LEONARDO SANTHIAGO TOPPEL
LUCIANO ALLA LAGOZA

PROJETO DE MOBILIDADE PARA DEFICIENTE VISUAL

Curitiba
2019

LEONARDO SANTHIAGO TOPPEL
LUCIANO ALLA LAGOZA

PROJETO DE MOBILIDADE PARA DEFICIENTE VISUAL

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Engenheiro Eletricista com ênfase em Sistemas Eletrônicos Embarcados da Universidade Federal do Paraná

Orientador: Prof. Dr. Marcos Vinicio Hass Rambo.

Curitiba
2019

TERMO DE APROVAÇÃO

LEONARDO SANTHIAGO TOPPEL
LUCIANO ALLA LAGOZA

PROJETO DE MOBILIDADE PARA DEFICIENTE VISUAL

Trabalho de conclusão de curso apresentado como requisito parcial para à obtenção do grau de Engenheiro Eletricista com Ênfase em Sistemas Eletrônicos Embarcados, do Setor de Tecnologia da Universidade Federal do Paraná, pela seguinte banca examinadora:

_____

Prof. Dr. Marcos Vinicio Haas Rambo
Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Sibilla Batista da Luz França
Departamento de Engenharia Elétrica, UFPR

_____

Prof. MSc. Ricardo Schumacher
Departamento de Engenharia Elétrica, UFPR

Curitiba, 22 de novembro de 2019.

AGRADECIMENTO

Agradecemos a todos pela colaboração e apoio de todos que de alguma forma contribuíram para que este trabalho fosse executado, em especial nossos familiares que nos apoiaram durante todo o decorrer do curso de Engenharia Elétrica, nos momentos de êxito e também nos momentos de frustração, em especial durante o decorrer do desenvolvimento de nosso TCC que demandou muita dedicação e perseverança para superar as barreiras que apareceram no decorrer do projeto.

Gostaríamos de agradecer também a todos os professores que nos apoiaram desde o início do curso e nos forneceram o conhecimento necessário para alcançarmos o ponto que estamos atualmente, em especial gostaríamos de agradecer ao professor orientador Marcos Vinicio Haas Rambo que nos suportou nas tomadas de decisão durante as alterações no projeto e também colaborou com sugestões que certamente contribuíram muito para obter o melhor resultado possível neste projeto.

Por fim fica o agradecimento a todos envolvidos nesta jornada, professores, técnicos, colegas de turma e de trabalho, a contribuição de todos em pequenas ações proporcionou nossa evolução não somente no âmbito acadêmico e técnico mas também no nosso desenvolvimento pessoal, proporcionando que ao final deste trabalho fossemos capazes de retribuir este investimento para a sociedade através da elaboração de uma proposta para oferecer mais independência a outros indivíduos.

RESUMO

O presente trabalho aborda o desenvolvimento de uma solução de tecnologia assistiva voltada para mobilidade urbana, especificamente para auxiliar pessoas com baixa visão com isso o desenvolvimento é dividido em três grandes vertentes sendo estas dispositivo microcontrolado, aplicativo *Android* e banco de dados, o dispositivo microcontrolado e o aplicativo irão servir como interface entre o usuário com baixa visão e o motorista dentro do ônibus, a interface *Android* será baseada em reconhecimento de voz uma vez que o público alvo necessita desta função, o dispositivo microcontrolado tem como objetivo alertar o motorista durante o percurso da distância até o usuário, estes dados são apresentados em um display e são obtidos através de GPS assim como o aplicativo necessita desta funcionalidade para possibilitar as comparações, o banco de dados funciona como um intermediário para disponibilizar as informações de localização e informações logicas de ambos os dispositivos. De modo a atingir os resultados esperados foi utilizado um banco de dados em tempo real, o *Firebase* que possibilita a interação dinâmica entre os dispositivos, outro recurso utilizado foi a tecnologia Wi-Fi para a conexão do dispositivo microcontrolado com o servidor. Foram desenvolvidos diversos artifícios lógicos para evitar erros e redundância no projeto, estes resultados são apresentados no final deste documento com os resultados de testes e validações realizadas. A metodologia do projeto se caracteriza como aplicada uma vez que ao final do projeto foi construído um protótipo funcional.

Palavras-chave: Mobilidade. Técnologia assistiva. Android. GPS. Wi-Fi. Reconhecimento de voz. Baixa visão.

ABSTRACT

The present work deals with the development of an assistive technology solution focused on urban mobility, specifically to assist visual impaired people. Thus, the development is split into three main aspects: microcontroller, Android application and database, microcontroller and the app will serve as an interface between the visually impaired user and the driver inside the bus. The android interface is based on voice recognition since the target audience needs this function. The microcontroller device aims to alert the driver during the ride showing the distance to the user, this data is shown on a display and is obtained through GPS just as the app needs this functionality to enable logical comparisons within the device, the database acts as an intermediary step to provide the location information and logical information of both devices. In order to achieve the expected results, a real-time database (Firebase) was developed, which enables dynamic interaction between devices. Another feature used was Wi-Fi technology for connecting the microcontrolled device to the server. Several logics were developed to avoid errors and redundancy in the project. These results are presented at the end of this document with the results of tests and validations performed. The project methodology is characterized as applied since a functional prototype was built at the end of the project.

Keywords: Mobility. Assistive technology. Android GPS Wi-Fi. Voice Recognition. Visual Impaired.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	-	Interface de programação de aplicações
APK	-	*Android Package*
DSSS	-	*Direct Sequence Spread Spectrum*
FHSS	-	*Frequency Hopping Spread Spectrum*
GPS	-	Sistema de Posicionamento Global
GSM	-	Sistema Global para Comunicações Móveis
IDE	-	Ambiente de desenvolvimento integrado
IoT	-	Internet das Coisas
MSC	-	*Mobile Switching Center*
NMEA	-	National Marine Electronics Association
RDBMS	-	Sistemas de Gerenciamento de Banco de Dados Relacionais
RF	-	Rádio Frequência
SSID	-	*Service Set Identifier*
SQL	-	*Structured Query Language*
TDMA	-	*Time Division Multiple Access*
TDMA	-	*Time Division Multiple Access*
UART	-	*Universal Asynchronous Receiver / Transmitter*
UMTS	-	*Universal Móbile Telecommunications System*

SUMÁRIO

1 INTRODUÇÃO

Com o passar do tempo a engenharia vem contribuindo cada vez mais para o desenvolvimento de soluções que tem como finalidade aperfeiçoar processos e dispositivos já existentes ou ainda criar novos processos e dispositivos para cada vez mais facilitar a vida das pessoas. Uma área da engenharia elétrica que vem crescendo nos últimos anos é a área relacionada à tecnologia assistiva que está passando a ter um papel social importante assim como o papel de auxílio para reabilitação.

Neste cenário, o uso de tecnologia como uma ferramenta para prover melhor qualidade de vida para o usuário e para possibilitar que o mesmo se torne cada vez mais independente, este projeto abordara uma aplicação para auxiliar pessoas com deficiência visual a se locomoverem de ônibus.

Esta solução será criada de forma a possibilitar que o usuário com deficiência visual consiga acessar com facilidade o aplicativo e que o mesmo funcione de forma amigável, avisando o usuário quando o ônibus estiver próximo, além disto um dispositivo será acoplado ao ônibus para que o mesmo sinalize ao motorista quando existe um usuário com deficiência visual próximo a rota do ônibus.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo do trabalho é desenvolver um sistema composto por um dispositivo acoplado ao ônibus e um aplicativo para o sistema operacional *Android* que possibilite que o usuário com deficiência visual selecione uma linha de ônibus e que o motorista seja avisado que existe um usuário com interesse em sua linha quando o mesmo estiver próximo.

O dispositivo que será acoplado ao ônibus deverá emitir um alerta visual para o motorista, além disto o dispositivo deverá enviar a posição do ônibus para um banco de dados que irá comparar a posição do ônibus e do usuário antes de emitir o alerta para o motorista e para o usuário.

O objetivo geral do projeto é elaborar o hardware a ser acoplado ao ônibus, o aplicativo com interface sonora e realizar a integração de ambos.

1.1.2 Objetivos Específicos

Dentre os principais objetivos específicos destacam-se:

1. Avaliar a aplicabilidade dos recursos disponíveis para reconhecimento de voz por meio do sistema *Android*.

2. Avaliar a aplicação da tecnologia GPS e o desempenho da mesma para a aplicação.

3. Especificar e desenvolver o hardware necessário para se comunicar com um servidor e para buscar as coordenadas do GPS.

4. Desenvolver o hardware de alerta visual, que ficara acoplado ao painel do ônibus.

5. Avaliar as soluções de bancos de dados disponíveis para a armazenagem das posições do celular e do dispositivo acoplado.

6. Desenvolver a comunicação entre o dispositivo Android e o banco de dados.

7. Desenvolver a interface em sistema *Android* para o usuário final.

8. Desenvolver o banco de dados que será usado como base e a lógica nele aplicada.

1.2 ESTRUTURA DO TRABALHO

A estrutura do trabalho se inicia com a fundamentação teórica que tem como objetivo contextualizar a aplicação do projeto além de fornecer o embasamento necessário para que o leitor entenda o funcionamento do projeto. A fundamentação teórica abordara assuntos relacionados a inclusão social, tecnologia assistiva, mobilidade, *Android*, GPS, banco de dados, Wi-Fi e GSM.

Após a fundamentação teórica, a metodologia do projeto será apresentada de forma a explicar o planejamento do projeto, as etapas a serem realizadas e o que será feito em cada uma delas e como o desenvolvimento irá prosseguir em caso de alguma mudança na estrutura do projeto.

No desenvolvimento será mostrada toda a evolução do projeto incluindo desde a escolha do *hardware*, *software*, as adaptações e desvios realizados para que o projeto possa ter continuidade.

Na seção resultados esperados serão apresentados os resultados qualitativos do projeto, quanto ao funcionamento e quanto a facilidade da utilização da interface.

2 FUNDAMENTAÇÃO TEÓRICA

O foco deste trabalho é elaborar uma solução baseada em tecnologia assistiva que é uma área de conhecimento com característica interdisciplinar que tem como objetivo dar mais autonomia, independência e qualidade de vida para o usuário com alguma limitação (Gorverno do Brasil, 2017).

Como abordado na introdução deste trabalho com o passar do tempo cada vez mais a eletrônica vem se tornando mais necessária para a melhoria na qualidade de vida da população em geral e principalmente com relação a população com deficiência visual, com isso em mente este projeto necessitará do estudo da situação social dos deficientes visuais, eletrônica e programação trabalhando em sinergia para que seja possível o desenvolvimento deste projeto.

Para o desenvolvimento de um projeto que realmente atenda à necessidade da população, com deficiência visual que utiliza o transporte coletivo, de modo a tornar este público mais independente no dia a dia, sem precisar se preocupar com a criação de um hardware para que o mesmo não se torne uma ‘caixa’ pouco ou nada ergonômica que precise ser carregada pelo usuário.

Com isso em mente a equipe decidiu optar pelo desenvolvimento de um aplicativo para o sistema *Android*, desta forma o usuário não terá de se preocupar com a utilização de um dispositivo extra. Este aplicativo terá a função de capturar a voz do usuário e codifica-la em texto para que o mesmo realize uma busca em um banco de dados que deverá associar a posição do celular, obtida por meio do GPS, com o nome da linha de ônibus falada pelo usuário.

O banco de dados terá uma função intermediaria, o mesmo deverá ser carregado com a informação de posição do celular após a escolha do usuário pela linha de ônibus, que deverá ser feita pelo reconhecimento de voz, além disto será carregado pela posição de GPS provida pelo dispositivo acoplado ao ônibus, e pelo próprio nome da linha do ônibus.

O dispositivo que será acoplado ao painel do ônibus e o aplicativo desenvolvido deverão também emitir um aviso para o condutor e para o usuário respectivamente, este aviso deverá ser emitido quando o ônibus estiver próximo ao usuário. O alerta será emitido através da consulta da posição gravada no banco de dados, quando a mesma mostrar que as coordenadas do ônibus e do usuário tem pouca diferença conforme será explicado no decorrer do desenvolvimento.

Para um melhor entendimento deste projeto a fundamentação teórica será apresentada sequencialmente, relacionando o estudo da inclusão social e da mobilidade para o público com deficiência visual, de modo a ressaltar a necessidade do projeto, na sequência serão apresentados tópicos relacionados a conhecimentos técnicos necessários para o desenvolvimento do projeto. Estas etapas foram divididas de modo a proporcionar um bom entendimento do funcionamento do dispositivo, bem como de sua construção seguindo a ordem da evolução do projeto no decorrer da disciplina.

2.1 PANORAMA DA DEFICIENCIA VISUAL

Segundo o senso 2010 realizado pelo IBGE (Instituto brasileiro de geografia e estatística) 23,9% da população brasileira declara ter algum tipo de deficiência sendo que 3,5% declaram algum tipo de deficiência visual o que faz com que este tipo de deficiência seja a mais presente no Brasil (Fundação Dorina, 2018).

Estes 3,5% da população representam mais de 6,5 milhões de brasileiros com deficiência visual sendo que destes 6,5 milhões, 528.624 são cegos e 6.056.654 são pessoas com baixa visão, ou seja, que apresentam dificuldade de enxergar. Na TABELA 1 é possível observar a distribuição destas pessoas no território nacional (Fundação Dorina, 2018).

TABELA 1 – ESTATÍSTICA DA DEFICIÊNCIA VISUAL

Pessoas com deficiência visual por região	Total	% população local
Norte	574.823	3,6
Nordeste	2.192.455	4,1
Sudeste	2.508.587	3,1
Sul	866.086	3,2
Centro-Oeste	443.357	3,2

FONTE: FUNDAÇÃO DORINA (2018).

Expandindo este panorama, globalmente estimasse que aproximadamente 1,3 bilhões de pessoas convivem com alguma forma de deficiência visual (Word Health Organization, 2018).

Com relação a visão de longa distância aproximadamente 36 milhões de

pessoas apresentam cegueira e 217 milhões apresentam baixa visão (Word Health Organization, 2018).

Com relação a visão de curta distância aproximadamente 826 milhões de pessoas apresentam deficiência (Word Health Organization, 2018).

2.2 MOBILIDADE E ACESSIBILIDADE EM CURITIBA

A mobilidade urbana pode ser definida como a capacidade das pessoas e agentes econômicos de se deslocarem para realizar atividades cotidianas levando em conta o espaço urbano e a complexidade das atividades que vão exercer e atingindo este objetivo de forma rápida, confortável e segura.

Segundo o artigo 15 do PlanMob de Curitiba: “Política municipal de mobilidade urbana e transporte, têm o compromisso de facilitar o transito de pessoas e bens no Município” (Prefeitura Municipal de Curitiba).

Dentre as diretrizes do artigo 15 consta como a primeira diretriz a priorização do espaço viário para o transporte coletivo em relação ao transporte individual. Além desta diretriz contam também diretrizes que tratam da expansão e integração do transporte público coletivo e da promoção da acessibilidade de modo a facilitar o deslocamento no município através de uma rede integrada de vias, ciclovias e ruas exclusivas para pedestres que proporcionem segurança, autonomia e conforto aos usuários, especialmente aos que possuem dificuldade de locomoção (Prefeitura Municipal de Curitiba).

2.2.1 Histórico da Mobilidade e Acessibilidade em Curitiba

As legislações que tocam no assunto acessibilidade tiveram início com a lei 6.989/87 que tratava da concessão de alvarás para construções apenas para as edificações que fornecessem condições de acesso a deficientes físicos nas dependências de edificações públicas. Nos anos seguintes foram implementadas melhorias como legislação para transporte escolar de alunos com deficiência, o estabelecimento de vagas reservadas para deficientes físicos estabelecendo mínimo de vagas e identificação das mesmas para este fim (Prefeitura Municipal de Curitiba).

No início da década de 1990 foram criadas as linhas diretas que tinham como objetivo realizar os percursos com poucas paradas e com embarque em nível através das estações-tubo de modo a reduzir o tempo de deslocamento. A primeira linha deste sistema teve o início de sua operação em 1991, no ano seguinte se iniciou a

implantação dos biarticulados para otimizar este sistema (Prefeitura Municipal de Curitiba).

Em 1997 foram sancionadas as leis municipais 9.121/2-97 e 9.132/97 que tratam da segurança de transito aos pedestres nas calçadas de Curitiba e sobre o programa comunitário de construção e melhoria de passeios. Este programa gerou a publicação do decreto 561/98 que estabelece padrões de pavimentação para a construção e revitalização de calçadas instituindo a obrigatoriedade da construção de rampas de travessia para deficientes (Prefeitura Municipal de Curitiba).

No ano seguinte, 1998, foram iniciados estudos referentes a implantação de linha tátil para os deficientes visuais que são instaladas nas calçadas do município. Nos anos seguintes mais leis e decretos foram sancionado, todos visando principalmente segurança e conforto para os usuários (Prefeitura Municipal de Curitiba).

Em suma o transporte coletivo, segundo o PlanMob, tem como objetivo ter um sistema que ofereça um serviço com regularidade, boa oferta, com prioridade na utilização do sistema viário, velocidade operacional, conforto, infraestrutura, segurança e racionalidade, tudo isso a um preço justo (Prefeitura Municipal de Curitiba).

2.3 MEIOS DE TRANSPORTE E DEFICIENCIA VISUAL

O objetivo desta seção é demostrar os impactos que pessoas com deficiência visual sentem socialmente e até mesmo profissionalmente no que tange ao uso de meios de transporte e capacidade de locomoção.

A deficiência visual causa grande dificuldade de mobilidade aos acometidos pela mesma, visto que quem possui baixa visão ou cegueira não tem a possibilidade de contar com um meio de transporte próprio e por consequência necessita do uso de transporte particular, seja este de familiares, taxis ou aplicativos, ou ainda de transporte público.

Segundo um estudo publicado em 2011 pelo IPEA (Instituto de pesquisa econômica aplicada) cerca de 65% da população que reside em capitais e em regiões metropolitanas utiliza o transporte público como principal meio de transporte (Gorverno do Brasil IPEA, 2017).

Devido à falta de uma massa de dados é difícil identificar quantas das pessoas utilizadoras de transporte público sofrem de alguma deficiência visual, entretanto é

notável que um percentual expressivo da população utiliza transporte público e também um percentual expressivo da população sofre de algum tipo de deficiência visual o que nos leva a crer que este é o meio de transporte utilizado por muitas destas pessoas (Gorverno do Brasil IPEA, 2017).

Com base no artigo (GOLLEDGE, MARSTON e COSTANZO, 2001) elaborado com base em uma pesquisa aplicada à 55 pessoas que apresentam baixa visão ou cegueira e residem na cidade de Santa Barbara nos estados unidos foi possível constatar que a necessidade de melhoria na acessibilidade para utilizadores de transporte público (GOLLEDGE, MARSTON e COSTANZO, 2001).

Este estudo foi realizado com uma distribuição relativamente homogênea entre faixas etárias com grupos de 20 até 39 anos (21,8%), de 40 até 59 anos (20%), de 60 até 79 anos (29,1%) e com 80 ou mais anos (20,1%) (GOLLEDGE, MARSTON e COSTANZO, 2001).

Os resultados deste estudo mostraram que 67% dos participantes se consideravam dependentes de ajuda para se transportar e 78% deles demonstraram bastante frustração relacionada a esta dependência (GOLLEDGE, MARSTON e COSTANZO, 2001).

A maioria dos participantes assumiu que tem familiaridade maior com algum meio de transporte porem apenas 58% deles concordaram que os meios que utilizam atendem plenamente suas necessidades (GOLLEDGE, MARSTON e COSTANZO, 2001).

Com relação ao transporte público 70% dos entrevistados assumiram que existe uma dificuldade em encontrar o local correto para embarcarem (ponto de ônibus). Além disto 50% dos participantes indicaram que é difícil reconhecer qual ônibus devem embarcar. A dificuldade em reconhecer o local de desembarque foi expressada por 54% dos entrevistados. Outra dificuldade notável é a dificuldade em encontrar pontos desconexos (fora de estações), 85% dos entrevistados relataram esta dificuldade (GOLLEDGE, MARSTON e COSTANZO, 2001).

O indicador que mais destaca a possibilidade da aplicação sugerida neste projeto é que 67% dos entrevistados concorda que a maneira mais efetiva para se obter auxilio é proveniente de mensagens de voz, sendo este o modo de auxilio considerado o mais eficaz pelos usuários (GOLLEDGE, MARSTON e COSTANZO, 2001).

2.4 BANCO DE DADOS

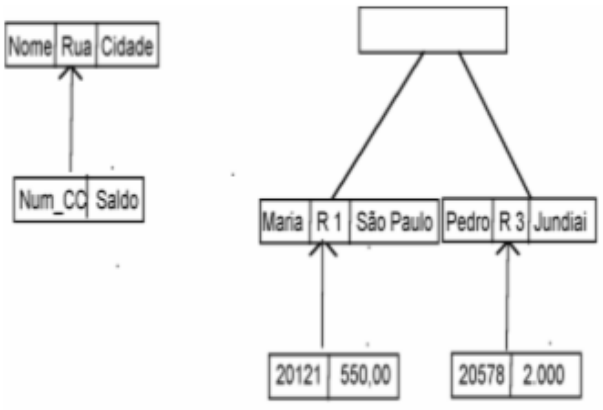
Neste projeto o banco de dados terá um papel fundamental visto que o mesmo será responsável por armazenar os dados enviados do GPS do celular e também dos dados enviados do microcontrolador. Existem diversos modelos de bancos de dados que serão abordados nesta seção de modo a justificar a escolha do modelo mais adequado para o projeto.

2.4.1 Banco de Dados Hierárquico

O modelo hierárquico foi o primeiro a ser reconhecido como um modelo de banco de dados, este modelo funciona com relações de 1:N, o acesso aos dados é feito sempre do nível superior para baixo (TAKAI, ITALIANO e FERREIRA, 2005).

A grande desvantagem deste modelo é que para associar registros diferentes é necessário a criação de réplicas o que faz com que o espaço ocupado não fique otimizado e também pode causar inconsistência nos dados quando acontecer alguma atualização. Na FIGURA 1 é possível observar o diagrama deste modelo (TAKAI, ITALIANO e FERREIRA, 2005).

FIGURA 1 – DADOS HIERÁRQUICO



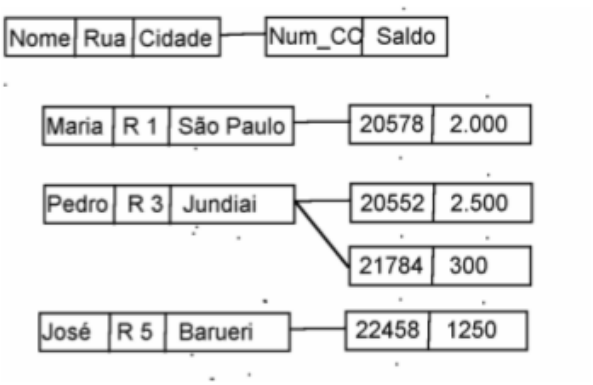
FONTE: TAKAI, ITALIANO E FERREIRA (2005).

