CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

Rodrigo Dos Santos Nobrega

Android Auto
Estudo para o Desenvolvimento de Aplicações Embarcadas

Santo André – São Paulo

2016

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

Android Auto
Estudo para o Desenvolvimento de Aplicações Embarcadas

Monografia apresentada ao Curso de Tecnologia em Eletrônica Automotiva da FATEC Santo André, como requisito parcial para conclusão do curso em Tecnologia em Eletrônica Automotiva.

Orientador: Prof. Msc. Murilo Zanini Carvalho

Santo André – São Paulo

2016

FICHA CATALOGRÁFICA

N754e

Nobrega, Rodrigo dos Santos

Estudo para o desenvolvimento de aplicações embarcadas / Rodrigo dos Santos Nobrega. - Santo André, 2017. – 58f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Eletrônica Automotiva, 2017.

Orientador: Prof. Murilo Zanini de Carvalho

1. Eletrônica automotiva. 2. Aplicativo. 3. Veículos. 4. Sistemas embarcados. 5. Smartphones. 6. Software. I. Estudo para o desenvolvimento de aplicações embarcadas.

621.389

LISTA DE PRESENÇA

SANTO ANDRÉ, 09 DE JULHO DE 2016.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA **"ANDROID AUTO – ESTUDO PARA O DESENVOLVIMENTO DE APLICAÇÕES EMBARCADAS"** DO ALUNO DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. MURILO ZANINI DE CARVALHO _____

MEMBROS:
PROF. EDSON CAORU KITANI _____

PROF. FERNANDO GARUP DALBO _____

ALUNO:

RODRIGO DOS SANTOS NÓBREGA _____ Rodrigo S. Nóbrega

DEDICATÓRIA

Dedico este trabalho a minha família e aos meus amigos que sempre estiveram próximos durante esta jornada.

AGRADECIMENTOS

Gostaria de agradecer a todos aqueles que direta e indiretamente contribuíram para a realização deste trabalho e principalmente a minha família, namorada e aos colegas de sala que mantiveram estímulos nos momentos mais árduos desta jornada.

Agradeço aos professores Murilo Zanini e ao Fernando Garup pelo apoio na decisão do tema e na realização deste trabalho, os bons professores que Fatec Santo André possui e aos colaboradores e funcionários que sempre estiveram prontos a nos ajudar.

"Não sei como posso parecer aos olhos do mundo, mas quanto a mim, vejo-me apenas como um menino brincando na praia e me divertindo em encontrar de quando em quando um seixo mais liso ou uma concha mais bonita, enquanto o grande oceano da verdade jaz incógnito à minha frente."

Isaac Newton

RESUMO

Atualmente o mundo gira através de seus *smartphone*s; empresários, professores, médicos todas as pessoas possuem um, se não quando mais que um, a todo o momento são disparadas mensagens, e-mails, ligações com finalidades diferentes, seja a trabalho ou para ouvir a voz de uma pessoa querida; depositamos nossa vida nos celulares, eles têm datas queridas, mensagens importantes, arquivos salvos, controle remoto para TV e cada dia surgem novas aplicações. Chegamos ao ponto que as pessoas não conseguem viver sem o celular, toda inovação traz consigo pontos positivos e negativos, e o ponto negativo fez com que as empresas fizessem uma analise, chegando ao ponto que viram que o uso do celular reduz a atenção do motorista e com isso causando acidentes, a fim de reduzir esse número uma empresa, na qual eu foquei em realizar meu trabalho de conclusão, investiu na criação de um aplicativo para momentos em que se estiver conduzindo um veículo para maior segurança e conforto.

Palavras chaves: *smartphone*, Android auto, sistema Android embarcado no automóvel.

ABSTRACT

Currently the world turns through their *smartphone*s; businessmen, teachers, doctors, all people have one, if not when more than one, all the time messages are triggered, emails, links with different purposes, either to work or to hear the voice of a loved one; we place our life in the cell, they have cherished dates, important messages, saved files, remote control for TV and each day brings new applications. Got to the point that people can not live without your cell phone, every innovation brings positives and negatives, and negative caused companies to do an analysis, to the point that they saw that cell phone use reduces the driver's attention and thereby causing accidents in order to reduce this number one company in which I focused in carrying out my term paper, invested in creating an application for times when you are driving a vehicle for greater safety and comfort.

Key words: *smartphone*, Android auto, Android embedded system in the car.

LISTA DE ABREVIATURAS

AA: Android Auto

DHU: Desktop Head Unit

OAA: Open Auto Alliance

TCC: Trabalho de Conclusão de Curso

ADB: Android Debug Bridge

SO: Sistema Operacional

OHA: Open Handset Alliance

INC.: Incorporation

ABS: Anti-lock Breaking System

MIDI: Musical Instruments Digital Interface

USB: Universal Serial Bus

CAN: Controller Area Network

OBD: On-Board Diagnostic

APP: Application

ID: Identification

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Por volta do ano de 2000 boa parte das pessoas tinha um celular que era muito utilizado para realizar ligações, mensagens de texto e possuía uma internet muito limitada, o celular tinha um visor que mostrava as informações porem de difícil interação para com o usuário.

No ano de 2003 Andy Rubin, Rich Miner, Nick Sears e Chris White fundaram a Android Inc., fruto de projetos que envolvia um sistema de segurança mais inteligente, começaram a desenvolver sistemas operacionais para celulares, definindo um novo conceito de celulares, os *smartphones*, "*celulares mais inteligentes e que estejam mais cientes das preferências e da localização do seu dono*".

Após dois anos no mercado de celulares a Google comprou a Android Inc., criando uma divisão do setor *mobile*, a Google manteve o lema do Android Inc. em deixar o sistema operacional em código aberto para que qualquer pessoa pudesse começar a produzir aplicativos, como estratégia para ganhar o mercado a Google criou programas de incentivos para programadores desenvolverem em Android. Em meado de 2014 a empresa do vale do silício comemorava 1 bilhão de celulares com Android, 1 milhão de aplicativos e 50 bilhões de *downloads*.

