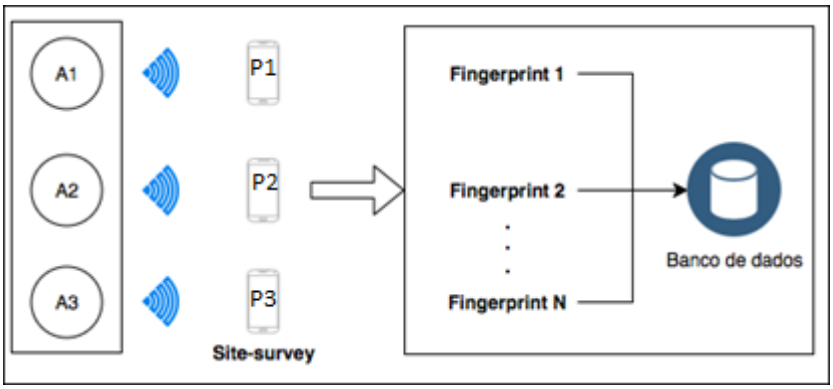
A fase do *site-survey* consiste basicamente em eleger pontos de interesse dentro do ambiente e coletar os sinais emitidos pelos pontos de acesso para cada um dos pontos de interesse. Na figura 2 temos o exemplo de um ambiente fechado composto por 3 salas e um corredor, os pontos de interesse P1, P2 e P3 eleitos para esse ambiente se encontram no corredor em frente a porta de cada uma das salas. Cada sala possui um ponto de acesso, respectivamente A1, A2 e A3 que emite sinais *Wi-Fi* dentro do ambiente. O *site-survey* para esse ambiente seria a coleta das intensidades dos sinais provenientes dos pontos de acesso A1, A2 e A3 para os pontos P1, P2 e P3. Como ilustrado na figura 3 realizamos a coleta dos sinais provenientes dos pontos de acesso nos pontos de interesse utilizando um dispositivo receptor capaz de captar os sinais, e a partir desses sinais construímos e armazenamos em um banco de dados os *Wi-Fi Fingerprints* de cada ponto de interesse. Dependendo da implementação é possível armazenar mais de um *Wi-Fi Fingerprint* para cada ponto de interesse.

Figura 2 – Pontos de acesso distribuídos no ambiente.



Fonte: Elaborada pelo autor

Figura 3 – *Wi-Fi Fingerprints* a partir dos pontos de interesse.

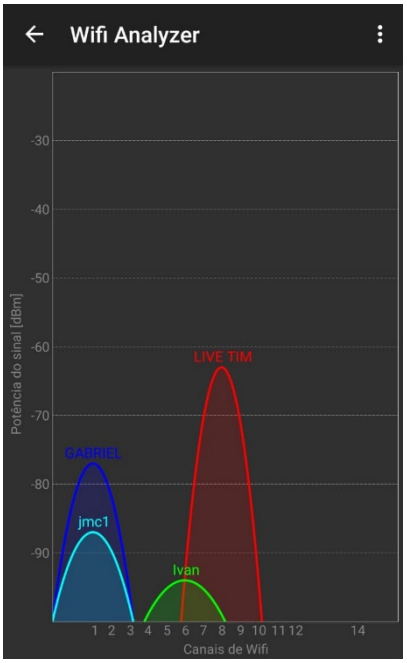


Fonte: Elaborada pelo autor

3.2.2 Construção do *Fingerprint*

Ao realizarmos a coleta de dados em um ponto de interesse P obtemos um vetor V (equação 1) de N valores de potência de sinal em dBm (dBm - indica a intensidade de potência de sinal em um determinado ponto) proveniente dos N pontos de acesso (A1 … An) cujos sinais incidem sobre a posição P, esse vetor V será o *Wi-Fi Fingerprint* para a localização P. No caso da figura 4, nosso vetor V seria constituído baseado nos valores de potência de sinal das quatro redes encontradas. Basicamente o banco de dados é constituído por uma série de vetores que são correspondentes aos pontos de interesse e serão utilizados para comparações com a posição atual.

Figura 4 – Intensidade dos sinais dos pontos de acesso.



Fonte: Medição realizada no aplicativo *WIFI ANALYZER*

$$V = [a1, \ldots, an] \quad (1)$$

3.2.3 Determinação da localização através do *Wi-Fi Fingerprint*

Existe uma série de técnicas utilizadas para a determinação da localização atual utilizando *Wi-Fi Fingerprint*; uma das mais utilizadas é a de aproximação por probabilidade chamada de Distância Euclidiana onde uma leitura dos sinais provenientes dos pontos de acesso que incidem na posição atual é realizada e um vetor S (equação 2) é construído.

$$S = [S1, \ldots, Sn] \quad (2)$$

A distância entre todos os vetores dos *Wi-Fi Fingerprints* presentes no banco de dados (V1… Vn) e o vetor atual do *Wi-Fi Fingerprint* S é calculada, o vetor mais próximo no banco de dados é escolhido. Para realizar o cálculo a seguinte fórmula é utilizada (3):

$$DistEuc(V,S) = \sqrt{\sum_{i=1}^{n} (V_i - S_i)^2} \quad (3)$$

Existem outros métodos de aproximação por probabilidade como o *Manhattan distance* (equação 4) que é definido no artigo *Performance analysis of fingerprinting algorithms used in wlan positioning system* (MRINDOKO; TITO; RUAMBO, 2016) como a soma das diferenças absolutas dos valores entre os *Wi-Fi Fingerprint* presentes no banco de dados (V1… Vn) e o *Wi-Fi Fingerprint* da posição atual (S). Por fim, um fator importante é o de que a ordem dos valores obtidos dos pontos de acesso no vetor precisa ser idêntica para que esses algoritmos funcionem.

$$ManDist(V,S) = \sum_{i=1}^{n} |V_i - S_i| \quad (4)$$

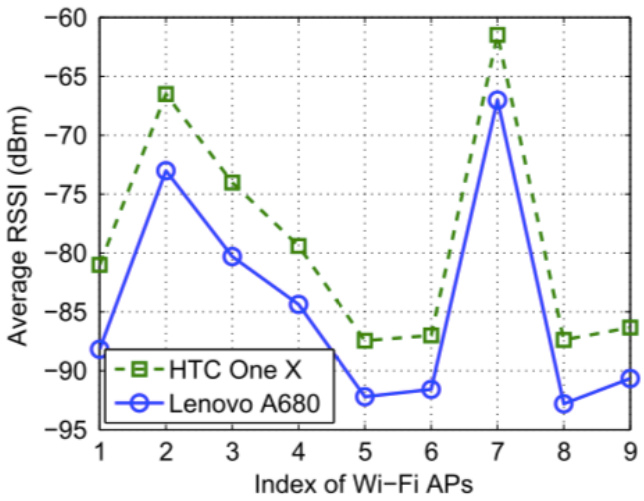
3.3 Problemas enfrentados com o uso do *Wi-Fi Fingerprint*

Como já citado anteriormente a disposição de obstáculos dentro de um ambiente interno muda constantemente. Um *Wi-Fi Fingerprint,* por ser um identificador gerado a partir de sinais, jamais é idêntico ao *Wi-Fi Fingerprint* gerado anteriormente para o mesmo ponto de interesse. Por essa razão o uso de algoritmos para comparação entre o *Wi-Fi Fingerprint* que acabou de ser gerado e os já armazenados é essencial. O resultado disso é que mesmo que um *site-survey* já tenha sido realizado no ambiente, os *Wi-Fi Fingerprints* dos pontos de interesse muito provavelmente terão que ser renovados periodicamente para continuarem válidos já que a disposição de obstáculos do ambiente possui influência direta na propagação do sinal. O artigo *Wi-Fi Localization Using RSSI Fingerprinting* (MAHAJAN; CHANANA, 2012) demonstrou que existe uma diferença enorme nos resultados obtidos em dois testes realizados com uma semana de diferença, sendo que o primeiro foi realizado logo depois do *site-survey*. É importante notar também que a intensidade de um sinal diminui conforme a distância do ponto de acesso (de onde o sinal é originário) aumenta. Outro fator crucial são os receptores de sinal: receptores diferentes são capazes de captar sinais com intensidades distintas, assim caso um site-survey do ambiente externo tenha sido realizado com um receptor mais sensível e o receptor utilizado para captar o *Fingerprint* da posição atual seja menos sensível existirá uma discrepância entre os valores. Essa diferença também existe entre receptores *Wi-Fi* de

celulares smartphones, como apontado no artigo *Wi-Fi Fingerprint-Based Indoor Positioning* (SUINING; CHAN, 2015). Fatores como: *chipset* dos smartphones, posição da instalação da antena e até mesmo a diferença entre as versões do sistema operacional podem influenciar na taxa de detecção e número de pontos de acesso detectados.

A figura 5 mostra a diferença de captação de sinal entre dois aparelhos smartphones HTC One X e Lenovo A680, aparelhos que utilizam versões diferentes do sistema operacional *Android*, lançados no ano de 2012 e 2014 respectivamente. Fica evidente o potencial impacto de utilização de diversos receptores diferentes durante a implementação e utilização de *Fingerprint.*

Figura 5 - Diferença de captação de sinal entre *smartphones*



Fonte: SUINING; SHUENG-HAN (2016, p.481)

3.4 Precisão do Wi-Fi Fingerprint

No artigo *An Improved Algorithm to Generate a Wi-Fi Fingerprint Database for Indoor Positioning* (SUINING; CHAN, 2015), testes foram realizados utilizando o mesmo dispositivo receptor dos sinais (tablet Lenovo X220) com diferentes métodos que englobavam formas de coleta, armazenamento e comparação entre os *Wi-Fi Fingerprints* para eleger a posição atual. Os resultados desses testes demonstraram discrepâncias de até um metro de diferença entre os métodos. Como apontado também pelo artigo *Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons*, os diferentes métodos citados no artigo como *Zee*, *XINS*, *Graph-*

Fusion, *HMM Fusion*, *Moloc* e *MapCraft*, possuem diferentes graus de precisão que neste caso variavam acima de um metro de diferença entre os resultados de um método e outro.

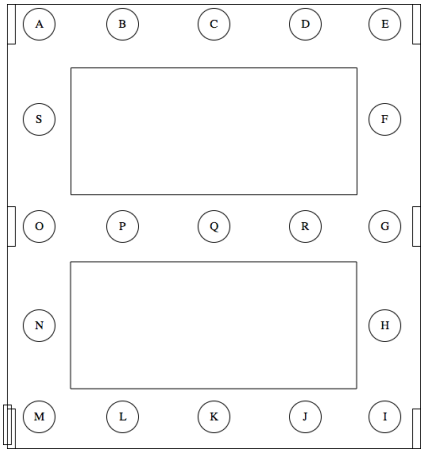
Para aumentar a precisão do *Wi-Fi Fingerprint*, podemos utilizar o auxílio de observações espaciais (padrões de caminhada dentro do ambiente ou padrões de sinal durante o trajeto) ou temporais (pico de leitura do sinal, onde um usuário caminha pelo ambiente enquanto uma série de leituras de sinais é realizada), o sistema então detecta o pico e encontra a localização correspondente através dos sinais armazenados. Muitas dessas soluções utilizam sensores já presentes dentro de dispositivos como celulares smartphones, porém nada impede o uso de outros sensores para aumentar a precisão.

3.5 Rotas

Um item muito importante que compõe um sistema de navegação é a criação de rotas, o software de navegação deve ser capaz de montar uma rota do ponto A ao B e guiar o usuário através dela. As figuras 6 e 7 mostram uma planta com um conjunto de pontos de interesse (A, B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R e S) e uma estrutura de grafo (figura 6) correspondente que relacionará os pontos de interesse que chamaremos de vértices e os corredores entre eles que serão as arestas. Através dessa estrutura podemos enxergar a ligação que existe entre cada ponto e como podemos navegar do ponto A ao ponto Q, por exemplo.

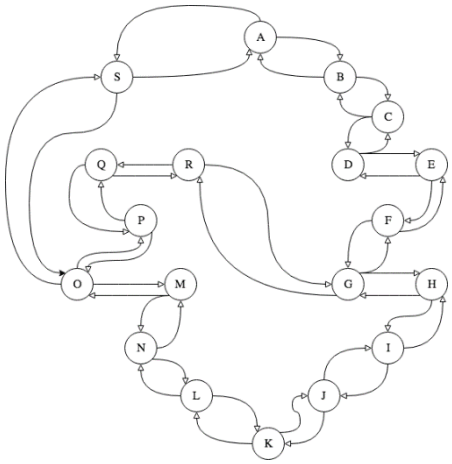
A partir da estrutura do grafo podemos aplicar algoritmos conceituados como *Dijkstra* ou A* (A - estrela) para resolução dos mesmos. Embora no exemplo da figura 7 o caminho seja simples, esses grafos podem se tornar estruturas bastante complexas, como por exemplo, mapas de países. Para esses casos é imprescindível o uso de algoritmos capazes de encontrar os caminhos entre os dois pontos de interesse.

Figura 6 - Planta



Fonte: Elaborada pelo autor

Figura 7 – Grafo baseado na planta



Fonte: Elaborada pelo autor

3.5.1 Algoritmo A*

O algoritmo A* é um algoritmo muito utilizado em *path finding* (busca de caminho). O algoritmo A* foi descrito pela primeira vez em 1968 e é uma extensão do algoritmo *Dijkstra* pois tem como objetivo encontrar o caminho ou atravessar o grafo sempre buscando o caminho mais curto. Diferentemente do algoritmo *Dijkstra,* o algoritmo A* sempre analisa as opções de caminho e considera o caminho com a menor distância além do custo de travessia entre os nós. Se tomarmos a estrutura de grafo da figura 7 como exemplo, poderíamos aplicar para as arestas que ligam os nós diferentes valores que representariam o custo de travessia, além de identificar a distância de cada nó em relação ao nó de destino, esses valores poderiam ser a distância em metros entre os nós, lembrando que cada nó equivale a um ponto de interesse da figura 6. Se aplicássemos o algoritmo A* para buscar o caminho entre dois

nós da estrutura de grafo ele traçaria uma rota verificando o custo de travessia entre os nós em conjunto com a distância entre o nó de destino e o nó atual. Dessa forma o algoritmo evita que caminhos mais longos, porém com custos de travessia entre nós menores sejam escolhidos. Para verificar esse custo de travessia diversas equações podem ser utilizadas, entre elas estão as equações já citadas no artigo, distância Euclidiana e *Manhattan distance* (equações 3 e 4 respectivamente).

Para realizar tal tarefa o algoritmo mantém uma pilha de prioridade, o algoritmo analisa os nós e seus caminhos considerando a distância de travessia entre os nós e a distância do nó em relação aos nós de destino e adiciona o nó em uma pilha de prioridade, essa pilha de prioridade é organizada pela proximidade dos nós em relação ao destino em ordem decrescente. O algoritmo também deve armazenar o caminho percorrido para chegar aos nós que estão na pilha de prioridade. Quando o nó de destino chega ao topo da pilha de prioridade basta verificar o nó anterior e assim sucessivamente para se realizar o caminho inverso e assim obter um trajeto.

3.6 *Machine Learning*

Machine Learning é um subcampo da ciência da computação, seu processo consiste em extrair informações úteis a partir de dados. Seu uso vem ganhando força nos últimos anos e está presente em muitas das aplicações com as quais interagimos no dia a dia, como por exemplo recomendações automáticas e reconhecimento de rostos. R.S. Michalski, J.G. Carbonell e T.M. Mitchell, na introdução do livro, "*Machine learning: An Artificial Intelligence Approach*" define "*Machine Learning*" como:

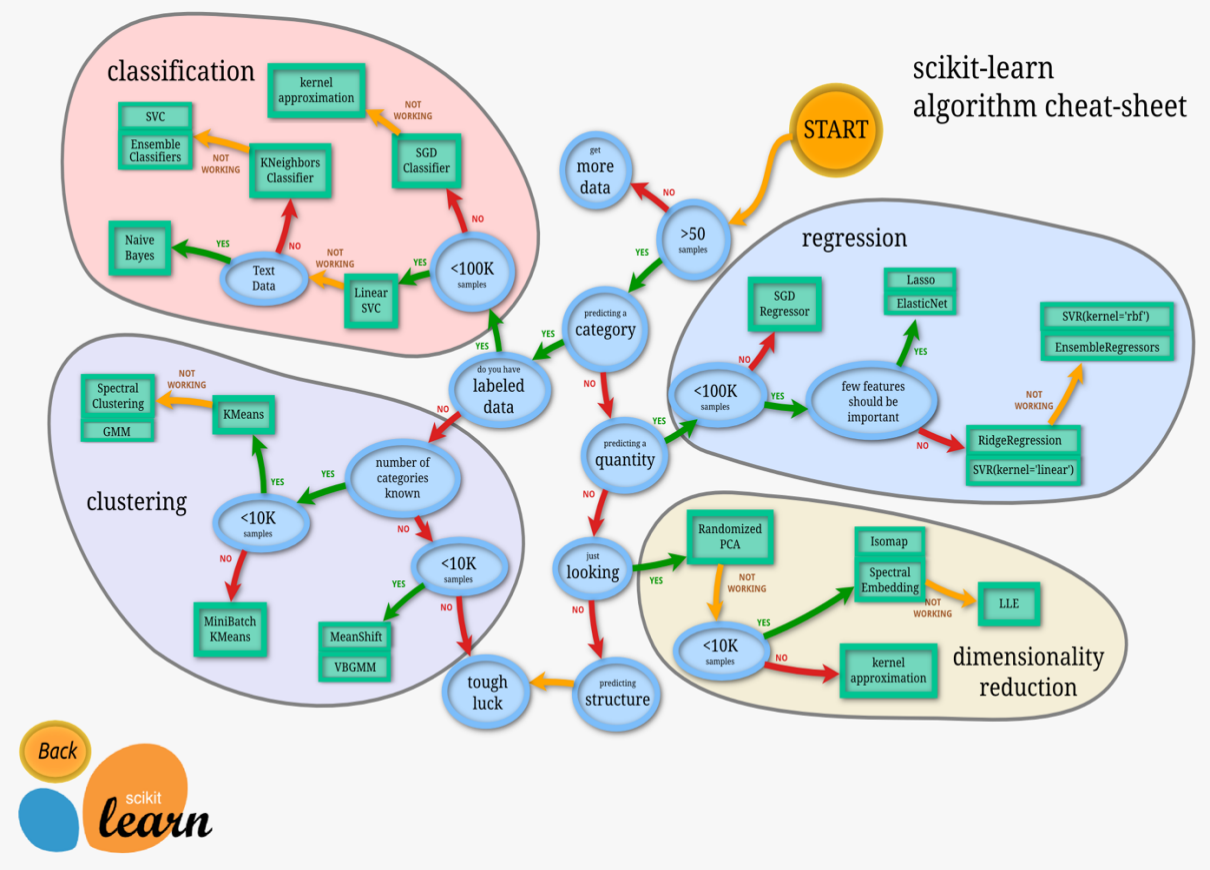
> Learning is a many-faceted phenomenon. Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observations and experimentation. Since the inception of the computer era, researchers have been striving to implant such capabilities into computers. Solving this problem has been, and remains, a most challenging and fascinating long-range goal in artificial intelligence (AI). The study and computer modeling of learning process in their multiple manifestations constitutes the subject matter of machine learning. (R.S.; J.G.; T.M., 2013, p. 3)

No campo acadêmico o uso de *Machine Learning* influenciou bastante a forma como pesquisas acadêmicas que dependiam bastante de análise de dados são realizadas. Nos primórdios da computação as aplicações consideradas como inteligentes usavam regras escritas à mão (*hard-coded*) para implementar os processos de decisão. Os dois grandes problemas que surgem a partir dessa técnica são:

1. A necessidade de o programador conhecer muito bem os dados e o processo de tomada de decisão.
2. Qualquer mudança no fluxo de tomada de decisão podia resultar na alteração do código inteiro.

Existem diversas maneiras diferentes de implementar técnicas de *Machine Learning* como mostrado na figura 8. O fator determinante para escolher uma técnica é o *Dataset*, o *Dataset* consiste basicamente dos dados dos quais desejamos extrair informação, por exemplo: uma série de fotos de animais onde o objetivo é reconhecer gatos. Outro fator muito importante é possuirmos ou não "*labeled data*" no nosso *Dataset*, ou seja, se as fotos que possuem gatos em suas imagens estão marcadas como positivo para a presença do animal. Caso as imagens não estão identificadas previamente com a presença do animal, será necessário agrupar as imagens identificando semelhanças entre as mesmas utilizando técnicas de "*clustering*" do *Machine Learning*. O tamanho do *Dataset* também é algo que deve ser levado em conta ao escolher uma técnica já que existem algoritmos mais otimizados para grandes volumes de dados, por fim vale lembrar que o conteúdo presente no *Dataset*, mais especificamente o tipo de dado a ser analisado também influencia na escolha do algoritmo a ser utilizado.

Figura 8– Exemplos de algoritmos de *Machine Learning*



Fonte: SCIKITLEARNING (2018)

O processo de *Machine Learning* começa com a preparação do *Dataset*, por exemplo: ordenar os dados ou remover informação desnecessária (escolher as características relevantes do *Dataset* para o treinamento). Posteriormente é necessário dividir o *Dataset*, 75% do mesmo deve ser destinado a treinamento do modelo, e 25% para teste da acurácia do mesmo, embora esses valores possam variar de acordo com o *Dataset* e a técnica implementada esses geralmente são bons valores iniciais. Como demonstrado no livro *Introduction to Machine Learning With Python* (GUIDO; MÜLLER, 2017), não devemos utilizar os mesmos dados usados no treinamento para avaliar o modelo pois o mesmo já possui as respostas para aqueles dados e assim sempre retornará a resposta correta. Diversos treinamentos podem ser realizados em um mesmo *Dataset* otimizando os parâmetros disponíveis em cada Algoritmo para obter maior acurácia. Uma parte importante do processo de treinamento é encontrar o ponto correto entre um modelo muito genérico e um modelo muito específico. Modelos cujos resultados de precisão nas fases de teste e treinamento são muito próximos geralmente indicam modelos muito genéricos e indicam casos de subajustes também chamados de *underfitting*. Já modelos com resultados

altos no treinamento, porém com performance ruim nos testes indicam um modelo muito específico e incapaz de lidar bem com novos dados submetidos ao mesmo, esses modelos são resultado do sobreajuste também chamado de *overfitting*.

Ao fim do processo de treinamento será criado um modelo para o qual podemos submeter novos dados (que devem obrigatoriamente estarem formatados da mesma forma que o *Dataset* inicial) e realizar predições.

3.6.1 Classificação

Como citado anteriormente, existem diversas técnicas de *Machine Learning*, uma das mais difundidas é a classificação. A classificação é um modelo de aprendizado supervisionado de *Machine Learning*, nela o objetivo é predizer uma classe baseado nos dados inseridos no modelo. A classe que também pode ser chamada de *label* (etiqueta) consiste na denominação correspondente a um determinado conjunto de dados, ou seja, todo conjunto de dados é correspondente a uma classe ou *label* (etiqueta). A classificação em *Machine Learning* se divide em dois grandes grupos: binária e multiclasse.

3.6.1.1 Classificação Binária

Na classificação binária possuímos apenas uma classe denominada positiva e outra denominada negativa, o exemplo anterior de reconhecer um gato em uma foto consiste justamente de uma classificação binária. Nesse exemplo a presença do gato ocasionaria no modelo entregando a classe positiva como resultado onde a classe positiva poderia ser equivalente ao *label* (etiqueta) “Existe um gato na imagem”. Já para o caso de o modelo não identificar nenhum gato na foto a classe resultante seria a negativa onde o *label* (etiqueta) seria equivalente “Não existe um gato na imagem”.

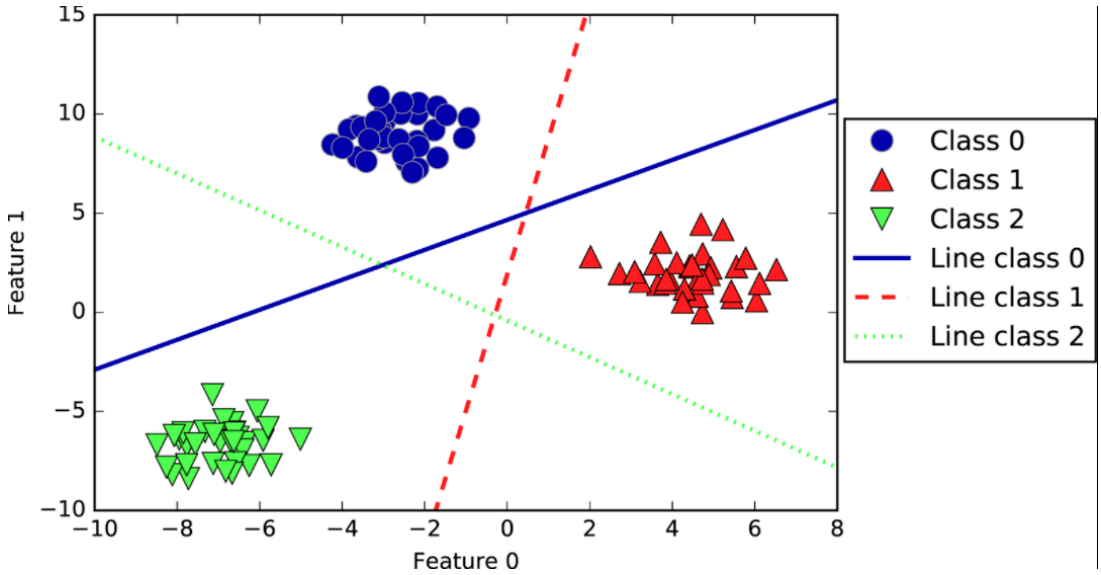
3.6.1.2 Classificação Multiclasse

A classificação multiclasse expande a classificação binária para n número de classes. Voltando ao exemplo do gato novamente podemos imaginar um modelo capaz de reconhecer a raça do gato da imagem, neste caso cada raça de gato possível seria equivalente a uma classe. Existem maneiras distintas de implementar algoritmos de classificação multiclasse, uma forma bem difundida é o conceito de

“One-vs.-All“ ou “Um contra todos”. Nesse modelo o algoritmo cria uma série de classificadores binários de tamanho equivalente ao número de casses presentes no modelo, cada classificador recebe uma das classes do modelo como classe positiva. Se considerarmos o modelo de classificação multiclasse para a raça de um gato em fotos, cada classificador analisaria a possibilidade do gato da foto pertencer a determinada raça (classe positiva) ou não pertencer a determinada raça (classe negativa), ou seja, todas probabilidades do gato pertencer a outra raça estariam contidas na classe negativa. O classificador multiclasse neste caso submeteria o mesmo conjunto de dados para todos os classificadores binários e então retornaria como predição o resultado com maior probabilidade entre todos os classificadores binários.