2.4.2 Banco de Dados em Rede

O modelo em rede veio como uma evolução do modelo hierárquico, este modelo elimina o conceito de hierarquia, permitindo várias associações diferentes entre registros. No modelo em rede os registros são organizados em grafos onde um único tipo de associação define uma relação 1 para N entre dois tipos de registros, registro proprietário e membro (TAKAI, ITALIANO e FERREIRA, 2005).

No modelo em rede é possível acessar qualquer registro diretamente através de qualquer nó e não necessita que o acesso parta da raiz como acontece no modelo hierárquico, o que confere grande vantagem no que diz respeito a segurança do banco de dados. Na FIGURA 2 é possível observar o diagrama do modelo em rede (TAKAI, ITALIANO e FERREIRA, 2005).

FIGURA 2 – DADOS DE REDE

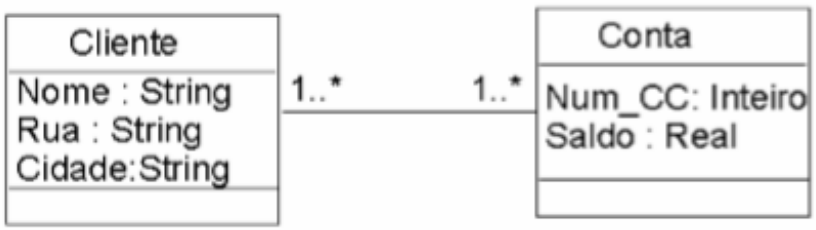


FONTE: TAKAI, ITALIANO E FERREIRA (2005).

2.4.3 Banco de Dados Orientado a Objetos

Os bancos de dados orientados a objetos são os mais recentes porem não tão difundidos devido à dificuldade na manipulação dos dados, comercialmente este tipo de banco de dados é utilizado em aplicações mais especificas, em que os dados são mais complexos um exemplo é a utilização para armazenar registros de softwares CAD e CAM. Na FIGURA 3 é possível observar um diagrama deste modelo (TAKAI, ITALIANO e FERREIRA, 2005).

FIGURA 3 – DADOS ORIENTADOS A OBJETOS



FONTE: TAKAI, ITALIANO E FERREIRA (2005).

2.4.4 Banco de Dados Relacional

O banco de dados relacional foi inicialmente proposto em 1969 por um pesquisador da IBM chamado Edgar Codd, desde então este modelo se tornou o mais

comum entre as aplicações comerciais. Hoje existem vários RDBMS (sistemas de gerenciamento de banco de dados relacionais), tanto pagos como Oracle, IBM DB2 e também *open-source* como o MySQL e mySQL (HOCK-CHUAN, 2010).

Um banco de dados relacional funciona organizando os dados em tabelas, como em uma planilha, de qualquer forma é possível criar relações entre tabelas o que faz com que este tipo de banco de dados seja eficiente para armazenar grande quantidade de dados e também de buscar os dados necessários para uma consulta. Foi criada a linguagem SQL (Structured Query Language) para trabalhar com este tipo de banco de dados (HOCK-CHUAN, 2010).

O modelo relacional se baseia na tabela em que as colunas são os campos e as instancias do sistema ou linhas são os registros em si, este modelo não tem caminhos definidos para acessar os dados como nos modelos de rede e hierárquico, este modelo organiza os dados em relações. Estas relações podem limitar o modelo visto que necessitam da exclusividade dos registros, ou serão duplicados todos os campos (TAKAI, ITALIANO e FERREIRA, 2005).

Na FIGURA 4 é possível observa um exemplo deste modelo.

FIGURA 4 – DADOS RELACIONAL

Cod_Cliente	Nome	Rua	Cidade
1	Pedro	A	São Paulo
2	Maria	B	Jundiai

Num_CC	Saldo
20121	1200
21582	1320
21352	652

Cod_Cliente	Num_CC
1	20121
2	21582
2	21352

FONTE: TAKAI, ITALIANO E FERREIRA (2005).

2.5 GPS

A tecnologia de GPS (*Global Positioning System*) será essencial para o desenvolvimento deste projeto, visto que será por meio dela que será definida a distância entre o veículo (ônibus) e o usuário.

O GPS assim como muitas outras tecnologias teve início com motivação militar, o sistema teve origem nos Estados Unidos e até hoje é sustentado pelo

governo norte americano, no início da década de 1980 a utilização do GPS pela população civil foi liberada após um incidente na união soviética ocorrido devido à falta de localização (LEICA GEOSYSTEMS INC., 1999).

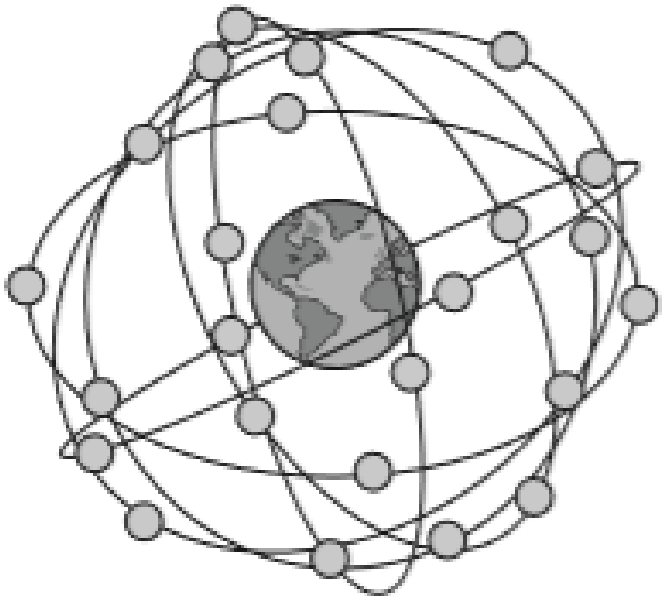
A utilização do GPS para aplicação civil trouxe muitas melhorias em tecnologias e aplicações, atualmente o GPS é utilizado não somente para navegação, como no caso de aplicativos de mapa mas também para aplicações muito mais especificas como medição de fatores sísmicos de terremotos, mineração, agricultura e demais aplicações que necessitam de um controle de posição acurado (LEICA GEOSYSTEMS INC., 1999).

O GPS é um sistema baseado em uma constelação de 24 satélites que dão a posição precisa para o usuário, a precisão é variável e dependente do tipo de receptor de GPS e da técnica utilizada. De modo a facilitar o entendimento serão abordadas três seções ou segmentos que juntos formam o sistema de GPS (LEICA GEOSYSTEMS INC., 1999).

2.5.1 Segmento Espacial – Satélites

O segmento espacial consiste em 24 satélites orbitando o planeta Terra e se movimentando aproximadamente 20200 km a cada 12 horas. Na FIGURA 5 é possível observar uma representação dos satélites orbitando a Terra (LEICA GEOSYSTEMS INC., 1999).

FIGURA 5 – SATÉLITES

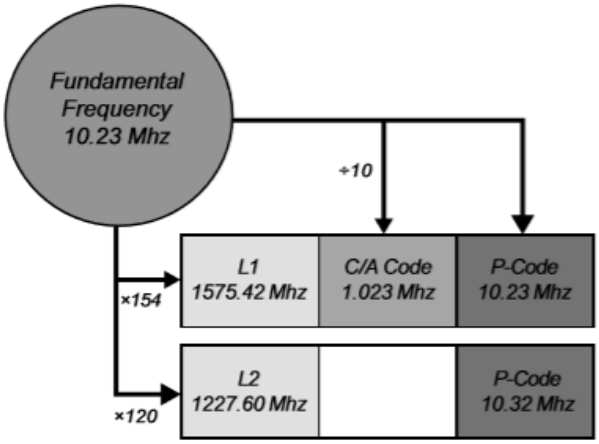


FONTE: LEICA GEOSYSTEMS INC. (1999).

Este segmento é desenhado de modo que sempre exista no mínimo 4 satélites visíveis em um ângulo de 15º em qualquer ponto do planeta, usualmente são visíveis de 5 até 7 satélites para esta mesma abertura (LEICA GEOSYSTEMS INC., 1999).

Os satélites fazem, constantemente, o broadcast de duas portadoras derivadas da frequência fundamental do *clock* do satélite que é extremamente preciso, os satélites utilizam diferentes codificações para que seja possível distinguir um satélite do outro e para serem utilizados como base para calcular a posição. Na FIGURA 6 é possível observar uma representação das portadoras geradas pelo satélite (LEICA GEOSYSTEMS INC., 1999).

FIGURA 6 – CLOCK DO SATÉLITE



FONTE: LEICA GEOSYSTEMS INC. (1999).

2.5.2 Segmento de Controle

O segmento de controle consiste em uma base de controle mestra, cinco estações de monitoramento e quatro antenas distribuídas entre as cinco estações que são situadas ao longo da linha do equador (LEICA GEOSYSTEMS INC., 1999).

A função do segmento de controle é rastrear os satélites, atualizar a posição deles na orbita, calibrar e sincronizar os *clocks* dos satélites, outra função importante deste segmento é a de prever o caminho dos satélites nas próximas 24 horas, esta informação é enviada para os satélites e permite que o receptor saiba onde aquele satélite deveria ser encontrado (LEICA GEOSYSTEMS INC., 1999).

Os sinais recebidos são lidos em Ascension, Diego Garcia e Kwajalein, as medidas são enviadas da estação de controle no Colorado e a informações são reenviadas pelas antenas e atualizadas nos satélites (LEICA GEOSYSTEMS INC., 1999).

FIGURA 7 – ESTAÇÕES DE CONTROLE DE SATÉLITE

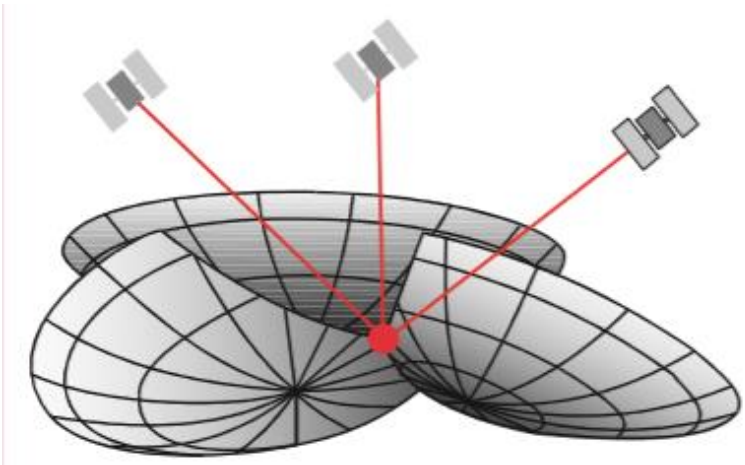
FONTE: LEICA GEOSYSTEMS INC. (1999).

2.5.3 Segmento do Usuário

O segmento do usuário compreende qualquer receptor de GPS que recebe sinal e determina a posição ou tempo através deste sistema como os exemplos supracitados (LEICA GEOSYSTEMS INC., 1999).

As posições são baseadas na medida da distância dos satélites em relação ao receptor, o receptor é o responsável por determinar esta distância, a ideia é basicamente utilizar a distância como o raio de uma esfera centrada no satélite (cuja posição foi transmitida pelo próprio satélite), deste modo são necessários três satélites para determinar a posição através da intersecção das três esferas como mostrado na FIGURA 8 (LEICA GEOSYSTEMS INC., 1999).

FIGURA 8 – POSIÇÃO DO GPS



FONTE: LEICA GEOSYSTEMS INC. (1999).

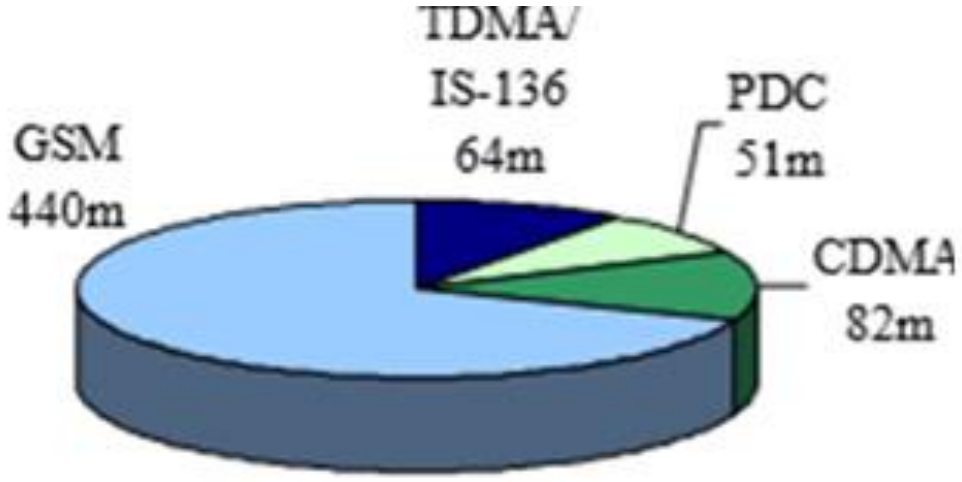
Além destes três satélites é necessário a presença de um quarto satélite para fornecer a quarta equação, visto que cada um dos satélites fornece uma distância e tempo, com quatro satélites e por consequência quatro equações é possível resolver este sistema e estimar a posição (LEICA GEOSYSTEMS INC., 1999).

2.6 GSM

Os sistemas de rádio baseados em celular tiveram início na década de 1970 nos laboratórios Bell nos Estados Unidos, uma década depois disto os sistemas foram introduzidos comercialmente e tiveram um acentuado crescimento, principalmente na Europa (SRIVASTAVA, PETRAZZINI e SELIAN, 2000).

Inicialmente as redes de celular tinham foco industrial e eram caracterizadas por pouca compatibilidade e tinham características de soluções locais de comunicação o que ressaltou e impulsionou o avanço deste tipo de tecnologias. Na FIGURA 9 podemos observar que este crescimento foi liderado pela tecnologia GSM já no início da década de 2000, o que mostra que o GSM está realmente se tornando um sistema global (SRIVASTAVA, PETRAZZINI e SELIAN, 2000).

FIGURA 9 – GSM



FONTE: SRIVASTAVA, PETRAZZINI E SELIAN (2000).

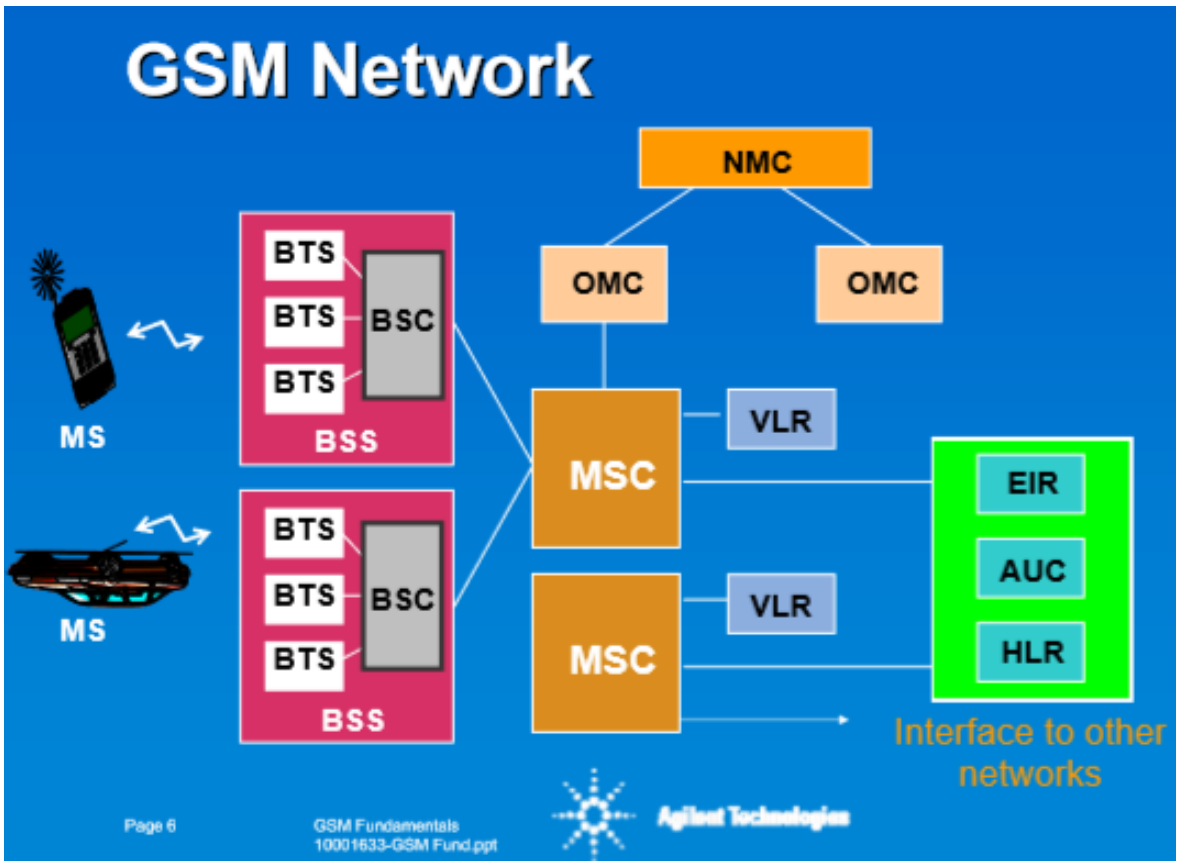
Os sistemas GSM são compostos por estações moveis, que podem ser celulares ou quaisquer outros dispositivos que suportem a tecnologia. As estações moveis se comunicam com a estação base através de uma interface de rádio frequência (RF) (SRIVASTAVA, PETRAZZINI e SELIAN, 2000).

A estação base é composta por um transceptor e um controlador da estação.

Os controladores fazem interface com os transceptores, tipicamente um controlador controla de 20 a 30 transceptores (AGILENT TECHNOLOGIES, 2000).

Um conjunto de estações base reporta para um MSC (*mobile switching center*) que é responsável por controlar o tráfico entre as diferentes células. As MSCs são conectadas a outras unidades responsáveis por registrar os usuários e realizar as validações de segurança, estes itens não serão abordados com profundidade neste tópico. Na FIGIRA 10 é possível observar um esquema das ligações entre os dispositivos (AGILENT TECHNOLOGIES, 2000).

FIGURA 10 – FUNCIONAMENTO DO GSM



FONTE: AGILENT TECHNOLOGIES (2000).

Um dos pontos mais importantes da tecnologia GSM é de usar por padrão o TDMA (*Time Division Multiple Access*) o que flexibiliza a o acesso para uma grande massa de fornecedores e aumenta o potencial dos aparelhos que utilizam este padrão de conseguirem um posicionamento no mercado com mais facilidade (AGILENT TECHNOLOGIES, 2000).

2.7 WI-FI

Wi-Fi é um conjunto de especificações para redes locais baseadas no padrão IEEE 802.11 que fornece as normas para a criação e uso de redes sem fio. A transmissão é feita por meio de sinais de radiofrequência que se propagam pelo ar, as frequências utilizadas dependem dos limites legais dos países nos quais os dispositivos estão inseridos (INFO WESTER, 2013).

Neste tópico serão introduzidos conceitos essenciais para o uso da tecnologia Wi-Fi neste projeto bem como a especificação das principais derivações do protocolo 802.11.

2.7.1 Access point, Station e SSID (*Service Set Indentifier*)

Uma rede Wi-Fi é definida pela comunicação de múltiplos dispositivos através de um fornecedor de acesso. Os dispositivos que acessam são denominados "*Station*" e os fornecedores de acesso são denominados Access Point (AP).

Na FIGURA 11 é possível identificar uma rede formada por diversos dispositivos em torno de um AP, neste caso representado pelo roteador (INFO WESTER, 2013).

FIGURA 11 – REDE COM STATIONS E AP

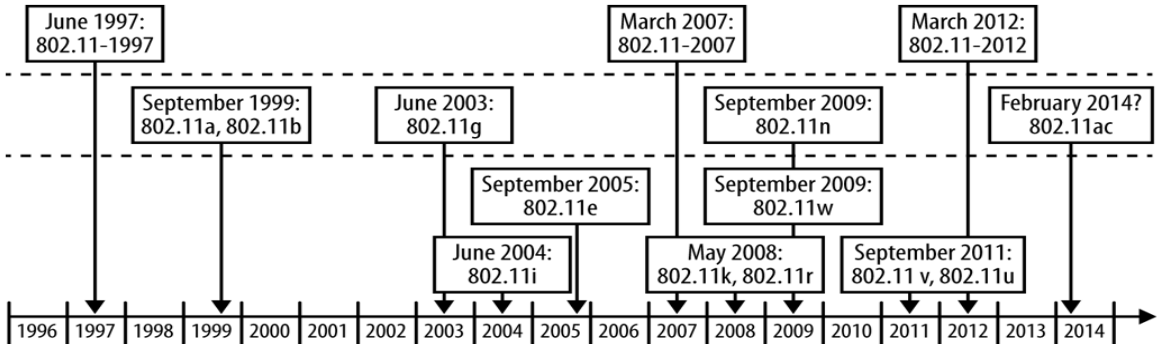
FONTE: INFO WESTER (2013).

SSID é uma tratativa de segurança que visa identificar diferentes redes Wi-Fi que ocupam um mesmo espaço, por exemplo, duas redes sem fio disponíveis no prédio PK na UFPR campus centro politécnico (Delt e UFPR_sem_fio). Ambas as redes podem ser acessadas, porém não se convergem, ou seja, dados não passam de uma rede para a outra diretamente, isso ocorre por conta do SSID que insere um conjunto de caracteres no cabeçalho de cada pacote de dados enviados na rede. (INFO WESTER, 2013).

2.7.2 PROTOCOLOS E TÉCNICAS DE MODULAÇÃO

Desde a sua criação o Wi-Fi vem evoluindo e criando novos protocolos com capacidade de transmissão cada vez maior e mais eficiente, está evolução proporcionou uma grande gama de protocolos como podemos observar no Time Line da FIGURA 12.

FIGURA 12 – EVOLUÇÃO DO WI-FI



FONTE: SILICON INVESTOR (2016).

Neste tópico serão explicados os principais protocolos e a funcionalidade de cada um deles.