A revolução dos *smartphones* atingiu todos os setores possíveis e hoje em dia dizemos que o celular faz "até chamadas" e não conseguimos mais sair de casa sem nosso *smartphone*. Tal revolução causou certo aumento da falta de atenção ao nos locomovermos, seja a pé ou de carro. Esse fato aumentou o número de acidentes veiculares causando avaliações negativas para as empresas de dispositivos móveis Para rebater tais criticas as empresas resolveram implementar as centrais multimídias criando um sistema de espelhamento do celular para o carro, capaz de reproduzir suas musicas, atender ligações através do viva voz.

O Android Auto chega para substituir as versões de espalhamento de celulares que já se encontram ultrapassadas. Ele aproveita das funções existentes e coloca em seu funcionamento, um sistema melhorado. O Android Auto é um aplicativo e não um sistema operacional, um aplicativo com uma interface conhecida pelas pessoas, de fácil interação com

o usuário, mas que ao fazer o “espelhamento” com o veiculo bloqueia alguns aplicativos, porém torna outros mais otimizados, como o serviço de navegação, *whats’app* comandado totalmente por voz, assim como recebimento de mensagens, um menu com ultimas ligações.

O Android Auto faz parte de mais uma fase da revolução que vivemos na tecnologia, trazendo consigo um grupo de benefícios. Acredita-se que o Android Auto veio apenas para reduzir o numero de acidentes causados por falta de atenção no trânsito, mas isso é apenas o inicio de uma nova tecnologia embarcada nos sistemas automotivos.

1.1 Objetivos e motivação

Dado o cenário apresentado anteriormente foi observado que o uso de *smartphones* cresce de forma contínua, tornando-se um mercado em potencial para o desenvolvimento de soluções embarcadas. Este trabalho tem como objetivos:

- Apresentar um novo método de multimídia que requer menos atenção/interação do motorista com o celular a fim de diminuir o número de acidentes causados por falta de atenção;
- Apresentar uma nova extensão do Android voltada para automóveis; realizar um estudo sobre o Android e suas características, pois para cada tipo de trabalho existe uma ferramenta certa que pode ser o que esteja faltando nos novos conceitos de veículos;
- Finalmente, apresentar as novas tendências do mercado.

1.2 Justificativas

Esse trabalho tem como principal finalidade apresentar e demonstrar estudos relacionados com a plataforma Android Auto, demonstrando que tal tecnologia é capaz de reduzir o numero de acidentes envolvendo automóveis devido à falta de atenção do motorista.

Partindo deste ponto, este trabalho busca apresentar soluções do que se pode ser feito com essa nova área e desenvolver um aplicativo que trabalhe entre a central multimídia e o celular.

1.3 Conteúdo

Este trabalho está assim dividido; o capitulo 2 apresenta a fundamentação teórica com maior ênfase na parte de eletrônica embarcada, sistemas operacionais para sistemas embarcados, *Android*, *Open Auto Aliance*, *Android Auto*; e assim seguindo para os capítulos 3 com matérias e métodos para desenvolvimento do TCC, capitulo 4 com resultados alcançados, capitulo 5 com considerações finais e capitulo 6 para referencias.

2 FUNDAMENTAÇÃO TEÓRICA

Ao longo deste capítulo, são apresentados os principais conceitos referentes ao desenvolvimento de aplicativos para a plataforma Android, bem como a história desse sistema operacional móvel. Dispositivos de monitoramento embarcado também são objetos de estudo deste trabalho. Um breve levantamento sobre sua utilização no mercado é apresentado.

2.1 Eletrônica Embarcada

Guimarães (2007) ressalta que a Eletrônica embarcada é a eletrônica desenvolvida para diversos setores do nosso meio ambiente, setores como, por exemplo: Automotivo, Aeroespacial, Agrícola, Naval e Móvel, dentro outros. Para este trabalho realizaremos uma integração dos setores **Automotivo e Móvel**, fornecendo uma aplicação móvel, com a finalidade de aproveitar a maior interação/facilidade do usuário com o *smartphone* integrando-o com o automóvel.

A eletrônica embarcada no sistema automotivo surgiu quase junto com o automóvel, nos primórdios, o veiculo era basicamente composto por engenharia mecânica e uma pequena parte por engenharia elétrica, com o tempo vimos que a parte mecânica estava ficando obsoleta e o automóvel precisava melhorar, as pessoas pediam mais conforto, a concorrência obrigava buscar novas melhorias e, contudo surgiram leis para restringir o aumento de emissões; para isso as montadoras foram atrás de uma tecnologia que começou a surgiu, a eletrônica embarcada.

Guimarães (2007) cita que a Eletrônica Embarcada foi dividida em subtemas, sendo eles: Arquiteturas Elétricas, Protocolos de Comunicação, Diagnóstico Veicular e Compatibilidade Eletromagnética. O Aprimoramento, avanço da eletrônica nos veículos se deu através de leis ambientais, era preciso ter um melhor controle do resultado da queima do combustível e para isso foi incrementado o método *On Board Diagnostic(OBD).*

Quando se trata de veículos, é preciso ter muito cuidado ao lançar um modelo sem testes, pois o carro é um equipamento de risco, podendo até matar pessoas. A fim de oferecer maior conforto, estabilidade, potencia, segurança, dentre outras coisas. A eletrônica começou

a ser mais usada dentro dos veículos e assim o automóvel ganhava módulos, como por exemplo: injeção eletrônica, freios ABS, sistemas de tração, *air bags* e muito mais.