A figura 9 mostra um modelo de *Machine Learning* para um classificador multiclasse de 3 classes que recebe como entrada duas características (*features*) que são números inteiros. Os círculos e triângulos representam pontos (compostos pelas duas características) distribuídos ao longo do gráfico. As linhas: azul, verde e vermelho representam a separação que cada um dos classificadores binários realizaria nos dados para considerar as classes positivas e o resto.

Figura 9 – Classificador multiclasse



Fonte: KIRK, Matthew (2015, p. 98)

3.7 *Python*, Pandas, NumPy e *Scikit-learn*

Como citado anteriormente uma boa parte do processo de treinar um modelo de *Machine Learning* reside na extração, organização, limpeza e formatação dos seus dados. *Python* é uma linguagem interpretada de alto nível lançada em 1991, um dos seus fundamentos é a simplicidade e a facilidade de leitura, por esses motivos e por conter um vasto ecossistema de bibliotecas que ajudam na manipulação de dados *Python* é uma das linguagens mais utilizadas na implementação de técnicas de *Machine Learning*. Uma das bibliotecas que ajudam a compor o ecossistema do *Python* é o Pandas, essa biblioteca disponibiliza uma estrutura de dados denominada DataFrame, um DataFrame é bem similar a uma tabela e por meio dessa estrutura de dados somos capazes de manipular nosso *Dataset* de diversas formas. Outro pacote fundamental para realizar computação cientifica com *Python* é o NumPy, esse pacote nos concede várias funcionalidades para manipular *arrays* de n dimensões (matrizes) e funções matemáticas de alto nível. Por fim, a biblioteca *Scikit-learn* oferta múltiplos algoritmos de técnicas de *Machine Learning* para que seja possível treinar um modelo para realizar predições.

4. MATERIAIS E MÉTODOS

4.1 Metodologia

O primeiro passo do projeto consistiu em encontrar o hardware base apropriado para o projeto pois a antena Wi-Fi é essencial para o projeto. Após a definição do hardware foi necessário encontrar uma forma de coletar dados sobre os pontos de acesso e partir dos mesmos construir *Wi-Fi Fingerprints*, esse processo é denominado *site-survey*. Para a realização do *site-survey* devem ser utilizados dispositivos com antenas Wi-Fi idênticas aquelas utilizadas pelo dispositivo final do usuário para que não haja qualquer discrepância. Uma ferramenta de linha de comando foi desenvolvida para que fosse possível automatizar o processo de *site-survey* em diversos dispositivos. Após a coleta dos dados foi necessário criar uma forma de armazenamento e consulta dessa informação.

Com os dados disponíveis foram realizados testes para encontrar uma forma eficiente de comparação entre os *Wi-Fi Fingerprints*, tanto em assertividade quanto em otimização computacional. Após diversas pesquisas, a técnica de *Machine Lear-*

ning como provável abordagem para o problema de comparação dos dados foi encontrada.

Variadas técnicas de *Machine Learning* foram estudadas e seguidamente algumas dessas técnicas foram escolhidas. Posteriormente a transformação dos dados armazenados para um formato aceitado pela biblioteca de *Machine Learning* surgiu como um requisito para a implementação dessas técnicas. Um processo de coleta dos dados persistidos no banco e transformação dos mesmos em um formato que serviria de entrada para os modelos de *Machine Learning* foi proposto e desenvolvido. Um outro processo de treinamento de uma série de modelos de Machine Learning utilizando os dados transformados e onde ao fim os modelos treinados seriam exportados como arquivos também foi desenvolvido.

Um processo de carregamento dos arquivos contendo os modelos e execução dos mesmos foi criado. Esse processo depois foi encapsulado em uma *API REST*, dessa forma os modelos poderiam ser executados através de requisições *HTTP*.

Por fim, o acoplamento dos sensores e atuadores ao hardware seguido pelo desenho do circuito na placa foi realizado. Com o hardware construído, alguns desenhos de carcaça foram esboçados no software *Solid Works*, uma versão final foi escolhida. O desenho foi então exportado para que pudesse ser impresso em 3D (3 dimensões) por uma das impressoras presentes no laboratório do Senac.

O último passo de elaboração do projeto consistiu em desenvolver o software presente no hardware responsável por guiar o usuário, receber entradas do mesmo e realizar a comunicação entre o dispositivo e a API.

Uma área de um dos prédios acadêmicos foi definida para os testes e as coletas de Wi-Fi Fingerprints foram realizadas nessa área. Um Dataset contendo todos os Wi-Fi Fingerprints dessa área foi montado e submetido como entrada para todas as técnicas de Machine Learning escolhidas anteriormente. O objetivo foi analisar as características dessas técnicas conforme variação do tamanho do Dataset e parametrização das mesmas. A técnica que apresentou o melhor desempenho foi esco-

lhida como técnica presente na API para que os testes com o dispositivo utilizassem o melhor algoritmo.

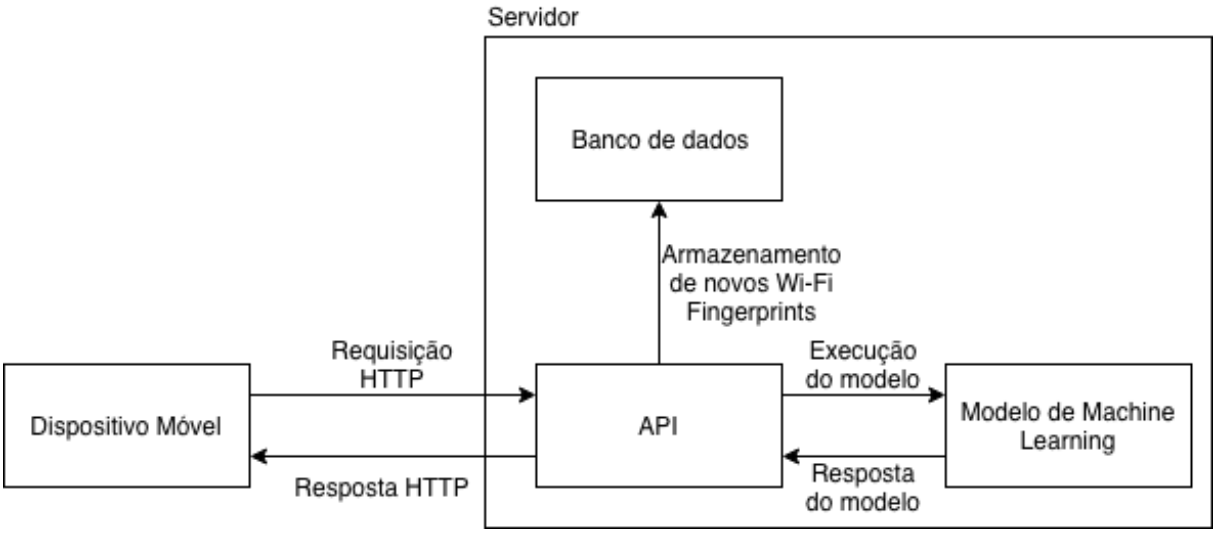
4.2 Visão geral do sistema

Para guiar usuários deficientes visuais pelo campus do Senac Santo Amaro, desenvolvemos um dispositivo portátil que o usuário carrega consigo. O aparelho é capaz de identificar sua posição atual dentro do campus e orientar os usuários por voz até um destino escolhido que pode ser uma sala ou laboratório dentro dos prédios acadêmicos 1 e 2. Para que o equipamento seja capaz de identificar sua posição atual, o mesmo envia os *Wi-Fi Fingerprints* para um servidor que é responsável por submeter esses dados para um modelo de *Machine Learning* previamente treinado. O servidor por sua vez retorna a resposta do modelo para o dispositivo móvel. Dessa forma o dispositivo pode se localizar dentro dos prédios acadêmicos e guiar um usuário até o destino utilizando algoritmos de busca de caminho e análises periódicas tanto do *Wi-Fi Fingerprint* quanto de outros sensores. A implementação da técnica do *Wi-Fi Fingerprint* requer um processo de *site-survey* inicial dos ambientes elegendo pontos de interesse dentro dos prédios como: salas, laboratórios e entradas para rampas de acesso. Como citado anteriormente é necessário manter os dados dos *Wi-Fi Fingerprints* atualizados para que o dispositivo continue funcionando corretamente. Para que seja possível a adaptação automática do sistema às diferenças advindas dos *Wi-Fi Fingerprints* armazenados no banco de dados em relação aos *Wi-Fi Fingerprints* atuais, os dispositivos enviarão para a API do servidor em conjunto com os *Wi-Fi Fingerprints* atuais, o ponto de interesse esperado para aquela posição. Caso o ponto de interesse resultante da análise do modelo de *Machine Learning* sobre os *Wi-Fi Fingerprints* seja idêntico ao passado como parâmetro pelo dispositivo, esses novos *Wi-Fi Fingerprints* serão adicionados no banco de dados do servidor e posteriormente quando o modelo de *Machine Learning* for treinado novamente esses dados estarão inclusos. Ao utilizar esse modelo o sistema sempre se mantém atualizado através do uso dos próprios usuários sem a necessidade de realizar um novo processo de *site-survey* no ambiente.

A figura 10 mostra uma visão geral do sistema relacionando os dois principais componentes do sistema. O dispositivo portátil é utilizado pelo usuário para navegação através do campus do Senac Santo Amaro, recebendo os comandos de usuário,

realizando a leitura dos sinais dos pontos de acesso (*Wi-Fi Fingerprints*) e sensores, que citaremos adiante, submetendo *Wi-Fi Fingerprints para a API do servidor*, montando rotas e devolvendo orientações para os usuários. Já o servidor possui o seu próprio banco de dados contendo os *Wi-Fi Fingerprints* mais atualizados, o objetivo é de que periodicamente o servidor treine novamente o seu modelo, para que o mesmo sempre reflita o estado atual do ambiente. Vários dispositivos em uso pelo campus podem submeter dados dos *Wi-Fi Fingerprints* encontrados para o servidor através da API.

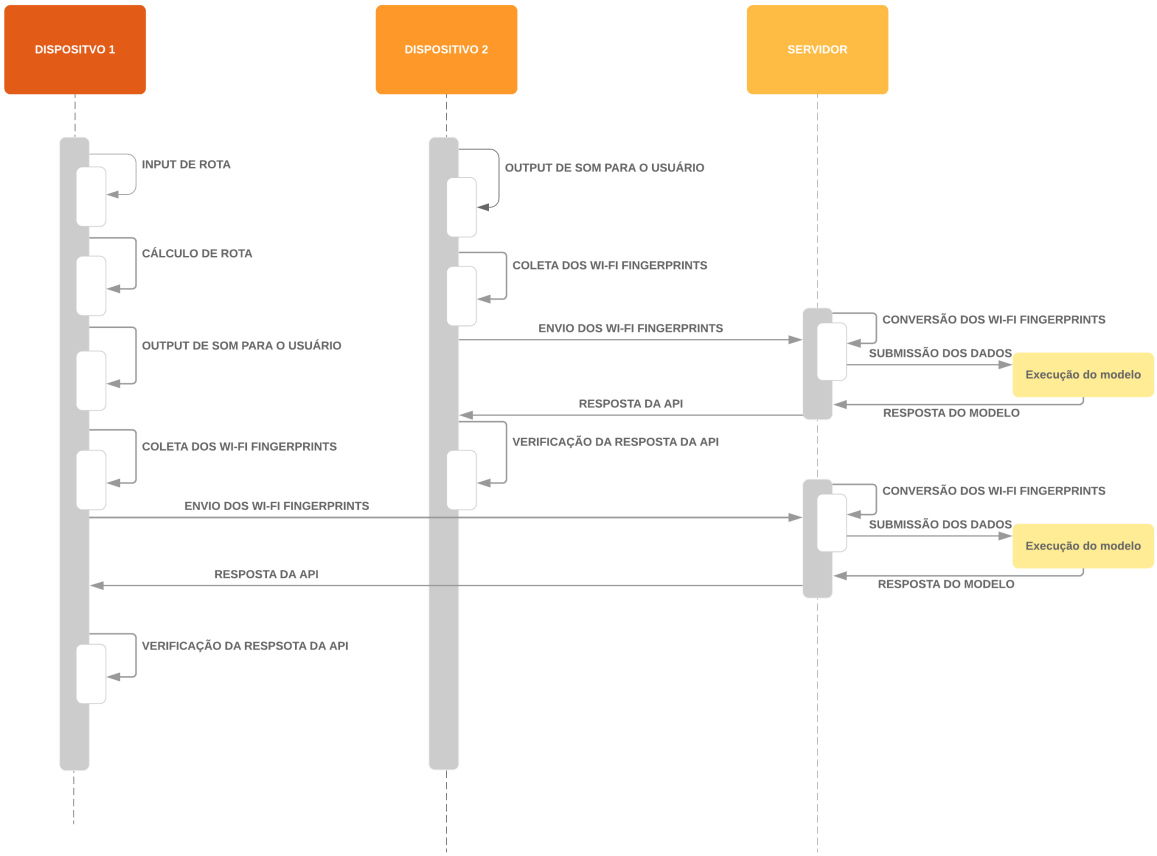
Figura 10 – Visão geral do sistema



Fonte: Elaborada pelo autor

A figura 11 apresenta, através de um diagrama de sequência como os dispositivos diferentes coletam e enviam *Wi-Fi Fingerprints* para serem executados no modelo de *Machine Learning* presente no servidor e aguardam sua resposta para continuarem o trajeto. Essa arquitetura de funcionamento possibilita a consulta continua do modelo treinado por diversos dispositivos além de possibilitar um modelo de *Machine Learning* e um *Dataset* para treinamento do mesmo, maiores do que o hardware de um dispositivo móvel aguentaria.

Figura 11 – Diagrama de sequência da interação entre os dispositivos e o servidor

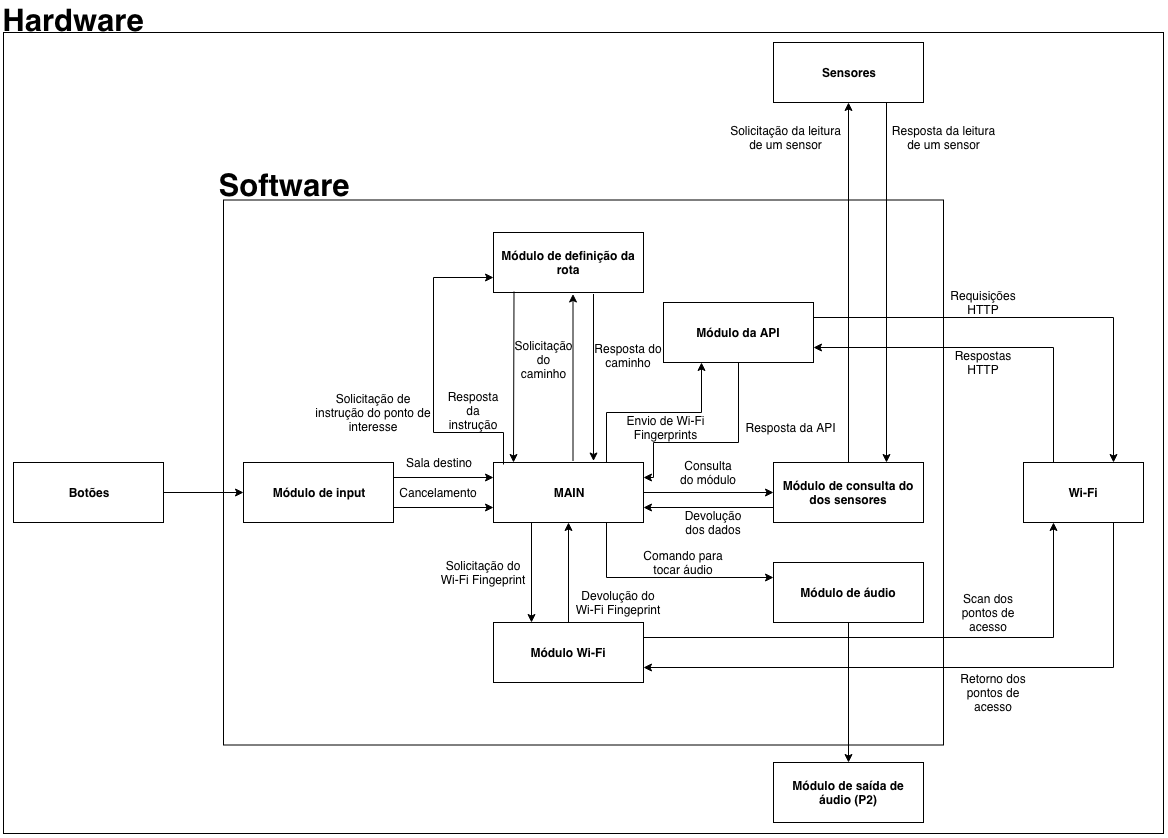


Fonte: Elaborada pelo autor

4.3 Diagrama do dispositivo

A figura 12 mostra o diagrama dos dispositivos que serão utilizados pelo usuário. O diagrama relaciona todos os módulos que compõem um dispositivo e troca de mensagens que existe entre cada um deles. Todos os dispositivos devem ser capazes de funcionar de forma independente utilizando sua própria cópia local dos dados dos *Wi-Fi Fingerprints* e seu conjunto de sensores que auxiliarão na determinação da localização.

Figura 12 - Diagrama do dispositivo



Fonte: Elaborada pelo autor

4.3.1 Raspberry

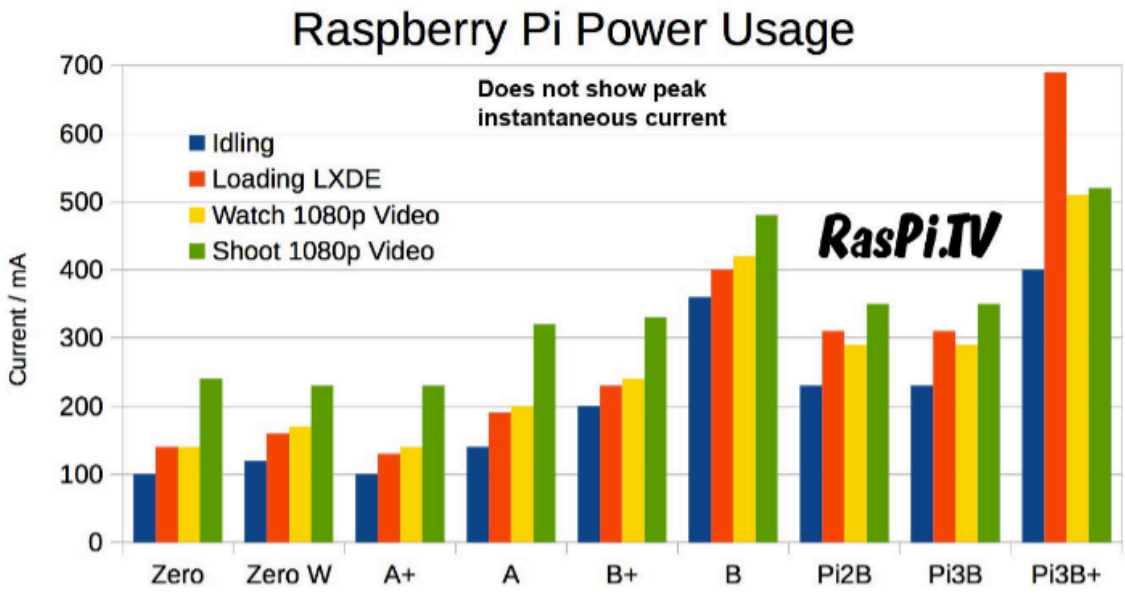
Os requisitos desse projeto para se determinar a placa de desenvolvimento eram: Os requisitos para determinar o hardware para o projeto eram:

- Conectividade *Wi-Fi* (embutida ou não).
- Capacidade para comportar um sistema operacional.
- Baixo custo.
- Baixo consumo de energia.

Realizamos uma pesquisa nas principais opções de compra e a que mais atendeu os requisitos foi o modelo Raspberry PI Zero W. O modelo em questão possui *Wi-Fi* on-board, preço baixo e seu consumo de energia é aceitável.

Conforme mostrado na Figura 13, o consumo em *idling* (ocioso) é o terceiro mais baixo, consumindo cerca de 0,62W perdendo somente para a sua versão sem *Wi-Fi*, Zero e para a versão A+, que consomem cerca de 0,51W.

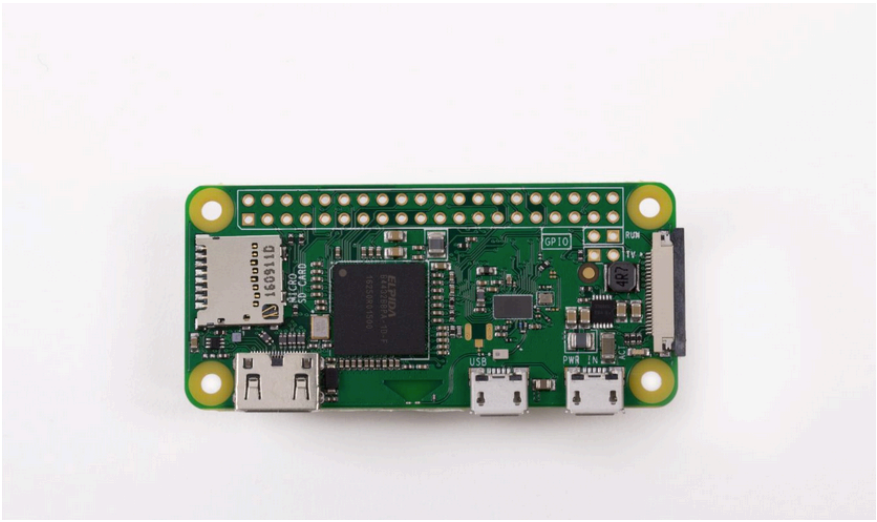
Figura 13 – Comparativo de consumo de energia entre os modelos *Raspberry Pi*



Fonte: RASPI.TV, 2017

Dentre os 3 modelos das placas Raspberry com menor consumo de energia os modelos Raspberry Pi Zero e Raspberry Pi Zero W são os mais baratos além de menores, cerca de metade do tamanho do modelo Raspberry Pi A+. A principal diferença entre os dois modelos é a presença do módulo Wi-Fi embutido na placa. Dessa forma foi escolhido que o modelo para o projeto seria a placa Raspberry Pi Zero W (figura 14).

Figura 14 – *Raspberry PI Zero W*



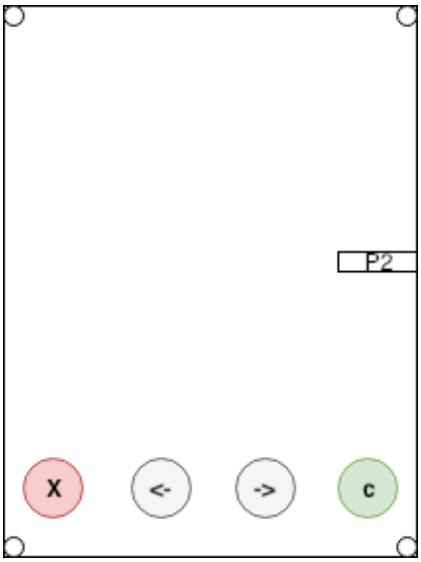
Fonte: RASPBERRYPI.ORG

Um ponto importante a ser levado em conta é que os modelos de Raspberry Pi executam um sistema operacional chamado Raspbian lite, baseado em Linux, e que pode ser usado para desenvolver em *Python*.