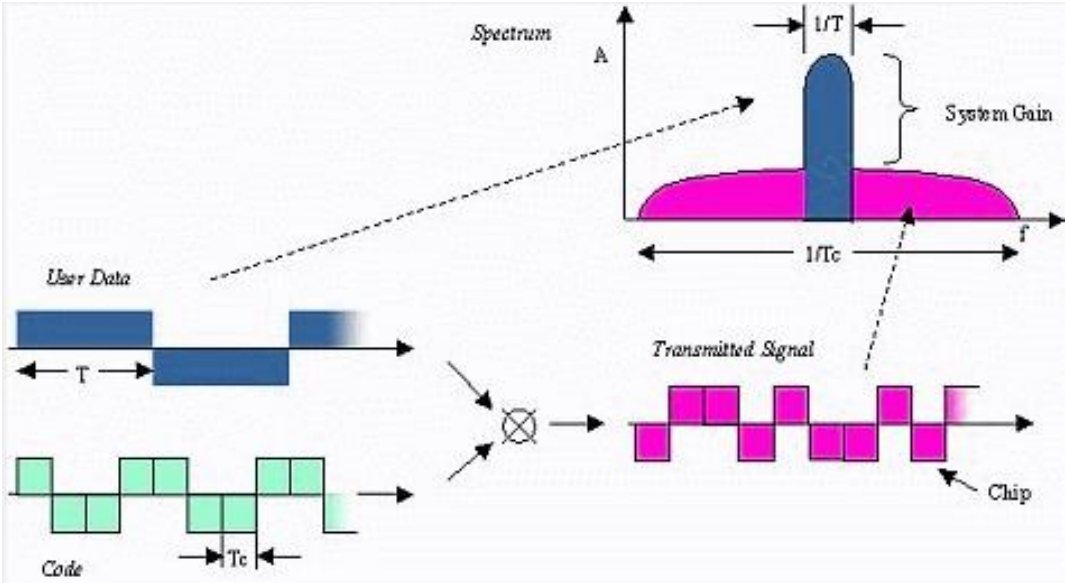
A primeira versão (de 1997), também conhecida como 802.11 *Legacy* conta com taxas de transmissão de 1 e 2 Mbps e faz o uso das técnicas de transmissão *Direct Sequence Spread Spectrum* (DSSS) e *Frequency Hopping Spread Spectrum* (FHSS). Esta versão também conta com três canais não sobrepostos na faixa de frequência ISM (destinada a fins industriais, científicos e médicos) 2,4GHz (INTEL CORPORATION, 2017).

A técnica de transmissão DSSS permite a utilização de vários canais dentro de uma mesma banda de frequência. É uma técnica de espalhamento espectral que multiplica os dados por um código PN (*peseudo-Noise*), esse código tem uma taxa muito superior à taxa dos dados (esta é a taxa de bits do código) (TELECON ABC,

2017).

Na FIGURA 13 é possível observar o efeito do espalhamento do sinal na frequência, A cor azul representa o sinal original e a cor rosa o sinal pós DSSS.

FIGURA 13 – DSSS



FONTE: TELECOM ABC (2017).

O DSSS melhora a proteção contrassinais de interferência, em especial os de banda estreita e faz com que o sinal se torne menos perceptível. DSSS também fornece uma boa segurança para os dados transmitidos uma vez que o código PN não seja conhecido pelo público (TELECON ABC, 2017).

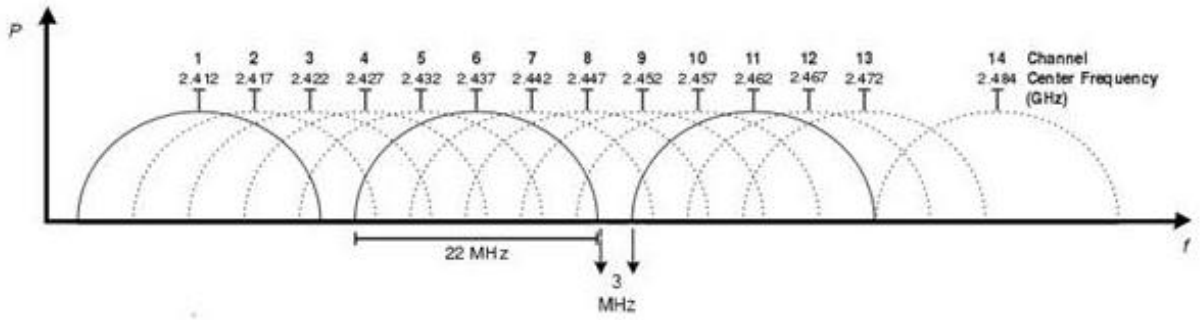
A técnica FHSS foi explicada anteriormente para a tecnologia *Bluetooth*, para o Wi-Fi segue o mesmo princípio de funcionamento, saltos de frequência durante a transmissão. A desvantagem em relação a DSSS é que esta técnica deixa um padrão de saltos de frequência que é possível de ser observado no espectro (TELECON ABC, 2017).

Após a versão 802.11 em 1999 foi lançada a versão 802.11 b que conta com diversos tipos de modulação e taxas de 1, 2, 5 e 11 Mbps, conta também com três canais não sobrepostos na faixa de frequência ISM a 2,4 GHz. Além disso conta com transmissão HR-DSSS (INTEL CORPORATION, 2017).

O HR-DSSS é uma evolução da técnica DSSS, esse sistema divide a banda em 14 canais com 22MHz em 2,4GHz. Cada canal começa 5 MHz depois do outro, ou seja os canais se sobrepõem como é possível identificar na FIGURA 14 (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

Como cada canal tem uma largura de 22MHz é possível observar que cada canal se sobrepõe com diversos outros, essa sobreposição pode gerar interferência, para minimizar este problema é necessário tentar coordenar o uso do canal com os gerentes da rede mais próxima (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

FIGURA 14 – HR-DSSS



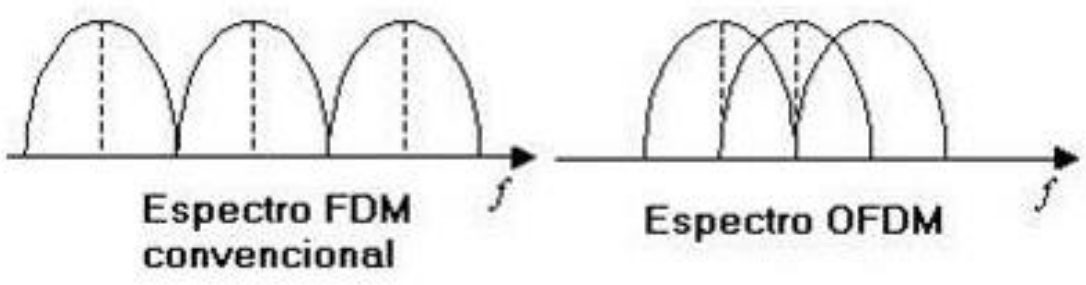
FONTE: TELECO INTELIGÊNCIA EM TELECOMUNICAÇÕES (2016).

Também em 1999 foi lançada a versão 802.11 a que tem taxas de 6, 9, 12, 18, 24, 36, 48 e 54 Mbps. Essa versão conta com 12 canais não sobrepostos na faixa não licenciada de 5 GHz. Também conta com OFDM com canais de subportadoras 52 (INTEL CORPORATION, 2017).

A técnica OFDM consiste na divisão do canal em várias subportadoras que são transmitidas paralelamente e independentemente, cada uma levando uma parcela da informação transmitida (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

O OFDM é uma evolução da técnica de multiplexação por divisão de frequência na qual as subportadoras ficavam separadas, no modelo OFDM elas se sobrepõem como mostrado na FIGURA 15.

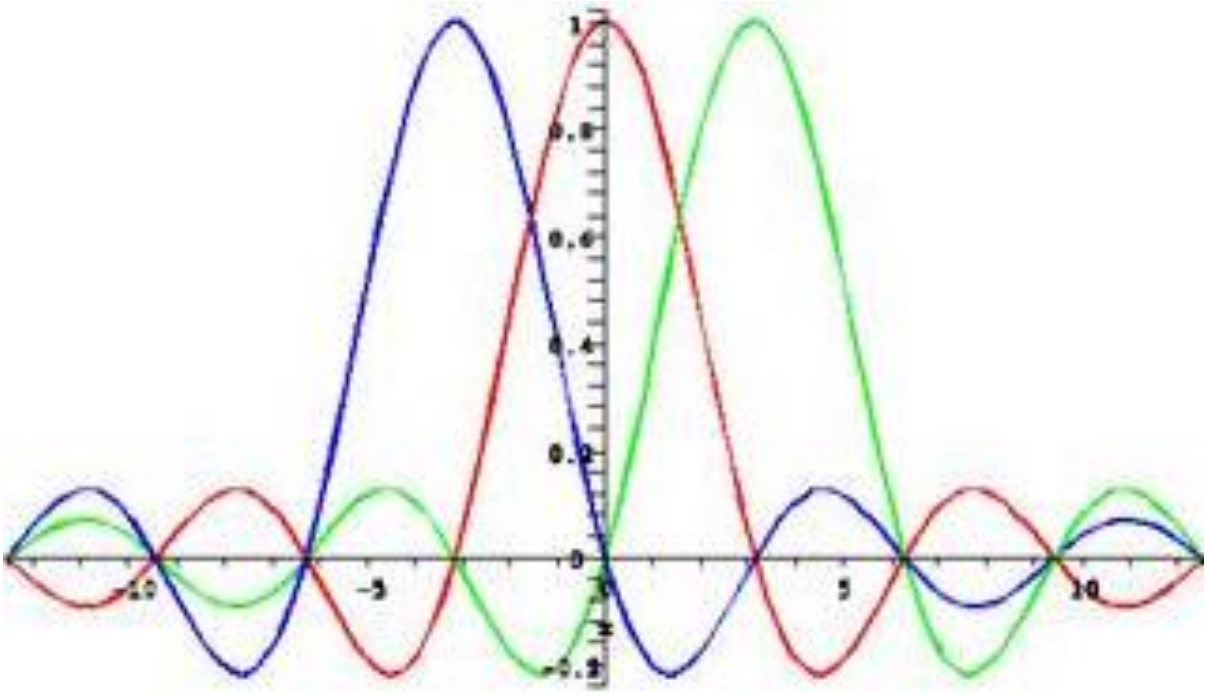
FIGURA 15 – MODULAÇÃO FDM E OFDM



FONTE: TELECO INTELIGÊNCIA EM TELECOMUNICAÇÕES (2016).

Essa sobreposição faz com que uma mesma quantidade de portadoras ocupe uma banda menor, ou seja, torna possível que mais informações possam ser transmitidas em uma mesma largura de banda. Essa sobreposição só é possível devido as portadoras serem ortogonais entre si. Na FIGURA 16 é possível observar um exemplo da distribuição de 3 subportadoras usando OFDM (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

FIGURA 16 – MODULAÇÃO OFDM



FONTE: TELECO INTELIGÊNCIA EM TELECOMUNICAÇÕES (2016).

A versão seguinte foi a 802.11 g que tem taxas de 6, 9, 12, 18, 48 e 54 Mps mas também pode reverter para 1, 2, 5 e 11 Mbps usando DSSS e CCK. Está versão também faz uso de OFDM com canais de subportadora 52 e é compatível com versões anteriores com 802.11 b através do uso de DSSS. A versão 802.11 g possui 3 canais não sobrepostos na faixa de frequência ISM a 2,4 GHz (INTEL CORPORATION, 2017).

A versão 802.11 n possui 3 canais não sobrepostos na faixa de frequência ISM a 2,4 GHz e 12 canais não sobrepostos na faixa não licenciada de 5 GHz. Esta versão conta com diversas taxas e que podem ser visualizadas na TABELA 2 (INTEL CORPORATION, 2017) .

Como é possível observar na TABELA 2 está versão suporta o uso de várias antenas de transmissão e recepção isso se dá pela técnica de transmissão utilizada

que é a MIMO-OFDM.

TABELA 2 – TAXAS DE TRANSMISSÃO 802.11 N

Modo	Taxa máxima	Transmissão de antena / Receber arranjos
1 x 1 20 MHz	72.2 Mbps	1 TX 1 RX
1 x 1 40 Mhz	150 Mbps	1 TX 1 RX
2 x 2 20 MHz	144.4 Mbps	2 TX RX 2
2 x 2 40 MHz	300 Mbps	2 TX RX 2
3 x 3 20 MHz	216.7 Mbps	3 TX RX 3
3 x 3 40 MHz	450 Mbps	3 TX RX 3

FONTE: INTEL CORPORATION (2017).

Como sugerido pela sigla MIMO (múltiplos inputs e múltiplos outputs) está tecnologia trabalha com diversas antenas que geram vários fluxos de dados, ou caminhos entre antenas e receptores, o que faz com que muitas informações possam ser transferidas simultaneamente. Na FIGURA 17 é possível observar um exemplo destes fluxos (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

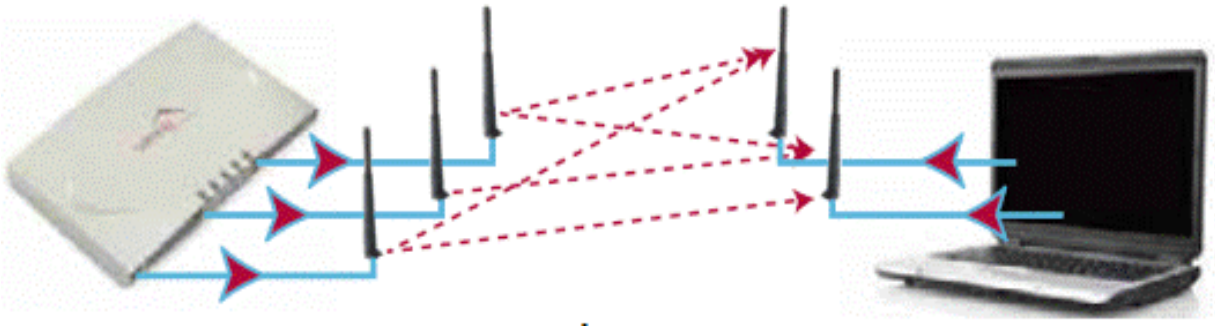
Com a existência de vários fluxos em um mesmo ambiente, todos transmitindo na mesma faixa de frequência é possível perceber que existe uma interferência entre os sinais assim como se tem várias redes operando em um mesmo espaço físico.

A maneira com que o MIMO soluciona este problema está baseada em algoritmos de alta complexidade que tiram proveito da reflexão dos sinais, já que as antenas estão espacialmente separadas e com isso os fluxos variam de uma para a outra. Desta maneira é possível identificar as informações e ordena-las na recepção do sinal (TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016).

Em 2014 foi lançada a versão 802.11 AC *wave* 1 e agrega os recursos da

802.11 n e os expande possuindo 24 canais não sobrepostos na faixa não licenciada de 5 GHz além de possuir taxas de 200, 400, 433, 600 e 867 Mbps conforme mostrado na TABELA 3 (INTEL CORPORATION, 2017).

FIGURA 17 – APLICAÇÃO MIMO



FONTE: TELECO INTELIGÊNCIA EM TELECOMUNICAÇÕES (2016)

TABELA 3 – TAXAS DE TRANSMMISSÃO 802.11 AC WAVE 1

Modo	Taxa máxima	Transmissão de antena / Receber arranjos
1 x 1 40 MHz	200 Mbps	1 TX 1 RX
2 x 2 40 MHz	400 Mbps	2 TX RX 2
1 x 1 80 MHz	433 Mbps	1 TX 1 RX
2 x 2 80 MHz	866 Mbps	2 TX RX 2

FONTE: INTEL CORPORATION (2017).

Em 2016 foi lançada a versão 802.11 AC *wave* 2 que trouxe novos recursos para os clientes de Wi-Fi, trouxe o MIMO para multiusuário e canais de 160 MHz, as taxas de transmissão são mostradas na TABELA 4 (INTEL CORPORATION, 2017).

Na TABELA 5 é possível observar uma comparação entre as versões apresentadas anteriormente no que se refere a frequência, largura de canal e taxa de transmissão que são alguns dos principais tópicos para comparar a eficiência de um método de transmissão (INTEL CORPORATION, 2017).

TABELA 4 – TAXA DE TRANSMISSÃO 802.11 AC WAVE 2

Modo	Taxa máxima	Transmissão de antena / Receber arranjos
1 x 1 40 MHz	200 Mbps	1 TX (transmissão, carregar) 1 RX (recepção, Download)
2 x 2 40 MHz	400 Mbps	2 TX RX 2
1 x 1 80 MHz	433 Mbps	1 TX 1 RX
2 x 2 80 MHz	866 Mbps	2 TX RX 2
1 x 1 160 MHz	866 Mbps	1 TX 1 RX
2 x 2 160 MHz	1, 73GHz Gbps	2 TX RX 2

FONTE: INTEL CORPORATION (2017).

TABELA 5 – COMPARATIVO DOS PROTOCOLOS

Protocolo	Frequência	Largura do canal	O RECURSO MIMO	Taxa máxima de dados (teórico)
802.11 AC wave2	5 GHz	80, 80 + 80, 160 MHz	(MU-MIMO) para vários usuários	1, 73GHz Gbps[1]
802.11 AC wave1	5 GHz	80 MHz	Usuário único (MIMO SU)	866.7 Mbps[1]
802.11 n	2,4 ou 5 GHz	20, 40MHz	Usuário único (MIMO SU)	450 Mbps[2]
802.11g	Faixa de 2,4 GHz	20 MHz	N/D	54 Mbps
802.11 a	5 GHz	20 MHz	N/D	54 Mbps
802.11 b	Faixa de 2,4 GHz	20 MHz	N/D	11 Mbps
Legacy 802.11	Faixa de 2,4 GHz	20 MHz	N/D	2 Mbps

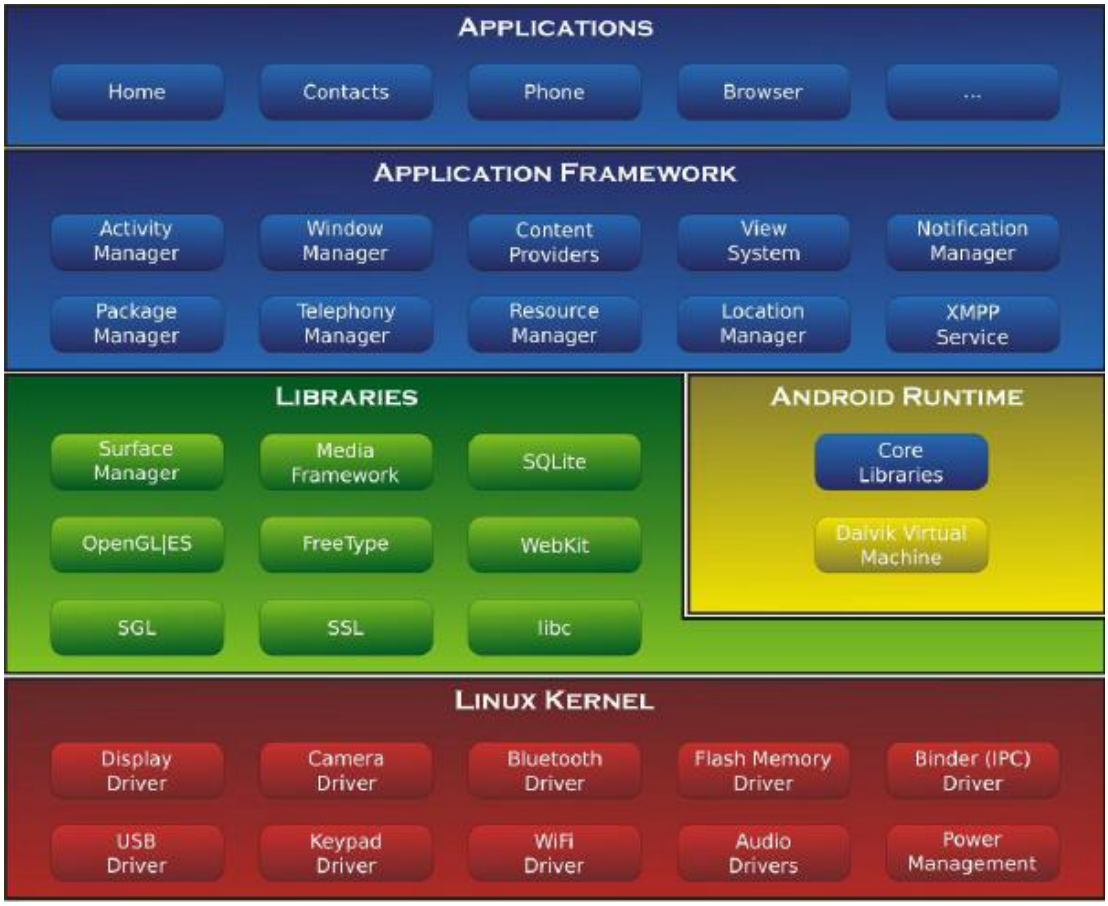
FONTE: INTEL CORPORATION (2017).

2.8 ANDROID

O sistema *Android* é atualmente o mais utilizado em plataformas móveis o que aumenta a capacidade de abrangência deste projeto, além disso a plataforma utiliza *kernel Linux* que tem as vantagens de permitir multitarefas, ou seja executar várias tarefas simultaneamente através da divisão do uso da memória. (GOMES, C., *et al.*, 2012).

O sistema operacional *Android* tem uma arquitetura dividida em várias camadas em que cada uma é formada por um grupo de vários componentes de programa, estas camadas compõem desde o sistema operacional até os aplicativos e cada uma delas pode fornecer serviços para as camadas superiores a mesma. Estas camadas estão representadas na FIGURA 18 (GOMES, C., *et al.*, 2012).

FIGURA 18 – CAMADAS DO SISTEMA OPERCIONAL ANDROID



FONTE: GOMES, C., ET AL. (2012 P. 8).

Os aplicativos para sistemas *Android* podem ser programados em Java que é uma linguagem de programação orientada a objetos. Existem vários métodos e IDEs, para o desenvolvimento em plataforma *Android* (ANDROID, 2014).

Após a compilação um arquivo *apk* é gerado, estes *apk* contêm o conteúdo do aplicativo e são os arquivos que os dispositivos *android* instalam (ANDROID, 2014).

Um aplicativo possui vários componentes, estes componentes funcionam como blocos de construção do aplicativo e cada um deles representa um ponto pelo qual o sistema pode acessar o aplicativo, funcionam como entidades independentes e desempenham funções especificas. Existem quatro tipos de componentes com finalidades diferentes e ciclos de vida também específicos, são eles componentes de atividades, serviços, provedores de conteúdo e receptores de transmissão (ANDROID, 2014).

Atividade consiste na interface que o usuário visualiza, uma atividade representa uma única tela, mas esta pode chamar outras atividades que mostraram outras telas para o usuário de modo que ele tenha uma boa experiência, por exemplo um aplicativo de e-mail pode abrir a câmera do celular, ou seja, uma atividade iniciando outra e assim por diante (ANDROID, 2014).

Serviços são componentes que ficam em segundo plano para realizar a execução por exemplo de tocar música sem que apareça uma interface para o usuário, desse modo permite que ele utilize outros aplicativos. Uma atividade pode iniciar um serviço e se manter vinculada para que seja realizada a interação com o mesmo (ANDROID, 2014).

Provedores de conteúdo são responsáveis por gerenciar dados armazenados pelo aplicativo. Os provedores permitem que outros aplicativos acessem e até modifiquem os dados de outros aplicativos desde que possuam as permissões necessárias (ANDROID, 2014).

Os receptores de transmissão são responsáveis por responder a anúncios de transmissão, as transmissões podem ter origem de aplicativos, por exemplo para informar que um arquivo foi baixado ou podem ter origem no próprio sistema mostrando por exemplo que a bateria está fraca (ANDROID, 2014).

Os receptores não mostram uma página na interface do usuário mas podem mostrar uma notificação na barra de status para que chame a atenção do usuário para algum aplicativo. Os receptores não costumam realizar muito trabalho são análogos a links que ao serem utilizados redirecionam para determinada atividade (ANDROID, 2014).