A grande revolução que o setor automotivo vem sofrendo é de grande parte da área da eletrônica, sendo que nem tudo que existe de inovador na eletrônica ainda esta sendo utilizado nos veículos. A introdução da eletrônica ganhou confiança e com isso foram implementados sistemas de entretenimento e conforto nos veículos, aumentando o campo para desenvolvimento de novas tecnologias como o Android Auto.

2.2 Sistemas Operacionais para Sistemas Embarcados

O Sistema Operacional é um conjunto de programas, sub-rotinas, utilizadas para gerenciar os recursos de um sistema fornecendo uma interface entre o computador e o usuário.

Com o avanço da tecnologia novos sistemas operacionais apareciam no mercado, nada ainda voltado para o setor automotivo, ganhando os usuários por suas *interfaces* e com um alto poder de processamento de dados, aumentando a segurança e confiabilidade do sistema, tornando possível sua expansão para diversos setores.

No momento, o que está sendo debatido no ramo da tecnologia veicular é se o uso de sistemas operacionais dentro dos veículos seria capaz de funcionar sem oferecer riscos aos usuários. Empresas do setor automotivo estão criando alianças com empresas de *smartphones* para juntos poderem realizar pesquisas sobre sistemas operacionais embarcados nos veículos. Um grande passo que está sendo dado é o espelhamento do *smartphone* com o veiculo, onde se pode criar uma *interface* através da central multimídia.

2.3 Android

Monteiro (2013) aponta que umas das tecnologias que mais está sendo usada é o Android, sendo utilizadas em celulares, tablets, geladeiras, televisões, vídeo games e também em automóveis. Com uma *interface* otimizada e de fácil compreensão para as pessoas. O Sistema Operacional Android conquistou o mercado.

Tudo começou há 13 anos na cidade de Palo Alto na Califórnia, em 2003, Andy Rubin, Rich Miner, Nick Sears e Chris White, todos empresários já iniciados no ramo da tecnologia fundaram a Android Inc.. No inicio Andy Rubin definiu os celulares como "*Dispositivos móveis mais inteligentes e que estejam mais cientes das preferências e da localização do seu dono*".

O Android Inc. ofereceu ao mercado um novo meio de sistema operacional móvel, um Sistema Operacional *Open Source*, baseado no *Kernel Linux*. A ideia foi lançar uma *interface* simples, funcional e que integra vários instrumentos, oferecendo um sistema gratuito e simples para desenvolvedores.

Rapidamente a empresa chamou a atenção de uma empresa grande. Então com apenas dois anos no ramo, a Android Inc. foi vendida para o Google, e assim em 2005 a Google iniciou uma nova divisão da empresa, o "*Google Mobile Division*", uma divisão de pesquisas em tecnologia móvel.

Monteiro (2013) ressalta que com o mercado mobile, causando certa desconfiança e duvidas na época, já que muitas pessoas achavam difícil uma competição direta com o Windows Mobile, da Microsoft, e o IOS, da Apple, e ainda empresas como *Symbian*. A Google conseguiu fechar parcerias com empresas de *softwares* e *hardwares*. Lançando seu primeiro protótipo em 2006, o "Sooner".

E nessa transição toda, a Google manteve Andy Rubin como responsável pelo setor mobile da empresa, mantendo a politica que se iniciou no Android Inc., como: *Código aberto, gratuito, fácil interface, mercado aberto para desenvolvedores*.

2.3.1 Evolução da Tecnologia

Monteiro (2013) diz que a história do Android no mercado *mobile* começa mesmo em 2006 quando a Google anuncia seu primeiro protótipo, o "Sooner", um celular parecido com um BlackBerry, não tendo tela *touchscreen* e operando através de um teclado *QWERTY*. Em 2006 e em 2007 o mercado *mobile* acabava de sofrer uma mudança, concorrentes como a LG e a Aplle lançavam o LGPrada e o IPhone com tela *touchsreen*.

O Sistema Operacional Android ganhou mercado em 2007 quando fabricantes como *Samsung*, *Sony*, *HTC*, operadoras americanas como *Sprint Nextel* e *TMoblie*, e *Texas Instrumentes*, além do próprio Google, reuniram-se em um consorcio de tecnologia e fundaram a *Open Handset Alliance*. Tendo como objetivo a criação de uma plataforma de código aberto para *smartphone*s, e o resultado foi o primeiro Sistema Operacional Android comercial do mercado, rodando em um *HTC Dream*, lançado no final de 2008. Nessa época o conceito de *smartphone*s foi remodelado e para atrair o mercado o Google ofereceu um prêmio de 10 milhões de dólares aos desenvolvedores que conseguissem projetar os melhores aplicativos, levando a primeira versão do *Android SDK* pública, Lecheta (2010).

Monteiro (2013) diz ainda que com o tempo o Android ganhou o mercado e começaram a projetar outras versões, todas as continuações estão em ordem alfabética e têm nome de doces exceto duas, as versões 1.0 e 1.1 conhecidas como *Astro(Alpha)* e *Battenberg(Beta)*, respectivamente.

2.3.2 Android Alpha e Beta (2007 ~ 2008)

A primeira foi utilizada somente pelos componentes da OHA e era chamada por nomes de robôs, como *Astro Boy*, *Bender* ou *R2-D2*. Já a versão *beta* foi a primeira a ser disponibilizada ao público, tendo um total de seis versões oficiais publicadas.