4.3.2 Módulo de entrada

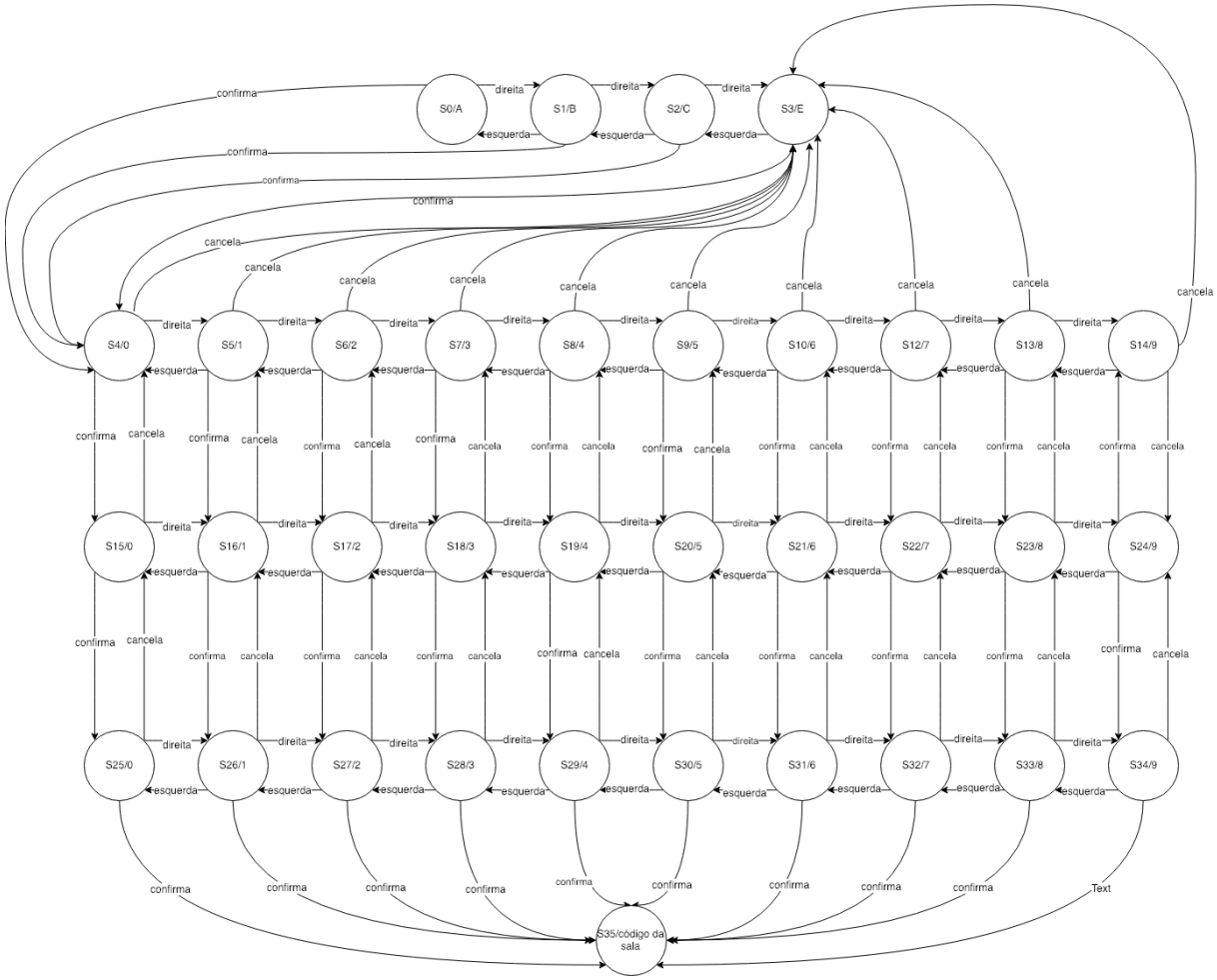
A figura 15 mostra o *Wireframe* do módulo do dispositivo com uma saída p2 para conexão do fone de ouvido e 4 botões: esquerda, direita, confirma e cancela. O dispositivo aceita um código de 4 caracteres onde o primeiro caractere é sempre uma letra seguido por 3 números, esses códigos constituem o identificador da ponto de interesse de destino, uma sala ou laboratório dos prédios acadêmicos 1 e 2. A inserção do destino ocorre a partir de 4 casas, na primeira casa o usuário pode utilizar os botões para esquerda e para direita para selecionar uma letra. Para selecionar uma letra o usuário deve apertar a tecla de confirmação, o módulo então seguirá para a próxima casa, caso o usuário confirme a letra ou número errado por engano ele pode apertar a tecla cancela para voltar para a coluna anterior, esse processo se repete até a quarta casa onde após a confirmação o dispositivo calcula a rota e começa a guiar o usuário, o processo de entrada do código da sala está representado na máquina de estados da figura 16. Como o foco do dispositivo é guiar o usuário com deficiência visual os botões são em braile e o dispositivo devolve um *feedback* de áudio sempre que o usuário altera uma letra ou número e ao confirmar ou cancelar o destino.

Figura 15 – *Wireframe* do dispositivo



Fonte: Elaborada pelo autor

Figura 16 – Maquina de estado do dispositivo



Fonte: Elaborada pelo autor

A tabela 1 mostra os códigos possíveis aceitos como destino válidos para o dispositivo. Cada um dos códigos de inserção possíveis corresponde a um laboratório ou sala de aula dos prédios acadêmico 1 e 2 do Senac.

Tabela 1 – Códigos das Salas e Laboratórios dos prédios acadêmicos 1 e 2

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J ***	**K ***
1º Andar	101 a 125	126 a 130	131 a 151	152 a 160	166 a 169	Nasa	301 a 317	318 a 325	327 a 347	351 e 353	352 a 363
2º Andar	201 a 227	228 a 230	231 a 257	258 a 271	285 a 287	272 a 284	401 a 425	426 a 437	439 a 469	471 a 481	478 a 487

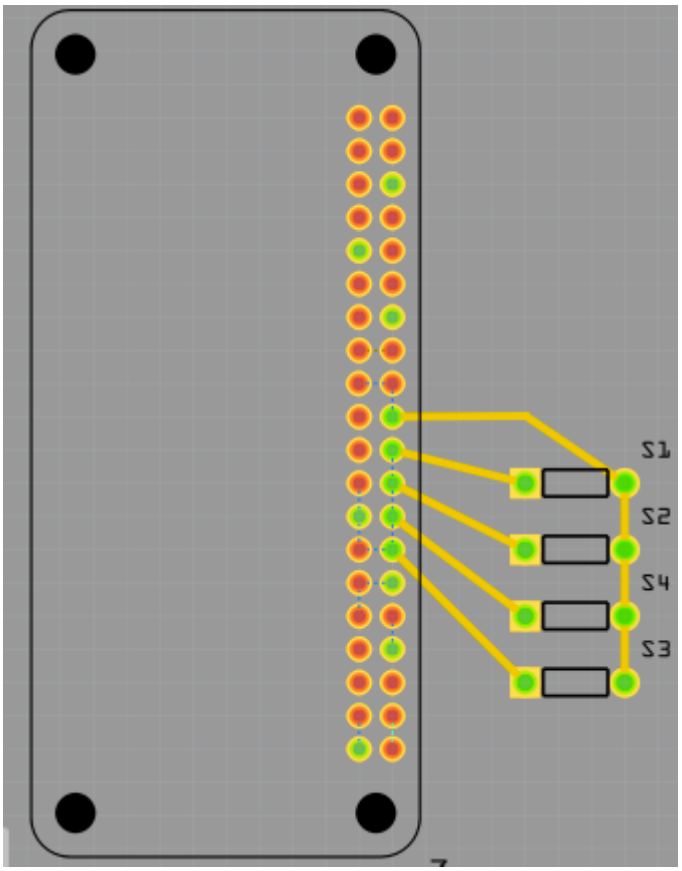
Fonte: Elaborada pelo autor

*Corredores intercalam números de salas.

4.3.3 Botoeira

Como forma de input de dados pelo usuário, desenvolvemos um módulo de botões capaz de receber qualquer tipo de retorno do usuário que seja solicitado. Foram utilizados 4 botões sem retenção, com uma entrada ligada diretamente na GPIO e a outra no GND da Raspberry. Quando um botão é pressionado, o valor do GND é transferido para a GPIO, sendo assim é possível identificar que ele foi pressionado lendo a entrada da GPIO. Conforme pode ser visto na Figura 17 as GPIOs utilizadas foram 1, 7, 8 e 25.

Figura 17 – Esquema elétrico botoeira



Fonte: Elaborada pelo autor

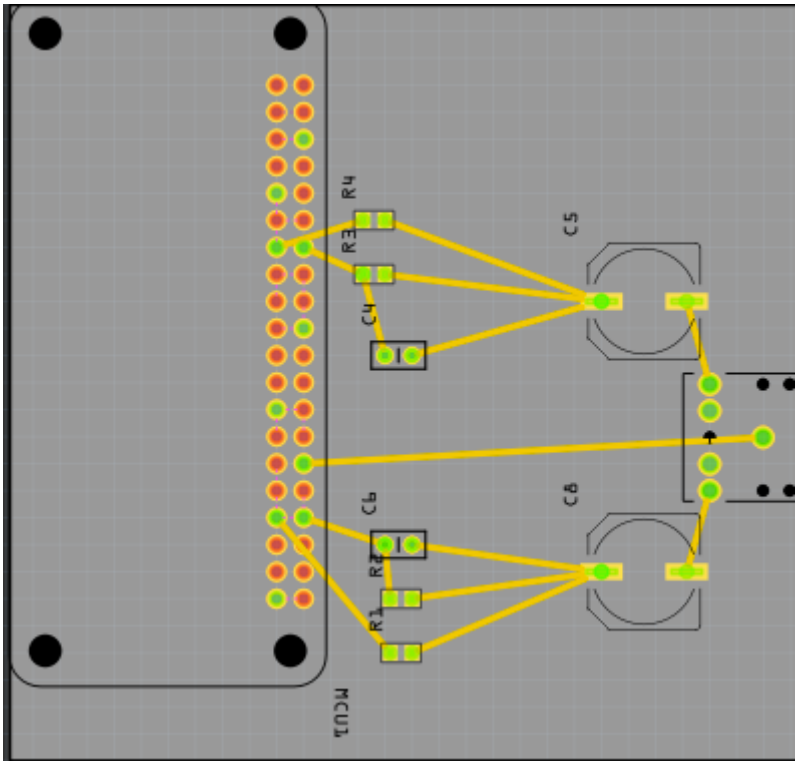
4.3.4 Módulo de *Output* de Som

Como no modelo da Raspberry escolhida não há um *output* de som nativo, foi necessário desenvolver uma solução para que o áudio pudesse ser transmitido para o usuário.

Utilizando duas portas PWM, foi criado um filtro do tipo passa baixa (low pass) para se utilizar as portas como saída de áudio estéreo. Cada porta PWM é responsável por um lado do som, esquerdo ou direito, cada lado têm um pequeno filtro para se limitar a passagem de altas frequências, possibilitando assim que se possa transmitir o áudio da Raspberry sem que hajam picos de frequência.

A Figura 18 mostra como o filtro foi criado. Foram utilizados um capacitor eletrolítico, um capacitor de cerâmica e dois resistores, de 270 ohms e 150 ohms respectivamente, para cada lado da saída de som.

Figura 18 – Esquema elétrico módulo de *Output* de som



Fonte: elaborada pelo autor

Há necessidade também de realizar algumas configurações no sistema operacional para que seja utilizado essas duas PWM como saída de áudio. A primeira foi adicionar a linha abaixo no arquivo /boot/config.txt:

```
dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4
```

Os pinos 18 e 13 são utilizados para a saída de som, os parâmetros “pwm-2chan”, “func=2” e “func2=4” são utilizados para indicar que o som será estéreo.

O próximo passo é forçar a saída de áudio na Raspberry pela conector 3,5mm jack, por isso executamos o comando “sudo raspi-config” para acessarmos as configurações da Raspberry, após isso acessamos as opções “7. Advanced Options”, “A4 Audio” e “1 Force 3.5mm (‘headphone’) jack”.

4.3.5 Módulo de sensores

Para aprimorar a precisão do *Wi-Fi Fingerprint* o dispositivo contará com dois sensores: BMP180 (pressão atmosférica) e HMC5883L (magnetrômetro).

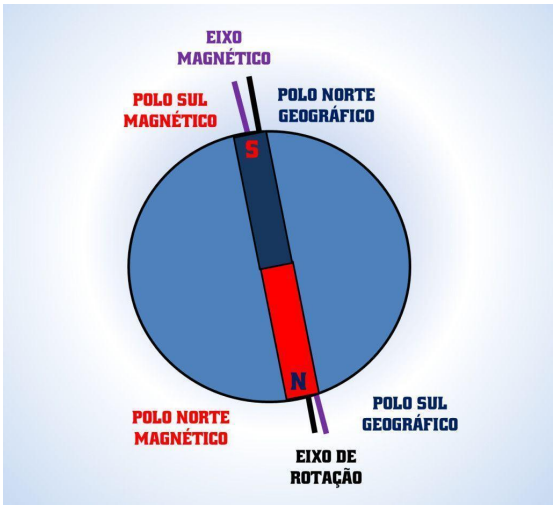
4.3.5.1 Sensor digital de pressão atmosférica

O primeiro deles é um sensor para medir pressão atmosférica, através da medição dos dados de pressão atmosférica é possível se obter a altitude em relação ao nível do mar, pois a pressão e a altitude são grandezas inversamente proporcionais. O objetivo é medir a altitude em relação ao nível do mar dos prédios acadêmicos 1 e 2 do campus Senac Santo Amaro no térreo e primeiro andar e dividir as salas e laboratórios nesses dois grupos. Com isso o sensor de digital de pressão será utilizado para que o dispositivo se localize na altitude do térreo ou do primeiro andar dos prédios acadêmicos 1 e 2, dessa forma ele pode desconsiderar todos os *Wi-Fi Fingerprints* do mesmo acadêmico que não se encontram no mesmo andar e otimizar a busca.

4.3.5.2 Módulo *Compass* - Magnetômetro

O segundo sensor é um magnetômetro, esse sensor é capaz de indicar o polo norte magnético da terra (figura 19). O objetivo é que todas os pontos de interesse tenham dados da sua posição em relação ao polo norte magnético da terra armazenados, com isso, o dispositivo será capaz de colocar o usuário sempre de frente para a sala ou laboratório no final do trajeto.

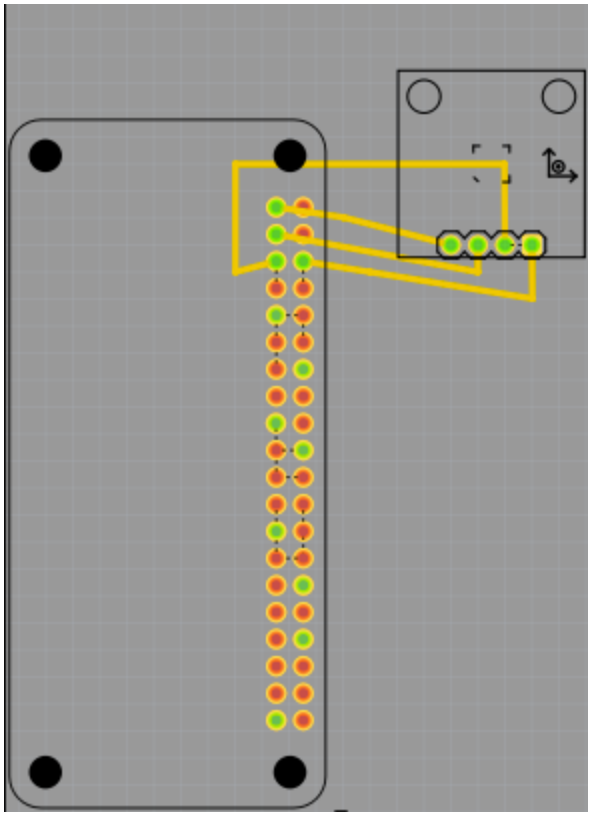
Figura 19 – Polos magnéticos e geográficos da terra



Fonte: CAROLINE, Luana

O sensor utilizado no projeto foi o HMC5883L com o objetivo de integrarmos uma bússola digital ao projeto. O HMC5883L utiliza o protocolo I2C (Inter-Integrated Circuit) das placas Raspberry Pi para se comunicar com as mesmas. A figura 20 mostra o esquemático para realizar a ligação entre o sensor e a Raspberry Pi Zero W. Embora a ligação do mesmo seja simples é preciso lidar com o processo de declinação magnética, a declinação magnética se deve ao fato de o campo magnético da Terra não estar alinhado corretamente com o seu eixo rotacional como mostrado na figura 19. Como resultado desse processo uma bússola pode não apontar para o norte verdadeiro, e é preciso calibrar a mesma. Para realizar a calibração automática da mesma o módulo de software do sensor é capaz de consultar a declinação magnética para uma determinada localização passando como parâmetros a latitude e longitude através de uma API disponibilizada pela *National Geophysical Data Center*.

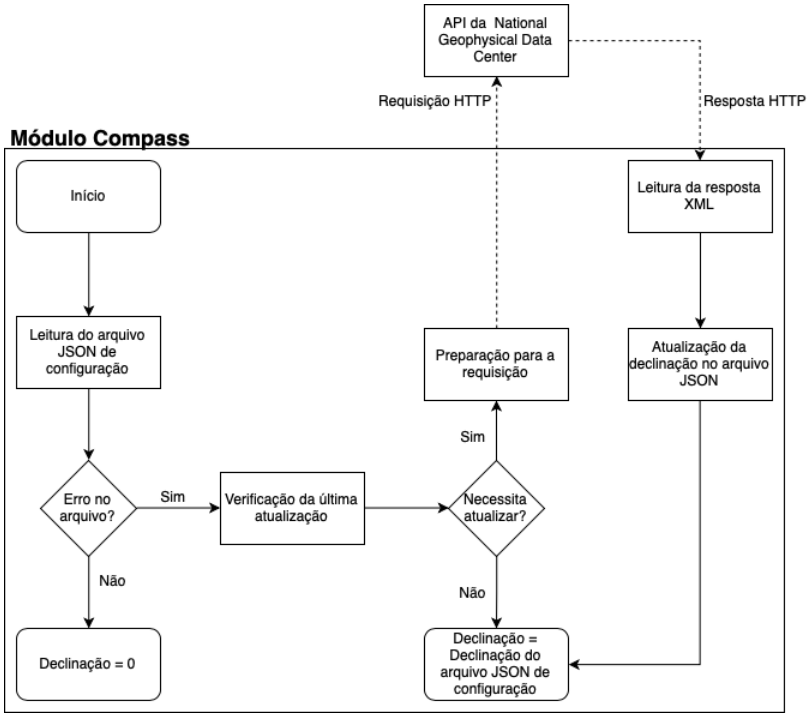
Figura 20 – Esquema elétrico Módulo *Compass*



Fonte: elaborada pelo autor

Dessa forma o sensor é capaz de realizar uma auto calibração baseada na latitude e longitude do arquivo de configuração sem que o usuário precise saber o que é declinação magnética, isso é importante pois a declinação magnética vai mudando com passar o tempo, o módulo construído é capaz de verificar a última atualização realizada e baseado nessa data realizar ou não uma chamada na API como mostrado na figura 21.

Figura 21 – Fluxo de decisão baseado no módulo *Compass*



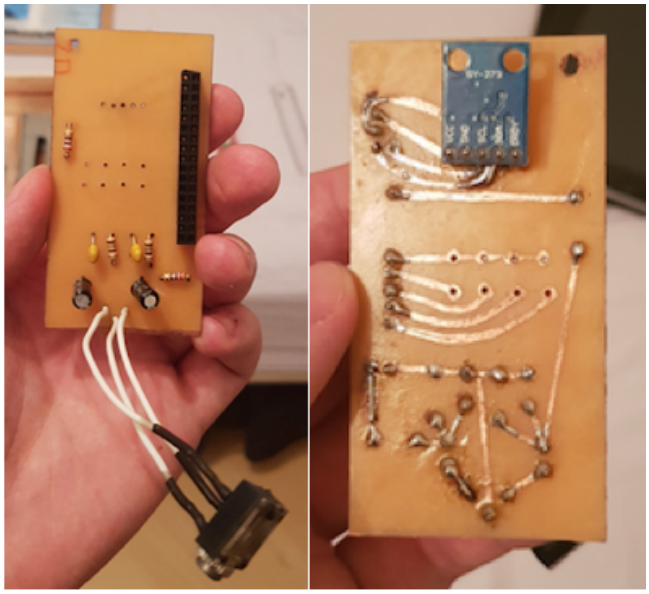
Fonte: Elaborado pelo autor

4.3.6 *Shield*

A principio, o POC seria realizado com um protótipo, por isso utilizaremos placas de prototipagem e jumpers para ligarmos os módulos, porém como esse tipo de técnica não é confiável quando não está parado em uma bancada de teste, identificamos a necessidade de criar algo que fosse estável e resistente a uma possível queda ou falha de alguma conexão por exemplo. Com isso, os módulos foram unidos em um *Shield*, com o intuito de reduzir o tamanho do circuito.

Após o desenvolvimento de todos os módulos separadamente, foi possível projetar um *Shield* em circuito impresso, para que seja possível inseri-lo e remove-lo da Raspberry a qualquer momento.

Figura 22 – *Shield* desenvolvido

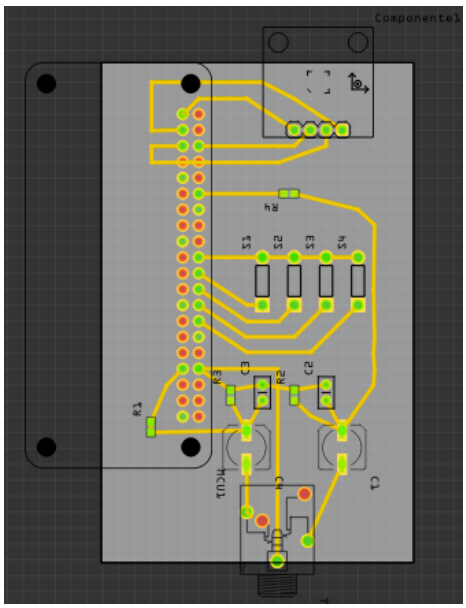


Fonte: Elaborado pelo autor

O primeiro passo para desenvolver o *Shield* foi traçar as trilhas em um rascunho, definindo onde cada componente iria ficar.

Após, foi realizado um projeto em um software para validar e simular todas as conexões.

Figura 23 – Esquemático do *Shield*



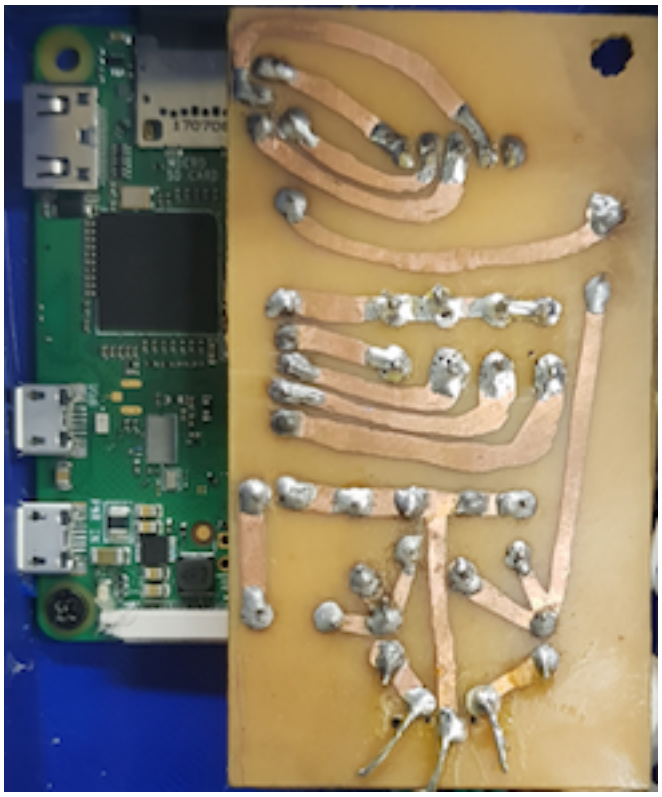
Fonte: Elaborado pelo autor

Começando o processo de impressão do circuito, a placa de fenolite foi cortada e furada nas devidas proporções. Passado então o desenho das trilhas a mão

para a placa utilizando a caneta condutiva para impedir que o cobre seja corroído mais tarde.

A placa foi mergulhada no percloreto de ferro, eliminando assim o cobre desnecessário, deixando apenas os que foram cobertos pela caneta. Para finalizar o processo, é retirado toda a tinta da caneta com palha de aço expondo as trilhas de cobre e os componentes são soldados na placa.

Figura 24 – *Shield* finalizado



Fonte: Elaborado pelo autor

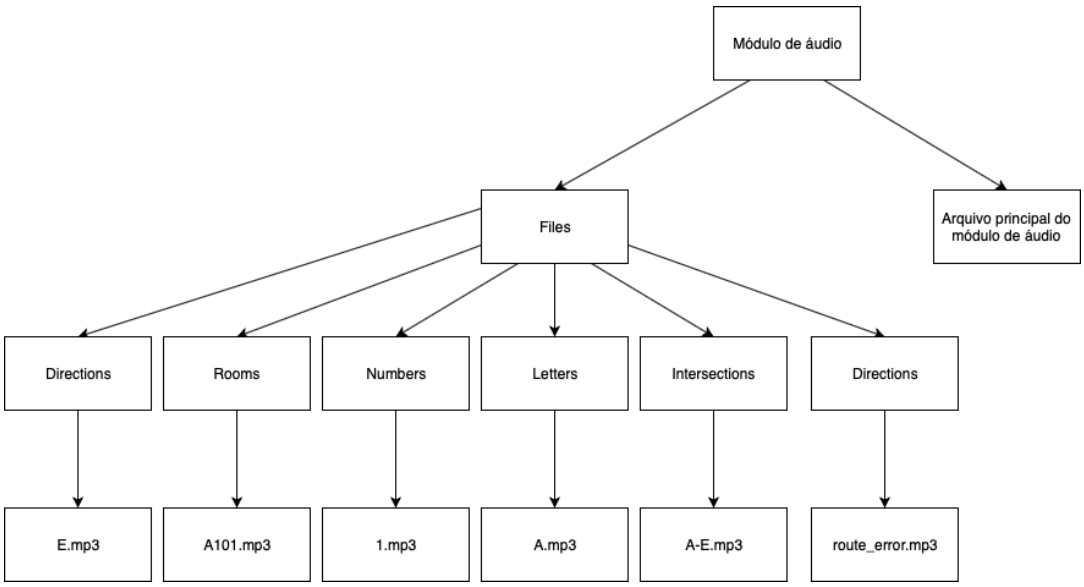
4.3.7 Módulo de áudio

O módulo de áudio é muito importante pois o dispositivo não possui uma tela, assim sendo o áudio é a única forma de *feedback* entre o dispositivo e o usuário. O áudio é necessário para auxiliar o usuário no processo de inserção da sala ou laboratório de destino, instruir o usuário durante o trajeto e até se necessário, reportar o usuário de erros ocorridos no software.

A figura 25 mostra a estrutura de pastas e arquivos do módulo de áudio. Os blocos finais são exemplos de arquivos mp3 contidos em cada uma das subpastas

da pasta “Files”. Todos os arquivos foram armazenados com nomes em inglês, pela simples questão de ser uma língua universal, porém os áudios foram todos gravados em português. Como exemplo, podemos citar o arquivo “E.mp3” (abreviação para leste) dentro da pasta “*Directions*”, que contém apenas a palavra “Leste”. Caso o idioma do áudio necessite ser alterado basta obter todas as frases em outro idioma e salvar as mesmas da mesma forma. O módulo de áudio constrói frases a partir dos arquivos de áudio da pasta “Files”, esse processo foi escolhido em detrimento de gravar exatamente todas as frases que poderiam ser ditas pelo dispositivo por uma questão de dinamismo e também por economizar espaço. Para gravar os áudios necessários o site Sound of Text foi utilizado, ele disponibiliza uma ferramenta capaz de transformar o texto inserido em arquivos de mp3 utilizando o motor “text to speech” do Google Tradutor.