Um ponto interessante relacionado a aplicativos Android é a capacidade de

um aplicativo acessar componentes de outro, como por exemplo um aplicativo de um jogo pode acessar a câmera do celular e quaisquer outros. Isto é feito por meio de ativação de componentes e será explicado mais adiante caso seja necessário a utilização no decorrer deste projeto (ANDROID, 2014).

A estrutura do *Android* disponibiliza *APIs* com diversas funções em diferentes áreas como multimídia, interface gráfica, gerenciamento de energia e armazenamento de dados. Estas *APIs* são bibliotecas desenvolvidas em Java e rodam no nível superior das bibliotecas do *Android* (*Aplication Framework*), estas bibliotecas utilizam as bibliotecas nativas dos sistemas e as otimizam para executar alguma função (BRAHLER, 2010).

Para este projeto as bibliotecas relacionadas a reconhecimento de voz e localização serão essenciais, as *APIs* relacionadas a estas funções serão abordadas na seção desenvolvimento (BRAHLER, 2010).

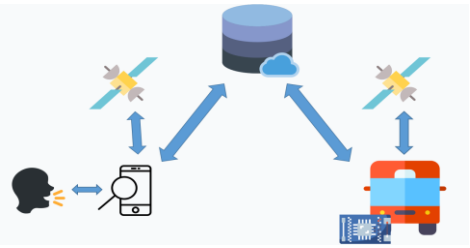
3 MÉTODO

Nesta seção serão abordadas as tecnologias empregadas e a interação entre elas de modo a clarificar a estrutura que o projeto irá seguir para alcançar a criação de um protótipo, assim como será apresentado as mudanças que ocorreram na metodologia devido a limitações encontradas durante o desenvolvimento do projeto.

O projeto conta com três principais vertentes, hardware implementado no ônibus que será responsável por enviar a informação e receber as mensagens que irão alertar o motorista, banco de dados que será responsável por armazenar os dados e a interface *Android* que será a guiada por voz, sendo assim é um ponto crítico do projeto.

Na FIGURA 19 é possível observar os fluxos de comunicação que ocorrem neste projeto. Existirá o fluxo de comunicação entre o usuário e o dispositivo *Android*, que consistira no reconhecimento de voz e resposta ao usuário, os fluxos entre satélite e microcontrolador e satélite e *Android* irão consistir na troca de coordenadas de GPS entre os dispositivos e informações utilizadas para os cálculos locais em cada dispositivo que enviam as informações pertinentes para o banco de dados para que fiquem disponíveis tanto para o microcontrolador como para o dispositivo *Android*.

FIGURA 19 – FLUXOS DE COMUNICAÇÃO



FONTE: OS AUTORES (2019).

Inicialmente a metodologia utilizava o banco de dados como o consolidador das informações e responsável pelos cálculos e tomadas de decisão, porém no decorrer do desenvolvimento foi optado por utilizar o banco de dados como um instrumento para comunicar o microcontrolador e o dispositivo Android de forma passiva.

Esta nova maneira de utilizar o banco de dados permite que ocorram relações de um para muitos, do microcontrolador para diversos dispositivos Android com problemas de concorrência minimizados, se comparado a abordagem inicial visto que o processamento será local nos dispositivos.

3.1 CRONOGRAMA DE EXECUÇÃO

O cronograma deste projeto se base a na divisão entre as três vertentes, banco de dados, Android e desenvolvimento do Hardware de forma que o cronograma atenda os entregáveis deste projeto de forma planejada, aplicando uma metodologia ágil de desenvolvimento em paralelo.

De forma a simplificar o entendimento do cronograma é apresentada a tabela a seguir com as interações entre as atividades apresentando as sucessoras e predecessoras de cada uma delas.

TABELA 6 – CRONOGRAMA

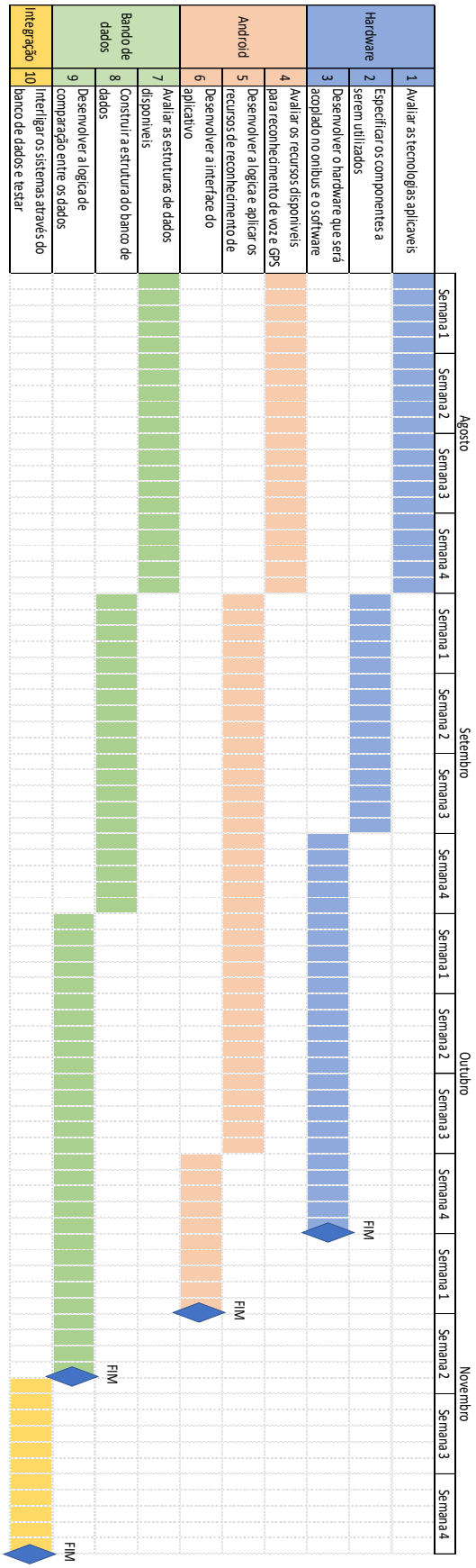
Vertente	Atividades		Predecessor	Sucessor
Hardware	1	Avaliar as tecnologias aplicaveis		2
	2	Especificar os componentes a serem utilizados	1	3
	3	Desenvolver o hardware que será acoplado no onibus e o software	2	10
Android	4	Avaliar os recursos disponiveis para reconhecimento de voz e GPS		5
	5	Desenvolver a logica e aplicar os recursos de reconhecimento de	4	6
	6	Desenvolver a interface do aplicativo	5	10
Bando de dados	7	Avaliar as estruturas de dados disponiveis		8
	8	Construir a estrutura do banco de dados	7	9
	9	Desenvolver a logica de comparação entre os dados	8	10
Integração	10	Interligar os sistemas através do banco de dados e testar	3; 6; 9	

FONTE: OS AUTORES (2019).

O cronograma inicial apresentado na sequência através do gráfico de gantt com a duração de cada uma das atividades no decorrer do semestre, as durações foram definidas com base na complexidade e tempo de desenvolvimento de cada uma das etapas estimado no início do projeto.

A execução não seguiu conforme o planejado devido a alteração de hardware e do conceito da solução do banco de dados, em geral a etapa mais custosa do projeto ao contrário do que a equipe estimava foi a integração das três vertentes que iniciou na semana quatro de outubro (adiantado) e se prolongou até o final de novembro.

TABELA 7 – GANTT



FONTE: OS AUTORES (2019).

4 DESENVOLVIMENTO

Nesta seção serão abordados todos os dispositivos, especificações utilizadas e ferramentas utilizadas no desenvolvimento do projeto. Como citado na metodologia o desenvolvimento será dividido em três principais pilares, banco de dados, *Android* e microcontrolador.

Os principais pontos foram a mudança na abordagem em relação a utilização do banco de dados e a alteração do módulo responsável pela aquisição do sinal de GPS, nesta sessão os motivos destas alterações serão abordados bem como as vantagens e desvantagens ao realizar estas mudanças.

4.1 DESENVOLVIMENTO DO BANCO DE DADOS

Inicialmente neste projeto o banco de dados era responsável por compilar as informações necessárias para o desenvolvimento do projeto e fazer comparações entre as informações. O método utilizado seria o banco de dados relacional SQL.

O sistema utilizado para o gerenciamento do banco de dados era o *MySQL* que é um banco de dados relacional de código aberto. A linguagem utilizada é o SQL.

O banco criado foi hospedado no 000webhost que é uma plataforma gratuita do *hostinger* para desenvolvimento de sites e bancos de dados, o 000webhost suporta desenvolvimento em PHP, e MySQL que inicialmente seriam peças chave para este projeto.

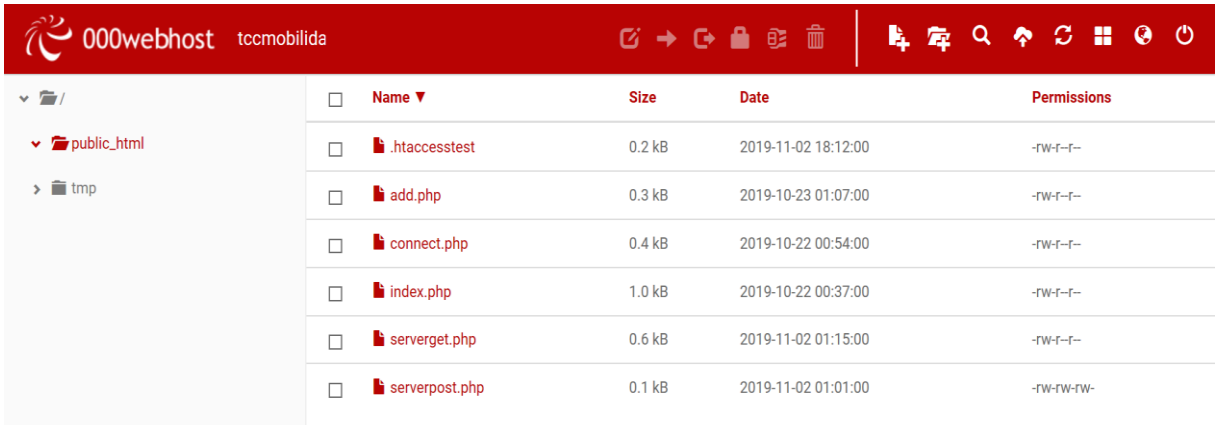
No início do desenvolvimento do servidor foram criados vários arquivos PHP, para receber os dados dos dispositivos, para atualizar uma página a qual os dispositivos iriam acessar para obter as informações e para atualizar os valores no banco de dados. Estes arquivos foram salvos dentro do gerenciador de arquivos do banco de dados e podem ser observados na FIGURA 20.

Inicialmente o arquivo responsável por receber as informações foram criados para receber comandos HTTP POST, porem durante o desenvolvimento do hardware foi observado que está implementação por meio do hardware (através de comandos AT) era muito complexa para a aplicação. Por conta disto foi decidido criar um arquivo para ler os arquivos através de comandos HTTP GET, o comando GET possibilita que os parâmetros sejam passados diretamente pelo navegador ou diretamente pelo comando AT+CISPSEND por parte do microcontrolador.

Na sequência é possível observar o código para o recebimento das requisições GET pelo servidor.

```
<?php
$servername = "localhost";
$username = "id11133742_android";
$password = "84351808";
$dbname = "id11133742_bancodedadosandroid";
 $porta = $_GET['porta'];
$posicao = $_GET['posicao'];
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO gpsonibus (posicao, porta)
VALUES ('$posicao', '$porta')";
if ($conn->query($sql) === TRUE) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "<br>" . $conn->error;
}
```

FIGURA 20 – SITE WEBHOST



FONTE: OS AUTORES (2019).

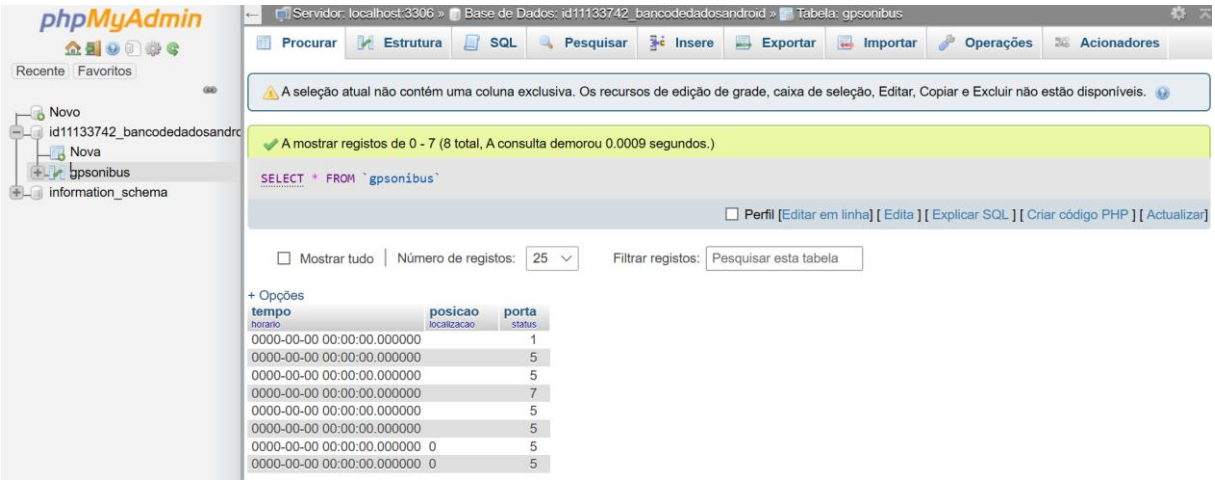
Este código nos permite adicionar informações ao servidor através de comandos GET como por exemplo no próprio navegador através do endereço que será utilizado como exemplo a seguir.

Para inserir valores em posição e porta através do método GET atribuindo valores as variáveis por meio do comando /serverget.php?variavel=valor, na sequência podemos ver um exemplo desta utilização e a resposta no banco de dados.

http://tccmobilidadeparacegos.000webhostapp.com/serverget.php?porta=5&posicao=0

Na FIGURA 21 é possível observar os dados adicionados ao banco de dados na última linha.

FIGURA 21 – SITE PHPMYADMIN



FONTE: OS AUTORES (2019).

No decorrer do desenvolvimento do hardware foi possível notar que o módulo SIM808 tem algumas limitações quanto aos protocolos utilizados nas redes de telefonia brasileira, todos os chips encontrados pela equipe necessitam da utilização do protocolo UTMS o qual o módulo SIM808 não suporta.

Dada as limitações de conexão não foi possível enviar os dados para o servidor SQL através do PHP, com isso o projeto foi alterado de modo a cumprir os objetivos específicos. O hardware foi alterado e a solução de banco de dados também.

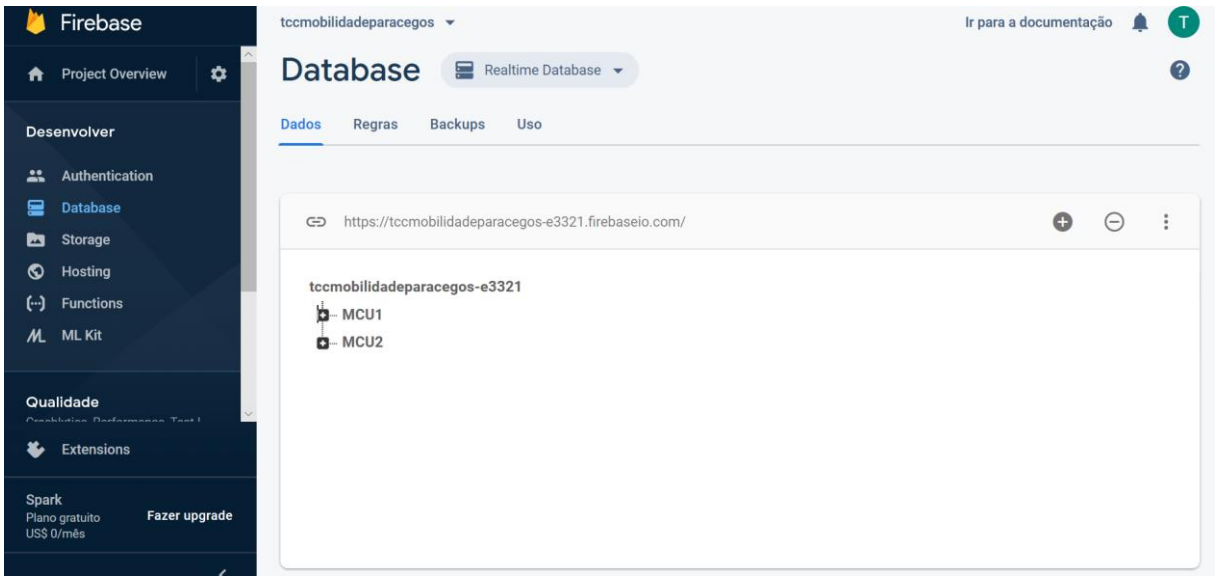
Foi decidido abordar uma solução mais moderna e flexível, um banco de dados *noSQL*, o *Firebase*, que é um banco de dados orientado a objetos. Conforme explanado na fundamentação estes bancos de dados são mais complexos em relação a realização de consultas e cálculos entre classes diferentes, porem com este tipo de banco de dados os dados podem ser trabalhados mais facilmente sem duplicações e redundâncias.

No decorrer do desenvolvimento foi notado que para a utilização em larga escala da solução a maneira mais intuitiva para o tratamento dos dados é fazendo-o

localmente e utilizando o banco de dados como um HUB com as informações necessárias para os dispositivos envolvidos porem sem realizar cálculos e comparações com estas informações.

O servidor é responsável por abrigar em sua raiz uma classe para cada microcontrolador, que fica vinculado a classe via código no próprio microcontrolador, os parâmetros necessários para os cálculos são construídos dentro das classes como subclasses. Na FIGURA 22 e FIGURA 23, podemos observar a construção do banco de dados no *Firebase*.

FIGURA 22 – FIREBASE



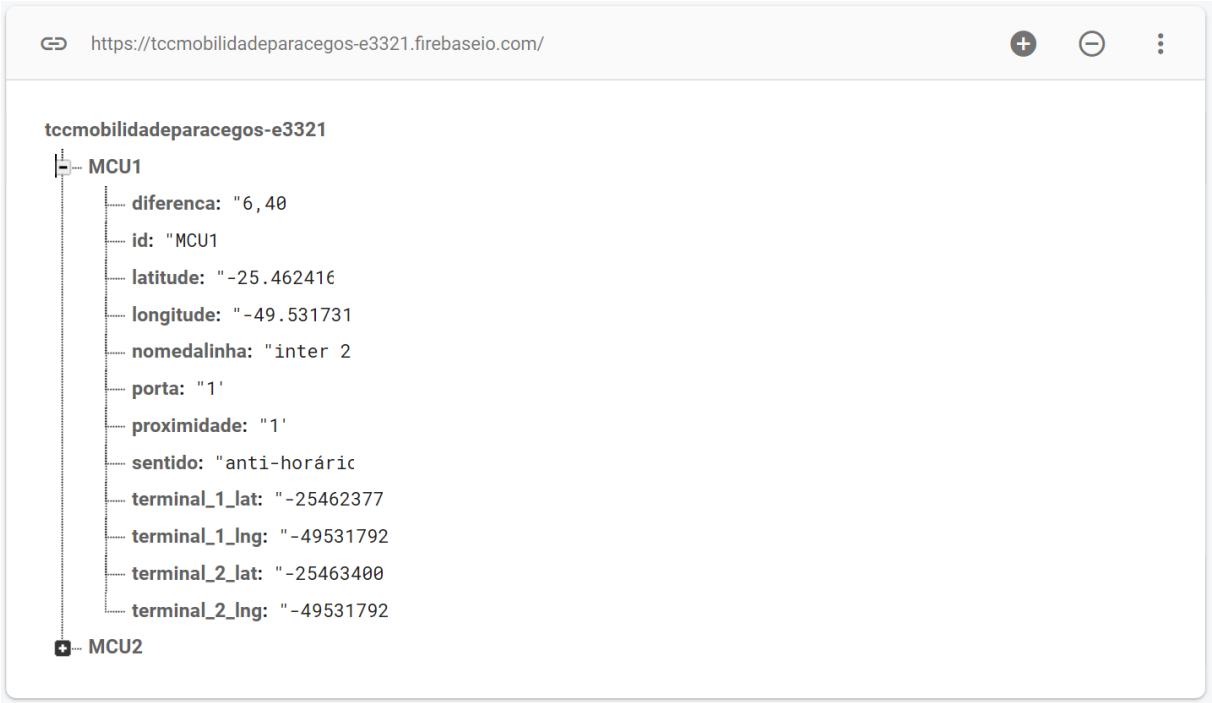
FONTE: OS AUTORES (2019).

O Firebase abriga dados fornecidos pelo dispositivo *Android*, pelo microcontrolador e dados que são alterados manualmente pelo administrador do banco de dados.

Para que estas edições sejam realizadas por inúmeros dispositivos as preferências de segurança do banco foram abertas conforme o código a seguir, permitindo que qualquer dispositivo se comunique com o banco.

```
{
/* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
"rules": {
".read": true,
".write": true }}
```

FIGURA 23 – FIREBASE DATABASE



FONTE: OS AUTORES (2019).

Os dados fornecidos pelo microcontrolador são os dados de latitude, longitude e porta que é o estado do sensor que indica a abertura ou não da porta, esta informação em conjunto com informações de proximidade do dispositivo *Android* indicam a chegada ou não do ônibus.

Os dados fornecidos pelo dispositivo *android* são a diferença que é a distância em metros entre o dispositivo *Android* e o microcontrolador e a *flag* de proximidade que é alterada pelo dispositivo quando está dentro de um raio pré-definido.

Para a conexão do aplicativo com o *Firebase* é criada uma SDK através do registro do aplicativo no próprio *Firebase*, após este registro o *package* contendo os fragmentos de código a seguir é importado para o aplicativo para realizar a conexão via JSON.

Este código é responsável por estabelecer a conexão e fornece os padrões como a chave de conexão ao servidor e os endereços para a conexões com o projeto dentro do *Firebase*.

{

"project_info": {

"project_number": "381819013000",

"firebase_url": "https://tccmobilidadeparacegos-e3321.firebaseio.com",

"project_id": "tccmobilidadeparacegos-e3321",
"storage_bucket": "tccmobilidadeparacegos-e3321.appspot.com"
},
"api_key": [
{"current_key": "AIzaSyCuiAx8WaoenR0fnPW4facbYL4Iv4poRbM"
}],
"services": {
"appinvite_service": {
"other_platform_oauth_client": [
{"client_id":"381819013000-
2du9621lnht1v9vlmr2lp15uuf6kd20m.apps.googleusercontent.com",
"client_type": 3
}] } } },
"services": {
"appinvite_service": {
"other_platform_oauth_client": [
{
"client_id":"381819013000-
2du9621lnht1v9vlmr2lp15uuf6kd20m.apps.googleusercontent.com",
"client_type": 3
}]
}}}],
"configuration_version": "1"}

Os dados pelos quais o administrador do banco de dados tem responsabilidade tornam a aplicação mais flexível e permitem com que cada dispositivo colocado em cada ônibus tenha seus parâmetros básicos alterados. Os parâmetros que são controlados pelo administrador do sistema são os parâmetros de coordenada dos terminais, terminal_1_lat, terminal_2_lat, terminal_1_lng, terminal_2_lng e nome da linha.