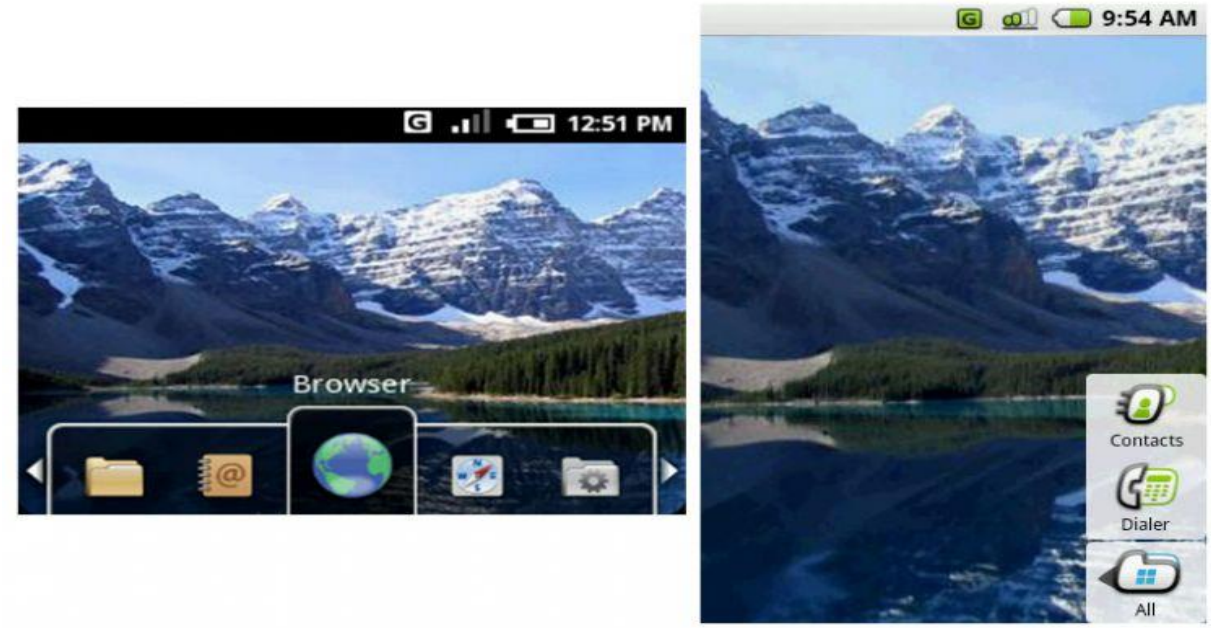

Figura 1– Interface do primeiro SO Android

Fonte: https://www.oficinadanet.com.br/post/13939-a-historia-do-android - Acessado em 24/03/15

2.3.3 Android 6.0 – Marshmallow (2015)

Lançado há pouco tempo e presente em apenas 0,3% dos aparelhos (dados de dezembro de 2015) o *Marshmallow* trouxe, entre outras, as seguintes melhorias:

- *Now on tap*: recurso que contextualiza o *Google Now* nos aplicativos com um toque no botão home;
- Modo Doze: recurso que economiza a bateria do dispositivo automaticamente quando em stand-by;
- Gaveta de aplicações na vertical, com busca alfabética;
- Barra de busca de aplicativos na gaveta de aplicações e destaque para aplicativos favoritos;
- Suporte nativo para leitores de impressão digital;
- Melhorias para compartilhamento de conteúdo entre aplicativos;
- Modo Não Perturbe;
- *Links* para apps para definição de abertura padrão de links em seus respectivos apps;
- Suporte para pastas de apps grandes, com páginas;
- Suporte para autorização de permissões de aplicativos sob demanda;
- Suporte para *USB Type-C*;

- *Backup* e restauração automática no *Drive* para dados e aplicativos;
- Modo de tela 4K para apps;
- Adaptação de memória externa (*SD cards*) como parte da memória interna;
- Suporte para *MIDI* em instrumentos musicais;
- Suporte experimental para visualização em multi-janela (habilitável somente em aparelhos *rooteados*).

Fonte: https://www.oficinadanet.com.br/post/13939-a-historia-do-android - Acessado em 24/03/15

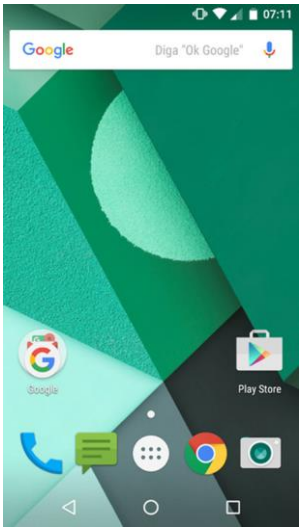

Figura 2– Interface do SO Android 6.0

Fonte: https://www.oficinadanet.com.br/post/13939-a-historia-do-android - Acessado em 24/03/15

A figura 2 ilustra a tela do Android *Marshmallow*. A figura 3 apresenta o gráfico dos Androids em utilização na primeira semana de dezembro de 2015:

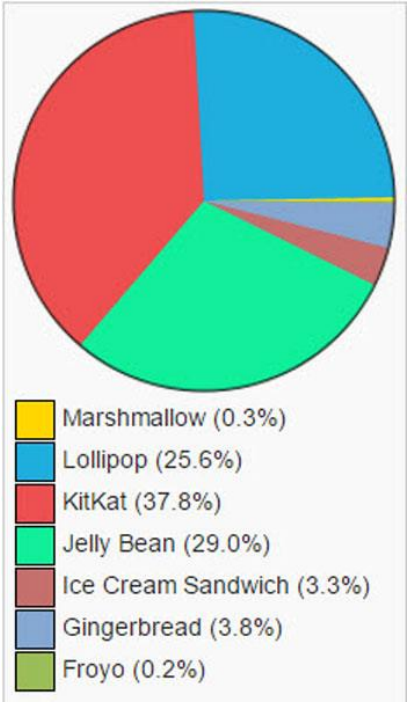

Figura 3– Grafico mostrando a quantidade de cada SO Android.

Fonte: https://www.oficinadanet.com.br/post/13939-a-historia-do-android - Acessado em 24/03/15

Já a figura 4 apresenta a evolução do Android de 2011 á 2015.