Figura 25 – Arquivos de áudio



Fonte: Elaborado pelo autor

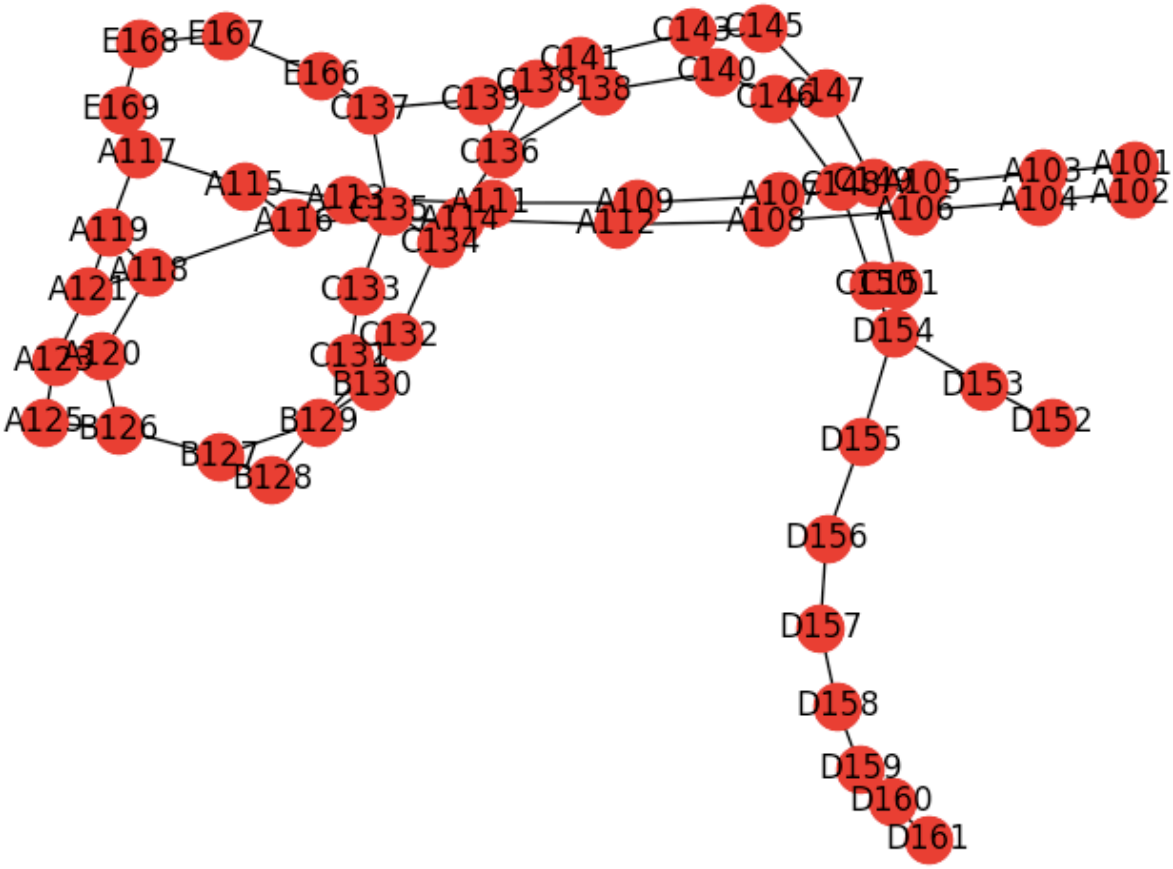
4.3.8 Módulo *PathSolver* - Definição de rota

Para guiar o usuário de um ponto ao outro o sistema deve primeiro construir um *Wi-Fi Fingerprint* para a posição atual, comparar o mesmo com os *Wi-Fi Fingerprints* do banco de dados e identificar o ponto de interesse correspondente para a posição atual. Após a identificação da localização atual o sistema deve montar uma rota entre o ponto de interesse atual até o ponto de interesse do destino, tanto o ponto de partida quanto o destino devem ser salas ou laboratórios de qualquer andar

de um dos prédios acadêmicos do campus Senac Santo Amaro. Para que seja possível realizar tal tarefa, construiremos uma estrutura de grafo representando os prédios acadêmicos 1 e 2 do campus. O campus Senac santo amaro possui escadas, rampas de acesso e elevadores como forma de transição entre os andares de um mesmo prédio acadêmico, as rampas de acesso foram escolhidas como meio de transição entre os andares por serem mais seguras do que escadas e mais rápidas do que aguardar elevadores. Cada sala ou laboratório dos prédios acadêmicos será considerado como um ponto de interesse, a partir disso uma estrutura de grafo será criada e os pontos de interesse que estiverem no mesmo corredor serão conectados. Entradas de rampas de acesso e interconexões entre os acadêmicos 1 e 2 serão considerados pontos de interesse também. Podemos dividir essa estrutura de grafo em 4 estruturas menores: térreo do acadêmico 1, primeiro andar do acadêmico 1, térreo do acadêmico 2, primeiro andar do acadêmico 2. Essas 4 estruturas serão conectadas pelos pontos de interesse de rampas de acesso e pontos de interesse de saída e entrada dos prédios acadêmicos 1 e 2. As transições de rampas de acesso podem ser percebidas através de sensores de altitude e as transições entre os acadêmicos serão percebidas através de *Wi-Fi Fingerprints* distintos, já que existem pontos de acesso diferentes entre um prédio acadêmico e outro.

A figura 26 mostra a estrutura de grafos criada para representar a ligações entre as salas e laboratórios do andar térreo do prédio acadêmico 1. É importante notar que cada uma dessas ligações entre os nós (salas e laboratórios) possui informações referentes à ligação, como por exemplo o custo de realizar essa travessia ou direção (norte, sul, leste oeste) entre os nós; esse tipo de informação é importante pois colabora para a elaboração de um caminho mais otimizado de travessia do grafo e de informações para auxiliar o usuário a realizar as travessias entre as salas.

Figura 26 – Conexões das salas



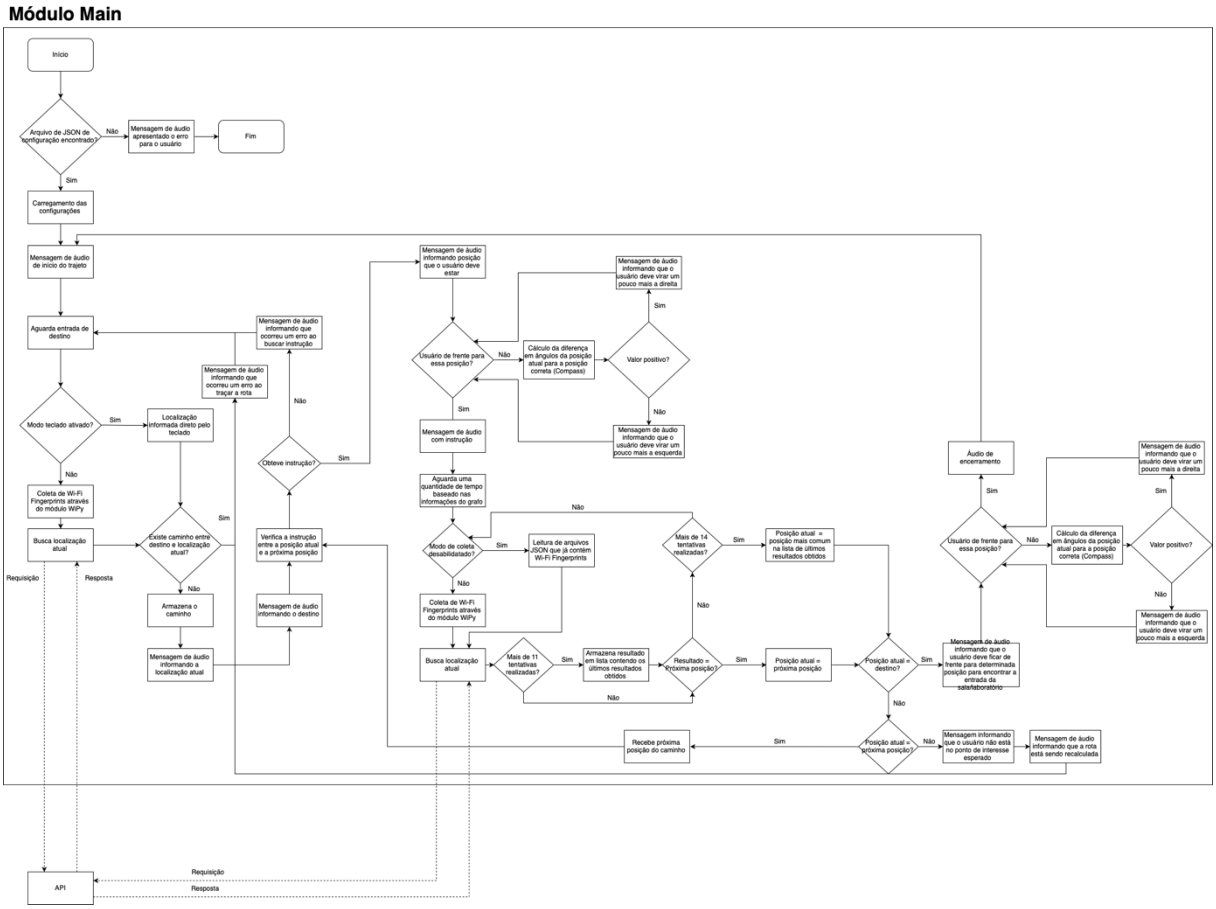
Fonte: Elaborado pelo autor

O módulo é capaz de retornar uma lista contendo os códigos das salas pelas quais o usuário deve passar para chegar ao destino, o módulo também disponibiliza uma função que retorna o tipo de ligação entre um nó e outro para que ele saiba qual direção seguir a partir da sala onde ele está.

4.3.9 Módulo *Main*

A figura 27 mostra o diagrama de execução do módulo *Main*, esse módulo é responsável por realizar todo o fluxo de trajeto e integrar as funcionalidades dos módulos descritos anteriormente para que o dispositivo seja capaz de guiar o usuário pelos prédios acadêmicos 1 e 2.

Figura 27 – Execução do módulo *Main*



Fonte: Elaborado pelo autor

A figura 28 mostra um exemplo do arquivo de configuração JSON que o módulo principal espera, as chaves desse documento JSON contém os valores que serão analisados pelo módulo *Main*, o principal objetivo do arquivo de configuração é habilitar opções para realizar testes nos módulos sem a necessidade de alterar o código.

Figura 28 – Arquivo *JSON*

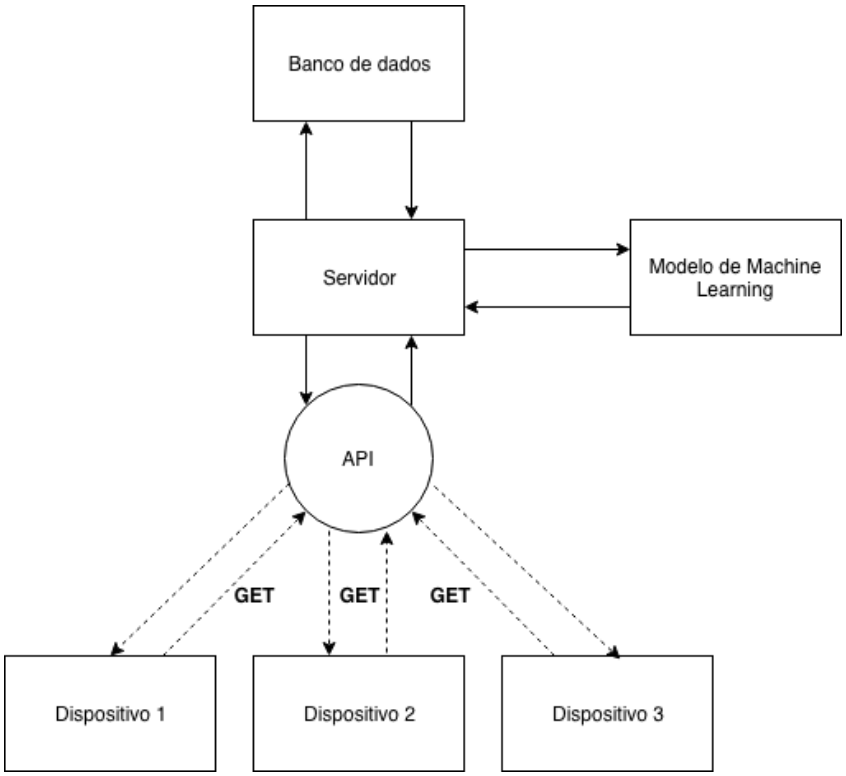
```
{
  "debug": true,
  "disableGetFingerprints": true,
  "keyboard": true,
  "disableIntialSearch": true,
  "forbidenMacAddresses": [
    "B8:27:EB:A6:7B:C5",
    "B8:27:EB:24:1B:E1"
  ],
  "compass_rose": {
    "N": {
      "min": 0,
      "max": 89.9
    },
    "E": {
      "min": 90,
      "max": 179.9
    },
    "S": {
      "min": 180,
      "max": 269.9
    },
    "W": {
      "min": 270,
      "max": 359.9
    }
  }
}
```

Fonte: Elaborado pelo autor

4.4 Servidor

Como definido anteriormente o servidor concentrará todas as informações necessárias e a versão mais atualizada dos *Wi-Fi Fingerprints*. Pelo fato de o servidor não estar presente localmente ele deverá ser acessado na internet pelos dispositivos através de uma *API WEB (Application Programming Interface)*.

Figura 29 – Processo de consulta no servidor



Fonte: Elaborado pelo autor

4.4.1 *API*

Uma *API* é uma interface de programação de aplicação, basicamente ela permite que sistemas diferentes se comuniquem entre si através de requisições e respostas como mostrado na figura 29. Nesse projeto a *API REST (Representional State Transfer ou Transferência de Estado Representacional)* é o meio de comunicação entre os dispositivos e o servidor, o servidor possui uma *API* que fica escutando requisições *HTTP* dos dispositivos, os dispositivos por sua vez realizam apenas requisições *POST*. A API é acessada automaticamente pelos dispositivos periodicamente, para checar a posição atual do usuário, os dispositivos anexam um *Array* JSON no corpo da requisição e nos parâmetros da URL enviam o ponto de interesse esperado para aqueles *Wi-Fi Fingerprints*. Caso o resultado da predição do modelo de *Machine Learning* seja igual ao passado pelo dispositivo, a API pode adicionar esses novos *Wi-Fi Fingerprints* ao seu banco de dados como forma de se manter atualizado.

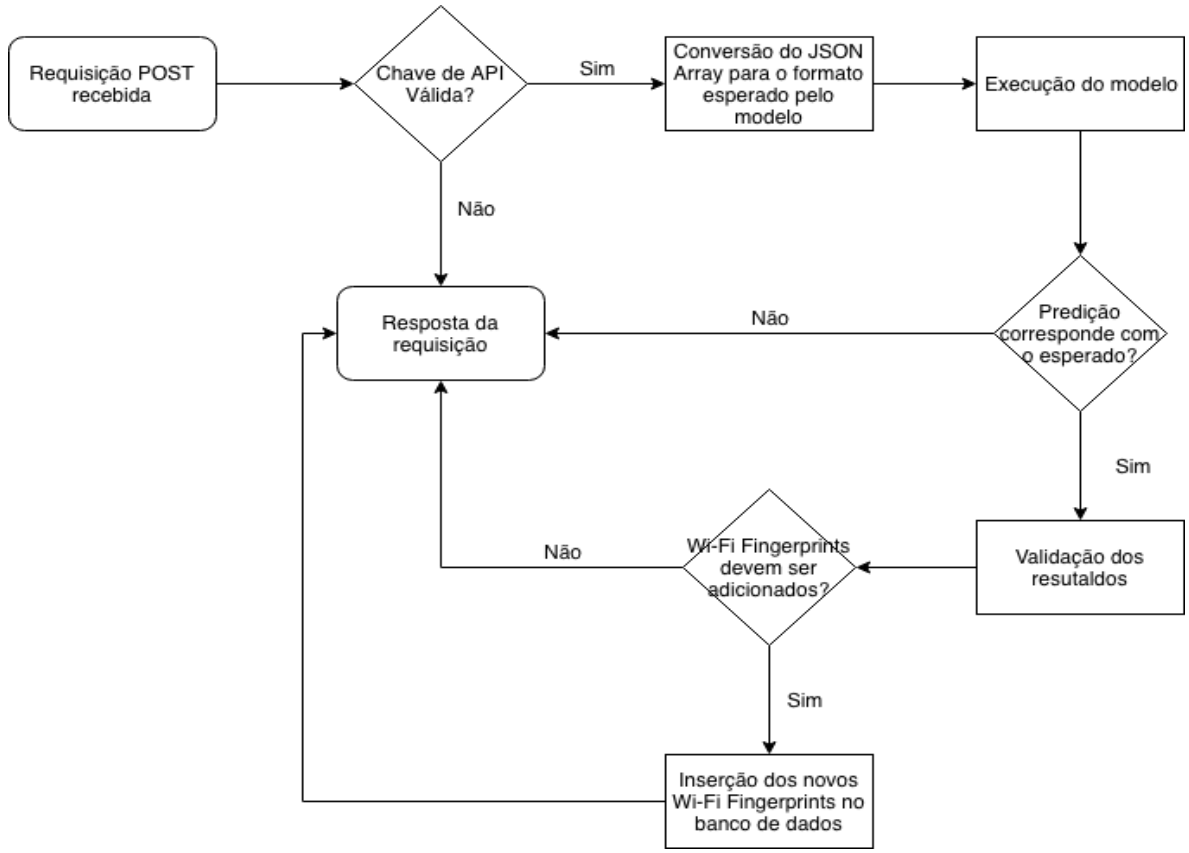
4.4.2 Métodos da API

A API disponibilizada para a comunicação com o servidor possui métodos de requisição responsáveis por indicar como serão acessados os recursos.

4.4.2.1 Método POST

O método POST disponibilizado na API é o caminho existente para que um dispositivo seja capaz de encontrar sua localização atual através da coleta de dos *Wi-Fi Fingerprints*. Como mostrado na figura 30 a API primeiramente checa se a chave de API é válida, caso a chave de API corresponda com o esperado os dados presentes no corpo da requisição são convertidos para o formato esperado pelo modelo para que possam ser submetidos. Na requisição também está presente um parâmetro no qual o dispositivo envia o ponto de interesse esperado para aquele conjunto de *Wi-Fi Fingerprints*, se a predição resultante do modelo for equivalente ao parâmetro da requisição a API pode adicionar esses novos *Wi-Fi Fingerprints* no banco de dados, para manter o mesmo atualizado.

Figura 30 – Processo do método *POST* da *API*



Fonte: Elaborado pelo autor

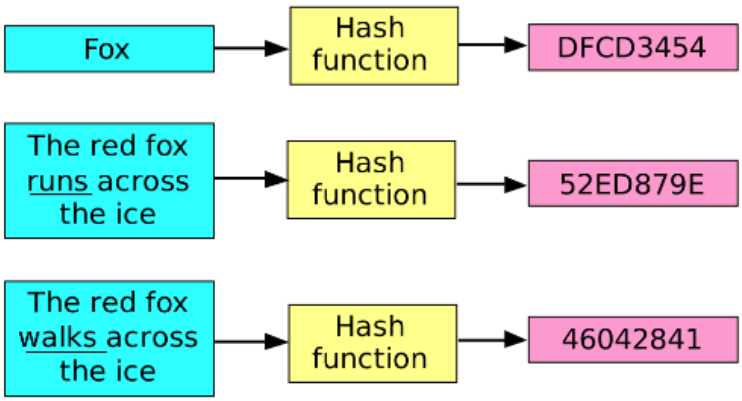
4.4.3 Segurança

Para garantir que os dados no banco de dados do servidor não sejam alterados por terceiros indevidamente algumas garantias foram tomadas.

4.4.3.1 Chave de API

Uma chave de *API* deve ser enviada junto a todas as requisições para que seja validada antes do servidor executar qualquer tarefa. Essa chave de *API* consiste de uma cadeia de caracteres sobre a qual será aplicado um conjunto de funções *hash* criptográficas SHA-2 (*Secure Hash Algorithm*). Serão aplicadas operações matemáticas sobre a chave de *API* que resultarão em um *hash*, caso esse *hash* seja idêntico ao presente no banco de dados a requisição é validada. As funções *hash* são importantes, pois são consideradas praticamente impossíveis de reverter, ou seja, a partir do *hash* gerar novamente o valor original. Outro fator que contribui para a segurança é o de que dois valores de entrada diferentes não podem gerar o mesmo *hash* como resultado. A figura 31 é um exemplo de funções de *hash* criptográficas sendo aplicadas a valores de entradas diferentes e resultando em *hashs* distintos.

Figura 31 – Exemplo de chaves *Hash* criptografadas



Fonte: LANDMAN; ROSS; WILLIAMS (2018)

4.4.3.1 HTTPS

HTTPS ou Hyper Text Transfer Protocol é uma implementação do protocolo HTTP sobre uma camada adicional chamada SSL (Secure Socket Layer ou Protocolo de camadas de socket seguras). Basicamente o SSL estabelece um link entre o servidor e os clientes para comunicação criptografada. Quando um cliente acessa um servidor que possui HTTPS, o servidor e o cliente estabelecem uma conexão através de um processo denominado *SSL Handshake*. Durante o *SSL Handshake*, o cliente requisita que o servidor se identifique e o servidor responde enviando um certificado SSL que contém uma chave pública embutida no mesmo. O cliente por sua vez pode checar se o certificado é assinado por um certificado confiável, caso o cliente confie no certificado o mesmo cria uma chave de sessão simétrica e criptografa a mesma com a chave pública enviada pelo servidor. Apenas a chave privada do servidor é capaz de decifrar o que foi criptografado pela chave pública do mesmo, ao decifrar a chave de sessão enviada pelo cliente, o servidor responde confirmando que recebeu a chave de sessão e a partir de então toda a informação trocada está criptografada por essa chave de sessão.

4.5 Banco de dados

Um banco de dados orientado a documentos será utilizado para armazenar os dados dos *Wi-Fi Fingerprints*. Diferentemente dos bancos de dados tradicionais que armazenam dados em linhas de tabelas e relacionam às mesmas, o banco de dados orientado a documentos armazena dados como uma série de objetos compostos por chaves e valores que podem ser separados em coleções diferentes. Uma característica importante é a de que não existe um esquema pré-definido para os documentos das coleções. Não estar preso a um esquema é bastante vantajoso nessa situação pois o conjunto de roteadores presentes nos prédios acadêmicos pode mudar a qualquer momento com adição ou remoção de aparelhos, se um banco relacional fosse utilizado o processo de adicionar ou remover pontos de interesse do banco de dados seria trabalhoso. Ao utilizarmos um banco orientado a documentos, podemos tratar o *Wi-Fi Fingerprint* como uma lista de objetos onde cada objeto corresponde a um ponto de acesso e seu valor RSSI. Caso um novo ponto de acesso seja instalado em algum lugar dos prédios acadêmicos do Senac o mesmo será adicionado automaticamente pelo banco de dados como qualquer outro ponto de

interesse sem a necessidade de intervenção humana. Na eventual substituição de um ponto de interesse basta desconsiderar esse item todas as vezes que ele aparecer dentro da lista de um *Wi-Fi Fingerprint*.

A figura 32 é um exemplo dos documentos que compõe a coleção “Pontos de interesse”. Nessa coleção serão armazenados um documento para cada sala ou laboratório dos prédios acadêmicos 1 e 2. Como essa é uma estrutura de chave e valor, as chaves serão idênticas para todos os documentos, porém os valores das mesmas serão modificados de acordo com a sala ou laboratório. Todo documento armazena a ordem na qual os sinais devem ser organizados para que os algoritmos de comparação possam ser executados, os sinais são identificados pelo *Mac Address*. *Mac Address* ou endereço *MAC* (*Media Access Control*) é um endereço único associado ao ponto de acesso, ou seja, todos os pontos de acesso dentro dos prédios acadêmicos 1 e 2 possuem um endereço *MAC* diferente. Apenas pelos endereços *MAC* que compõem o *Wi-Fi Fingerprint* atual é possível descobrir em qual acadêmico o dispositivo se encontra já que cada prédio contém seu próprio conjunto de pontos de acesso distintos.

Figura 32 – Exemplo de documento contido no banco de dados


```
{
  "_id": "1",
  "room": "A119",
  "floor": 0,
  "fingerprints": [
    [
      {
        "macAddress": "24:79:2A:3D:1B:18",
        "RSSI": -59
      },
      {
        "macAddress": "24:79:2A:FD:1B:18",
        "RSSI": -59
      },
      {
        "macAddress": "24:79:2A:BD:1E:88",
        "RSSI": -55
      },
      {
        "macAddress": "24:79:2A:FD:1E:88",
        "RSSI": -56
      },
      {
        "macAddress": "24:79:2A:BD:1B:18",
        "RSSI": -59
      }
    ],
    [
      {
        "macAddress": "24:79:2A:3D:1B:18",
        "RSSI": -61
      },
      {
        "macAddress": "24:79:2A:FD:1B:18",
        "RSSI": -60
      },
      {
        "macAddress": "24:79:2A:FD:1E:88",
        "RSSI": -60
      }
    ]
  ],
  "fingerprints_order": [
    "24:79:2A:3D:1B:18",
    "24:79:2A:FD:1B:18",
    "24:79:2A:BD:1E:88",
    "24:79:2A:FD:1E:88",
    "24:79:2A:FD:1E:88",
    "24:79:2A:BD:1B:18"
  ],
  "observations": "Sala A119 do prédio Acadêmico 1"
}
```


Fonte: Elaborado pelo autor

4.6 Site Survey

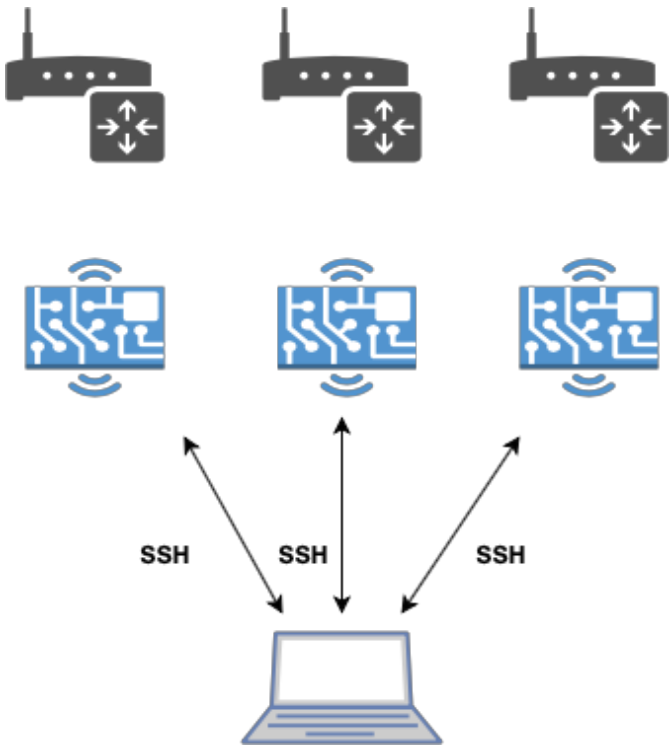
Uma das bases para o sucesso de um projeto envolvendo *Machine Learning* são os dados. No caso de um projeto que utiliza *Wi-Fi Fingerprints*, o *site-survey* para aquisição dos *Wi-Fi Fingerprints* é parte essencial do projeto.