Por fim o banco de dados funciona de forma passiva como uma camada intermediaria para compartilhar os dados entre múltiplos dispositivos, ou seja, não executa funções lógicas e matemáticas, foi uma decisão tomada no decorrer do projeto para tornar o mesmo mais robusto e diminuir possíveis problemas relacionados a concorrência, duplicidade de dados e complexidade lógica.

4.2 DESENVOLVIMENTO DO HARDWARE

Para o desenvolvimento do código embarcado foi utilizado a IDE *PlataformIO*, esta plataforma foi concebida para suportar o uso em desenvolvimento voltado para *IoT*, atualmente a plataforma suporta, atualmente, 741 placas, 33 plataformas de desenvolvimento, 20 frameworks e possui 6843 bibliotecas (PLATFORMIO, 2014).

O hardware consiste em um microcontrolador que é responsável por verificar o estado de um sensor que será posicionado na porta do ônibus e acionar um indicador que irá alertar o motorista da presença de um deficiente visual próximo.

Inicialmente o microcontrolador seria responsável por acessar o banco de dados para consultar o estado da *flag* de proximidade, caso a mesma estivesse indicando proximidade o indicador visual iria acender no painel do motorista.

O sensor é responsável por monitorar a abertura da porta, quando o microcontrolador identificar a abertura o mesmo irá alterar a *flag* de controle da porta no banco de dados para desassociar o microcontrolador do dispositivo *Android*, indicando que o ônibus chegou e o passageiro está embarcando. O sensor será um *switch* de contato similar ao apresentado na FIGURA 24.

FIGURA 24 – CHAVE DE FIM DE CURSO

FONTE: RYDACK COMPONENTES (2019).

No projeto inicial o microcontrolador escolhido para a implementação deste projeto era o MSP4302553 da *Texas Instruments*. O microcontrolador tem uma CPU

de 16 bits com arquitetura RISC, a CPU é integrada a 16 registradores o que reduz o tempo de instrução, 12 destes registradores são de propósito geral e 4 deles são utilizados exclusivamente como *program counter, stack pointer, status register* e *constant generator.* O microcontrolador conta também com 16kB de memória flash, memória utilizada para o programa, e 512 bits de memória RAM (MSP430G2 LaunchPad Development kit, 1995).

Na FIGURA 25 é possível observar a placa de desenvolvimento MSP430 launchpad (MSP430G2 LaunchPad Development kit, 1995).

FIGURA 25 – MSP430

FONTE: MSP430G2 LAUNCHPAD DEVELOPMENT KIT (1995).

Em conjunto com este microcontrolador a equipe planejou utilizar o módulo SIM 808 que é um módulo que tem integrado a tecnologia GSM/GPRS e GPS. Este módulo seria utilizado para enviar as informações para o banco de dados e para fornecer as informações de localização para o mesmo.

O SIM 808 funciona através de comandos AT que são enviados para o mesmo através de comunicação serial. Inicialmente para realização de testes foi utilizado o *putty* para comunicação direta com o módulo pelo terminal. Desta forma foi possível entender o funcionamento dos comandos necessários para acessar as funções do SIM 808 que seriam utilizadas no projeto.

Na FIGURA 26 é possível observar o módulo SIM808.

FIGURA 26 – SIM808

FONTE: SIM900 (2018).

No início do desenvolvimento do projeto foi notada que a memória RAM disponível no MSP430 era um limitante grave para o projeto, este detalhe foi notado no momento em que foram inseridas as bibliotecas necessárias para estabelecer a comunicação por protocolo AT através da interface UART do microcontrolador.

Com o MSP430 foi possível obter as informações de localização e conectar o sistema a internet via GSM, porém a ocupação de memória RAM chegou a 80% da capacidade o que pode tornar o hardware instável, além disto nesta etapa do projeto era necessário o desenvolvimento de grande parte da lógica e dos processos para leitura e armazenamento de dados do banco de dados. Neste momento foi optado pelo uso de outro microcontrolador o ESP8266.

O microcontrolador utilizado para a implementação deste projeto foi o ESP8266 da *Espressif*. O microcontrolador é focado em *IoT* e com isso possui nativamente suporte à conexão Wi-Fi. Possui uma CPU de 32 bits com arquitetura RISC. O microcontrolador conta também com no máximo (depende do modo de operação) 50kB de memória RAM, memória utilizada para o programa, outra porção do *flash* é utilizada para a conexão Wi-Fi, e 16Mbits de memória flash. O microcontrolador ESP8266 pode ser observado na figura abaixo (ESPRESSIF

SYSTEMS, 2019).

FIGURA 27 – SIM808

FONTE: ESPRESSIF SYSTEMS (2019).

Como apresentado acima é possível observar que em termos de processamento o ESP8266 tem uma capacidade consideravelmente maior e o mesmo não apresentou problemas relacionados a capacidade de armazenamento de programa.

Com a utilização do ESP8266 e do módulo SIM808 foi possível obter a localização do GPS e foi iniciada a conexão instável com a internet, porém não foi possível enviar os dados para o banco de dados através dos comandos AT foram enviados comandos que realizam *http GET* e *http POST*, a forma em que o servidor foi construído para receber estes dados será apresentada na seção 4.4.

Ao buscar as causas desta não conexão foi observado que o problema acontecia devido a incompatibilidade da tecnologia, o SIM808 apesar de ser um módulo completo e confiável suporta apenas o padrão de rede GSM para 2G que é diferente do padrão utilizado pelas operadoras que atuam no ambiente nacional que necessitam do padrão UTMS (Sistema universal de telecomunicações celulares) (GEMALTO A THALES COMPANY, 2019).

Como não foi possível utilizar o SIM808 devido a esta incompatibilidade de tecnologias a equipe optou por fazer uso da interface Wi-Fi nativa do ESP8266 e enviar os dados para outro servidor que será apresentado no decorrer deste projeto.

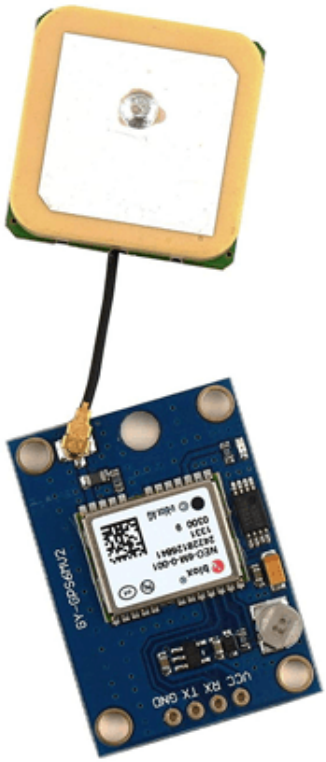
Como a função do GSM do SIM808 não foi utilizada a equipe optou por alterar o GPS para o Neo-6M que não necessita de uma fonte de alimentação externa ao microcontrolador, é menor e apresenta uma precisão de horizontal de 2,5 metros.

Este módulo apresenta os dados através da interface serial com protocolo definido pela NMEA (*National Marine Electronics Association*), esta padronização foi

concebida para compatibilizar equipamentos de georreferenciamento utilizados em equipamentos da marinha, consiste em um padrão para envio de dados com separações predefinidas e posições para os conteúdos também predefinidos (GPS INFORMATION, 2019).

Na FIGURA 28 é possível observar o módulo Neo-6m (U-BLOX AG, 2019).

FIGURA 28 – U-BLOX



FONTE: U-BLOX AG (2019).

Outra etapa do desenvolvimento do hardware é a criação do indicador para o motorista, inicialmente foi desenvolvido um indicador com LED baseado em uma *flag* que é postada no banco de dados pelo dispositivo *Android*. Durante o desenvolvimento foi percebido que seria mais interessante se o motorista conseguisse uma informação mais precisa da proximidade, por isto foi utilizado um display *oLed* para indicar a distância entre o usuário e o ônibus em metros, uma vez que o ônibus se encontra em um raio de 200 metros do usuário. Foi utilizado um display de 0,96 polegadas como o mostrado na FIGURA 29.

Para tornar este indicador mais eficiente foi desenvolvida uma lógica que utiliza além da informação de distância calculada pelo dispositivo *Android* e postada

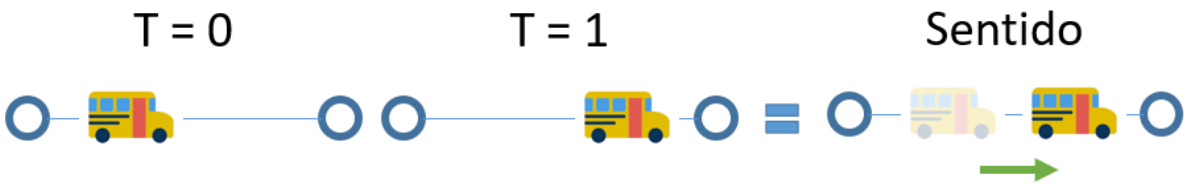
no servidor, mas também a informação da *flag* de proximidade que também é definida pelo dispositivo Android quando atinge um limiar mínimo, quando esta flag é ativada o display começa a piscar para alertar o motorista.

FIGURA 29 – DISPLAY OLED

FONTE: OS AUTORES (2019).

Além das funções apresentadas acima um outro parâmetro que foi observado durante o desenvolvimento do projeto foi a necessidade da informação do sentido do ônibus, para isto a estratégia utilizada foi realizar cálculos de distância entre dois pontos em diferentes momentos para que seja possível traçar a direção do deslocamento através da comparação da intensidade dos vetores de deslocamento conforme a FIGURA 30.

FIGURA 30 – SENTIDO DA LINHA DO ÔNIBUS



FONTE: OS AUTORES (2019).

Para não tornar este cálculo dispendioso para o microcontrolador foram estabelecidas equações para definir a distância quadrada de dois pontos de referência e a comparação da intensidade destes vetores conforme o código abaixo.

difvterminal1 = (1.852 * 60 * (terminal1lat – latvelho)* (terminal1lat – latvelho)) / 1000000 + (1.852*60*(terminal1lng - lngvelho)*(terminal1lng - lngvelho))/1000000;

difnterminal1 = (1.852 * 60 * (lat_str * 1000000 - terminal1lat) * (lat_str*1000000 - terminal1lat)) /1000000 + (1.852*60* (lng_str * 1000000 -

terminal1lng) * (lng_str * 1000000 - terminal1lng)) /1000000;

difvterminal2 = (1.852*60*(terminal2lat - latvelho) *(terminal2lat - latvelho)) /1000000 + (1.852*60*(terminal2lng - lngvelho)*(terminal2lng - lngvelho))/1000000;

difnterminal2= (1.852*60* (lat_str*1000000 - terminal2lat) * (lat_str*1000000 - terminal2lat)) /1000000 + (1.852*60* (lng_str * 1000000 - terminal2lng) * (lng_str * 1000000 - terminal2lng)) /1000000;

if((difnterminal1>difvterminal1)&&(difnterminal2<difvterminal2))

Firebase.setString("MCU1/sentido", "horário");

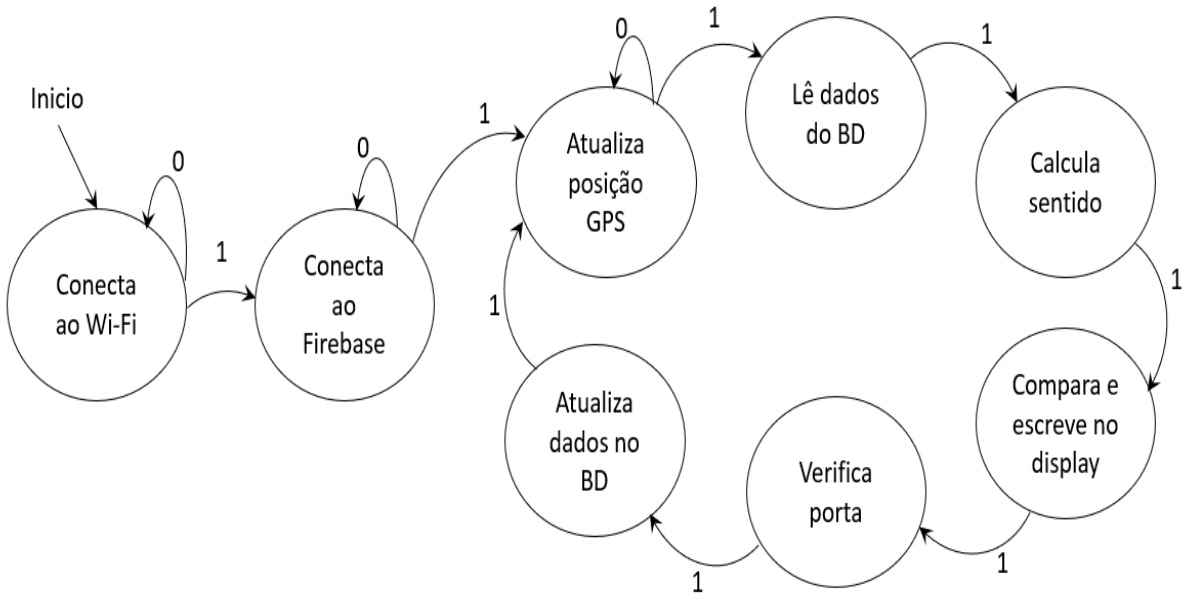
if((difnterminal1<difvterminal1)&&(difnterminal2>difvterminal2))

Firebase.setString("MCU1/sentido", "anti-horário");

Lembrando que no decorrer do projeto foi optado por utilizar o microcontrolador como uma interface de visualização para o motorista e de comparação para o ônibus, dito isto o microcontrolador é responsável por editar apenas as variáveis inerentes ao funcionamento do próprio ônibus, as variáveis atualizadas pelo dispositivo *Android* serão apenas lidas pelo microcontrolador.

Na FIGURA 31 é possível observar o comportamento do hardware envolvido em forma de uma máquina de estados em que os ramos denotados com zero são os caminhos caso a ação não consiga ser executada e os ramos denotados com um, os em que a ação foi executada com sucesso.

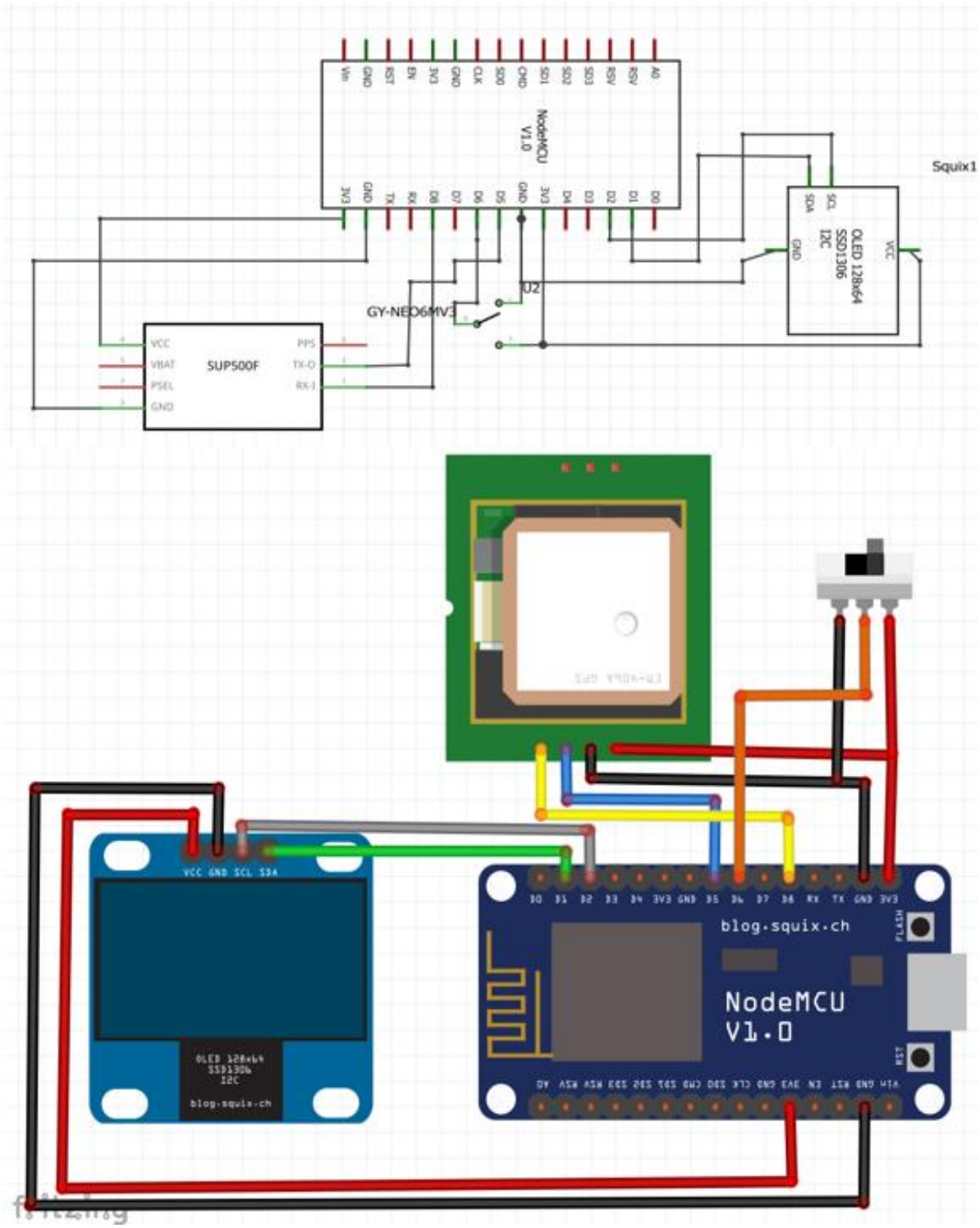
FIGURA 31 – MAQUINA DE ESTADOS DO HARDWARE



FONTE: OS AUTORES (2019).

A construção do hardware pode é representada pelo esquemático e pelo diagrama de ligações criado através do software Fritzing que pode ser observado na FIGURA 32 onde o sensor da porta é representado por um switch geral e os demais componentes pelas suas representações presentes no software utilizado.

FIGURA 32 – PROTÓTIPO ACOPLADO AO ÕNIBUS



FONTE: OS AUTORES (2019).

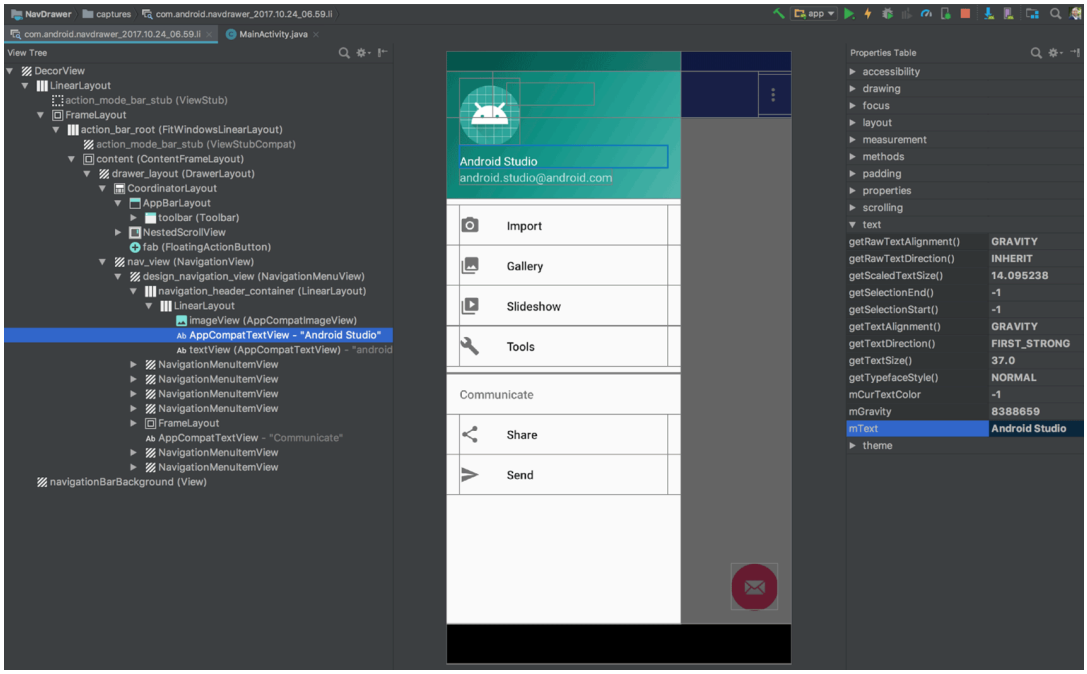
4.3 DESENVOLVIMENTO DA INTERFACE

A interface utilizada será um aplicativo *Android* com foco no tratamento dos comandos por voz, visto o público alvo do projeto. Na FIGURA 33 é possível observar um exemplo da interface do *Android Studio*, *IDE* (ambiente de desenvolvimento) que será utilizada para desenvolver este projeto.

Inicialmente o aplicativo era responsável por transcrever a voz do usuário e comparar o que o usuário disse com o nome das linhas de ônibus disponíveis no banco de dados, após a comparação o aplicativo deveria confirmar para o usuário o nome da linha selecionada (através do som), a segunda etapa do processo era atualizar um campo do banco de dados com o ID do celular de modo a associar o usuário a um ônibus.

Como a solução de banco de dados foi alterada, assim como o conceito de comparação os cálculos passaram a ser realizados localmente no dispositivo *Android* logo não se faz necessário atrelar o dispositivo a uma linha, desta forma é possível estabelecer uma relação de um microcontrolador para diversos dispositivos Android com mais facilidade fazendo uso do banco de dados orientado a objeto.

FIGURA 33 – ANDROID STUDIO



FONTE: ANDROID STUDIO (2019).

Para facilitar o entendimento do funcionamento do aplicativo será explicado passo a passo a interação entre o usuário, o aplicativo e o banco de dados. Os

artifícios técnicos utilizados para garantir estas interações serão explicados posteriormente para que a aplicação dos mesmos já esteja no contexto.

Para facilitar o entendimento do funcionamento da interface primeiramente serão explicados os artifícios utilizados para que seja possível comparar os dados tanto com o banco de dados quanto com os dados provenientes dos comandos de voz do usuário.

Para realizar comunicação de alto-falante do programa desenvolvido utiliza a biblioteca *textToSpeech e seus métodos. C*ria um contexto para iniciar um processo de *override* que é uma implementação obrigatório da biblioteca, utiliza a definição do idioma que vai transmitir ao usuário é igual a definida como padrão no *smartphone*.

Com base no reconhecimento do idioma do usuário o processo de transformar a frase do usuário em um formato de *string* é realizado através da utilização da biblioteca *SpeechRecognizer*.

O *SpeechRecognizer* é chamado através do botão do tipo *touch*, ou seja, um botão que aperte e solta, este é o momento que o usuário terá que informar a linha de ônibus desejada, a biblioteca *SpeechRecognizer* irá atuar de forma a interpretar a som da voz do usuário em uma *string*.