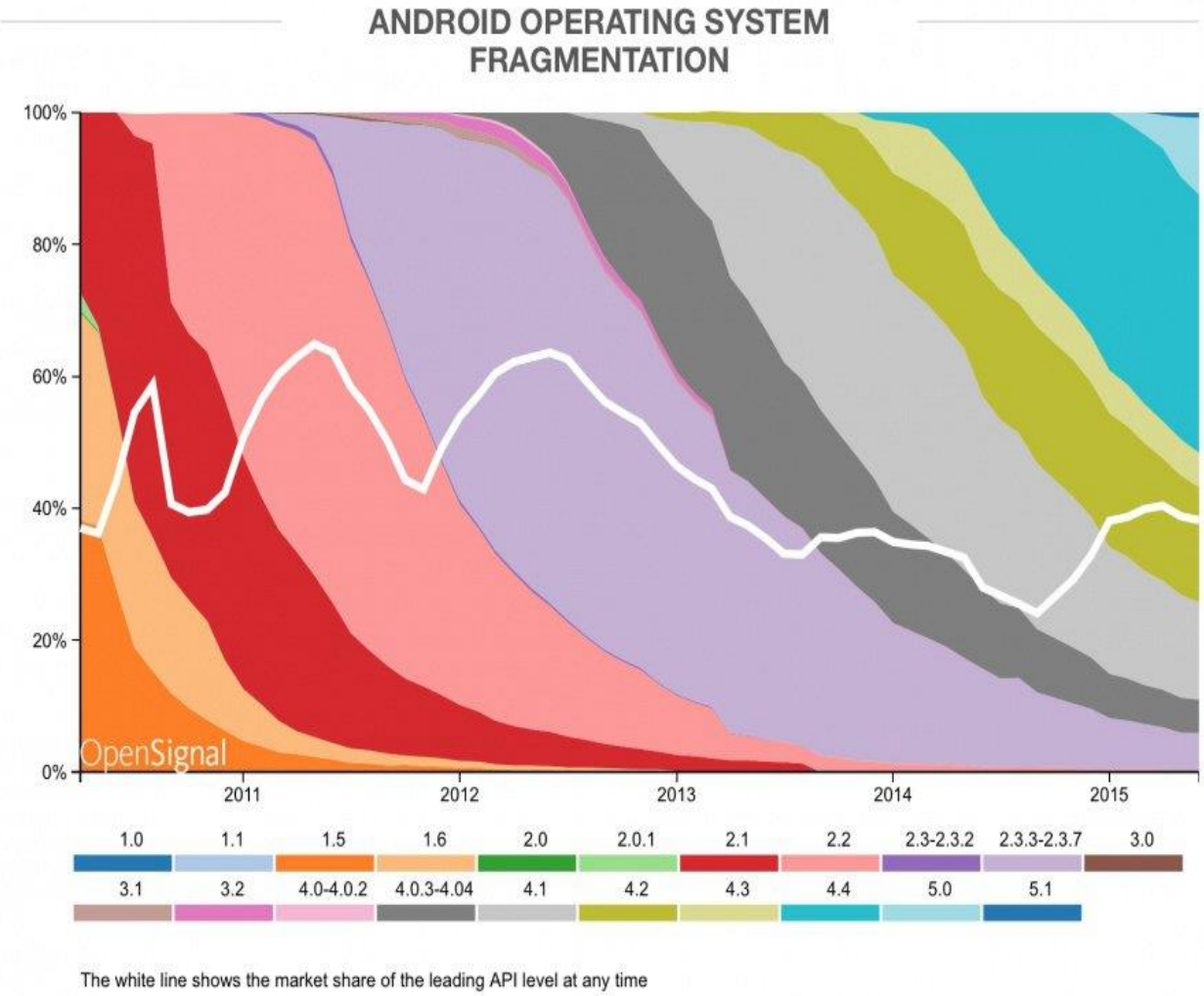

Figura 4– Grafico mostrando a evolução do Android

Fonte: http://www.tudocelular.com/android/noticias/n58941/android-fragmentacao-lollipop.html- Acessado em 14/07/16

2.3.4 Características principais do sistema

Monteiro, (2013) cita que o Android é uma plataforma composta de um sistema operacional, *middlewares* e um conjunto de aplicativos principais como os Contatos, Navegador de Internet e o Telefone propriamente dito.

O sistema operacional do Android foi desenvolvido baseando se no *Kernel* do *Linux*, tendo como regra ser "*Open Source*" para os desenvolvedores. A ideia do Android é possuir uma *interface* simples e funcional. Segundo "O que é Kernel?" (AMARALEM, 2009).

2.4 Open Auto Alliance

2.4.1 História da Open Auto Alliance

Antes da Google lançar sua plataforma Android no mercado foram realizadas alianças para desenvolver as alternativas para o mercado mobile e assim foram feitas pesquisas para então conseguirem desenvolver seu primeiro celular. A aliança mostrou que foi possível trabalhar com diversas empresas e todas falando a mesma língua, a da Tecnologia.

Hoje, temos um mercado automotivo, que até pouco tempo estava fechado para as áreas de informática, TI, etc. Talvez por receio de integração ou por se achar competente na sua determinada área, mas o que o mercado automotivo muitas vezes esquece é que já foi preciso em tempos atrás fazer uma integração de outras áreas como a da mecânica e da elétrica/eletrônica. Por outro lado temos o mercado de telecomunicações, informática e TI que estão expandido suas raízes para outras áreas, como por exemplo: eletrodomésticos, saúde. A união de diversos setores favorece o usuário e como resultante temos o avanço da tecnologia.

A OAA é a união de marcas de diversas áreas, como montadoras de carros, empresas de componentes eletrônicos, centrais multimídias; buscando melhorar seus produtos a fim de melhorar seus índices de vendas e desenvolver novos produtos. Essa união determinou o

nome da próxima tecnologia envolvida no ambiente automotivo, conhecida como Android Auto, uma *interface* de fácil interação do usuário envolvendo seu *smartphone*, a central multimídia e seu automóvel, expondo as funcionalidades do *smartphone* do usuário na central multimídia do automóvel.