4.6.1 Conexão com os dispositivos

Uma das bases para o sucesso de um projeto envolvendo *Machine Learning* são os dados. No caso de um projeto que utiliza *Wi-Fi Fingerprints*, o *site-survey* para aquisição dos *Wi-Fi Fingerprints* é parte essencial do projeto. Para realizar a coleta desses dados foram utilizados 3 dispositivos conectados a baterias portáteis contendo uma ferramenta CLI (*Command Line Interface* ou Interface de linha de comando). Os dispositivos utilizados eram as placas escolhidas para a construção do

hardware, já que a antena Wi-Fi é um fator de determinante como visto anteriormente. Cada placa continha um o sistema operacional Raspbian Lite (sem interface gráfica) instalado. O Raspbian é um sistema operacional livre baseado na distribuição Debian do Linux para a arquitetura ARM (arquitetura dos processadores presentes nas placas escolhidas para o Hardware). Em cada um dos dispositivos foram habilitadas as sessões SSH e cada dispositivo foi configurado como um ponto de acesso único: dispositivo1, dispositivo2 e dispositivo3. Utilizando essa configuração era possível utilizar um computador ou mesmo um smartphone para nos conectarmos ao ponto de acesso da placa de coleta e então realizarmos uma conexão SSH pelo terminal. O SSH (*Secure Shell*) consiste basicamente de um protocolo de rede criptográfico para operação de serviços. Ao utilizarmos o SSH abrimos uma sessão criptografada via terminal entre o cliente e o servidor como mostrado na figura 33, através do terminal somos capazes de utilizar todos os recursos do servidor.

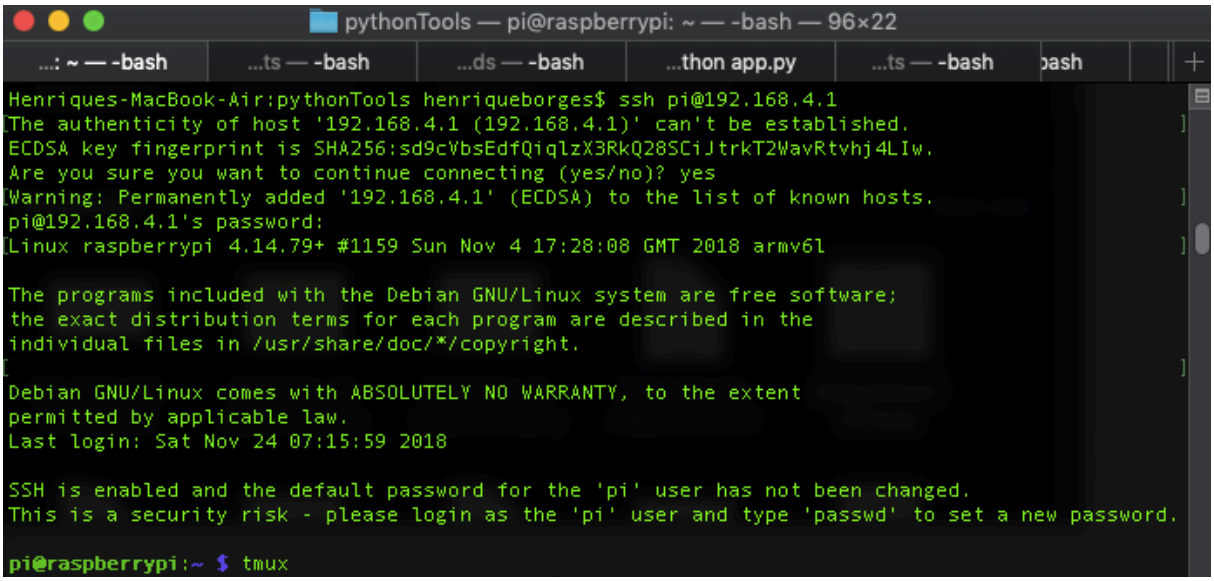
Figura 33 – Exemplo da conexão *SSH* realizada entre os dispositivos (servidores) e um computador(cliente)



Fonte: Elaborado pelo autor

No caso do *site-survey* realizado todos os dispositivos eram servidores e os computadores utilizados para realizar a conexão SSH eram os clientes. A figura 34 mostra um *print* de uma janela do terminal que iniciou uma sessão com um dos dispositivos utilizados para a realização do *site-survey*. Após iniciar a conexão com o dispositivo bastava iniciar a ferramenta

Figura 34 – Exemplo de conexão *SSH* via terminal



Fonte: Elaborado pelo autor

4.6.2 siteSurvey-CLI – Ferrramenta de linha de comando do *site-survey*

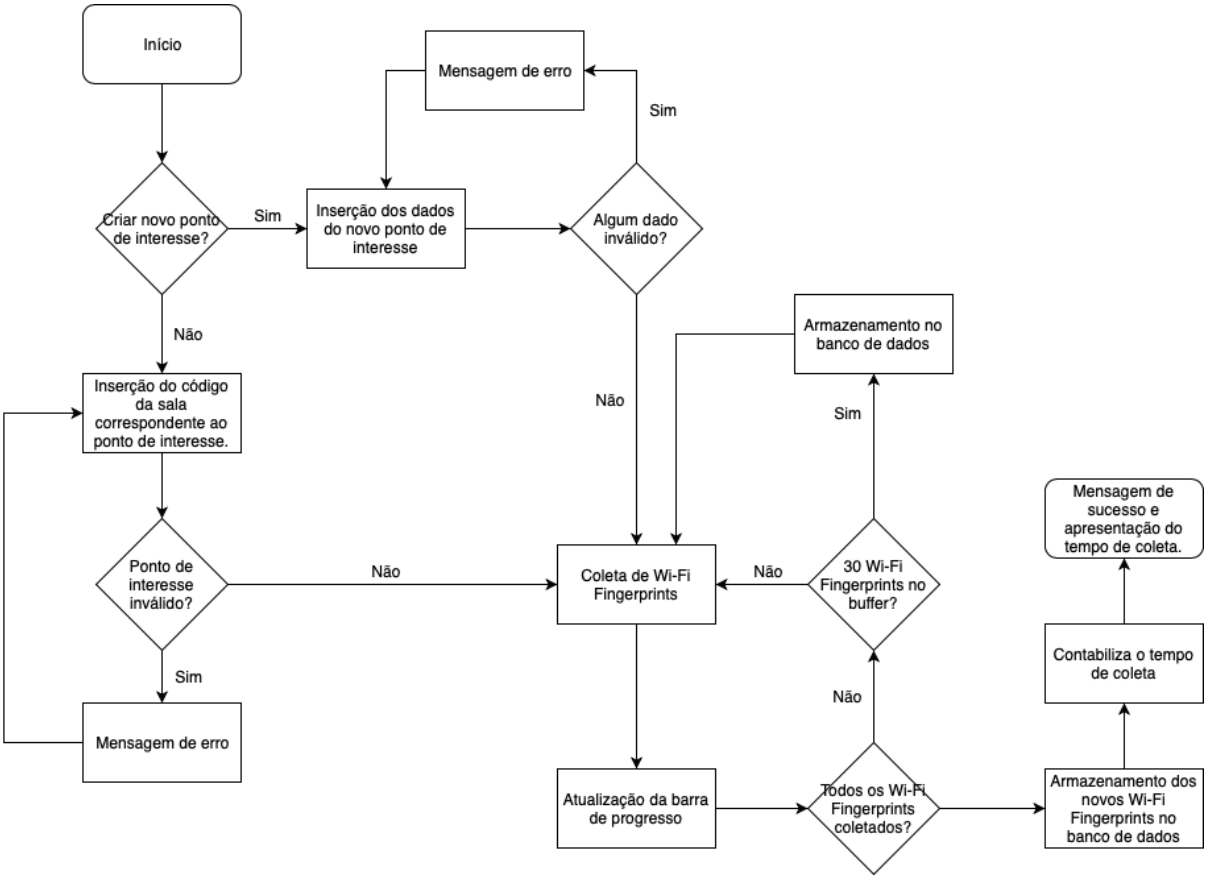
A siteSurvey-CLI foi a ferramenta de linha de comando desenvolvida em *Python* para a automatização do processo de coleta de *Wi-Fi Fingerprints*. Um dos objetivos da ferramenta é possibilitar que um usuário que não possui conhecimento em programação possa realizar a tarefa de *site-survey* de forma menos trabalhosa. A ferramenta aceita 5 comandos:

- addf – Adicionar Fingerprint (A opção “-s” seguido de um número pode ser utilizada para adicionar uma série Fingerprints para a mesma localização do tamanho equivalente ao inserido na opção “-s”).
- rmf – Remover um determinado Fingerprint.
- help – Obter ajuda.
- Exit – Para encerrar a aplicação.

A figura 35 demonstra o fluxograma do comando mais importante da ferramenta de captura o “addf”. Através do uso dessa ferramenta de captura foi possível automatizar o processo de coleta de *Wi-Fi Fingerprints* para um ponto de interesse, todas os dispositivos eram colocados em frente ao ponto de interesse e utilizando a ferramenta de linha de comando podíamos ordenar a coleta para um novo ponto de interesse ou simplesmente adicionar mais *Wi-Fi Fingerprints* para o mesmo. Também era possível verificar o status da tarefa de coleta através de uma barra de pro-

gresso e analisar o tempo total de coleta para uma determinada quantidade de *Wi-Fi Fingerprints* ao final do processo. Cada *Wi-Fi Fingerprint* leva em média 1 segundo para ser coletado, ao ser coletado o *Wi-Fi Fingerprint* era armazenado em um buffer, após 30 inserções no buffer os *Wi-Fi Fingerprints* contidos nele eram armazenados no banco e o buffer era esvaziado novamente.

Figura 35 – Fluxo do comando addf



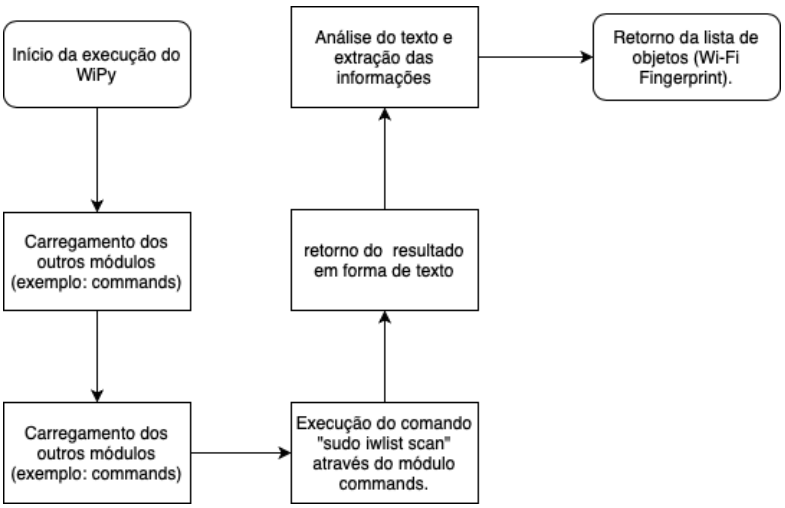
Fonte: Elaborado pelo autor

4.6.3 WiPy – Módulo de *Wi-Fi Fingerprints*

O módulo WiPy (sigla para *Wi-Fi Fingerprints Python*) é um programa cujo objetivo é empacotar todas as tarefas relacionadas a *Wi-Fi Fingerprints*. Esse módulo é responsável por coletar os dados da placa de rede dos dispositivos e transformá-los para o formato esperado pelos outros; uma lista de objetos contento pontos de interesse e seus respectivo valores de RSSI. Para coletar os *Wi-Fi Fingerprints* é necessário acessar a placa de rede do dispositivo, no ambiente do sistema operacional

Raspbian o comando “sudo iwlist scan” pode ser utilizado para obtermos informações da interface sobre as redes Wi-Fi. O “scan” é um parâmetro que necessita de permissões de usuário root (Administrador do sistema) por isso a palavra “sudo” é utilizada antes do comando. A opção “scan” é a responsável por retornar uma lista de pontos de acesso e uma série de informações a respeito desses pontos como: qualidade, frequência, canal.

Figura 36 – Fluxo da execução do módulo WiPy



Fonte: Elaborado pelo autor

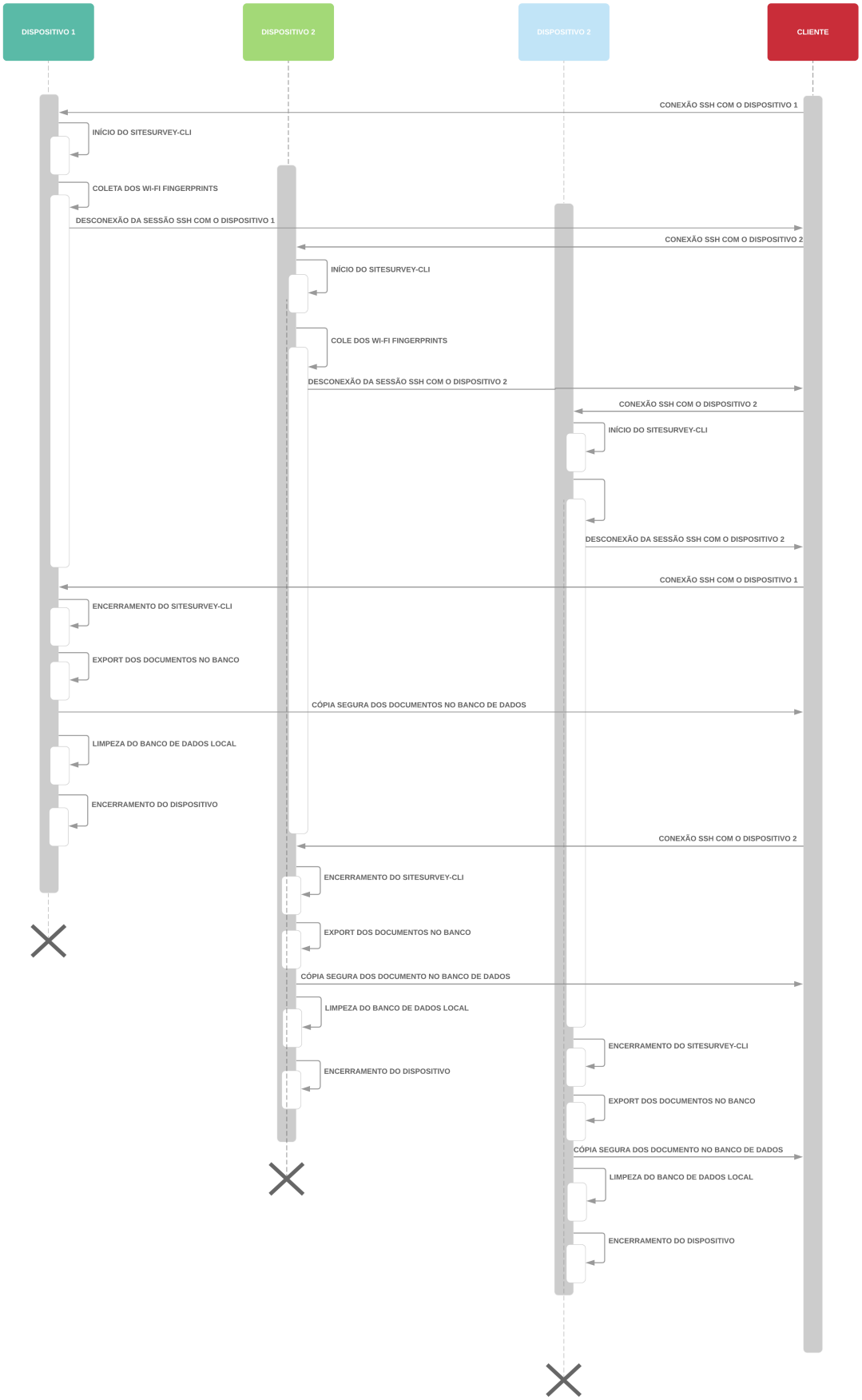
O módulo WiPy utiliza o pacote “commands” do *Python* para executar esse comando e capturar o seu retorno. O retorno da execução do comando é um texto contendo todas as informações que a placa de rede tem capacidade de fornecer, utilizando as ferramentas disponíveis no *Python* esse texto é analisado e todos os *Mac Address* e seus respectivos RSSI são extraídos do texto e adicionados a uma lista de objetos onde cada objeto possui duas chaves: MacAddress e RSSI. A lista de objetos retornada é o *Wi-Fi Fingerprint* para aquela posição onde o dispositivo se encontra.

4.6.4 Envio dos dados coletados no *site-survey*

Cada dispositivo utilizado no *site-survey* utilizava a ferramenta de linha de comando desenvolvida para o projeto siteSurvey-CLI, essa ferramenta por sua vez fazia uso do módulo WiPy para coletar os *Wi-Fi Fingerprints* de um ponto de interesse e armazena-los em uma instância do banco de dados local do dispositivo. Ao fim

das coletas realizadas nos dispositivos os documentos eram exportados como arquivos JSON que por sua vez eram enviados do dispositivo para o computador cliente através do comando “scp”. O comando “scp” é uma sigla para segure copy ou cópia segura, já que a transmissão do arquivo é realizada através da conexão SSH. Todo o processo de *site-survey* nos dispositivos e envio dos *Wi-Fi Fingerprints* para o cliente via SSH é mostrado na figura 37.

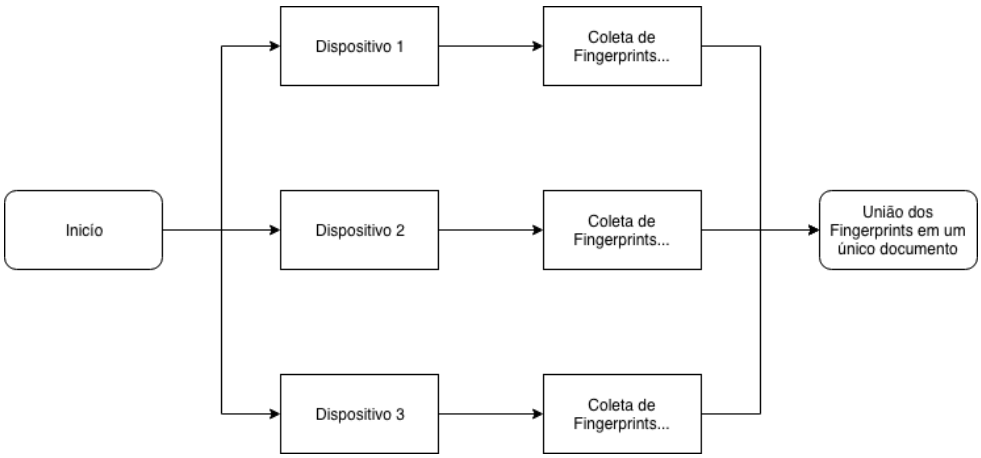
Figura 37 – Diagrama de sequência da coleta e envio dos *Wi-Fi Fingerprints*



Fonte: Elaborado pelo autor

Muitas vezes os *Wi-Fi Fingerprint*s de um mesmo ponto de interesse foram coletados em dispositivos diferentes e era necessário agrupá-los em um único documento, a figura 38 mostra esse processo de agrupamento. O banco de dados MongoDB possui um método de exportação dos documentos de uma coleção no formato JSON. Todos os documentos referentes a uma mesma posição eram agrupados em um único documento contendo todos os *Wi-Fi Fingerprints*, um script foi criado para automatizar a tarefa.

Figura 38 – Agrupamento de coletas diferentes para um mesmo ponto de interesse



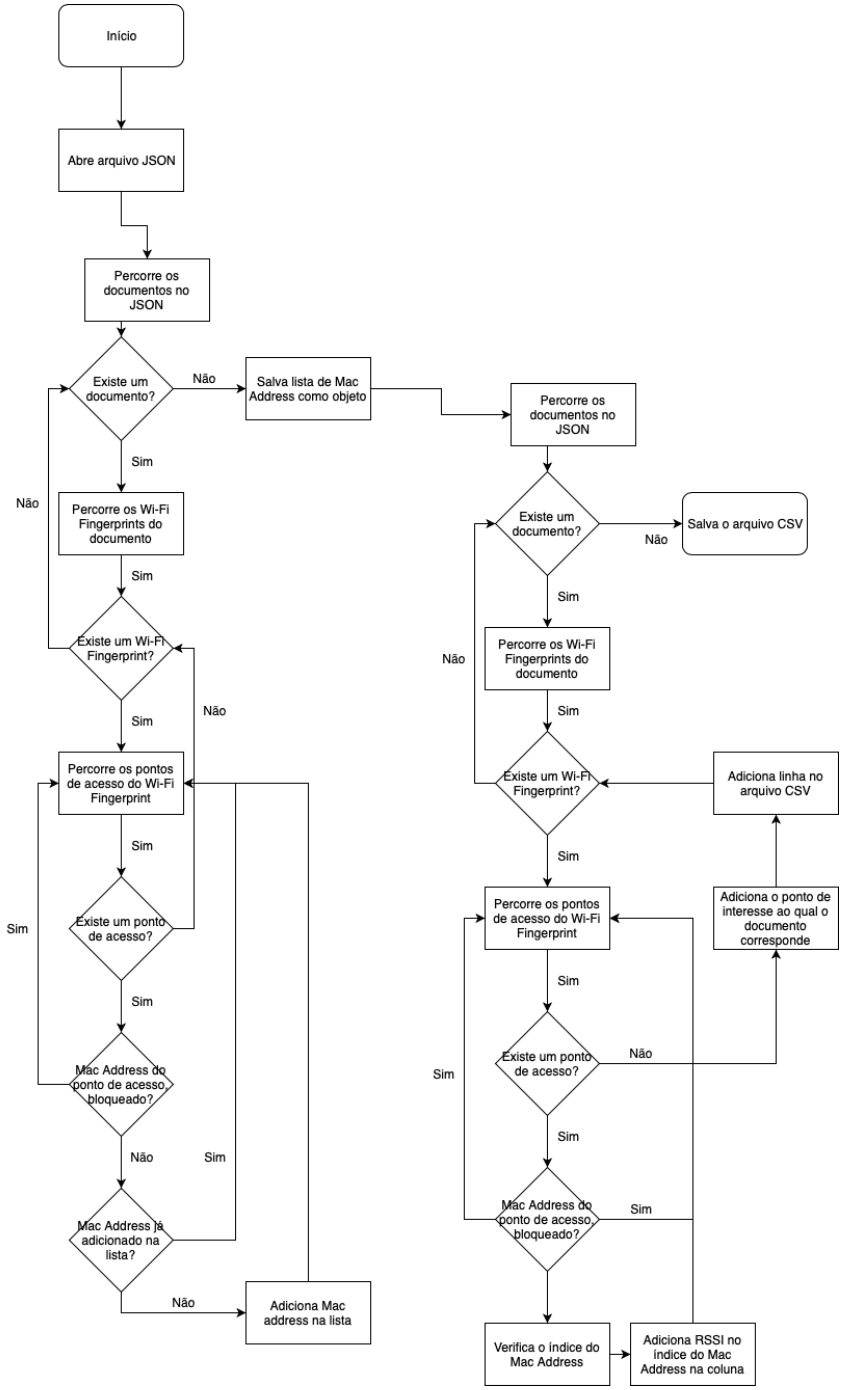
Fonte: Elaborado pelo autor

4.6.5 Transformação dos dados coletados

Como citado anteriormente os modelos de *Machine Learning* são treinados a partir de *Datasets*, os algoritmos disponíveis no *Scikit-Learn* esperam dados organizados por linhas e colunas e não organizados em documentos contendo chaves e valores como é o caso dos dados armazenados no MongoDB. Um script de conversão de dados foi escrito para transformar os dados coletados de JSON para CSV, o CSV sigla para *Comma Separeted Values* ou Valores Separados por vírgula ordena os todos os valores, separando-os pela virgula, conforme sugerido pelo nome do formato em uma ordem pré-determinada. A ordem dos valores do CSV pode ser imaginada como uma série de colunas, essas colunas por sua vez são características (*features*) que o modelo de *Machine Learning* utilizará para distinguir os *Wi-Fi Fingerprints*. A primeira parte do processo consiste em varrer todos os documentos coletando todos os *Wi-Fi Fingerprints* encontrados no processo de *Site-Survey*, nesse ponto é possível implementar uma lista de *Mac Address* bloqueados ou uma lista de *Mac Address* aceitos para determinar quais pontos de acesso serão incluídos no

Dataset. Após reunir uma lista de todos os *Mac Address* que serão incluídos na coleção o mesmo cria um objeto em *Python* (dicionário de dados) onde cada *Mac Address* da lista corresponde a uma chave e cada chave possui um valor que vai de 0 ao tamanho da lista menos um, o objeto é salvo e pode ser carregado na memória posteriormente, o fluxograma do script de conversão dos documentos para CSV pode ser visto na imagem 39. O objeto construído é importante pois ele corresponde à ordem em que as colunas de *Mac Address* serão dispostas no arquivo CSV, ele está intimamente ligado a operação de conversão dos dados e acompanhará os mesmos como forma de orientação nas etapas de treinamento e execução do modelo.

Figura 39 – Fluxograma de conversão de documentos *JSON* para arquivo *CSV*



Fonte: Elaborado pelo autor

Um exemplo da transformação dos dados é a figura 40 e a tabela 2. As duas armazenam exatamente a mesma informação, porém em cada uma delas a informação está estruturada de formas diferentes. Um ponto importante é o fato de que onde não existir valor RSSI lido o mesmo deve ser igual a zero para que o modelo possa computar o *Dataset* nas fases de treinamento, teste e execução.