O texto interpretado é armazenado localmente e transcreve o texto para um *arraylist* que armazena a string, caso a interpretação for bem-sucedida esta string é gravada em uma variável local.

Para realizar a inicialização do banco de dados do Google que é o *Firebase* precisamos implementar o fragmento do código.

FirebaseApp.initializeApp(MainActivity.this);

FirebaseDatabase = FirebaseDatabase.getInstance();

DatabaseReference = firebaseDatabase.getReference();

Esse trecho do código irá inicializar a comunicação com o *Firebase,* também é necessário configurar o *Gradle* que atua como o compilador do *Android Studio* também se faz necessário inserir os dois trechos de código a seguir no compilador para que o *Firebase* possa utilizar todas as bibliotecas disponíveis.

Implementation 'com.google.firebase:firebase-database:19.2.0'

Classpath 'com.google.gms:google-services:4.3.2'

Para que o *Firebase* reconheca a URL e a chave do servidor é necessário adicionar um arquivo gerado pelo próprio *Firebase* que se chama *google-service.json*, esse arquivo é chamado atua no *background* do compilador do *Android Studio.*

Quando ocorre uma modificação no *Firebase* é necessário que o aplicativo seja informado, para isso é utilizado o trecho do código a seguir.

```
DatabaseReference.addValueEventListener(new ValueEventListener() {
  @Override
  public void onDataChange(DataSnapshot dataSnapshot) {
  listOnibus.clear();
  //Vai trazer informações por ordem que estão no dataBase
  for (DataSnapshot objSnapshot:dataSnapshot.getChildren()){
  Onibus onibus = objSnapshot.getValue(Onibus.class);
  listOnibus.add(onibus);
  } }
…);
```

Este trecho mostra a adição de um método chamado toda vez que a alteração de algum dado é realizada no banco de dados, é feito uma limpeza no *arraylist* da variável do tipo da classe ônibus, logo a seguir é carregado as variáveis que estão no banco de dados em um *arraylist* auxiliar com todos os dados das linhas contidas no banco.

Ao pressionar o botão na tela o sistema reconhece que o usuário apertou o botão através do movimento do evento *action down* responsável por realizar o chamado da função *Speech Recognizer,* quando o usuário solta o botão o *Speech Recognizer* é finalizado*.* No momento do primeiro clique o aplicativo emite um bipe que indica que o aplicativo está apto ao receber as informações do usuário.

Após o reconhecimento da fala do usuário o aplicativo realiza as manipulações necessárias e as comparações com o *arraylist* carregado do banco de dados e em caso negativo devolve para o usuário a negativa em através do uso da *Textospeech.*

Caso seja encontrada a linha desejada pelo usuário, o aplicativo avalia quantas posições existem com o nome da linha solicitada no servidor, na sequencia realiza uma comparação com o sentido desejado e analisar se esse sentido existe, caso o sentido da linha de ônibus desejado pelo usuário exista o programa irá chamar a função para iniciar o monitoramento do GPS.

Se o usuário informar um sentido inexistente o programa irá informar que o sentido não foi encontrado.

Após o usuário escolher a linha desejada do ônibus e o sentido da linha de

ônibus será necessário obter os dados de localização do GPS do *smartphone* do usuário, a localização é obtida por meio do fragmento de código a seguir.

```
public void configurarServico(){
try {
LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
LocationListener locationListener = new LocationListener() {
public void onLocationChanged(Location location) {
atualizar(location); }
…
};
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDE
R, 0, 0, locationListener);
}catch(SecurityException ex){
Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
}}
```

A biblioteca da função *location manager* realiza o chamado da função de serviço através do *get system service*, após essa linha é necessário realizar chamado de *new location listener* e depois implementar os métodos, o método mais importante utilizado neste aplicativo é *on location changed* que significa que todo o movimento que o usuário realiza é atualizado a váriável de chamada de *location,* a atualização é realizada através da requisição *request location update*, caso existe alguma falha é necessário implementar o *try catch* para trata-la.

O fragmento do código da função configurar serviço é a parte do programa que faz todo o processo de localização do usuário com a linha desejada, também é aonde se realiza todo o cálculo pode ser observado a seguir.

```
public void atualizar(Location location){
Double latPoint = location.getLatitude();
Double lngPoint = location.getLongitude();
latFire = Double.parseDouble(listOnibus.get(posicao).getLatitude());
lonFire = Double.parseDouble(listOnibus.get(posicao).getLongitude());
```

Após o usuário ter escolhido a linha de ônibus desejada e o sentido da linha de ônibus desejado, as informações do GPS fornecidas pelo *smartphone* serão adicionadas nas variáveis do tipo double chamada latPoint e lngPoint que

representam a latitude e longitude que o usuário está no momento.

As variáveis latFire e lonFire representam a posição que o ônibus está, posição armazenada no *Firebase*. Para o cálculo da distância entre o ônibus e o usuário é utilizado o fragmento de código a seguir.

```
LatLng posicaoInicial = new LatLng(latPoint,lngPoint);
LatLng posicaoFinal = new LatLng(latFire,lonFire);
distancia =
SphericalUtil.computeDistanceBetween(posicaoInicial, posicaoFinal);
```

A biblioteca *Spherical Util possibilita uma maneira simples para realizar* o cálculo desta distância e obter a resposta em metros.

Após obter a distância calculada entre o usuário até a linha de ônibus desejado é realizada a comparação com o valor de distância presente no banco de os, caso a distância for menor do que a distância que se encontra no banco de dados o aplicativo atualiza este valor com a distância calculada, caso contrário o valor antigo é mantido, esse fragmento do código é necessário pois pode existir dois usuários próximos e o que pode escrever a distância no banco de dados é aquele cujo tem a menos distância.

```
String difenca = listOnibus.get(posicao).getDiferenca();
if(distancia <= Double.parseDouble(difenca.replaceAll(",",".") )) {
databaseReference.child(listOnibus.get(posicao).getId()).child("diferenca").
setValue(String.valueOf(String.format("%,.2f", distancia))); }
```

Quando a distância for menor que 100 metros o aplicativo irá informar ao usuário que a linha desejada está próxima, para isto é utilizado o seguinte fragmento de código.

```
if(distancia <= 100.00){
if(flag==0) {
textToSpeech.speak("ônibus chegando", TextToSpeech.QUEUE_FLUSH,
null);
 flag = 1;
}
```

Quando o motorista abrir a porta ao usuário o aplicativo irá informar que a linha chegou, pois pode existir mais de um ônibus que para no mesmo, após esta ação o aplicativo é reiniciado para que esteja disponível para selecionar outra linha de ônibus, o código responsável por esta reinicialização pode ser observado na

sequencia.

if(Integer.parseInt(listOnibus.get(posicao).getPorta()) == 0) {

databaseReference.child(listOnibus.get(posicao).getId()).child("proximidade") .setValue("1");

}else

if(Integer.parseInt(listOnibus.get(posicao).getPorta()) == 1){

if (flagchegou == 0){

textToSpeech.speak(

String.valueOf(listOnibus.get(posicao).getNomedalinha()+" "+"chegou !"),

TextToSpeech.QUEUE_FLUSH, null);

flagchegou ++;

}

}}}}

Um ponto importante sobre o *Android* é que por questão de segurança e para ter acesso ao hardware do smartphone que são o microfone, GPS e internet é necessário solicitar as aprovações do usuário. Para ter acesso a esses recursos é necessário solicitar acesso manifesto do *android*, o manifesto é aonde se gerencia todas as permissões.

<uses-permission android:name = "android.permission.

ACCESS_FINE_LOCATION" />

<uses-permission-sdk-23 android:name=

"android.permission.ACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.RECORD_AUDIO" />

Essas permissões são necessárias, pois sem acesso ao hardware e internet o aplicativo não irá funcionar corretamente. Há um fragmento no código que quando ainda não foram dadas as permissões é solicitado ao usuário que permita ter acesso a esse hardware.

private void checkPermission()

{

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

if (!(ContextCompat.checkSelfPermission(this,

Manifest.permission.RECORD_AUDIO) ==

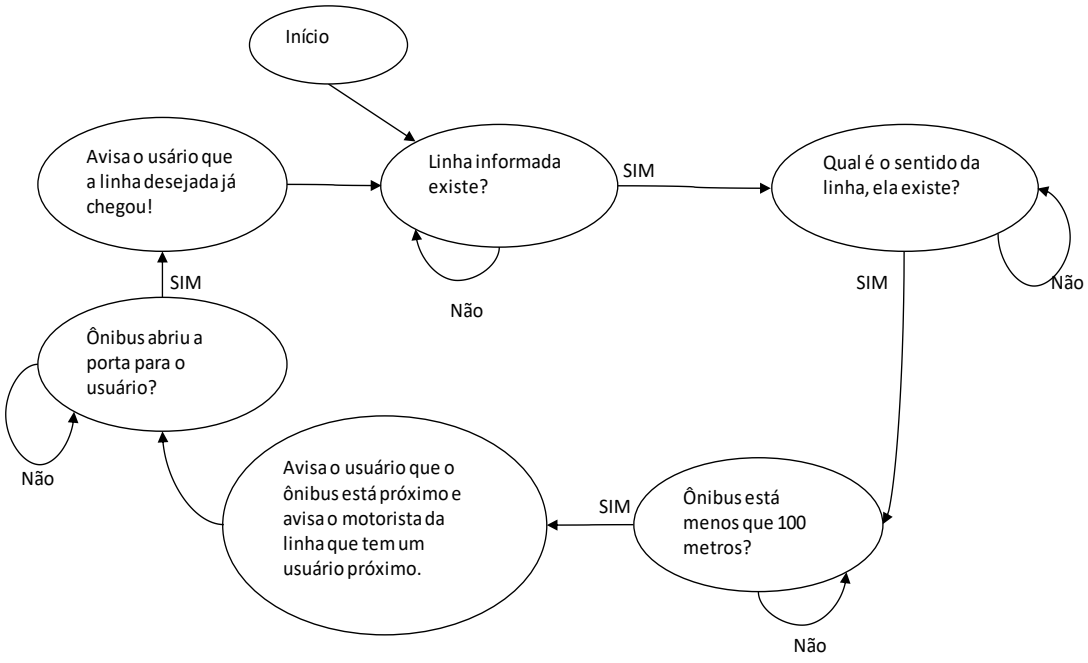
PackageManager.PERMISSION_GRANTED)) {Intent intent = new

```
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
Uri.parse("package:" + getPackageName()));
startActivity(intent); finish();}}
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
}}}
```

Se as permissões foram concedidas pelo usuário do smartphone o aplicativo irá funcionar, se usuário não conceder as permissões não será possível utilizar o aplicativo até que as permissões sejam concedidas.

Na FIGURA 34 é possível observar o comportamento do android envolvido em forma de uma máquina de estados em que os ramos denotados com não são os caminhos caso a ação não consiga ser executada e os ramos denotados com sim, os em que a ação foi executada com sucesso.

FIGURA 34 – MAQUINA DE ESTADOS DO ANDROID



FONTE: OS AUTORES (2019).

4.4 INTEGRAÇÃO

A integração entre os sistemas desenvolvidos é a chave para este projeto funcionar conforme o esperado, desta forma nesta sessão será abordado inicialmente o funcionamento geral do sistema unindo microcontrolador, aplicativo e banco de dados e na sequencia serão apresentadas as soluções para os principais problemas encontrados no decorrer do projeto.

O banco de dados tem a função neutra de apenas armazenar os dados sem realizar alterações sobre os mesmos.

Conforme ilustrado na sessão 4.1 a estrutura do banco é dividida em classes, uma para cada microcontrolador, esta foi a maneira encontrada para não necessitar da criação de vínculos entre os dispositivos *Android* e os microcontroladores, a criação dos vínculos poderia criar duplicidade de dados e problemas de concorrência.

A solução para evitar os problemas de concorrência foi definir as lógicas localmente e independentemente de dispositivo, por exemplo a proximidade é calculada pelo dispositivo *android*, o mesmo envia uma *flag* de proximidade para o servidor, porém ao comparar com a *flag* da porta o dispositivo *Android* compara o valor da porta armazenado no banco de dados com o valor da *flag* de proximidade local para só então concluir o ciclo reiniciando o aplicativo e alterando a *flag* de proximidade para zero.

Outra estratégia está relacionada aos valores postados no banco de dados sobre a distância calculada pelo dispositivo que são apresentadas pelo microcontrolador no display oLed, para realizar a atualização o aplicativo verifica se a distância que já está no banco de dados é menor que a distância atual, em caso afirmativo a distância não sobrescreve a anterior em caso negativo sobrescreve. O uso desta sobre descrição se faz funcional uma vez que ao fim do processo o microcontrolador atualiza um valor de distância alto no servidor (valor acima do limiar que inicia a apresentação no display), desta forma caso exista outro usuário próximo ele irá sobrescrever este registro rapidamente.

O microcontrolador por si só não gera situações de concorrência ou duplicidade uma vez que cada um é representado por uma classe exclusiva, os parâmetros para o microcontrolador são voltados principalmente para visualização e para o envio de dados que servem como broadcast para todos os aplicativos conectados a um determinado microcontrolador.

Com base nestas precauções extras podemos concluir que o sistema conta com uma integração robusta que não deverá apresentar erros relacionados a interferências nos fluxos de dados entre os componentes do sistema visto que todas as informações compartilhadas por dispositivos *Android* possuem pelo menos uma estratégia para evitar sobreposição de dados indevida.

5 RESULTADOS

Nesta seção serão abordados os resultados provenientes das atividades desenvolvidas e apresentadas na seção anterior.

5.1 BANCO DE DADOS

As estruturas de banco de dados foram avaliadas e inicialmente se optou por uma estrutura de banco de dados relacionais, porem durante o desenvolvimento do projeto foi notado que conforme citado no desenvolvimento para uma solução mais robusta seria necessário realizar as comparações localmente dispositivo a dispositivo e não diretamente no banco de dados conforme planejado inicialmente.

Com isso a equipe decidiu optar pelo desenvolvimento no *Firebase* que é um banco de dados não relacional, *noSQL* que organiza os dados em uma estrutura orientada a objeto.

A estratégia da equipe foi utilizar classes e subclasses para cada um dos dispositivos microcontrolados, como estes dispositivos são únicos por ônibus, a cada linha nova o registro deve ser feito manualmente no banco de dados. Uma vez registrado o banco armazena informações de ambos os dispositivos sendo que cada classe representa um microcontrolador e cada subclasse o conjunto de dados significativos para as trocas de informações entre microcontrolador e dispositivo *Android*.

O desenvolvimento da lógica para comparação foi realizado diretamente nos dispositivos, assim como a lógica para evitar situações de concorrência durante a execução do aplicativo e desta forma possibilitando conexões múltiplas.

Um detalhe importante relacionado aos dados enviados para o banco de dados é de que todos sejam passados como *string*, este padrão foi adotado para facilitar o carregamento de uma lista para o aplicativo e a busca nesta lista pelos parâmetros indicados pelo usuário.

Com isto o objetivo específico cinco, relacionado a definição do banco de dados foi concluído assim como o objetivo específico oito que trata da estruturação do banco, a lógica aplicada foi movida para os dispositivos porem ainda assim está presente logo este objetivo foi atendido.

A estrutura do banco de dados foi apresentada no desenvolvimento deste projeto e como o banco de dados funciona de forma neutra neste projeto o funcionamento do mesmo pode ser observado nos testes do Hardware e do

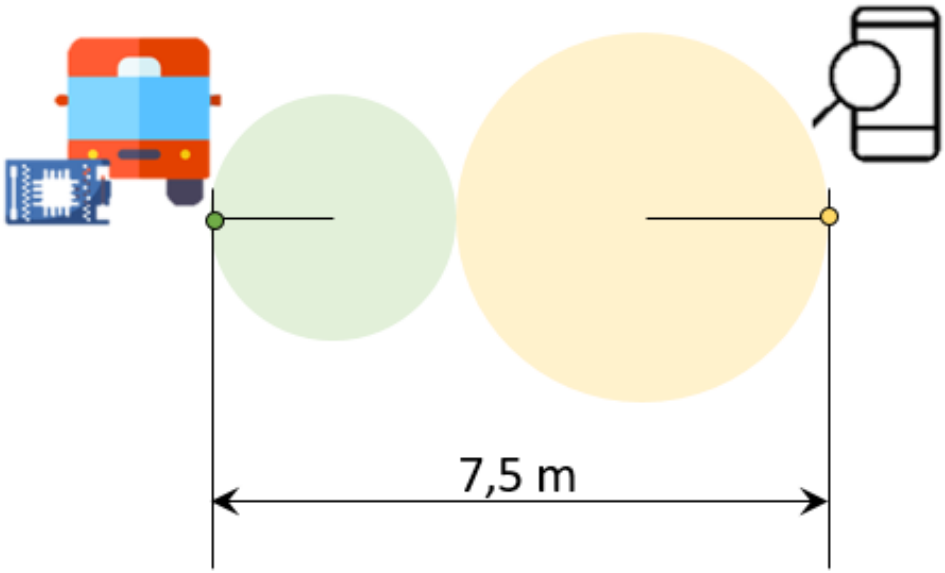
dispositivo *Android*.

5.2 HARDWARE

Um dos grandes pontos focais deste projeto é a habilidade de realizar comparações com coordenadas GPS, esta tecnologia foi utilizada através de módulos prontos, inicialmente o módulo SIM808, e posteriormente o módulo Neo-6M que possui uma precisão de 2,5 metros.

Para a aplicação esta precisão não afeta o funcionamento e o atendimento a proposta com isso a tecnologia é aplicável uma vez que combinada a precisão apresentada pelo GPS do dispositivo *Android* o erro é em média de até cinco metros, podendo variar com o dispositivo. Na FIGURA 35 podemos observar o erro máximo do sistema (U.S. AIR FORCE, 2017).

FIGURA 35 – PRECISÃO DO GPS



FONTE: OS AUTORES (2019).

Além disto existe a redundância do sensor de monitoramento da porta o que minimiza a probabilidade de o usuário não embarcar por conta de posicionamento inadequado. Com isso a tecnologia foi avaliada como aplicável e o objetivo especifico dois foi atingido.

O objetivo especifico trata da especificação do hardware. O início do desenvolvimento do hardware para este projeto foi a definição dos módulos e

tecnologias que seriam aplicadas, logo no início do desenvolvimento do projeto a equipe se deparou com dificuldades relacionadas a memória de programa disponível no MSP430, com isso a equipe decidiu alterar o microcontrolador para o ESP8266 conforme apresentado no desenvolvimento deste projeto.

Apesar da alteração do microcontrolador a principal dificuldade conforme citada durante o desenvolvimento foi a impossibilidade de realizar a comunicação com o servidor via módulo SIM808. A solução foi realizar a comunicação diretamente pelo ESP8266 por consequência o dispositivo necessita de uma conexão Wi-Fi

A necessidade da comunicação por uma rede Wi-Fi não se torna um limitante no panorama atual, visto que este tipo de rede pode ter como *access point* um celular ou um módulo conversor de 4G para Wi-Fi.

Outra alteração foi a substituição por completo do módulo SIM808 pela combinação entre o Wi-Fi disponível no ESP8266 e o GPS Neo-6m que é um módulo menor e que não precisa de uma fonte de alimentação externa ao microcontrolador, o que facilita o desenvolvimento do Hardware. Esta alteração não tem interferência no projeto uma vez que a resposta de ambos os módulos é a localização em coordenadas GPS. Com isso o objetivo especifico três foi concluído com sucesso.

O objetivo quatro trata do indicador visual para o motorista e também da última alteração em termos de Hardware neste projeto. Esta alteração foi a substituição do indicador para o motorista, inicialmente seria utilizado um *LED* para indicar a proximidade, no decorrer do desenvolvimento foi entendido que seria uma necessidade do cliente a possibilidade de observar a distância do usuário com deficiência visual para se orientar mais facilmente. Para solucionar este problema o *LED* foi substituído por um display que mostra a distância em metros do dispositivo *Android* e do microcontrolador. Com isso o objetivo quatro foi concluído com resultados além do planejado.

Além destes objetivos durante o desenvolvimento do projeto foi notada a necessidade de dar a opção para o usuário selecionar o sentido do ônibus, esta solução foi executada através da comparação de distancias no decorrer do tempo conforme apresentado no TÓPICO 4.2.

Desta forma foi desenvolvido um hardware com uma solução robusta que atende aos objetivos definidos no início do projeto e também atende demandas dos clientes que foram notadas durante o desenvolvimento.

Outro ponto importante foi que as mudanças propiciaram uma redução no

custo do projeto que inicialmente era composto por R$50 do MSP430, R$170 do módulo SIM808 e R$20 de periféricos como chave *led* e caixa patola para R$20 do ESP8266, R$40 do módulo Neo-6m, R$40 do display e R$20 de periféricos totalizando uma redução de R$130 por dispositivo e chegando a aparência final conforme o apresentado na FIGURA 36.

FIGURA 36 – HARDWARE

FONTE: OS AUTORES (2019).

5.2.1 TESTE E VALIDAÇÃO DO HARDWARE

Para testar e validar o funcionamento foram testadas as situações lógicas possíveis a qual o sistema pode ser submetido, como as comparações e cálculos, apesar de utilizarem dados do banco de dados são realizadas localmente e existe uma classe exclusiva para cada microcontrolador não a risco de interferência entre microcontroladores.

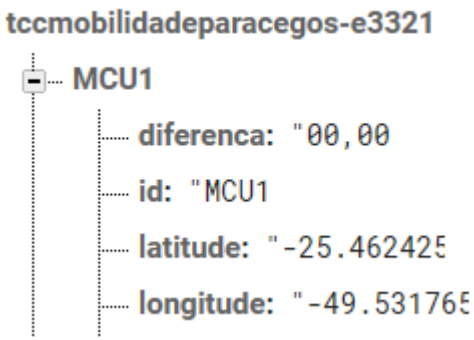
Os testes foram realizados em ambiente interno para evitar ruídos devido a variação da localização atualizada recorrentemente pelo GPS.

Os testes foram realizados manualmente, com isso nesta seção serão apresentados as ações e alterações executadas no banco de dados e a resposta do microcontrolador e vice-versa apresentando o instante antes da alteração e a reposta após a alteração (quando aplicável).