"A OAA tem como objetivo acelerar a inovação com uma abordagem que oferece a abertura, a personalização e a escala, princípios-chave que já fizeram Android uma parte familiar de milhões de vidas das pessoas. Este modelo de desenvolvimento mútuo permitirá que os fabricantes de automóveis possam trazer tecnologia de ponta para os seus condutores, e criar novas oportunidades para os desenvolvedores oferecerem experiências poderosas para condutores e passageiros de forma segura e escalável." (http://www.openautoalliance.net/#press 16/04/2016).

2.4.2 Concorrência no Mercado Automotivo

A OAA é a união de empresas para ajudar no desenvolvimento de nova tecnologia, não dispensando que cada empresa desenvolverá sua própria inovação, fazendo uma comparação com os produtos de interatividade disponíveis no mercado podemos notar algumas diferenças:

FIAT: -"UConnect".

O sistema UConnect é o que o grupo FCA apresenta de mais novo no mercado brasileiro, possui uma plataforma própria da empresa que conta com sistema de comunicação via *bluetooth*, do celular com a central multimídia, equipada com sistema de navegação, tela *touch*, comandos pelo volante ou voz e aceita reprodução de músicas tanto de Android quanto Apple.(http://www.fiat.com.br/conectividade/u-connect.html 18/07/2016).

FORD: -“FORD SYNC”,-“Projeto MOOBI”.

A Ford apresenta no mercado brasileiro a plataforma *FORD SYNC*, sistema equipado com comunicação via *bluetooth*, tela *touch*, comandos pelo volante ou por voz, podendo reproduzir músicas de *smartphones*, a plataforma conta com uma integração com o App Link, expandindo alguns aplicativos para o modo da central multimídia, algumas versões de automóveis estão saindo com o Ford DOCK, seria um estojo para celular, onde prenderia o aparelho e conseguiria visualizar sua tela. Fora do mercado a Ford em parceria com a Intel está buscando desenvolver um sistema de segurança para o veículo que será possível saber quem está dirigindo seu carro e onde ele está, apresentado como Projeto Moobi. (http://www.ford.com.br/ford/applink/fordsync 18/07/2016).

GM:-“Mylink”

A GM conta no mercado brasileiro com a nova versão do sistema MyLink, sistema que possui: navegação, tela *touch*, integração com *smartphones* através do Android Auto e Car play, reprodução de músicas via *bluetooth*, controles pelo volante ou voz. (http://www.chevrolet.com.br/universo-chevrolet/mylink-nova-geracao.html 18/07/2016).

VW:- “App Connect”

A VW apresenta para o mercado brasileiro sua plataforma, o App Connect, sistema multimídia que possui: serviço de chamadas, enviar mensagens por comando de voz, checar agenda, acessar aplicativos, integração com *smartphones* através do Android Auto, Car play e Mirror Link, aplicativos próprios da VW, entradas SD Card e através de modem de *internet* é possível criar uma rede Wi-Fi no veículo.

2.5 Android Auto

2.5.1 Descrição e Configurações do Android Auto

A mais nova tecnologia sobre a integração *mobile* e automotivo, lançado ao mercado no ano de 2015, o Android Auto veio para substituir o modo com que usamos nosso *smartphone*, sem a necessidade de rotacionar sua tela, sacar o celular do suporte toda vez que o caminho mudar, atender o celular ou enviar mensagens de texto digitadas, procurar seu aplicativo de áudio dentre vários aplicativos, levar multa por estar dirigindo e utilizando o celular ao mesmo tempo; seu modo de viver e interagir dentro do carro passará a mudar a partir do momento em que começar a usar o Android Auto.

Considerações para funcionamento do Android Auto:

- Necessário possuir um celular com Android 5.0 ou maior;
- Conexão via *USB* com a central multimídia;
- Central multimídia que consiga reproduzir o Android Auto.

Para ajudar na condição de motorista ou de passageiro, o conceito principal do Android Auto é "*No Hands*", ser possível de usar todas as funções do seu celular apenas por comandos de voz, deixando assim as mãos diretas no volante, assim o motorista consegue enviar mensagens para seus contatos ou escolher seu *player* de musicas com comandos de voz, caso preferir ainda é possível utilizar reprodução do celular através da central multimídia e escolher suas funções através do "*touch*", lembrando que quando o celular entra no modo Android Auto ele restringe algumas funções do aparelho *smartphone* para garantir segurança na condução.

A Google simplesmente não só desenvolveu o Android Auto mas também uma plataforma para tal, fazendo parte do novo conceito de plataforma de desenvolvimento, os "*Weareble*". Quem desenvolve para Android irá sentir uma diferença em comparação com o

Android Auto (AA), pois em razão da segurança dos passageiros, muitas funções do Android estão restritas, permitindo apenas dois tipos de serviços no AA:

- Serviços de mensagens, notificações;
- Serviços de *players* de musicas.

Foi preciso usar a criatividade e relacionar o mercado mobile e automotivo para desenvolver para essa nova plataforma, lembrando que o Android Auto é um aplicativo e uma plataforma para desenvolvimento, para execução e testes (no DHU) de seu aplicativo será necessário o aplicativo do Android Auto para entrar em modo de desenvolvedor. Para expansão de grandes aplicativos conhecidos a Google facilitou uma operação de extensão dos aplicativos para os carros, conhecido como "*Car extender*" é possível utilizar no Android Studio (software de desenvolvimento Android).

Como não poderia ficar de fora do mercado, as concorrentes do setor mobile entraram na corrida pela integração do celular com o carro, após o lançamento do Android Auto a Apple lançou o "*Car Play*" com o mesmo padrão que o Android Auto, criando um espelhamento do serviço *mobile* e utilizando comandos de voz para utilização; em seguida a Microsoft lança o "*Mirror Link*" com o serviço de espelhamento.