Figura 40 – Exemplo de documento *JSON*


```
{
  "_id": "1",
  "room": "A121",
  "floor": 0,
  "fingerprints": [
    [
      {
        "macAddress": "24:79:2A:BD:2F:D8",
        "RSSI": -80
      },
      {
        "macAddress": "24:79:2A:BD:B0:78",
        "RSSI": -84
      },
      {
        "macAddress": "24:79:2A:3D:21:08",
        "RSSI": -85
      }
    ],
    [
      {
        "macAddress": "24:79:2A:BC:94:68",
        "RSSI": -77
      },
      {
        "macAddress": "74:3E:2B:78:2A:A8",
        "RSSI": -86
      },
      {
        "macAddress": "74:3E:2B:B8:2A:A8",
        "RSSI": -88
      }
    ],
    [
      {
        "macAddress": "24:79:2A:BD:AB:58",
        "RSSI": -73
      },
      {
        "macAddress": "24:79:2A:3D:C5:98",
        "RSSI": -44
      },
      {
        "macAddress": "24:79:2A:BD:C5:98",
        "RSSI": -44
      }
    ]
  ],
  "fingerprints_order": [
    "24:79:2A:BD:2F:D8",
    "24:79:2A:BD:B0:78",
    "24:79:2A:3D:21:08",
    "24:79:2A:BC:94:68",
    "74:3E:2B:78:2A:A8",
    "74:3E:2B:B8:2A:A8",
    "24:79:2A:BD:AB:58",
    "24:79:2A:3D:C5:98",
    "24:79:2A:BD:C5:98"
  ],
  "observations": "Sala A121 do prédio Acadêmico 1"
}
```


Fonte: Elaborado pelo autor

Tabela 2 – Agrupamento de coletas

room	24:79:2A:BD:2F:D8	24:79:2A:BD:B0:78	24:79:2A:3D:21:08	24:79:2A:BC:94:68	74:3E:2B:78:2A:A8	74:3E:2B:B8:2A:A8	24:79:2A:BD:AB:58	24:79:2A:3D:C5:98	24:79:2A:BD:C5:98
A121	-80	-84	-85	0	0	0	0	0	0
A121	0	0	0	-77	-86	-88	0	0	0
A121	0	0	0	0	0	0	-77	-44	-44

Fonte: Elaborado pelo autor

4.6.6 Treinamento do modelo de *Machine Learning*

A escolha de um algoritmo de *Machine Learning* depende basicamente do seu *Dataset*, seja no tipo dos dados ou no volume do mesmo. No caso dos *Wi-Fi Fingerprints* coletados os dados estavam “etiquetados”, também chamados de “*labeled data*”, o *Dataset* CSV pode ser modelado em um *Array* de características X que se relacionavam a uma classe Y. As classes são formadas pelos pontos de interesse onde foram realizadas as coletas de dados, já os *Wi-Fi Fingerprints* correspondem as características desses pontos de interesse. No *Dataset* coletado cada ponto de interesse contém inúmeras leituras (linhas no arquivo). O fato de o *Dataset* estar organizado da maneira descrita anteriormente caracterizou a tarefa de reconhecimento do ponto de interesse como um problema de classificação em *Machine Learning*, onde esperamos encontrar a classe (ponto de interesse) correspondente a partir de um conjunto de *Mac Address* e seus respectivos valores RSSI.

A partir dessa premissa alguns algoritmos foram testados, levando em consideração: tempo de treinamento, tempo para execução, acurácia da predição e escalabilidade. Entre os algoritmos testados estavam: *Nearest Neighbours*, *Linear SVC*, *Decision Tree*, *Random Forest* e *SGD Classifier*. Os treinamentos iniciais foram realizados nos próprios dispositivos, porém já era antecipado o fato de que conforme o volume de dados aumentasse o tempo e consumo de recursos para treinamento dos modelos iria crescer também. Diversas tentativas de treinar um modelo em uma máquina e exportar o mesmo para o dispositivo posteriormente foram realizadas sem sucesso, pois não foi possível criar uma máquina virtual com o sistema operacional Raspbian Lite já que o sistema utiliza uma arquitetura ARM que é diferente das arquiteturas utilizadas na esmagadora maioria dos computadores. Conforme o *Dataset* aumentou, os dispositivos de prototipagem com especificações modestas de 512MB de memória RAM e um processador de um núcleo apenas com velocidade de 1GHz não foram capazes de comportar sequer o treinamento dos modelos mais leves testados. Com isso a necessidade da construção de uma API que realizasse a intermediação entre os dispositivos portáteis e um ambiente mais robusto capaz de comportar a execução e treinamentos constantes do modelo ficou evidente.

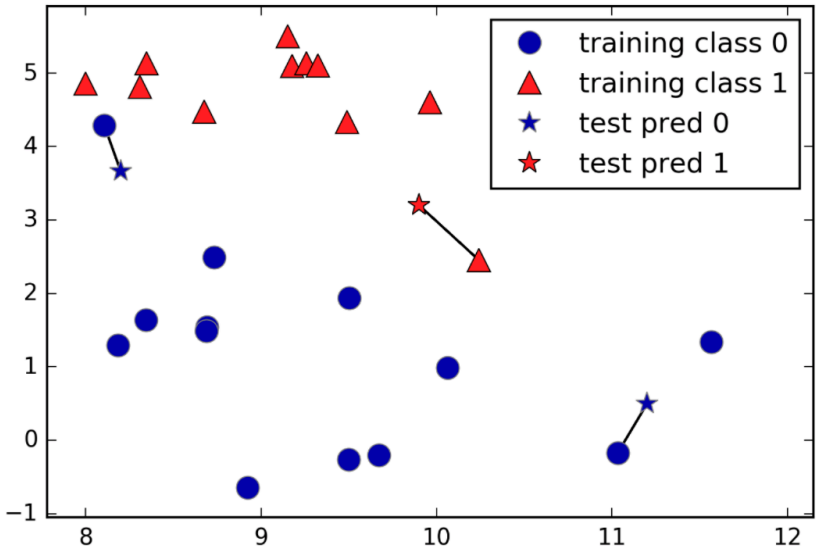
Todas as técnicas utilizadas obtiveram resultados acima de 70% de chance de assertividade da classe. Dentre os resultados, o melhor resultado foi obtido pelo

algoritmo Decision Forest, seguido pelo algoritmo Linear SVC e por fim o algoritmo *K-Nearest Neighbors*.

4.6.7 K-Nearest Neighbors

O algoritmo *K-Nearest Neighbors* (K-ésimo vizinhos mais próximos) ou k-NN é considerado um dos algoritmos mais simples de *Machine Learning* disponíveis. Um modelo treinado de k-NN consiste basicamente do próprio *Dataset*; ao realizar uma predição o modelo busca o ponto mais próximo daquele submetido para o modelo. A figura 41 mostra um exemplo de um modelo de classificação binária k-NN, que recebe como entrada de dados duas características correspondentes a números inteiros, essas duas características formam um ponto, baseado nesse ponto o algoritmo k-NN busca o ponto mais próximo no *Dataset* submetido na fase de treinamento do modelo.

Figura 41 – *K-Nearest Neighbors – K = 1*

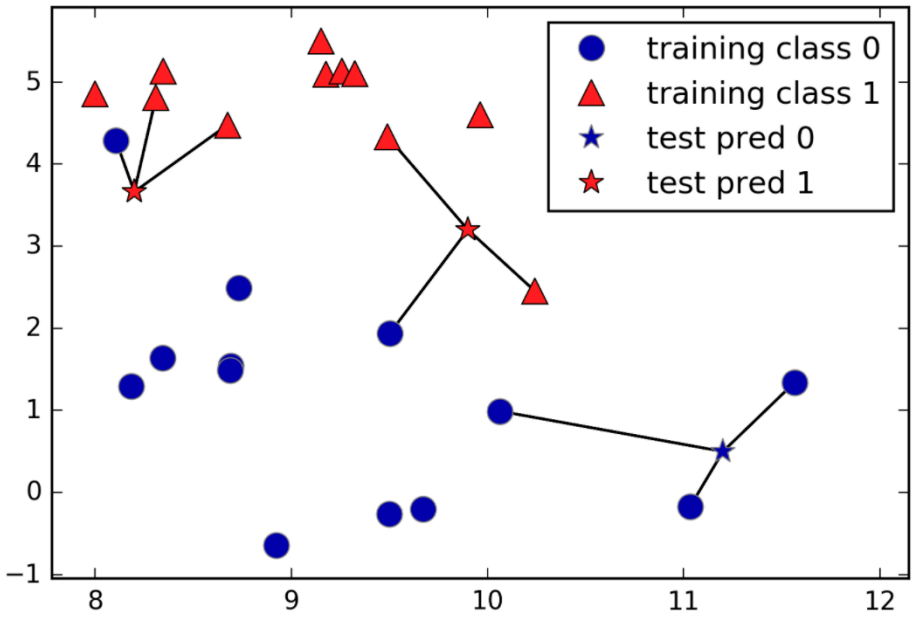


Fonte: GUIDO; MÜLLER (2017, p 112)

O algoritmo k-NN é capaz de considerar não apenas um único ponto mais próximo, a quantidade de vizinhos próximos a serem considerados é configurada pelo próprio usuário. A letra k no nome do algoritmo faz referência ao número de vizinhos a serem considerados, em casos em que k > 1 todos os vizinhos são somados e a classe que possuir mais pontos próximos é escolhida pelo modelo como resultado. Uma das formas de cálculo de distância entre os pontos no algoritmo K-NN

que pode ser utilizada é o cálculo da distância Euclidiana já citada anteriormente na equação 3.

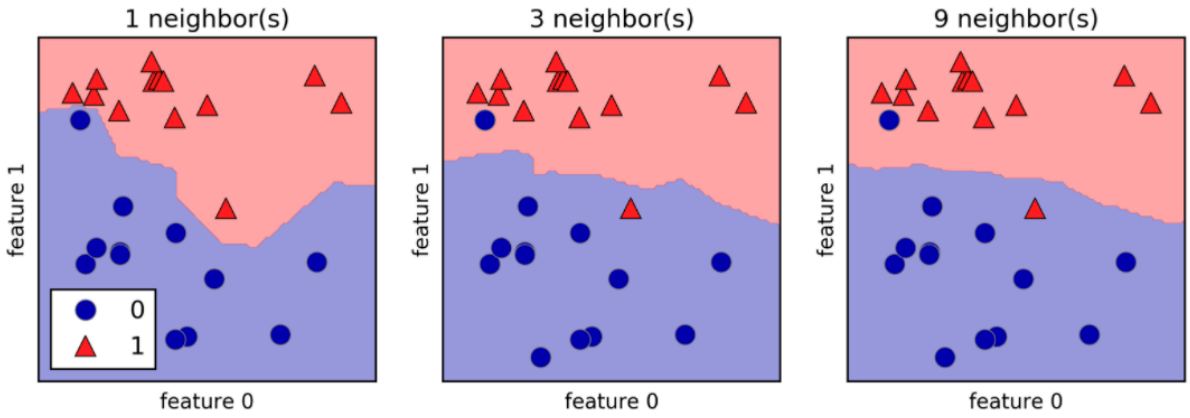
Figura 42 – *K-Nearest Neighbors - K = 3*



Fonte: GUIDO; MÜLLER (2017, p 112)

A figura 43 mostra a divisão da área do *Dataset* conforme o número de vizinhos considerados aumenta. É importante notar que o número de vizinhos possui uma relação inversamente proporcional a complexidade do modelo, um exemplo claro disso é que se considerarmos o número de vizinhos igual ao número de itens presentes no *Dataset* todas as predições realizadas no modelo teriam o mesmo resultado, o fator determinante para a classe resultante nesse caso seria a mesma possuir mais itens no *Dataset*.

Figura 43 – Divisão de *Dataset* no *K-Nearest Neighbors*



Fonte: GUIDO; MÜLLER (2017, p 117)

Embora considerado simples o k-NN é um algoritmo que apresenta bons resultados e um tempo de treinamento menor do que a média. Entretanto o processo de classificação é mais lento do que outros algoritmos e a escalabilidade do mesmo também é ruim já que é necessário armazenar o *Dataset* inteiro para a execução do modelo.

4.6.8 Linear SVC

Linear SVC ou *Support Vector Classification* é um modelo de classificação linear de *Machine Learning* bastante difundido. Esse modelo distribui os dados como pontos em um espaço de *n* dimensões onde o número de dimensões corresponde ao número de características (*features*) do *Dataset*.

Esse algoritmo utiliza uma equação construída a partir do conjunto de características (*features*) do *Dataset* para realizar as predições de classes. Na equação 5 x[0] até x[p] equivalem a todas as características (*features*) presentes no *Dataset*, ou seja, o *Array* x será utilizado para predição de uma classe y, o termo b é chamado de termo constante e o conjunto de w[0] a w[p] são os coeficientes angulares dos termos presentes em x. Caso o y resultante seja maior que zero a classe considerada será a classe positiva, do contrário a classe negativa será o resultado da predição do modelo. Muitos Classificadores lineares multiclasse na verdade utilizam a técnica de uma classe contra o resto citada no capítulo 3.7.

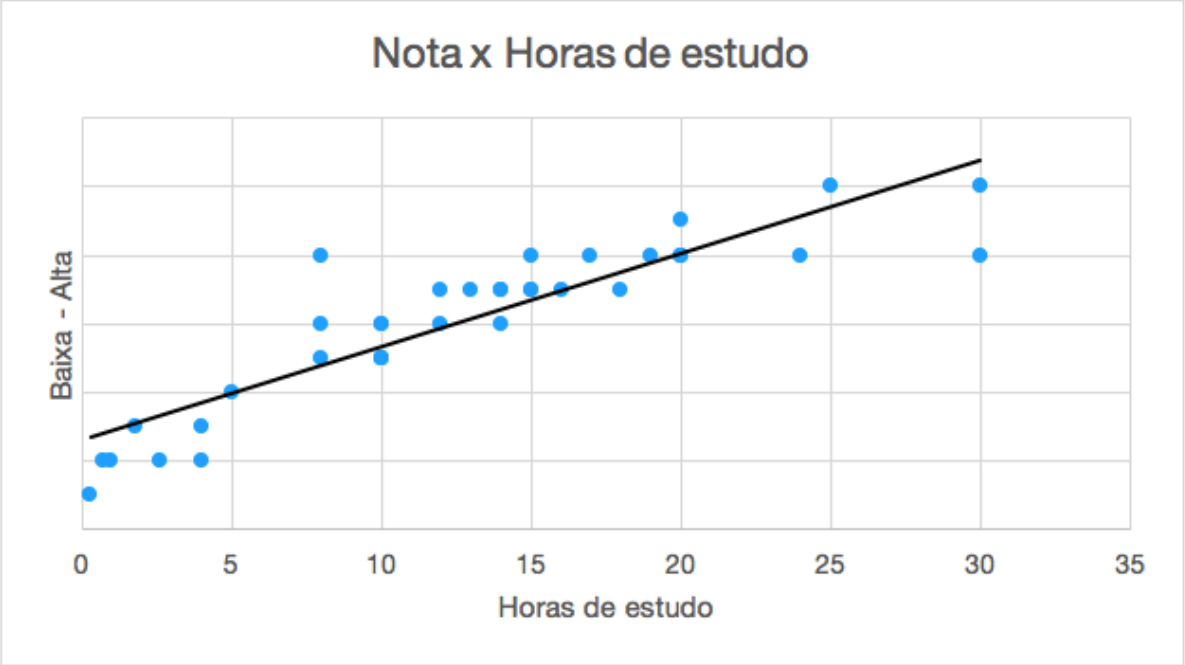
$$y = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b > 0 \qquad (5)$$

Quando o *Dataset* possui apenas uma característica (feature) ou seja o tamanho do *Array* x é igual a 1, a equação é uma reta que recebe o x como entrada e retorna um valor y como mostrado na equação 6. O exemplo da figura 44 relaciona as notas obtidas por alunos em uma prova com o tempo de estudo em horas dos mesmos, o modelo SVC neste caso buscaria montar uma equação reta que melhor relacionasse os valores de entrada x (horas de estudo) com suas respectivas classes, na tabela 2 todos os pontos com nota igual à 6 ou superior correspondem aos alunos que ficaram com notas altas, já os pontos abaixo disso correspondem aos alunos que ficaram com notas baixas. Para montar a equação de uma reta que melhor relacione as notas dos alunos com as horas de estudo, o SVC executará a fun-

ção da reta nos dados disponíveis para testes e então comparará o erro através de uma função conhecida como função de perda. Após isso o mesmo manipulará o coeficiente angular (w) e o termo constante (b) para adequar a reta, esse processo se repetirá por todo o *Dataset* disponível para treinamento. No *Dataset* dos *Wi-Fi Fingerprint*s coletados cada *Mac Address* seria equivalente a um índice do *Array* X onde os valores RSSI estariam presentes.

$$y = w[0] * x[0] + b > 0 \quad (6)$$

Figura 44 – Exemplo Linear *SVC*



Fonte: Elaborado pelo autor

O Linear SVC possui alguns parâmetros que podem ser alterados para treinar um modelo mais adequado para o *Dataset*. Alguns deles são:

Penalty: Esse parâmetro configura uma técnica de regularização dos coeficientes angulares w da equação. Regularização é um artificio usado para impedir o sobreajuste dos modelos. As duas técnicas aceitas são as técnicas denominadas L1 e L2 que correspondem a regressão Lasso e a regressão Ridge respectivamente. A técnica Lasso faz com que muitos coeficientes angulares (w) sejam exatamente 0, diminuindo dessa forma o número de características (*features*) utilizadas do modelo e tornando-o mais fácil de interpretar. A *Ridge Regression* por sua vez também força

a diminuição da magnitude dos coeficientes angulares (w) porém sem efetivamente zerá-los. Quando esse parâmetro não é explicitamente definido a regularização L2 é automaticamente escolhida.

Loss: O parâmetro Loss configura a técnica utilizada para realizar nas predições do modelo durante a fase de treinamento. As opções para esse modelo são as funções Hinge e Squared Hinge (ao quadrado). O Linear SVC divide o *Dataset* nas classes positivas (1) e negativas (-1), todos os valores computados pelo modelo na fase de treinamento estão entre esses valores e o resultado da predição é a classe mais próxima do mesmo, exemplo: se o resultado for equivalente à 0.5 então a classe positiva é retornada como resultado. A função Hinge possui uma margem em relação ao valor 0 e caso a distância do resultado (mesmo que certo) esteja dentro dessa margem um erro é computado pois o resultado está muito distante da classe certa.

C: O parâmetro C equivale a um número que define a força com que as regularizações serão aplicadas no modelo em relação ao dado. Por exemplo, um alto valor em C fará com que o modelo tente se ajustar melhor a cada dado fornecido do *Dataset* de treinamento, por outro lado um baixo valor de C forçará o mesmo a se ajudar melhor em relação a maioria. Ou seja, o parâmetro C é essencial para controlar o subajuste e o sobreajuste que podem ocorrer no modelo como descrito anteriormente.

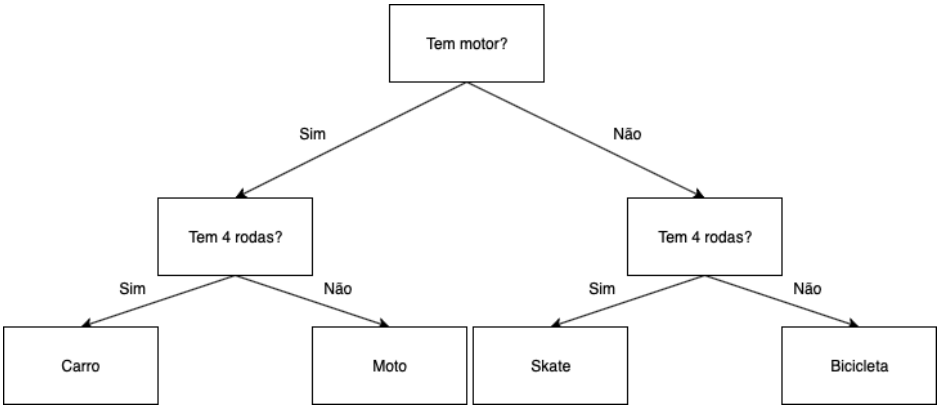
O algoritmo Linear SVC é muito poderoso e tende a se tornar mais preciso conforme a quantidade de características (*features*) aumenta além de trabalhar bem com dados esparsos. Seu tempo para treinamento é bom e o mesmo tem um tempo de execução para realizar a predição menor que o K-NN em grandes *Datasets* por exemplo. Infelizmente o Linear SVC necessita de bastante memória e não escala tão bem com grandes amostras de dados.

4.6.9 Decision Tree

Os modelos de *Machine Learning* chamados de *Decision Tree* basicamente implementam uma estrutura hierárquica de *Ifs* e *Elses* (Se e Senão). Como mostrado na figura 45 a estrutura que podemos chamar de árvore binária é montada de

cima para baixo onde cada bloco do diagrama é chamado de nó ou folha, a folha inicial é chamada de raiz, as outras folhas podem ser consideradas terminais ou podem possuir ligações com outras folhas abaixo. A partir dessa estrutura podemos realizar uma predição baseado em um conjunto de dados de entrada.

Figura 45 – Exemplo de *Decision Tree*



Fonte: Elaborado pelo autor

Geralmente os dados não podem ser agrupados através de características (*features*) binárias de verdadeiro ou falso como na figura 45. As figuras 46, 47, 48 e 49 demonstram como o algoritmo do *Decision Tree* percorre um *Dataset* onde as características são valores contínuos e procura uma forma de agrupar os dados repartindo o mesmo diversas vezes, quando uma repartição possui apenas dados de uma única classe a folha na estrutura é considera pura, cada repartição realizada aumenta o nível de profundidade (*deph*) do modelo. Nas figuras 47, 48 e 49 as profundidades são 1, 2 e 9 respectivamente.

Figura 46 Figura 47 Figura 48 Figura 49

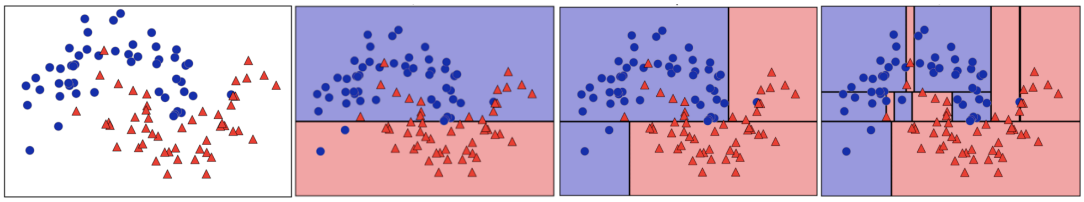


Figura 45 - Fonte: GUIDO; MÜLLER (2017, p 194).

Figuras 46, 47 e 48 - Fonte: GUIDO; MÜLLER (2017, p. 196)

Para entender o melhor processo podemos imaginar um plano cartesiano bidimensional com diversos pontos (x, y) demarcados. No caso de um classificador binário, cada um dos pontos poderia pertencer exclusivamente a classe positiva ou

negativa, o algoritmo de *Decision Tree* repartiria esse plano diversas vezes para tentar agrupar melhor os dados distribuídos nele. Ao realizar uma nova predição procuraríamos demarcar o ponto baseado nas coordenadas x e y e verificaríamos qual classe é maioria dentro daquela repartição demarcada no processo de treinamento do modelo.

Algoritmos que utilizam estruturas hierárquicas de árvore tendem ao sobreajuste criando modelos que são muito específicos. A figura 48 demonstra bem esse caso, o modelo sempre busca criar folhas puras, isso pode resultar em repartições muito específicas para um determinado ponto no plano. Existem duas técnicas para contornar o sobreajuste de um modelo, interromper a continuação da criação da árvore dada uma condição (técnica denominada *pre-pruning*), por exemplo: a profundidade da árvore alcançar um tamanho pré-determinado, número mínimo de pontos no espaço antes de continuar a repartição ou limitar o número máximo de folhas da estrutura. Também é possível permitir que o modelo realize o treinamento de forma livre, porém após isso realizar um processo de retirada das pequenas repartições que contém poucos pontos (pruning).

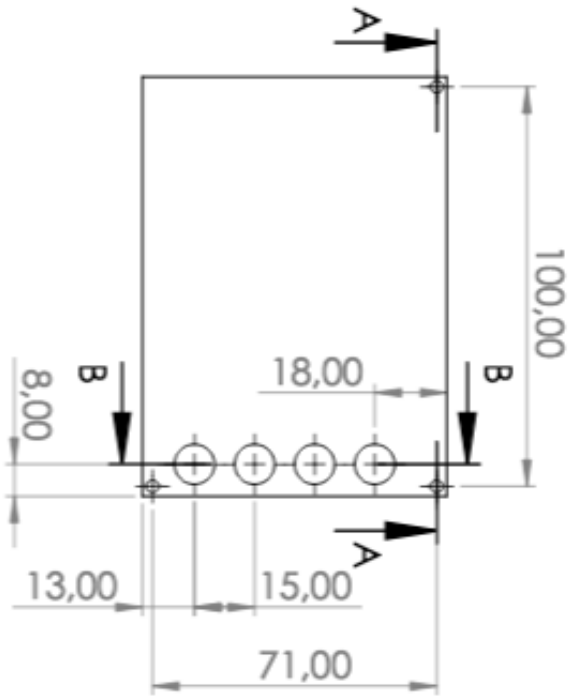
Os modelos de *Decision Tree* disponibilizam ferramentas interessantes que permitem uma análise visual da árvore obtida através do treinamento do modelo, o tempo de treinamento e execução desses modelos é relativamente baixo se comparado com os modelos anteriores. *Decision Tree* também são muito boas para treinar modelos que possuem um *Array* de características (*features*) x onde as características do mesmo possuem tipos de dados diferentes como: dados contínuos, dados binários ou dados em escalas completamente distintas. O maior problema do *Decision Tree* continua sendo sua forte tendência a sobre ajustar o modelo mesmo quando as técnicas citadas são aplicadas

4.7 Case

Um dos passos finais do projeto foi desenvolver uma case que coubesse em uma só mão, com espaço para a ventilação da Raspberry, com saída de áudio, dos botões e entrada para energia.