- Validação da coordenada GPS

Na FIGURA 37 podemos observar a latitude e longitude recebidas pelo Neo-6m e enviadas para o banco de dados

FIGURA 37 – VALIDAÇÃO DO FIREBASE

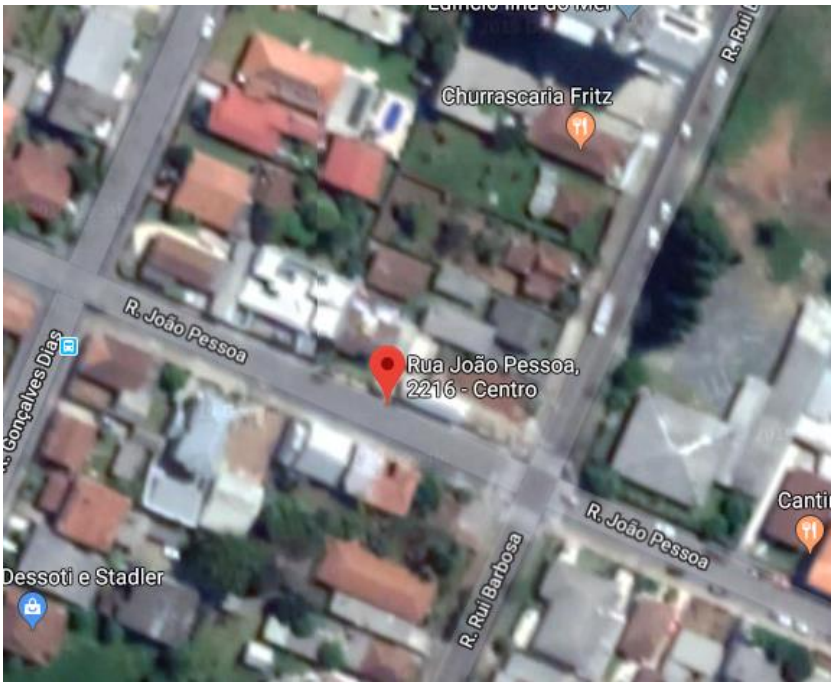


FONTE: OS AUTORES (2019).

A localização dos testes foi na coordenada -25.4624187, -49.5323868 que pode ser observada na FIGURA 38 ou através do link a seguir:

https://www.google.com.br/maps/place/R.+João+Pessoa,+2216+-+Centro,+Campo+Largo+-+PR,+83601-060/@-25.4624187,-49.5323868,299m/data=!3m1!1e3!4m5!3m4!1s0x94dd177e9595fa63:0x74f04516688799f4!8m2!3d-25.4622881!4d-49.5317239

FIGURA 38 – GOOGLE MAPS



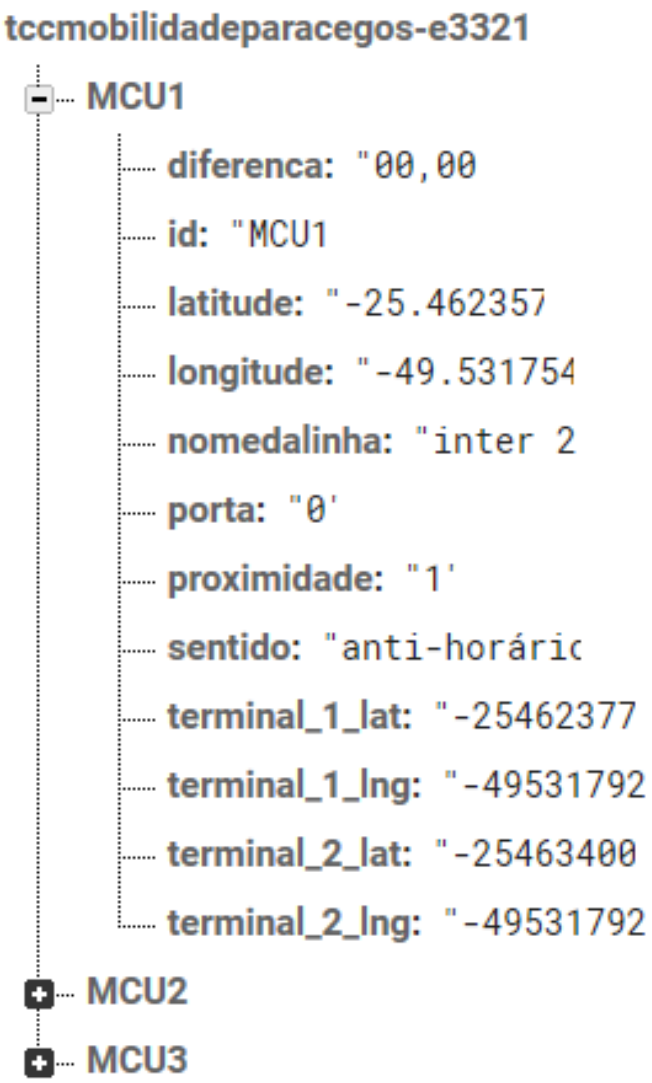
FONTE: OS AUTORES (2019).

- Teste do cálculo do sentido

Com o dispositivo parado para o cálculo do sentido foram variadas as referências das coordenadas em torno do ponto medido pelo Neo-6m caso esteja se distanciando do terminal 1 e se aproximando do terminal 2 o sentido é o horário, caso esteja se aproximando do terminal 1 e se distanciando o cálculo do sentido deve indicar anti-horário. Para simplificar a apresentação do teste foi alterado apenas uma das coordenadas, a latitude de ambos os terminais.

Na FIGURA 39 podemos observar o estado inicial (guardando as últimas informações).

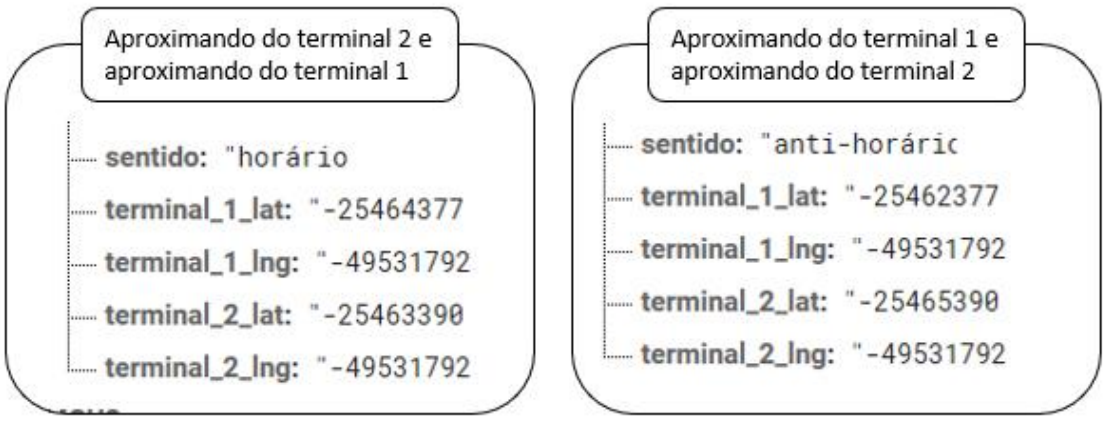
FIGURA 39 – FIREBASE ID MCU1



FONTE: OS AUTORES (2019).

Na FIGURA 40 podemos observar os resultados dos dois cenários aplicados.

FIGURA 40 – FIREBASE TESTE DE APROXIMAÇÃO



FONTE: OS AUTORES (2019).

- Teste do acionamento do display apresentando distância.

Quando o valor da distância no *Firebase* é menor do que 200 metros a mesma é apresentada no display conforme ilustrado na FIGURA 41.

FIGURA 41 – FIREBASE TESTE COM DISPLAY

FONTE: OS AUTORES (2019).

- Teste do alerta de proximidade (display piscando)

Quando o *Firebase* abriga uma distância abaixo do limiar de apresentar no display (duzentos metros) e com a *flag* de proximidade ativa, menor do que 100 metros o display começa a piscar conforme a FIGURA 42.

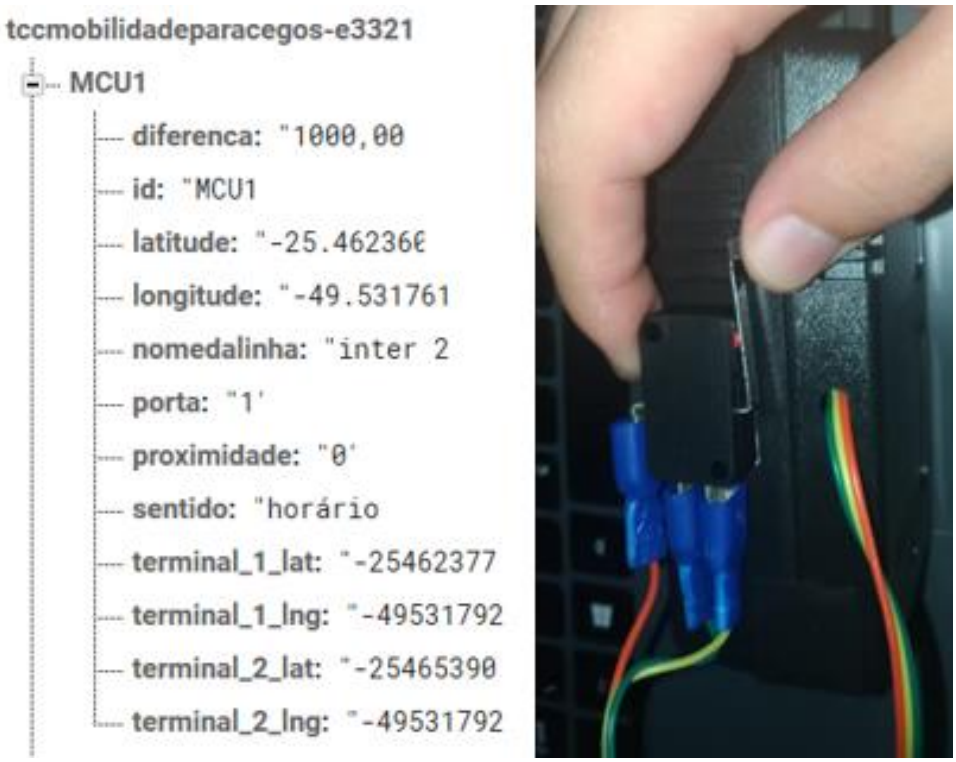
FIGURA 42 – FIREBASE TESTE COM DISPLAY PISCANDO

FONTE: OS AUTORES (2019).

- Teste da porta

Ao pressionar a chave da porta o indicador no banco de dados é alterado para um conforme a FIGURA 43.

FIGURA 43 – FIREBASE TESTE COM CHAVE DA PORTA



FONTE: OS AUTORES (2019).

5.3 INTERFACE ANDROID

No que se refere a interface foi possível desenvolver a lógica necessária para realizar o reconhecimento de voz que será utilizado pelo usuário para solicitar a linha de ônibus que interessa o mesmo e também a aplicação de um feedback sonoro por meio do uso de bibliotecas voltadas ao reconhecimento de voz e transcrição das informações conforme descrito no desenvolvimento.

O reconhecimento de voz é eficiente e não apresenta muitos erros, de forma a assegurar a assertividade da execução da seleção da linha, após o usuário falar a linha desejada o aplicativo executa a comparação com as linhas cadastradas nos microcontroladores disponíveis e informa para o usuário caso a linha não foi encontrada, caso o nome da linha exista o aplicativo repete o nome informado pelo usuário para o próprio usuário através da interface de som.

Na sequência o aplicativo repete esta estratégia para a definição do sentido do ônibus pelo usuário sendo que o mesmo pode optar por horário e anti-horário. Considerando as precauções adotadas o objetivo específico um que trata da avaliação da aplicabilidade das soluções para reconhecimento de voz do *Android* foi concluído de forma satisfatória uma vez que foi possível isolar os casos de erro de comparação garantindo a aplicabilidade dos recursos ao projeto.

No que se trata do desenvolvimento da interface, visto que o projeto se baseia em tecnologia assistiva e tem como foco pessoas com baixa visão a interface foi desenvolvida de modo a facilitar a utilização do aplicativo. O aplicativo conta com um botão de ação que cobre toda a tela do dispositivo *Android* e é ativada pelo toque em qualquer parte da tela.

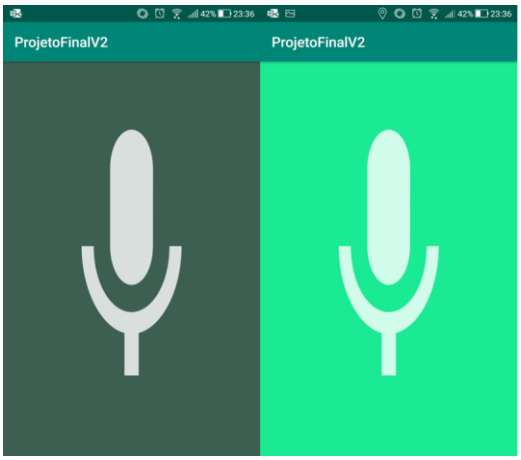
Os feedbacks para o usuário são realizados por meio da interface de áudio, a única alteração na tela é a mudança no brilho no momento que o botão está ativado, sendo assim pode servir de referência para pessoas que ainda possuem uma pequena parcela da visão. Com isso o objetivo sete foi alcançado uma vez que a interface ficou intuitiva e com necessidade de pouca interação.

Na FIGURA 43 podemos observar a interface final para o usuário com o botão pressionado e sem pressionar.

No que diz respeito a utilização do GPS a mesma foi realizada através de métodos conforme apresentado no desenvolvimento, os cálculos e comparações lógicas que são usadas como gatilho são realizadas no próprio dispositivo *Android*

através da utilização das informações contidas localmente e no banco de dados hospedado no *Firebase*.

FIGURA 44 – ANDROID TESTE DE BOTÃO



FONTE: OS AUTORES (2019).

A conexão com o banco de dados foi estabelecida conforme apresentado no desenvolvimento do projeto, o *Firebase* é um banco de dados em tempo real, logo as alterações são refletidas diretamente do aplicativo para o banco de dados e vice-versa, com isso o objetivo seis foi alcançado uma vez que atende as necessidades do projeto mantendo ambos os dispositivos envolvidos atualizados regularmente.

5.3.1 TESTE E VALIDAÇÃO DO APLICATIVO

Para testar e validar o funcionamento do aplicativo foram testadas as situações lógicas possíveis a qual o sistema pode ser submetido, como as comparações e cálculos.

Os testes foram realizados estaticamente para evitar ruídos devido a variação da localização atualizada recorrentemente pelo GPS.

Os testes foram realizados manualmente, com isso nesta seção serão apresentados as ações e alterações executadas no banco de dados e a resposta mostrada no banco de dados para cada situação assim como descrito o comportamento sonoro do aplicativo.

- Validação da coordenada GPS e cálculo de distancia

Para validar foi observado durante o desenvolvimento as coordenadas através de campos criados para teste conforme apresentado na FIGURA 44, onde a posição coincide com a do local do teste no momento.

FIGURA 45 – ANDROID TESTE DE GPS

FONTE: OS AUTORES (2019).

De modo a tornar os testes unificados foram realizados juntos e sequencialmente primeiro apenas hardware e posteriormente adicionando o aplicativo como ambos estão em coordenadas próximas, mesma mesa, a distância fica próxima (apenas erro de precisão de ambos os dispositivos) o cálculo da distância mostrado no banco de dados pode ser observado na FIGURA 45 onde a distância (em metros) é representada pelo campo diferença.

FIGURA 46 – ANDROID TESTE DE DISTÂNCIA

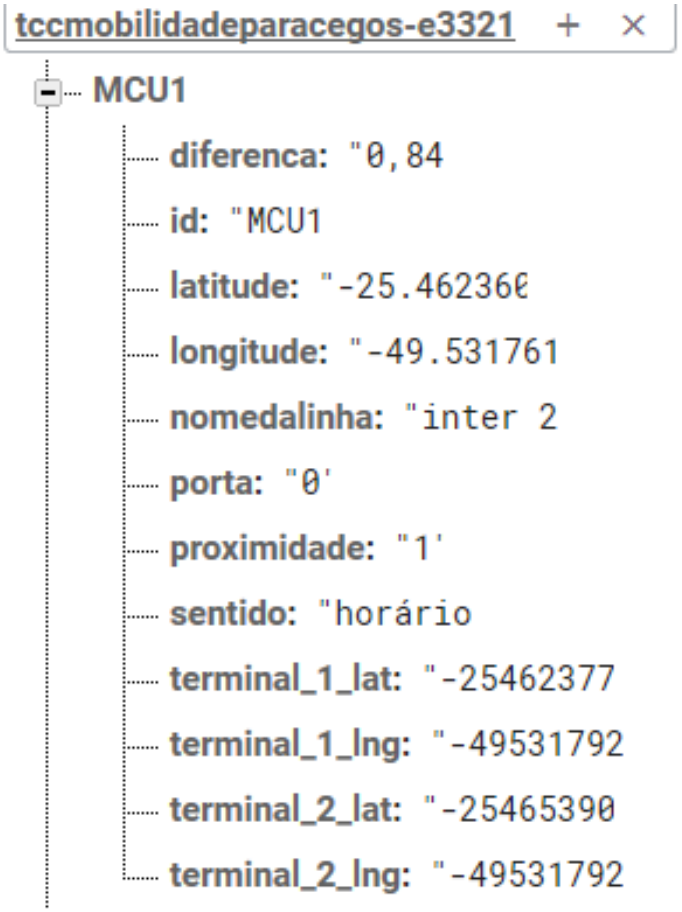
FONTE: OS AUTORES (2019).

- Teste do alerta de proximidade
- Teste da detecção da menor distancia

Conforme abordado no desenvolvimento uma das estratégias para evitar concorrência é a verificação pelo aplicativo do valor da distância presente no servidor de modo que o motorista seja avisado sempre da distância do usuário mais próximo, dessa forma o aplicativo só sobrescreve no *Firebase* caso a distância seja menor.

Neste teste que foi realizado estaticamente a distância tende a convergir com a variação das coordenadas que são recalculadas a todo momento uma vez que não irá sobrescreve valores maiores. O exemplo o funcionamento desta estratégia pode ser observado na convergência da distância mostrada na FIGURA 47.

FIGURA 47 – ANDROID TESTE DE APROXIMIDADE

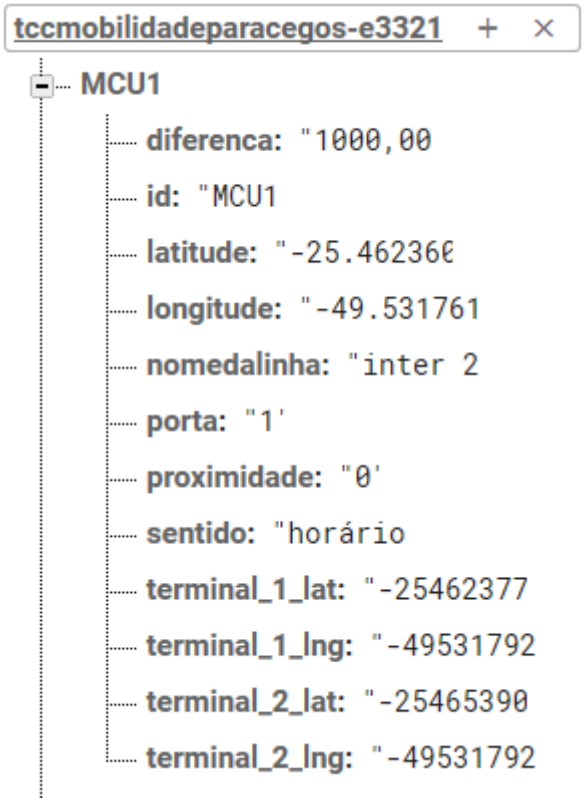


FONTE: OS AUTORES (2019).

- Teste da desvinculação do aplicativo (chegada)

Seguindo o teste anterior a porta foi pressionada, com isso o aplicativo respondeu com a chegada do ônibus e desvinculou o celular resetando a *flag* de proximidade e alterando a distância para 1000 metros conforme a FIGURA 48.

FIGURA 48 – ANDROID TESTE DE CHEGADA



FONTE: OS AUTORES (2019).

6 CONCLUSÃO

Durante o desenvolvimento deste projeto foram encontrados muitos desafios e dificuldades, a proposta da equipe foi voltada para o desenvolvimento de uma solução para problemas de mobilidade e inclusão social desde o princípio, no entanto as ferramentas e tecnologias aplicadas para desenvolver esta solução não eram conhecimentos dominados pelos membros da equipe.

Durante o desenvolvimento a falta de domínio da equipe proporcionou um grande desenvolvimento para os envolvidos uma vez que desde o início do projeto foi necessário a pesquisa de ferramentas e artifícios para integrar as três frentes em que o projeto foi desenvolvido, hardware, aplicativo e banco de dados.

Além da pesquisa durante a execução foram realizadas alterações por necessidade por exemplo como na situação da troca do módulo GPS e GSM pela utilização de outro módulo GPS junto com o uso de Wi-Fi, até esta etapa por exemplo os desenvolvimentos das outras frentes do trabalho já haviam iniciado o que proporcionou para a equipe a oportunidade de aprender os fundamentos de PHP, SQL comandos AT entre outros mesmo não sendo aplicados no projeto final.

Durante o projeto também foram notadas necessidades do cliente que proporcionaram mudanças de paradigma como por exemplo a alteração do núcleo do projeto que inicialmente estava no banco de dados e no projeto final se dividiu entre os dispositivos *Android* e o microcontrolador.

Esta mudança de paradigma foi oportuna para a equipe aprofundar o desenvolvimento nas plataformas moveis citadas anteriormente assim como na alteração do tipo de banco de dados que foi para um modelo relativamente novo no mercado que propicia aos envolvidos uma atualização sobre novas soluções disponíveis.

Por fim o projeto também propiciou que conhecimentos desenvolvidos no decorrer do curso fossem aplicados no projeto, desde conhecimentos em eletrônica e programação até conhecimentos relacionados a metodologia e gerenciamento de projetos que ajudaram a equipe a manter o rumo do projeto mesmo com os imprevistos e alterações que ocorreram durante o desenvolvimento.

Contudo o mais importante foi a execução do projeto de forma que o mesmo possa ser aplicado na prática de forma simples e barata, uma vez que o projeto não teve um viés econômico e sim um viés social que proporciona utilizar os

conhecimentos adquiridos durante o curso para contribuir com uma solução que pode facilitar a locomoção e auxiliar pessoal com deficiência visual a alcançar mais independência e integração.

Durante o desenvolvimento a equipe procurou desenvolver soluções para possíveis problemas que poderiam ser causados por falhas no aplicativo ou na lógica porem muitas melhorias ainda precisam ser realizadas e certamente outras ainda precisam ser imaginadas, na sessão relacionada a trabalhos futuros estes pontos serão abordados.

6.1 TRABALHOS FUTUROS

Futuramente para a implementação deste projeto em escala será necessário incorporar ao hardware um circuito regulador de tensão estável para poder conectar o circuito diretamente a bateria do ônibus por exemplo.

Outro ponto interessante é a inclusão da tecnologia GSM com alguma estratégia para tornar este hardware ainda mais genérico.

Aplicar outras tecnologias para expandir a aplicabilidade, desenvolver por exemplo um aplicativo com uma versão para pessoas com visão que auxilie outras necessidades como por exemplo pessoas com deficiência motora que necessitam de auxílio para embarcar em um ônibus fora do ponto ou tem dificuldade para chegar ao mesmo.

Por fim para que este projeto seja viável e mantenha o cunho social a redução do custo para a implementação do hardware com um projeto dedicado em PCI e com design especifico para colaborar com a viabilidade econômica do projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

AGILENT TECHNOLOGIES. **GSM Fundamentals**. Agilent Technologies. Califórnia, p. 94. 2000. (10001633-GSM).

ANDROID. **ANDROID**, 2014. Disponivel em: <https://developer.android.com/guide/components/fundamentals.html>. Acesso em: 08 Março 2018.