Através da aliança "*Open Auto Alliance*", empresas do setor automotivo começaram a desenvolver seus produtos para utilização do Android Auto, incrementando com sistemas mais apurados para ajudar na interpretação do Android Auto ou serviços de busca próprios das empresas.

No Brasil, o primeiro carro a chegar com o sistema embarcado Android Auto foi o *Fox* da VW na metade de 2015. Em menos de um ano, veículos de diversas marcas saíram com os modelos parecidos como itens de série dos novos carros. (http://www.androidpit.com.br/android-auto-volkswagen-fox 18/07/2016)

O *AUTOMATE* é praticamente uma simulação do Android Auto, porem sem a necessidade de precisar de uma central multimídia ou do DHU, sendo apenas um aplicativo e

podendo ser baixado em qualquer dispositivo Android. Possui uma *interface* gráfica praticamente igual ao Android Auto, mas com uma maior liberdade de funções permitidas por ser apenas um aplicativo, o *Automate* está em uma versão *beta* grátis mas já é possível comprar a versão *Premium*.

2.5.2 Concorrente do Android Auto

Assim como a Google entrou no mercado automotivo com o Android Auto suas concorrentes do mercado *mobile* resolveram expandir seus sistemas para os veículos também.

O Car Play principal concorrente do Android Auto, pertencente a Apple, possui finalidades parecidas, é um sistema que permite expandir os aplicativos para a central multimídia, tendo uma tela padrão e possuindo um serviço de comandos de voz através da Siri para ligações, mensagens e outros serviços. (http://www.apple.com/br/ios/carplay/ 18/07/2016)

Mirror Link, o sistema de espelhamento de *smartphone* que está há mais tempo no mercado, desenvolvido pela Nokia em parceria com o grupo *Consumer Eletronics for Automotive* (CE4A).

Para basear a concorrência em dados, foi realizado dois gráficos:

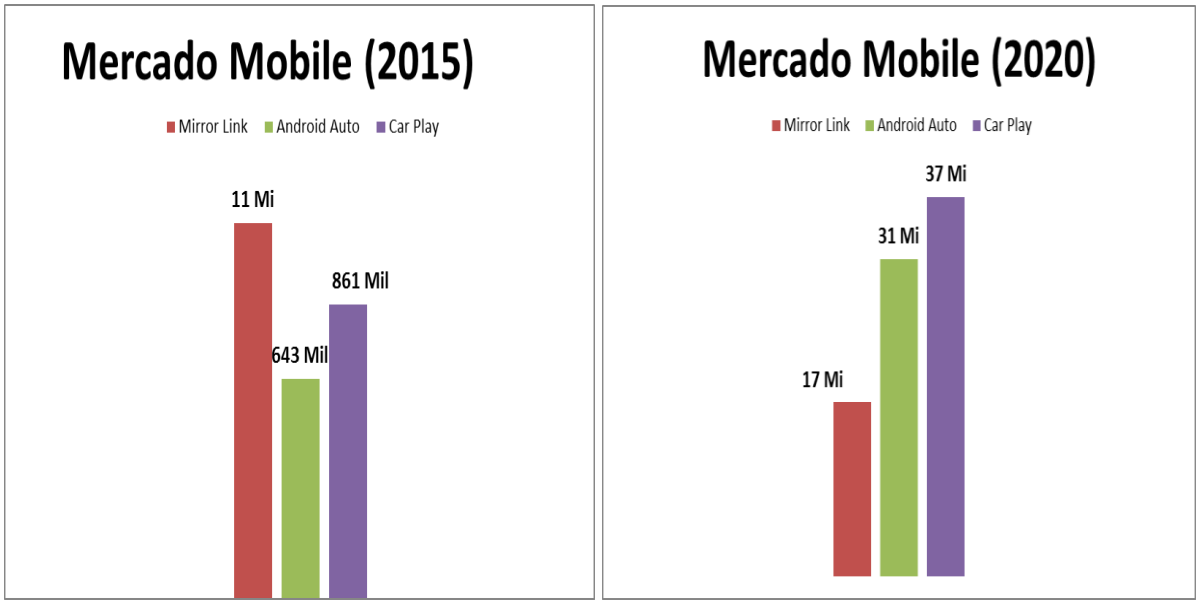

Figura 5- Gráfico da concorrência dos serviços de multimídia. Fonte: IHS Automotive.

Segundo os dados da IHS Automotive, o Mirror Link que atende a mais tempo no mercado possui um número maior dispositivos, porém com a projeção para 2020 estima-se que o número de dispositivos com o Car Play seja maior, seguido pelo Android Auto.

3 MATERIAIS E METÓDOS

Durante este capítulo, é apresentado o desenvolvimento do aplicativo "*4Car*", proposto como resultado do estudo realizado das tecnologias existentes no mercado e das possibilidades levantadas durante a revisão bibliográfica do tema.

O "*4Car*", estudo de aplicação para o Android Auto desenvolvido no projeto, utiliza o serviço disponível de mensagens e notificações do Android Auto para realizar uma comunicação entre o usuário/motorista, o automóvel e o *smartphone*. É possível que com essa interação dos demais, seja feita uma "Conversa", sendo possível o celular receber informações do automóvel, como avarias, períodos de revisão, informações de velocidade, e mostrar essas informações para motorista a fim de evitar acidentes.

A ideia é o "4Car" funcionar junto com o Android Auto, aproveitando todas as funções que o Android Auto permite. Para funcionamento são necessários:

- Aplicativo do Android Auto;
- Aplicativo do "4Car";
- Central multimídia que emule o Android Auto;
- Conexão via USB do celular com a central multimídia;
- Hardware ELM237 via *Bluetooth*.