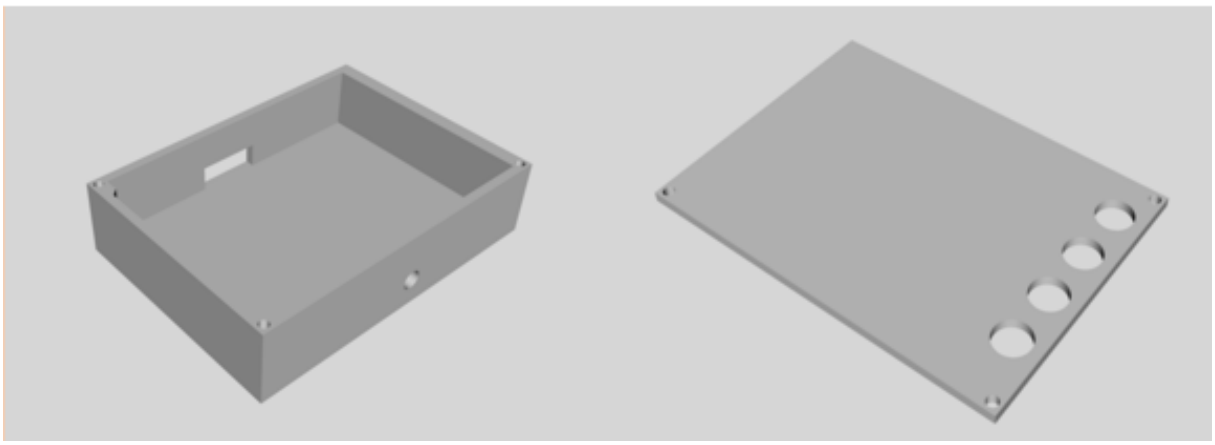
Foi desenvolvida primeiro em desenhos 2D no papel e foi incorporado em um modelo 3D para que pudesse ser gerada por uma impressora 3D.

Figura 50 – Desenho 2D visão da tampa



Fonte: Elaborado pelo autor

Figura 51 – Projeto 3D caixa e tampa



Fonte: Elaborado pelo autor

O resultado final pode ser visto na figura 52, onde no final o tamanho desenvolvido foi de 10,5x7,6x3,5 (LxAxP), com 4 parafusos, um em cada extremidade. Cerca de 10 horas foram exigidas para que a impressora finalizasse a impressão.

Figura 52 – Impressão 3D

Fonte: Elaborado pelo autor

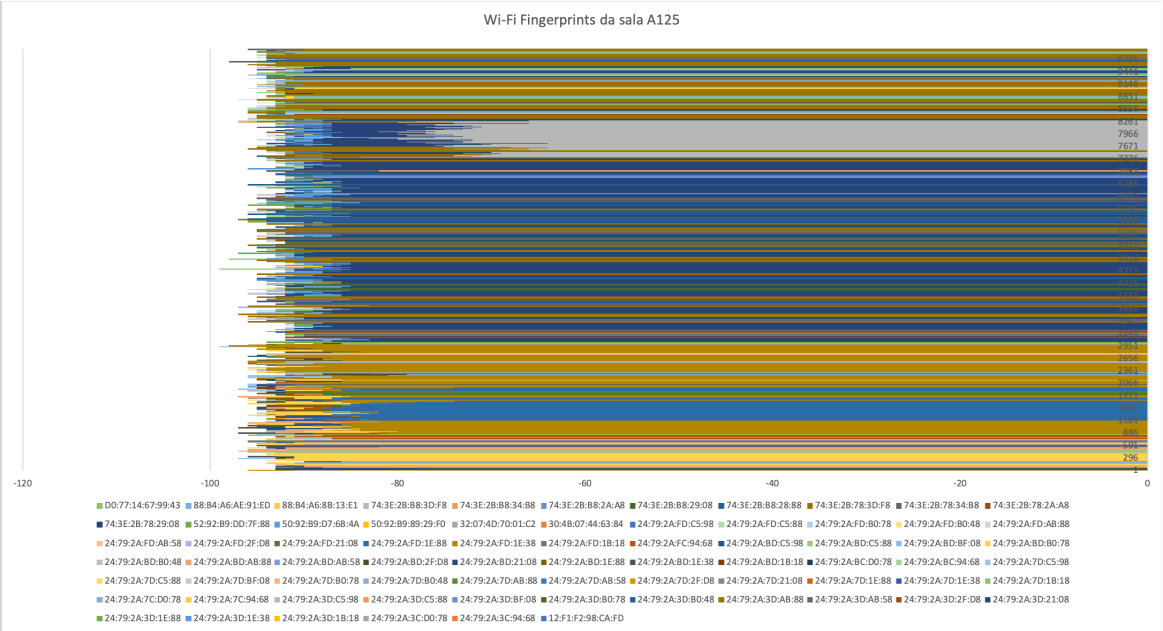
5. RESULTADOS

Os algoritmos de *Machine Learning* disponibilizados pela biblioteca *Scikit-Learn* possuem uma forma de validação do modelo. O *Dataset* deve ser repartido entre o conjunto de *features* (características) e suas respectivas *labels* (classes) e posteriormente como descrito no capítulo 3.7 é necessário dividir o *Dataset* entre dados de treinamento e dados de teste. Dessa forma teremos nossas *features* (características) que podemos chamar de x, e nossas *labels* (classes) que podem ser chamadas de Y agrupadas por treinamento e teste, exemplo: x_treinamento, x_testes, y_treinamento, y_testes. As variáveis x_teste e y_teste são essenciais pois nos permitem averiguar o quão acuradas são as predições do modelo para dados não presentes no *Dataset* de treinamento do mesmo, já que podemos comparar as *labels* (classes) resultantes das predições com as *labels* (classes) corretas presentes em y_teste. Os testes automáticos realizados no modelo são a forma de averiguar o mesmo.

5.1 Amostra dos dados coletados no *site-survey*

O gráfico da figura 53 mostra todos os 10.000 *Wi-Fi Fingerprints* coletados para o ponto de interesse correspondente à sala A125 do prédio acadêmico 1. Foram encontrados 72 pontos de acessos diferentes que juntos compõem os *Wi-Fi Fingerprints* desse ponto de interesse. É importante enfatizar que durante o processo de conversão dos dados presentes no documento JSON para o formato de dados CSV todos os pontos de acesso correspondentes aos outros dispositivos de coleta foram retirados através de uma lista de itens *Mac Adddress* bloqueados, dessa forma os dados não foram influenciados pelos outros dispositivos de coleta.

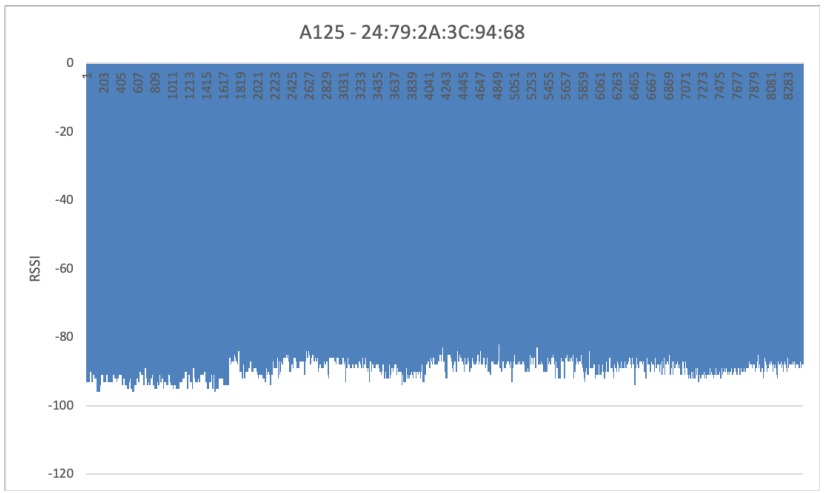
Figura 53 – *Wi-Fi Fingerprints* sala A125



Fonte: Elaborado pelo autor

A imagem abaixo (Figura 54) mostra um gráfico com todas 8.462 leituras realizadas para o *Mac Address* “24:79:2A:3C:94:68”. As médias para os valores RSSI encontrados para esse *Mac Address* foram de -88,5, porém a variância entre os dados foi de 8,2. A variância alta entre os valores RSSI foi um dos principais motivos para o uso de modelos de *Machine Learning* pois indica uma dispersão alta.

Figura 54 – Leituras do *Mac Address* 24:79:2A:3C:94:68



Fonte: Elaborado pelo autor

5.2 Testes realizados nos algoritmos

Primeiro serão analisados os resultados dos 3 algoritmos de *Machine Learning* apresentados anteriormente utilizando um *Dataset* menor em dois testes, o primeiro sem realizar nenhuma configuração para o treinamento e o segundo realizando ajustes para otimização. O último teste consiste em analisar os resultados dos modelos treinados pelos 3 algoritmos com a mesma otimização do teste anterior, porém para um *Dataset* maior que foi construído para a prova de conceito.

O primeiro teste para de treinamento de modelos *Machine Learning* utilizava um *Dataset* com 78.942 *Wi-Fi Fingerprints* formados por 161 pontos de acesso. Esses *Wi-Fi Fingerprints* estavam organizados para 23 pontos de interesse.

A tabela 3 mostra os resultados obtidos pelos modelos utilizando o mesmo *Dataset* descrito anteriormente sem realizar a configuração de nenhum parâmetro. Nenhuma configuração foi realizada nos modelos antes do treinamento. É importante atentar ao fato de que a nota alta no algoritmo de *Decision Tree* é resultado de o algoritmo ter sido capaz de particionar as áreas até o ponto em que cada uma delas se torna pura, isso é perigoso pois pode ser um sinal de sobreajuste do modelo. O K-NN também obteve resultados de treinamento e teste muito próximos o que também indica subajuste do modelo. Já o Linear SVC também obteve notas muito próximas no treinamento e no teste indicando um provável sinal de subasjuste do modelo.

Tabela 3 – Comparação entre os modelos

Modelo	**Tempo de treinamento**	**Nota no treinamento**	**Nota no teste**
K-Nearest Neighbors	2,417654s	0,996645	0,993061
Decision Tree	1,060243s	1,000000	0,995241
Linear SVC	10,620901s	0,988791	0,988124

Fonte: Elaborado pelo autor

Foram realizados ajustes nos parâmetros dos algoritmos dos modelos, os resultados estão na tabela 4. O Linear SVC teve seu método de regularização alterado para L1. Essa técnica remove algumas características (*features*) do modelo, e é útil quando o mesmo possui muitas características (*features*) e muitas delas não são tão

relevantes. O aumento no tempo de treinamento é devido à técnica L1 ser mais lenta que a técnica L2. O parâmetro que controla a regularização L1 é chamado de C, o que controla o quanto a regularização forçará o modelo para uma direção mais genérica ou mais específica. Um parâmetro C maior foi utilizado com o intuito de tornar o modelo mais específico, é interessante notar que testes foram realizados com valores de C igual a 100, C igual a 1.000 e C igual a 10.000, embora teste com valores de C maiores que 100 tenham obtido resultados muito próximos ao C igual a 100, o tempo de treinamento aumentava em conjunto com o aumento do parâmetro C. O K-NN por sua vez teve seus números de vizinhos diminuídos de 5 para 3, tornando o modelo menos genérico do que anteriormente. O algoritmo de *Decision Tree* teve seu número de amostras mínimas para criação de uma repartição alterada de 1 para 50, essa técnica como visto anteriormente é importante para tentar controlar o sobreajuste do modelo.

Tabela 4 – Comparação entre os modelos após ajustes

Modelo	**Tempo de treinamento**	**Nota no treinamento**	**Nota no teste**	**Tempo para execução do modelo**
K-Nearest Neighbors	2,332524s	0,99838	0,995908	0.0199592113495s
Decision Tree	0,981559s	0,996378	0,993551	0.00269603729248s
Linear SVC	133,150768s	0,9943	0,993106	0.00242877006531s

Fonte: Elaborado pelo autor

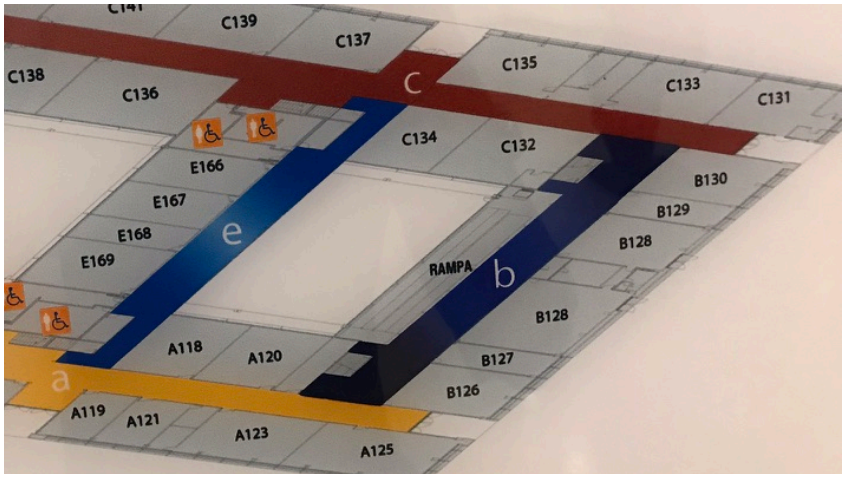
5.2.1 Prova de conceito

Ter um grande volume de dados é essencial para o sucesso de um modelo de *Machine Learning*, uma média de 3.000 *Wi-Fi Fingerprint*s foi estipulada para o modelo inicialmente. Durante os primeiros testes os resultados foram bem insatisfatórios devido à pequena quantidade de dados, pois poucos dados resultam em baixa diversidade no *Dataset*. Um modelo treinado com baixa diversidade de dados acaba se tornando muito específico para aqueles dados utilizados e é incapaz de realizar boas predições para os novos dados. Com o intuito de maximizar a quantidade de amostras por pontos de interesse reduzimos o escopo para realizar uma prova de conceito e aumentamos o número de *Wi-Fi Fingerprints* coletados por ponto de inte-

resse para 10.000 lembrando que 25% desses dados serão destinados à testes. Essa decisão foi tomada pelo fato de não haver tempo hábil para realização do *site-survey* para todos os pontos de interesse. Um *Wi-Fi Fingerprint* leva em média 1 segundo para ser coletado, para realizar a coleta de uma média de 10.000 *Wi-Fi Fingerprints* por sala levaríamos mais de 3 horas, e considerando todas as salas de ambos os acadêmicos levaríamos cerca de 903 horas de *site-survey*.

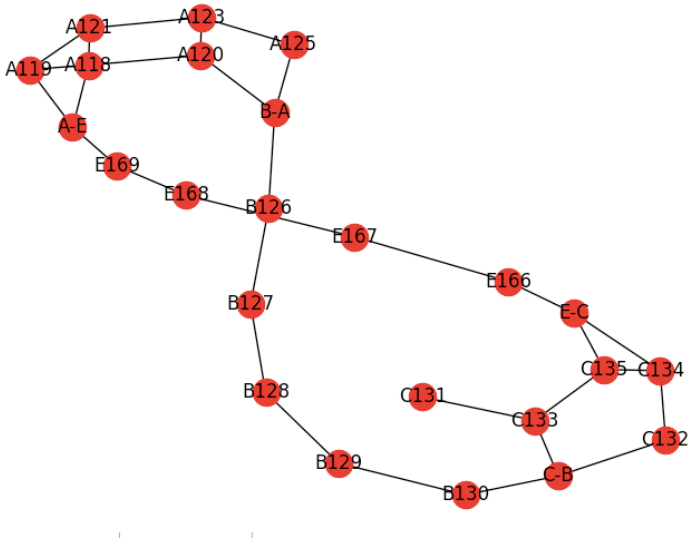
A figura 55 mostra a área do prédio acadêmico 1 considerada para a prova de conceito. Foram considerados como pontos de interesse as salas: E169, E168, E167, E166, C135, C134, C133, C132, C131, B130, B129, B128, B127, B126, A125, A123, A122, A121, A120, A119, A118, E169, E168, E167 e E166. Para que o usuário pudesse mudar entre os corredores A, B, C e E, foram considerados mais quatro pontos de interesse que realizam a intersecção entres os corredores, são eles: A-E, B-A, C-B, E-C. A figura 56 mostra a estrutura de grafo construída a partir dos pontos te interesse considerados.

Figura 55 – Circuito considerado no teste



Fonte: Elaborado pelo autor

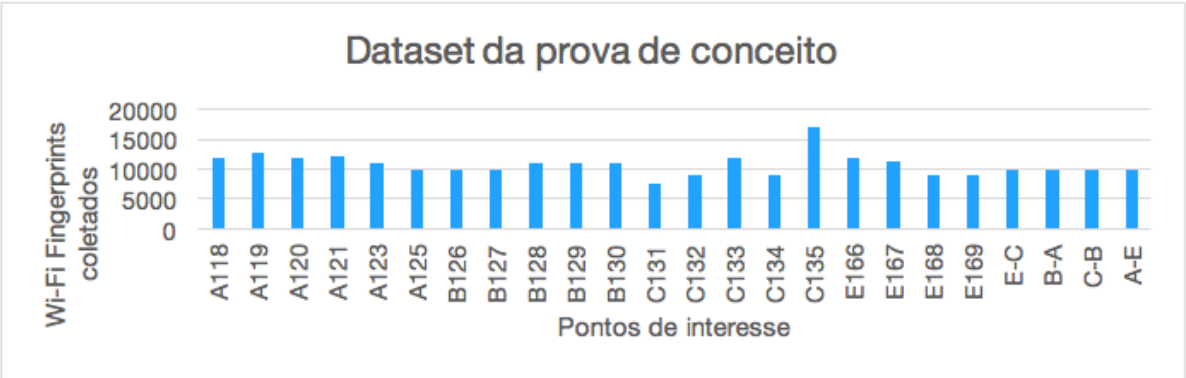
Figura 56 – Grafo dos pontos de interesse



Fonte: Elaborado pelo autor

O *Dataset* final utilizado para os testes continha 258.369 *Wi-Fi Fingerprints* constituídos por 229 pontos de acesso diferentes encontrados para um total de 24 pontos de interesse. A Figura 57 mostra o total de *Wi-Fi Fingerprints* coletados para cada ponto de interesse do escolhido para a prova de conceito.

Figura 57 – Total de *Wi-Fi Fingerprints* para cada ponto de interesse



Fonte: Elaborado pelo autor

A tabela 5 mostra os resultados obtidos pelos modelos treinados utilizando as mesmas configurações dos testes anteriores para o *Dataset* construído para a prova de conceito. É importante notar o aumento no tempo de treinamento agora com um *Dataset* 227,29% maior que o anterior. O tempo de treinamento do K-NN aumentou cerca de 1.583,25%, embora ainda continue com um tempo baixo, fica evidente que o algoritmo não escala bem. O algoritmo Linear SVC também teve um aumento no tempo de treinamento de 5.548,07%, um tempo que já era o maior dos 3. Existem

alternativas para o algoritmo Linear SVC como o algoritmo SGDClassifier que possui otimizações para grandes *Datasets*. Embora todos os modelos tenham obtido bons resultados o *Decision Tree* foi o algoritmo mais assertivo dos 3 nos testes realizados com o dispositivo.

Tabela 5 – Resultados obtidos com os modelos treinados

Modelo	**Tempo de treinamento**	**Nota no treinamento**	**Nota no teste**	**Tempo para execução do modelo**
K-Nearest Neighbors	39,262101s	0,998177	0,994860	0,348535060883
Decision Tree	6,512316s	0,994921	0,991223	0,00488209724426
Linear SVC	599,875477s	0,985195	0,985049	0,0099949836731

Fonte: Elaborado pelo autor

A biblioteca *Scikit-Learn* disponibiliza um recurso muito interessante chamado matriz de confusão (confusion matrix). Esse recurso recebe como parâmetro duas listas de classes (labels), a primeira lista contém a classes corretos, já a segunda lista contém os resultados das predições do modelo. A matriz de confusão consiste em relacionar a classes corretas com as classes resultantes um a um, por isso é importante que ambas as listas estejam na ordem. Durante a fase de treinamento do modelo, uma parte do *Dataset* foi destinada para realizar os testes e dividida entre as características (*features*) X e as classes correspondentes (labels) Y, essa fração é responsável pelo resultado do teste. A matriz de confusão é montada a partir desses dados e é uma forma mais detalhada para análise da performance do modelo em cada um dos pontos de interesse. A matriz de confusão contendo todos os resultados do teste pode ser encontrada no apêndice desse documento, porém a tabela 6 mostra os resultados da matriz de confusão para as classes: A-E, A118, A119, A120, A121 e A123.

Tabela 6 – Matriz de confusão dos testes do modelo *Decision Tree*

label	A-E	A118	A119	A120	A121	A123
A-E	2985	4	0	0	0	0
A118	0	3515	7	11	74	8
A119	0	12	3825	0	41	0
A120	0	5	0	3616	1	3
A121	0	80	26	0	3615	0
A123	0	22	0	9	5	3241

Fonte: Elaborado pelo autor

Durante os testes finais realizados com o usuário sendo guiado pelo dispositivo todos os algoritmos obtiveram uma performance pior ao receber *Wi-Fi Fingerprints* atuais. Primeiramente o modelo obteve uma chance ligeiramente maior de acertar o ponto de interesse quando o dispositivo estava distante do usuário, o que contribuiu para isso é o fato de que durante o processo de *site-survey* do ambiente o dispositivo de coleta não possuía interferência humana próxima. O corredor A foi de longe o corredor mais problemático para o modelo, um ponto que contribuiu para isso foi a alta incidência de sinais de pontos de acesso distintos como mostrado pelo gráfico da figura 53 onde a sala A125 que foi coletada durante um único dia encontrou mais de 72 pontos de acesso distintos. Em geral as salas no corredor C obtiveram resultados melhores onde quando as predições do modelo estavam erradas geralmente eram a sala ao lado ou em frente. Alguns pontos de intersecção como o ponto entre os corredores E-C se provaram muito ruins pois estavam muito próximos das salas, cerca de menos de 3 metros, o que gerava resultados de predições quase sempre errados apontando para as salas mais próximas como C134 ou C135.

6. CONCLUSÃO

O objetivo desse projeto era desenvolver um sistema capaz de guiar os alunos com deficiência visual pelas salas dos prédios acadêmicos 1 e 2 do Centro Universitário Senac. Para isso a técnica de localização através do uso de *Wi-Fi Fingerprints* foi escolhida, pois um dos pontos mais atrativos dessa técnica era a possibilidade de implementar todo o sistema sem a necessidade de mudar nada na estrutura dos prédios pois a técnica apenas faria uso do que já está presente, ou seja, os pontos de acesso.

Para realizar o reconhecimento através do *Wi-Fi Fingerprint* primeiro foi necessário escolher um hardware que pudéssemos utilizar para a construção dos dispositivos portáteis. Encontramos uma ótima opção com um custo benefício alto que atendia as necessidades iniciais do projeto. Após a definição do hardware desenvolvemos um software específico para realizar o *site-survey* e logo ficou claro que as técnicas como cálculo da distância Euclidiana sozinhas não eram suficientes para resolver o problema de reconhecimento do *Wi-Fi Fingerprint* devido a enorme dispersão dos dados.

Depois de pesquisar alternativas, encontramos a técnica de *Machine Learning* que até então era totalmente desconhecida para nós. Foi necessária muita pesquisa na área, conversas com alguns profissionais do ramo e testes para entender a melhor forma de agrupamento dos dados e aplicação das técnicas disponibilizadas. Após isso foi necessário criar todo um fluxo de coleta de dados através do *site-survey* necessário para gerar um volume suficiente para o uso das técnicas de *Machine Learning* apresentadas no documento, esse processo além de complicado é demorado e só foi possível pois utilizamos 3 placas diferentes e revezamos em turnos para vigiar as placas enquanto elas realizavam as coletas.

Para aprimorar a localização do usuário foi necessário incluir um sensor de bússola HMC5883L fazendo com que nosso dispositivo consiga determinar qual a direção o mesmo está apontando, podendo corrigir sua trajetória com mais precisão. Sobre o usuário, foi necessário adicionar a funcionalidade de saída de áudio por fone de ouvido P2 uma vez que a placa escolhida não possuía. Tudo isso foi acoplado

em um *Shield* desenvolvido à mão para que o hardware ficasse mais confiável e ocupasse menos espaço.

Quando decidimos utilizar modelos de *Machine Learning* no projeto pensávamos em executar os mesmos direto nos dispositivos e no início ocorreu dessa forma. Porém conforme o volume de dados aumentava os dispositivos não foram capazes de continuar realizando o treinamento devido aos seus recursos limitados. Foram realizadas inúmeras tentativas de treinar esses modelos em máquinas virtuais e exportá-los para os dispositivos utilizando Scikit-Learn, porém não obtivemos sucesso visto que os modelos estão fortemente atrelados a arquitetura em que foram treinados e não foi possível criar uma máquina virtual com o mesmo sistema operacional Raspbian e recursos suficientes para o processo de treinamento.

Reduzimos o escopo para coletar um volume de dados suficiente em tempo hábil e podermos testar os dispositivos submetendo *Wi-Fi Fingerprints* para uma API que executava o modelo. Obtivemos sucesso construindo uma interface com quatro botões e áudio para que um usuário pudesse utilizar sem a necessidade de uma tela. Também propomos uma forma de contornar o problema de os dados envelhecerem e não refletirem mais o estado real do ambiente já que após o *site-survey* inicial o sistema é capaz de se manter atualizado através dos dispositivos que conforme são usados pelo usuário submetem novas amostras que podem ser utilizadas para treinar novamente o modelo.

Por fim é importante notar que no *Dataset* geral contém um número muito grande de pontos de acesso, cerca 320 dispositivos diferentes. Esse grande número é devido ao fluxo de alunos e funcionários estavam com pontos de acesso em seus celulares e outros dispositivos ligados e foram captados pelos dispositivos durante o *site-survey*. Embora seja fácil criar uma lista contendo os itens *Mac Address* que serão aceitos, essas informações não foram disponibilizadas para nós. A diversidade dos dados é um fator chave para um alto desempenho da técnica de *Wi-Fi Fingerprints com Machine Learning*, embora tenhamos alcançado uma precisão alta nos testes essa precisão no *Dataset* não reflete completamente o cenário real já que a quantidade de variáveis como fluxo de pessoas é enorme e precisaríamos realizar uma coleta extensa que não seria possível dentro do prazo do projeto. Contudo o sistema foi capaz de identificar a posição do usuário diversas vezes e nas vezes em

que predições erradas foram realizadas o resultado era uma sala ao lado ou em frente, cerca de 2-3 metros de distância. Ao todo mais de 313.001 *Wi-Fi Fingerprints* foram coletados ao longo desse projeto.