BRAHLER, S. **Analysis of the Android Architecture**. Karlsruhe Institute of Technology. Karlsruhe, p. 43. 2010.

ESPRESSIF SYSTEMS, 2019. Disponivel em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 05 Outubro 2019.

FUNDAÇÃO Dorina, 2018. Disponivel em: <https://www.fundacaodorina.org.br/a-fundacao/deficiencia-visual/estatisticas-da-deficiencia-visual/>. Acesso em: 2019 Maio 05.

GEMALTO A THALES COMPANY, 2019. Disponivel em: <https://www.justaskgemalto.com/br/qual-e-diferenca-entre-redes-umts-3g-e-edge/>. Acesso em: 05 Outubro 2019.

GOLLEDGE, R. G.; MARSTON, J. R.; COSTANZO, C. M. Attitudes of Visually Impaired Persons Toward the Use of Public Transportation. **Attitudes of Visually Impaired Persons Toward the Use of Public Transportation**, Berkeley, 01 dez. 2001. 18. Disponivel em: <https://escholarship.org/uc/item/5pv2k256>. Acesso em: 2019 Março 29.

GOMES et al. MIDIACOM, 2012. Disponivel em: <http://www.midiacom.uff.br/~natalia/2012-1-sisop/tgrupo1.pdf>. Acesso em: 06 Setembro 2017.

GORVERNO do Brasil, 2017. Disponivel em: <http://www.brasil.gov.br/noticias/educacao-e-ciencia/2010/08/tecnologia-assistiva >. Acesso em: 14 Maio 2019.

GORVERNO do Brasil IPEA, 2017. Disponivel em: <http://www.brasil.gov.br/governo/2011/05/estudo-do-ipea-mostra-que-65-da-populacao-usam-transporte-publico-nas-capitais>. Acesso em: 2019 Março 20.

GPS INFORMATION, 2019. Disponivel em: <https://www.gpsinformation.org/dale/nmea.htm>. Acesso em: 20 Outubro 2019.

HOCK-CHUAN, C. **Nanayng Technological University**, 2010. Disponivel em: <https://www.ntu.edu.sg/home/ehchua/programming/sql/relational_database_design.html>. Acesso em: 2019 Maio 15.

INFO WESTER, 2013. Disponivel em: <https://www.infowester.com/wifi.php>. Acesso em: 01 Maio 2018.

INTEL CORPORATION, 2017. Disponivel em: <https://www.intel.com.br/content/www/br/pt/support/articles/000005725/network-and-i-o/wireless-networking.html>. Acesso em: 02 Maio 2018.

LEICA GEOSYSTEMS INC. Introduction to GPS (Global Positioning System). **GPS Basic**, Switzerland, v. 1, n. 1.0, p. 63, 1999. ISSN 713282.

MSP430G2 LaunchPad Development kit. **TI**, 1995. Disponivel em: <http://www.ti.com/tool/MSP-EXP430G2>. Acesso em: 23 Maio 2019.

PLATFORMIO, 2014. Disponivel em: <platformio.org>. Acesso em: 23 Outubro 2019.

PREFEITURA Municipal de Curitiba. **Plano de Mobilidade Urbana e Transporte Integrado PlanMob Curitiba**. Disponivel em: <http://redpgv.coppe.ufrj.br/index.php/es/informacion/banco-de-estudo-de-impactos/641-plano-de-mobilidade-curitiba/file>. Acesso em: 2019 Abril 20.

REMOTEM SQL, 2019. Disponivel em: <https://remotemysql.com/>. Acesso em: 5 Julho 2019.

RYDACK Componentes. **Rydack Componentes**, 2019. Disponivel em: <https://www.ryndackcomponentes.com.br/micro-switch-fim-de-curso/1273-chave-micro-switch-250v-16a-com-alavanca-de-27mm-kw11-7-fim-de-curso.html?search_query=switch&results=16>. Acesso em: 23 Maio 2019.

SILICON INVERSTOR, 2016. Disponivel em: <www.siliconinvestor.com/readmsg.aspx?msgid=30505872>. Acesso em: 30 Abril 2018.

SIM900. **Robotshop**, 2018. Disponivel em: <https://www.robotshop.com/media/files/PDF/gprs-shield-sld33149p.pdf>. Acesso em: 2019 Maio 2019.

SRIVASTAVA, L.; PETRAZZINI, B.; SELIAN, A. **GSM Case Study**. International Telecommunication Union. Japão, p. 50. 2000.

TAKAI, O. K.; ITALIANO, I. C.; FERREIRA, J. E. **INTRODUÇÃO A BANCO DE DADOS**. USP. São Paulo, p. 119. 2005.

TELECO INTELIGÊNCIA EM TELECOMUNICAÇÃO, 2016. Disponivel em:

<http://www.teleco.com.br/tutoriais/tutorialwifimanaus2/pagina_3.asp>. Acesso em: 02 Maio 2018.

TELECON ABC, 2017. Disponivel em: <www.telecomabc.com/d/dsss.html>. Acesso em: 02 Maio 2018.

U.S. AIR FORCE, 2017. Disponivel em: <https://www.gps.gov/systems/gps/performance/accuracy/>. Acesso em: 22 Outrubro 2019.

U-BLOX AG, 2019. Disponivel em: <https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf>. Acesso em: 21 Outubro 2019.

URBS. URBS S.A, 2019. Disponivel em: <https://www.urbs.curitiba.pr.gov.br/itibus>. Acesso em: 20 Maio 2019.

WORD Health Organization, 2018. Disponivel em: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. Acesso em: 2019 Abril 05.

APÊNDICE 1 – CÓDIGO DO PROGRAMA DO MICROCONTROLADOR

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <Wire.h>
#include "SSD1306Wire.h"

#define FIREBASE_HOST "tccmobilidadeparacegos-e3321.firebaseio.com"
#define FIREBASE_AUTH "1T4PamLpwUZbp2BiqQOZq9XVnuJxUsaOi7ciWdWH"
#define WIFI_SSID "LEOTOPPEL"
#define WIFI_PASSWORD "84351808"

static const int RXPin = 14, TXPin = 15;
String date_str , time_str, portaaux, difcelular;
char latstr[8], lngstr[8];
float lat_str , lng_str, latvelho, lngvelho, difvterminal1, difnterminal1, difvterminal2, difnterminal2;
int graus, minutos, segundos, centesimos, proximidade, porta, terminal1lat, terminal1lng, terminal2lat, terminal2lng;

TinyGPSPlus gps;

SoftwareSerial ss(RXPin, TXPin);
SSD1306Wire display(0x3c , D1, D2);
void setup()
{
  Serial.begin(9600);
  ss.begin(9600);
  display.init();
  display.flipScreenVertically();
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  Serial.println(F("DeviceExample.ino"));
  Serial.println(F("A simple demonstration of TinyGPS++ with an attached GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. "));
Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
  Serial.println();
}

void telainicial()
{
  //Apaga o display
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  //Seleciona a fonte
  display.setFont(ArialMT_Plain_24);
  //display.drawString(63, 10, );
  if ((difcelular.toInt()) < 100) {
    display.drawString(63, 20, difcelular);
  }
  else
    display.drawString(63, 20, "");
  //display.drawString(63, 45, "Display Oled");
  display.display();
```

```
}

void graficos()
{
  display.clear();
  //Desenha um quadrado
  display.fillRect(0, 0, 128, 64);
  display.display();
}
int n = 0;
int count = 0;
int count2 = 0;
void loop()
{
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("sem GPS"));
    while (true);
  }
}

void displayInfo()
{
  Serial.print(F("Localizacao: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    lat_str = gps.location.lat();
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
```

```
  lng_str = gps.location.lng();
}
else
{
  Serial.print(F("ERRO"));
}

Serial.print(F("  Date/Time: "));
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print(F("/"));
  Serial.print(gps.date.day());
  Serial.print(F("/"));
  Serial.print(gps.date.year());
}
else
{
  Serial.print(F("ERRO"));
}

Serial.print(F(" "));
if (gps.time.isValid())
{
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(F(":"));
  if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(F(":"));
  if (gps.time.second() < 10) Serial.print(F("0"));
  Serial.print(gps.time.second());
  Serial.print(F("."));
  if (gps.time.centisecond() < 10) Serial.print(F("0"));
```

```
  Serial.print(gps.time.centisecond());
 }
 else
 {
  Serial.print(F("ERRO"));
 }
 Firebase.setString("MCU1/latitude", dtostrf(lat_str, 8, 6, latstr));
 Firebase.setString("MCU1/longitude", dtostrf(lng_str, 8, 6, lngstr));
 terminal1lat = Firebase.getString("MCU1/terminal_1_lat").toInt();
 terminal1lng = Firebase.getString("MCU1/terminal_1_lng").toInt();
 terminal2lat = Firebase.getString("MCU1/terminal_2_lat").toInt();
 terminal2lng = Firebase.getString("MCU1/terminal_2_lng").toInt();
 Serial.println(terminal1lat);

 difvterminal1 = (1.852 * 60 * (terminal1lat - latvelho) * (terminal1lat - latvelho)) /
1000000 + (1.852 * 60 * (terminal1lng - lngvelho) * (terminal1lng - lngvelho)) / 1000000;
 difnterminal1 = (1.852 * 60 * (lat_str * 1000000 - terminal1lat) * (lat_str * 1000000 -
terminal1lat)) / 1000000 + (1.852 * 60 * (lng_str * 1000000 - terminal1lng) * (lng_str *
1000000 - terminal1lng)) / 1000000;
 difvterminal2 = (1.852 * 60 * (terminal2lat - latvelho) * (terminal2lat - latvelho)) /
1000000 + (1.852 * 60 * (terminal2lng - lngvelho) * (terminal2lng - lngvelho)) / 1000000;
 difnterminal2 = (1.852 * 60 * (lat_str * 1000000 - terminal2lat) * (lat_str * 1000000 -
terminal2lat)) / 1000000 + (1.852 * 60 * (lng_str * 1000000 - terminal2lng) * (lng_str *
1000000 - terminal2lng)) / 1000000;

 if ((difnterminal1 > difvterminal1) && (difnterminal2 < difvterminal2))
  Firebase.setString("MCU1/sentido", "horário");
 if ((difnterminal1 < difvterminal1) && (difnterminal2 > difvterminal2))
  Firebase.setString("MCU1/sentido", "anti-horário");

 porta = digitalRead(D6);
 if (porta == 1)
  portaaux = "1";
 if (porta == 0)
```

```
    portaaux = "0";
  Firebase.setString("MCU1/porta", portaaux);
  //Serial.println(porta);
  Serial.println(latvelho);
  Serial.println(lat_str * 1000000);
  proximidade = Firebase.getString("MCU1/proximidade").toInt();

  Serial.println(proximidade);
  Serial.println(difnterminal1);
  Serial.println(difnterminal2);
  difcelular = Firebase.getString("MCU1/diferenca");
  if (((gps.time.second()) + 5) > count) {
    count = gps.time.second();
    latvelho = lat_str * 1000000;
    lngvelho = lng_str * 1000000;
  }

  if ((difcelular.toInt()) < 200) {
    telainicial();
    if (proximidade > 0) {
      graficos();
      delay(400);
    }
  }
  telainicial();
  //display.clear();
  Serial.println(difcelular);

}
```

APÊNDICE 2 – CÓDIGO DO PROGRAMA ANDROID

```
package com.llagoza.projetofinalv2;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.speech.tts.TextToSpeech;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.Toast;

import com.google.android.gms.maps.model.LatLng;
import com.google.firebase.FirebaseApp;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.maps.android.SphericalUtil;
import com.llagoza.projetofinalv2.modelo.Onibus;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;

import static com.google.firebase.database.DatabaseReference.goOffline;
import static com.google.firebase.database.DatabaseReference.goOnline;


public class MainActivity extends AppCompatActivity {

    private TextToSpeech textToSpeech;
    private Context context;
    public String texto;
    private ImageButton btn;
    private int posicao;//caso exista duas linhas com mesmo nome precisa escolher um
sentido
    private List<Integer> arrayPosicoes = new ArrayList<Integer>();
    private String armazenaNomedaLinha;
    private String sentidodaLinhaOnibus;
    private int flagSelecao; //Para ver o nome da linha e o sentido;


    FirebaseDatabase firebaseDatabase;
```

```
    DatabaseReference databaseReference;

    double latFire; //latitude do Firebase
    double lonFire; //longitude do Firabase
    double distancia; //calcular a distancia do Firabase
    int flag; ///variavel da voz proximidade e reset
    int flagchegou;
    String difenca;

    private List<Onibus> listOnibus = new ArrayList<Onibus>();
//    private ArrayAdapter<Onibus>  arrayAdapterOnibus;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        flag = 0;
        flagchegou = 0;



//        listV_dados = (ListView) findViewById(R.id.listV_dados);
//        edtNome = (EditText) findViewById(R.id.edtNome);

        inicializarFirebase();
        eventoDatabase();
        checkPermission(); ///PERMISSAO DO MICROFONE CHECAR

        final SpeechRecognizer mSpeechRecognizer =
SpeechRecognizer.createSpeechRecognizer(this);

        final Intent mSpeechRecognizerIntent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
```

```
mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
        Locale.getDefault());
    mSpeechRecognizer.setRecognitionListener(new RecognitionListener() {
      @Override
      public void onReadyForSpeech(Bundle bundle) {

      }

      @Override
      public void onBeginningOfSpeech() {

      }

      @Override
      public void onRmsChanged(float v) {

      }

      @Override
      public void onBufferReceived(byte[] bytes) {

      }

      @Override
      public void onEndOfSpeech() {

      }

      @Override
      public void onError(int i) {

      }
```

```
            @Override
            public void onResults(Bundle bundle) {
                //getting all the matches
                ArrayList<String> matches = bundle
                        .getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);

                //displaying the first match
                if (matches != null) {
                    texto = matches.get(0);
                    comparacao();
                }
            }


            @Override
            public void onPartialResults(Bundle bundle) {

            }

            @Override
            public void onEvent(int i, Bundle bundle) {

            }
        });
//////////////////////////CONFIGURANADO A PARTE DO TEXTO//////////////////
        context = getApplicationContext();
        textToSpeech = new TextToSpeech(context, new TextToSpeech.OnInitListener()
{
            @Override
            public void onInit(int status) {
                if(status == TextToSpeech.SUCCESS){
                    textToSpeech.setLanguage(Locale.getDefault());
```

```
            }
        }
    });
////////////////////////////BOTAO TOUCH//////////////////
    findViewById(R.id.button).setOnTouchListener(new View.OnTouchListener() {

        @Override
        public boolean onTouch(View view, MotionEvent motionEvent) {

            switch (motionEvent.getAction()) {
                case MotionEvent.ACTION_UP:
                    mSpeechRecognizer.stopListening();
                    break;

                case MotionEvent.ACTION_DOWN:
                    //String fala  = "Fale normalmente";
                    //textToSpeech.speak(fala, TextToSpeech.QUEUE_FLUSH, null);
                    //delay(700);
                    if(flagSelecao == 0){
                        arrayPosicoes.clear();
                    }
                    mSpeechRecognizer.startListening(mSpeechRecognizerIntent);

                    break;
            }

            return false;
        }

    });


}
```

```
    private void comparacao() {
//        edtNome.setText(texto);
        Toast.makeText(getApplicationContext(),texto,Toast.LENGTH_SHORT).show();
        if(flagSelecao == 0){
            for (int i = 0; i < listOnibus.size();i ++) {
                if (listOnibus.get(i).getNomedalinha().equalsIgnoreCase(texto)) {
                    arrayPosicoes.add(i);
                    posicao = i;
                    textToSpeech.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    for (int j = posicao; j <listOnibus.size(); j ++){
                        if (listOnibus.get(j).getNomedalinha().equalsIgnoreCase(texto)) {
                            arrayPosicoes.add(j);
                        }
                    }
                    i = listOnibus.size();
                    armazenaNomedaLinha = texto.toLowerCase();
                    textToSpeech.speak("Informe o sentido da linha de ônibus",
                            TextToSpeech.QUEUE_FLUSH, null);
                    flagSelecao ++;

                } else {
                    String naoachei = "Linha não localizada";
                    textToSpeech.speak(naoachei, TextToSpeech.QUEUE_FLUSH, null);
                    flagSelecao = 0;
                }
            }
        }else if (flagSelecao == 1){

            for(int i = 0; i < arrayPosicoes.size(); i ++){
                if
(listOnibus.get(arrayPosicoes.get(i)).getSentido().equalsIgnoreCase(texto)) {
                    textToSpeech.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    armazenaNomedaLinha = texto.toLowerCase();
                    posicao = arrayPosicoes.get(i);
```

```
                i = arrayPosicoes.size();
                flagSelecao = 0;
                configurarServico();
            }else {
                String naoachei = "Sentido do ônibus não encontrado";
                textToSpeech.speak(naoachei, TextToSpeech.QUEUE_FLUSH, null);
            }
        }
    }


}

public void configurarServico(){
    try {
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        LocationListener locationListener = new LocationListener() {
            public void onLocationChanged(Location location) {
                atualizar(location);
            }

            public void onStatusChanged(String provider, int status, Bundle extras) { }

            public void onProviderEnabled(String provider) { }

            public void onProviderDisabled(String provider) { }
        };

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
locationListener);
    }catch(SecurityException ex){
        Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
```

```
        }
    }

    public void atualizar(Location location){
        Double latPoint = location.getLatitude();
        Double lngPoint = location.getLongitude();
        latFire = Double.parseDouble(listOnibus.get(posicao).getLatitude());
        lonFire = Double.parseDouble(listOnibus.get(posicao).getLongitude());


        LatLng posicaoInicial = new LatLng(latPoint,lngPoint);
        LatLng posicaoFinal = new LatLng(latFire,lonFire);

        distancia = SphericalUtil.computeDistanceBetween(posicaoInicial, posicaoFinal);

        difenca = listOnibus.get(posicao).getDiferenca();
        if(difenca.indexOf(".") !=-1){
            difenca = difenca.replace(".","");
        }

        if(distancia <= Double.parseDouble(difenca.replaceAll(",",".") )) {

databaseReference.child(listOnibus.get(posicao).getId()).child("diferenca").setValue(
String.valueOf(String.format("%,.2f", distancia)));
        }


        if(distancia <= 100.00){

            if(flag==0) {
            textToSpeech.speak("ônibus  chegando",  TextToSpeech.QUEUE_FLUSH,
null);
            flag = 1;
          }
```

```
        if(Integer.parseInt(listOnibus.get(posicao).getPorta()) == 0) {

databaseReference.child(listOnibus.get(posicao).getId()).child("proximidade").setVal
ue("1");
        }else
        if(Integer.parseInt(listOnibus.get(posicao).getPorta()) == 1){
            if (flagchegou == 0){
                textToSpeech.speak(
String.valueOf(listOnibus.get(posicao).getNomedalinha()+" "+"chegou !"),
                        TextToSpeech.QUEUE_FLUSH, null);

                flagchegou ++;


            }



databaseReference.child(listOnibus.get(posicao).getId()).child("proximidade").setVal
ue("0");
            if(flag ==1) {

databaseReference.child(listOnibus.get(posicao).getId()).child("proximidade").setVal
ue("0");

databaseReference.child(listOnibus.get(posicao).getId()).child("diferenca").setValue("
1000,00");
                posicao = listOnibus.size() - 1;
                latPoint = 0.0;
                lngPoint = 0.0;
//                exit(MainActivity.this);
                  doRestart(MainActivity.this);

                flag ++;
```

```
            }
        }
    }
}

public void exit(MainActivity view)
{
    android.os.Process.killProcess(android.os.Process.myPid());
}

public static void doRestart(Activity anyActivity) {
    anyActivity.startActivity(new Intent(anyActivity.getApplicationContext(),
MainActivity.class));
}

private void inicializarFirebase() {
    //inicialização Firebase
    FirebaseApp.initializeApp(MainActivity.this);
    firebaseDatabase = FirebaseDatabase.getInstance();
    databaseReference =  firebaseDatabase.getReference();
}

private void eventoDatabase() {
    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            listOnibus.clear();
            //Vai trazer informações por ordem que estao no dataBase
            for (DataSnapshot objSnapshot:dataSnapshot.getChildren()){
                Onibus onibus = objSnapshot.getValue(Onibus.class);
                listOnibus.add(onibus);
            }
//                                                  arrayAdapterOnibus = new
ArrayAdapter<Onibus>(MainActivity.this,android.R.layout.simple_list_item_1,listOnib
```

```
us);
//              listV_dados.setAdapter(arrayAdapterOnibus);
//                edtNome.setText(listOnibus.get(1).getTerminal_2_lat());

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
  }

  private void checkPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      if (!(ContextCompat.checkSelfPermission(this,
Manifest.permission.RECORD_AUDIO) ==
PackageManager.PERMISSION_GRANTED)) {
        Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
            Uri.parse("package:" + getPackageName()));
        startActivity(intent);
        finish();
      }
    }
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
      && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
    }}}
```

APÊNDICE 3 – CÓDIGO DO PROGRAMA ADROID – ARRAYLIST

```
package com.llagoza.projetofinalv2.modelo;

public class Onibus {
    private String id;
    private String nomedalinha;
    private String diferenca;
    private String latitude;
    private String longitude;
    private String porta;
    private String proximidade;
    private String sentido;
    private String terminal_1_lat;
    private String terminal_1_lng;
    private String terminal_2_lat;
    private String Terminal_2_lng;

    public Onibus(){

    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getNomedalinha() {
        return nomedalinha;
    }
```

```
public void setNomedalinha(String nomedalinha) {
    this.nomedalinha = nomedalinha;
}

public String getDiferenca() {
    return diferenca;
}

public void setDiferenca(String diferenca) {
    this.diferenca = diferenca;
}

public String getLatitude() {
    return latitude;
}

public void setLatitude(String latitude) {
    this.latitude = latitude;
}

public String getLongitude() {
    return longitude;
}

public void setLongitude(String longitude) {
    this.longitude = longitude;
}

public String getPorta() {
    return porta;
}

public void setPorta(String porta) {
    this.porta = porta;
```

```
}

public String getProximidade() {
    return proximidade;
}

public void setProximidade(String proximidade) {
    this.proximidade = proximidade;
}

public String getSentido() {
    return sentido;
}

public void setSentido(String sentido) {
    this.sentido = sentido;
}

public String getTerminal_1_lat() {
    return terminal_1_lat;
}

public void setTerminal_1_lat(String terminal_1_lat) {
    this.terminal_1_lat = terminal_1_lat;
}

public String getTerminal_1_lng() {
    return terminal_1_lng;
}

public void setTerminal_1_lng(String terminal_1_lng) {
    this.terminal_1_lng = terminal_1_lng;
}
```

```
    public String getTerminal_2_lat() {
        return terminal_2_lat;
    }

    public void setTerminal_2_lat(String terminal_2_lat) {
        this.terminal_2_lat = terminal_2_lat;
    }

    public String getTerminal_2_lng() {
        return Terminal_2_lng;
    }

    public void setTerminal_2_lng(String terminal_2_lng) {
        Terminal_2_lng = terminal_2_lng;
    }

    @Override
    public String toString() {
        return nomedalinha;
    }
}
```