7. TRABALHOS FUTUROS

Para desenvolver modelos de *Machine Learning* capazes de realizar predições assertivas é necessário entender bem os dados que temos em mãos, assim podemos criar um fluxo de constante treinamento do modelo visto que a disposição do ambiente pode mudar. Os trabalhos futuros necessários para otimizar o projeto são:

- Diferenciar os pontos de acesso do Senac dos pontos de acesso provenientes de dispositivos móveis de alunos e funcionários.
- Comparar resultados dos modelos treinados com e sem esses pontos de acesso.
- Analisar o declínio da assertividade do modelo ao longo do tempo para propor um fluxo de novos treinamentos para o modelo já presente na API.
- Retrabalhar a distribuição de pontos de interesse para distanciar os mesmos com o objetivo de aumentar a assertividade do modelo.
- Redesenhar o projeto da case para incluir uma bateria portátil para o dispositivo.

Todo o código desenvolvido para esse projeto está documentado e disponível em conjunto com os dados adquiridos em um repositório do *GitHub*. O link para do repositório pode ser encontrado em:
https://github.com/HenriqueLBorges/USO-DE-TECNOLOGIA-ASSISTIVA-PARA-GUIAR-ALUNOS-COM-DEFICIENCIA-VISUAL-NO-CAMPUS-SENAC-SANTO-AMARO.git

REFERÊNCIAS

WIFI ANALYZER, Aplicativo. Disponível em <https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer&hl=pt_BR>. Acessado em 20 de maio de 2018.

BRASIL, Artigo 4º lei Nº 9.394. Disponível em <https://www.jusbrasil.com.br/topicos/11696628/artigo-4-da-lei-n-9394-de-20-de-dezembro-de-1996 >. Acessado em 22 de abr. 2018.

BRASIL, Cartilha Censo 2010. Disponível em <http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/cartilha-censo-2010-pessoas-com-deficienciareduzido.pdf>. Acessado em 22 de abr. 2018.

CAROLINE, Luana. Bússola. Disponível em <https://www.todoestudo.com.br/geografia/bussola>. Acessado em 30 de maio de 2018.

CHENSHU, Wu; ZHENG, Yang; ZIMU, Zhou, YUNHAO, Liu; MINGYAN, Liu, DorFin: WiFi Fingerprint-based Localization Revisited, 2013. Disponível em <https://arxiv.org/pdf/1308.6663.pdf>. Acessado em 17 de maio de 2018.

CONSOLO, Ana Carolina Pecin; GIANULLO, Wilson. Direito dos Portadores de Deficiência física, 2011. Disponível em <http://www.mackenzie.com.br/fileadmin/Pesquisa/pibic/publicacoes/2011/pdf/dir/ana_carolina_pecin.pdf>. Acessado em 22 de abr. 2018.

GAMES, Red Blob. Introduction to A*, 2018. Disponível em <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>. Acessado em 15 de junho de 2018.

GCC Indoor Location-based Services (LBS) Market Analysis by Product, by Technology, by Application (Monitoring, Navigation, Proximity, Tracking), by End-use, by Country and Segment Forecasts, 2014 – 2025. 2017. Disponível em

<https://www.researchandmarkets.com/research/zflfgn/gcc_indoor>. Acessado em 10 de maio 2018.

GUIDO, Sarah; MÜLLER, Andreas C. Introduction to Machine Learning with *Python*, 2017. Disponível em <https://books.google.com.br/books/about/Introduction_to_Machine_Learning_with_Py.html?id=qjUVogEACAAJ&redir_esc=y>. Acessado em 25 de novembro de 2018.

INSTITUTO NEWTON C BRAGA, Filtros ativos usando amplificadores operacionais. Disponível em <http://www.newtoncbraga.com.br/index.php/artigos/49-curiosidades/1748>. Acessado em 25 de setembro de 2018.

KIRK, Matthew, Thoughtful Machine Learning, 2015. Disponível em <https://doc.lagout.org/science/Artificial%20Intelligence/Machine%20learning/Thoughtful%20Machine%20Learning_%20A%20Test-Driven%20Approach%20%5BKirk%202014-10-12%5D.pdf>. Acessado em 25 de novembro de 2018.

LANDMAN, Nathan; ROSS, Eli; WILLIAMS, Christopher. Secure Hashing Algorithms, Brilliant.org, 2018. Disponível em <https://brilliant.org/wiki/secure-hashing-algorithms/>. Acessado em 22 de junho de 2018.

MAHAJAN, Anupam; CHANANA, Madhur, Wi-Fi Localization using RSSI in Indoor Environment via a smartphone, 2012. Disponível em <https://www.ijecs.in/index.php/ijecs/article/download/39/35/>. Acessado em 10 de maio de 2018.

MARKET RESEARCH FUTURE, Global Indoor Positioning and Navigation System Market Research Report- Forecast 2022, Market Research Future, 2018. Disponível em <https://www.marketresearchfuture.com/reports/indoor-positioning-navigation-system-market-1775>. Acessado em 26 de maio de 2018.

MORAIS, Fernando Antônio de Andrade. A Importância da Acessibilidade na Cidade. Disponível em

<https://semanaacademica.org.br/system/files/artigos/artigo_33.pdf>. Acessado em: 16 abr. 2018.

MRINDOKO, Nicholaus R.; TITO, Stanley L.; RUAMBO, Francis A., PERFORMANCE ANALYSIS OF FINGERPRINTING ALGORITHMS USED IN WLAN POSITIONING SYSTEM, 2016. Disponível em <http://www.ijsrp.org/research-paper-1116.php?rp=P595965 >. Acessado em 10 de maio de 2018.

NOSRATI, Masoud; KARIMI, Ronak; HASANVAND, Hojat Allah. Investigation of the * (Star) Search Algorithms- Characteristics, Methods and Approaches, Vol 2 N4, 2012. Disponível em <https://pdfs.semanticscholar.org/831f/f239ba77b2a8eaed473ffbfa22d61b7f5d19.pdf >. Acessado em 15 de junho de 2018.

RAJA, Deepti Samant. Bridging the Disability Divide through Digital Technologies. Background Paper, 2016. Disponível em <http://pubdocs.worldbank.org/en/123481461249337484/WDR16-BP-Bridging-the-Disability-Divide-through-Digital-Technology-RAJA.pdf>. Acessado em: 20 abr. 2018.

RASPBERRY, Raspberry PI zero W. Disponível em <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>. Acessado em 20 de novembro de 2018.

RASPI.TV, How much power does Pi Zero W use, 2017. Disponível em <https://raspi.tv/2017/how-much-power-does-pi-zero-w-use>. Acessado em 20 de novembro de 2018.

SCIKITLEARNING, Machine Learning. Disponível em <https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>. Acessado em 20 de novembro de 2018.

SILVA, KELE CRISTINA. Condições de Acessibilidade na Universidade, 2016. Disponível em <https://repositorio.unesp.br/bitstream/handle/11449/138845/silva_kc_me_mar.pdf?sequence=3>. Acessado em 29 de abr. 2018.

SOCIAL COMPARE BETA, RaspberryPI models comparison, 2018. Disponível em <http://socialcompare.com/en/comparison/raspberrypi-models-comparison>. Acessado em 26 de junho de 2018.

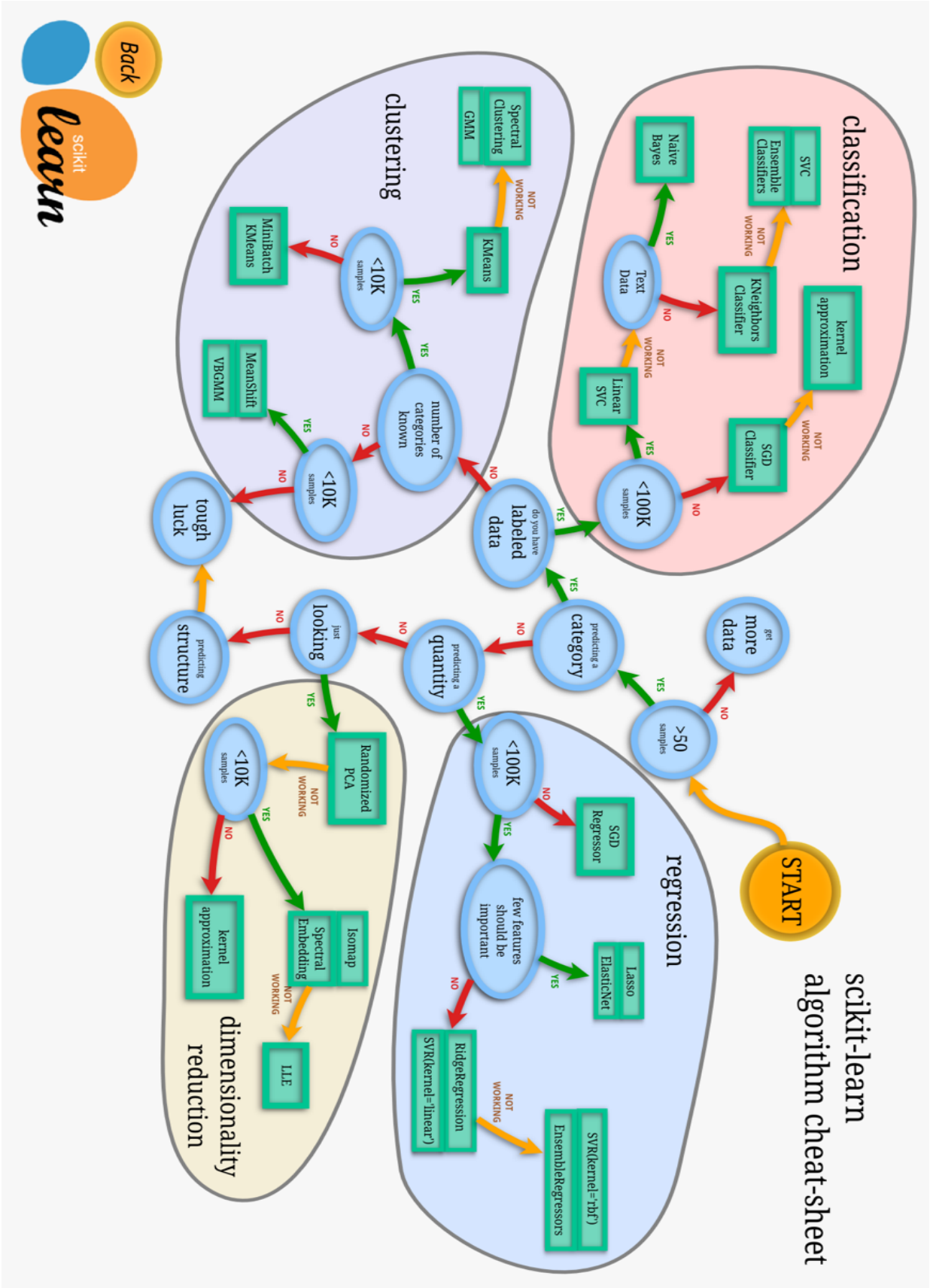
SOUNDOFTEXT, Converter text to áudio. Disponível em <https://soundoftext.com>. Acessado em 20 de novembro de 2018.

SUINING, HE; CHAN, S.-H. Gary, Wi-Fi Fingerprint-Based Indoor Positioning, 2015. Disponível em <https://ieeexplore.ieee.org/document/7174948/ >. Acessado em 17 de maio de 2018

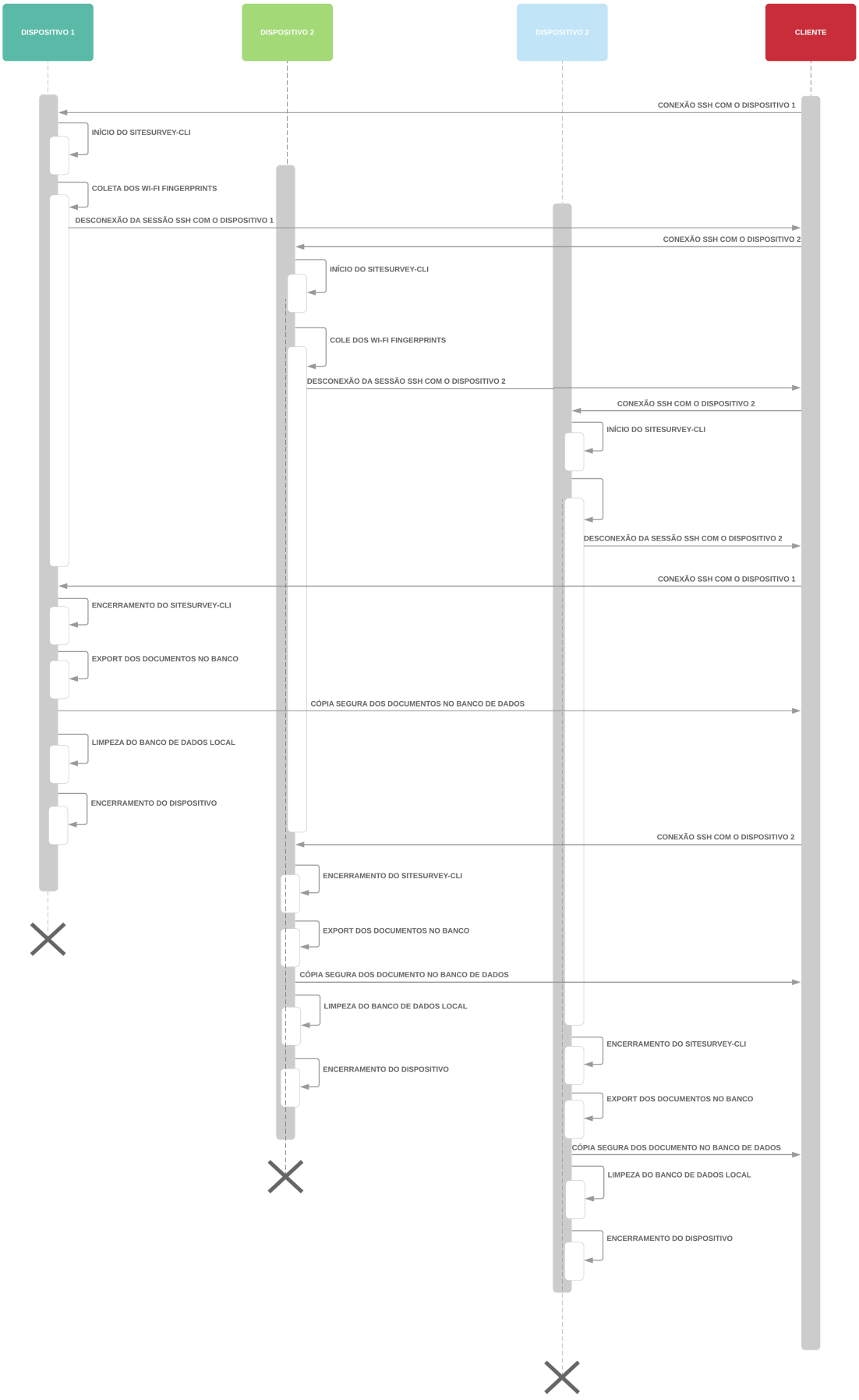
SUINING, He; SHUENG-HAN, Gary Chan, Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons, 2016. Disponível em <https://www.semanticscholar.org/paper/Wi-Fi-Fingerprint-Based-Indoor-Positioning%3A-Recent-He-Chan/3e7aabaffdb4b05e701c544451dce55ad96e9401?tab=abstract>. Acessado em 20 de maio de 2018.

R.S, Michalski; J.G, Carbonell; T.M, Mitchell, Machine Learning, 2013. Disponível em <https://books.google.com.br/books?hl=pt-BR&lr=&id=-eqpCAAAQBAJ&oi=fnd&pg=PA2&dq=paper+about+machine+learning&ots=Wl3POE1Df3&sig=pzTXR1Qp5Y_Am2XlozP-8kf3VOQ#v=onepage&q=paper%20about%20machine%20learning&f=false>. Acessado em 10 de dezembro de 2018.

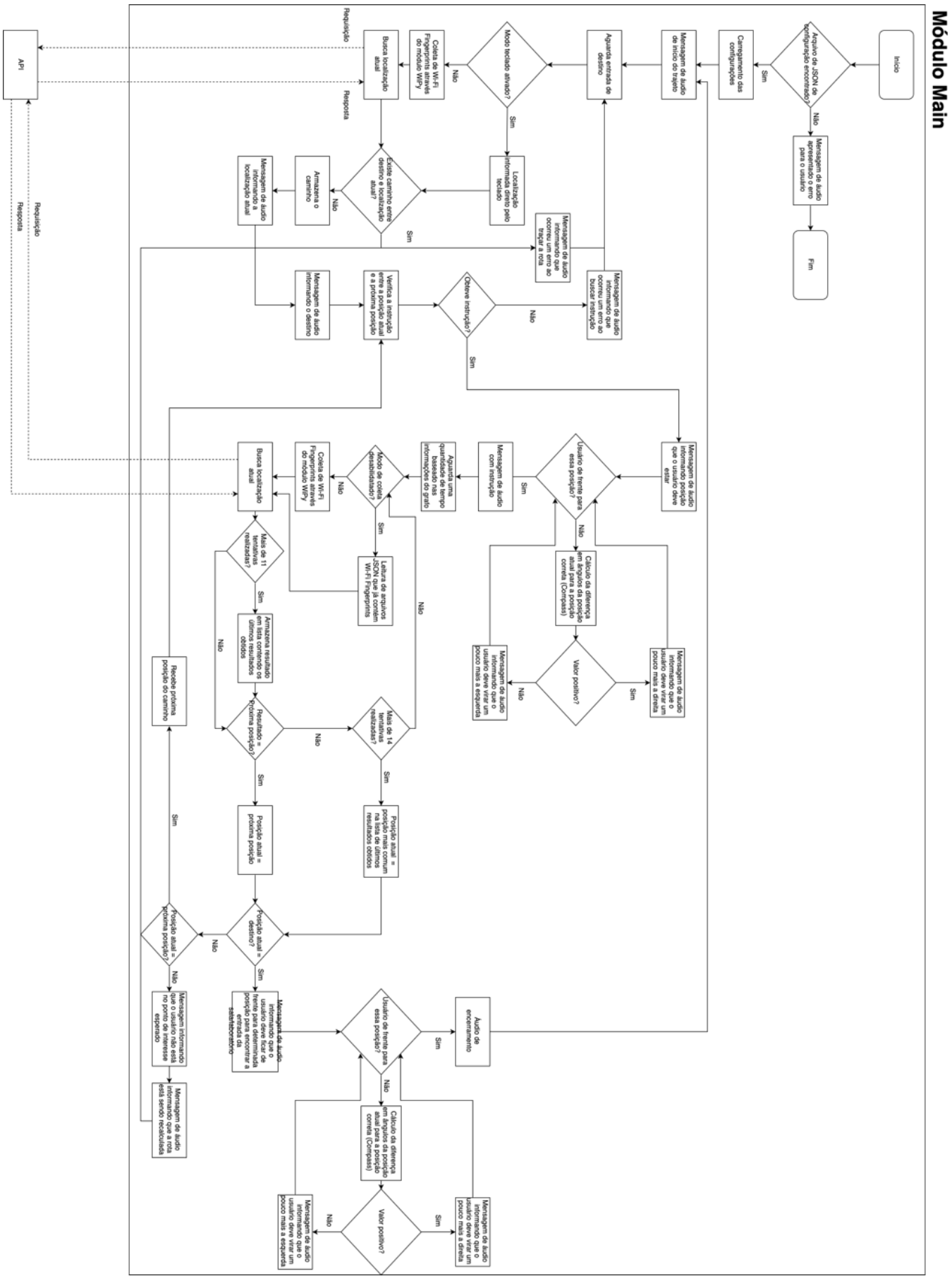
APÊNDICE A – Algoritmos de Machine Learning



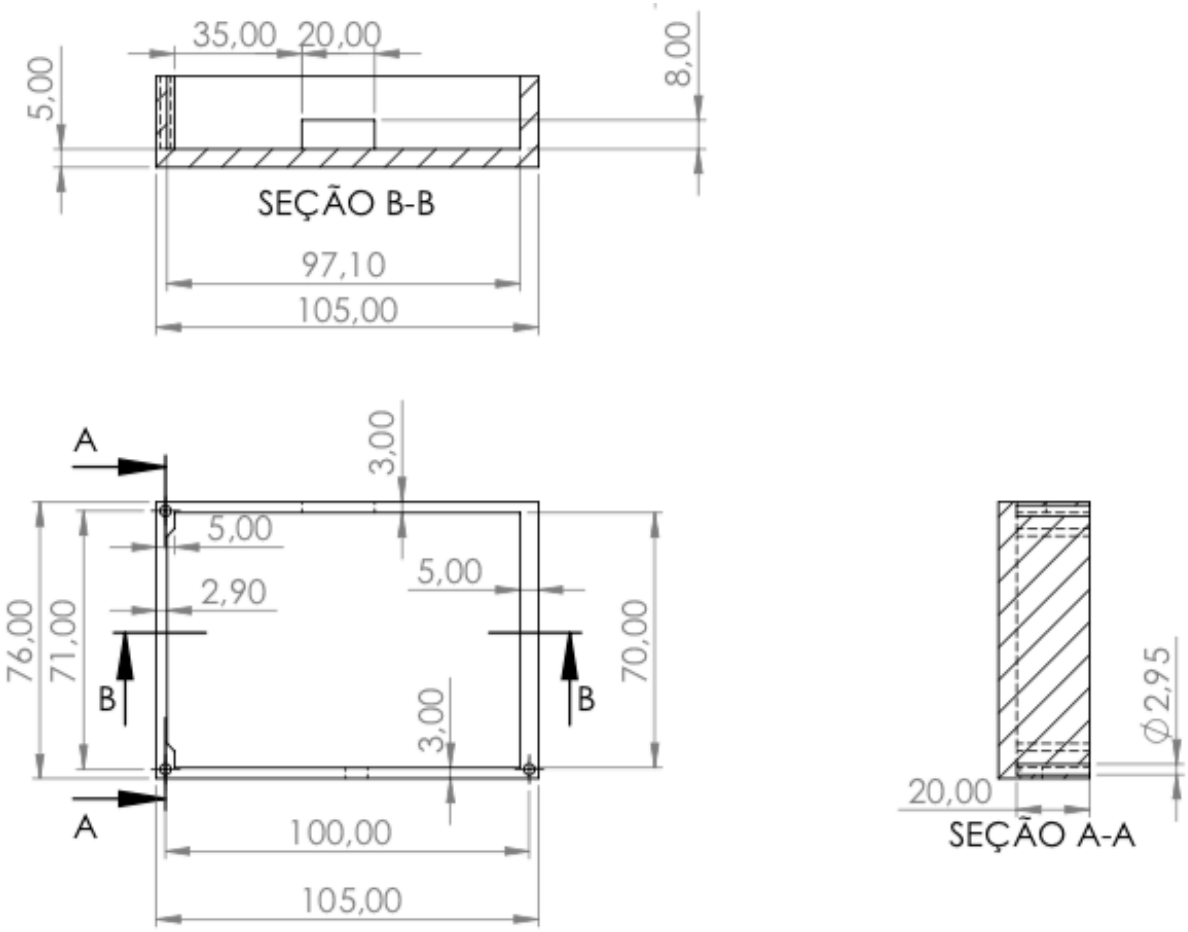
APÊNDICE B – Diagrama de sequência da coleta e envio dos *Wi-Fi Fingerprints*



APÊNDICE C – Diagrama módulo *Main*



APÊNDICE D – Projeto Case 2D



APÊNDICE E – Matriz de confusão dos testes do modelo *Decision Tree*

label	A-E	A118	A119	A120	A121	A123	A125	B-A	B126	B127	B128	B129	B130
A-E	2985	4	0	0	0	0	0	0	11	0	0	0	0
A118	0	3515	7	11	74	8	0	0	0	0	0	0	0
A119	0	12	3825	0	41	0	0	0	0	0	0	0	0
A120	0	5	0	3616	1	3	1	1	4	0	0	0	0
A121	0	80	26	0	3615	0	0	0	0	0	0	0	0
A123	0	22	0	9	5	3241	2	1	0	0	0	2	0
A125	0	0	0	0	0	0	2869	0	2	0	0	0	0
B-A	0	2	0	0	0	3	0	2970	0	1	0	0	0
B126	0	0	0	0	1	0	0	2	2994	2	0	0	0
B127	0	0	0	0	0	0	1	0	0	2976	5	0	0
B128	0	0	0	0	0	0	0	0	0	0	3306	3	0
B129	0	0	0	0	0	0	0	0	0	0	0	3267	5
B130	0	0	0	0	0	0	1	0	0	0	0	13	3252
C-B	0	0	0	0	0	0	0	0	0	0	0	0	2
C131	0	0	0	0	0	0	1	0	0	0	3	0	7
C132	0	0	0	0	0	0	0	0	0	0	0	2	10
C133	0	0	0	0	0	0	0	0	0	0	0	0	7
C134	0	0	0	0	0	0	0	0	0	0	0	3	2
C135	0	0	0	0	0	0	0	0	0	3	0	5	1
E-C	0	0	0	0	0	0	0	0	0	0	0	0	0
E166	0	0	0	0	0	0	0	0	0	0	0	0	0
E167	0	0	0	0	1	0	0	0	0	0	1	0	0
E168	0	0	0	0	1	0	0	0	0	1	0	0	0
E169	0	0	0	0	1	0	0	0	0	0	0	0	0

C-B	C131	C132	C133	C134	C135	E-C	E166	E167	E168	E169
0	3	2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0
0	1	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
2	2	6	3	0	0	0	0	0	0	0
2952	1	0	4	1	0	0	0	0	0	0
0	2250	0	5	0	0	0	0	0	3	0
0	0	2648	5	1	16	0	0	0	0	0
0	22	10	3552	0	3	0	0	0	0	0
0	0	6	0	2594	15	7	0	0	1	0
1	0	10	4	18	5138	3	0	0	0	0
0	0	0	1	11	3	2954	0	0	0	0
0	0	1	0	0	0	0	3653	0	4	0
0	0	0	0	0	0	0	1	3299	2	0
0	1	0	0	0	0	0	0	0	2643	19
0	1	0	0	0	0	0	0	0	22	2744