3.1 Materiais de Desenvolvimento

Em muitos casos de trabalhos o alto custo de desenvolvimento o torna inviável, neste projeto de conclusão, conseguimos utilizar os materiais disponíveis e que atendiam as especificações necessárias, deixando o custo em praticamente zero.

Da parte dos materiais utilizados, destacam-se: plataformas de desenvolvimento como o Android Studio, simuladores do Android Auto, *smartphone Samsung S5* com Android 5.0, hardware *ELM237 via Bluetooth* exclusivo para aplicação do "4CAR". ELM237 conhecido como um escâner automotivo que extrai as informações, códigos de avarias via protocolo CAN e envia via Bluetooth para o celular.

3.1.1 Android Studio

Android Studio é um software de programação especializado para linguagem Android, que inclui linguagens como JAVA e XML, após a primeira utilização do mesmo é possível notar que é um software de programadores para programadores, por sua *interface*, funções de auto completar, dicas toda vez que inicia uma nova sessão, dentre outros artifícios. A integração das linguagens de programação, JAVA e XML, permitem ao programar em dois tipos de linguagens, por código e blocos de desenho, facilitando a percepção do programador ao que o projeto necessita.

3.1.2 Android SDK

Android SDK é uma ferramenta do Android Studio que nos permite realizar testes do aplicativo a ser desenvolvido podendo utilizar o aparelho celular ou explorar opções de simuladores, é possível simular a tela de aparelhos móveis ajustando as opções gráficas e de programação, segundo Lecheta (2010).

3.1.3 API Android Utilizada

Lecheta (2010) cita que a API Android é a versão da plataforma do Android no *smartphone*, para uso do Android Auto, é preciso que API seja a 21 com o android 5.0 ou maiores, no caso são os celulares mais novos.

Api’s que podem ser utilizadas para o serviço Android Auto e suas configurações são:

- Android 5.0(5.0.1 e 5.0.2)
- Android 5.1(5.1.1) – Lollipop (2015)
- Android 6.0 – Marshmallow (2015)

3.1.4 “4CAR”

3.1.4.1 Motivação e Justificativa

A motivação para pesquisa e desenvolvimento deste aplicativo surgiu diante do alto número de pessoas que estão adquirindo um automóvel e não possuem conhecimento no grau de necessidade de uma manutenção periódica.

A manutenção, sendo veicular ou não, é dividida em três:

- Preventiva: Seria uma manutenção programada, como troca de óleo, ela utiliza o tempo de vida de alguns componentes para poder realizar uma troca, evitando falhas futuras;

- Preditiva: Manutenção referente a substituição de peças através de testes;

- Corretiva: Este tipo de manutenção é realizado quando um ou mais componentes, peças deixam de funcionar precisando trocar.

“A manutenção preventiva é no mínimo 40% mais barata que a corretiva”, segundo José Palacio, auditor técnico do Instituto de Qualidade Automotiva (IQA). [Disponível em: http://g1.globo.com/Noticias/Carros/0,,MUL155595-9658,00.html 18/07/2016].

O cuidado com o tempo de uma revisão muitas vezes é esquecido pelo usuário do veículo, onde para ele o que importa é o carro estar andando.

O desenvolvimento do 4CAR remete-se a lembrar as pessoas do dia da revisão, quando foi realizado e até através da expansão do serviço para quando um componente apresentar falhas antes de parar de funcionar completamente.

3.1.4.2 Introdução ao 4CAR

Quando se deseja criar um aplicativo para o modo Android Auto é preciso manter o aplicativo do AA no celular para seu funcionamento, quando um aplicativo for desenvolvido é válido lembrar que o app funcionará fora do modo AA, sendo assim, podendo explorar maiores opções para atrair o usuário.

O “4CAR”, aplicativo desenvolvido em Android com extensão para Android Auto foi elaborado para melhor interação entre o usuário e o veiculo, o fundamento para a criação deste aplicativo é criar um canal de comunicação entre o veículo e o usuário.

Pontos que o aplicativo “4CAR” fornecerá :

- Através de uma *interface*, via *bluetooth*, são retirados dados diretos da rede *CAN* e transmitidos para o *smartphone*;
- Agenda no aplicativo para criar eventos, integrado com a quilometragem, como troca de óleo, revisões, dentre outras, para que o aplicativo lembre o usuário, fazendo com que tenha uma manutenção preventiva;
- Através do código de erro do sistema *OBD* é possível avisar o usuário sobre possíveis falhas;
- Notificações para o usuário.

Até o final deste trabalho foi possível apenas realizar o serviço de notificação para o usuário.

Vale lembrar que o aplicativo também funcionará quando estiver fora do modo Android Auto e quando estiver no modo AA, vale lembrar, que o aplicativo “4CAR” apenas funcionará com o serviço de mensagens e notificações.

3.2 Metodologia

O projeto proposto tem natureza aplicada na integração de aplicativos móveis com as mensagens de monitoramento do veículo. Seus objetivos são exploratórios, na construção de padrões que possam ser replicados posteriormente em outros projetos de *interface* entre esses sistemas. Já o procedimento para o levantamento dos dados tem natureza experimental.

Inicialmente foi realizada uma pesquisa para determinar o escopo do estudo e da aplicação que seria desenvolvida. Informações básicas sobre Java e XLM, linguagens que o Android utiliza para programar foram sumarizadas, para depois implementar as funcionalidades de interatividade entre o *smartphone* o automóvel.

Esse trabalho busca mostrar o funcionamento do Android Auto, seu sistema operacional Android e uma aplicação da plataforma Android Auto.

4 DESENVOLVIMENTO DO APLICATIVO

Ao longo desse capítulo, será relatado todo o procedimento utilizado para criar o aplicativo 4CAR, desde sua criação básica no Android Studio, até o seu desenvolvimento final.

Antes de iniciar o desenvolvimento do aplicativo, foi necessário elaborar o fluxo de dados de como ele deveria se comportar. O desenvolvimento do fluxo de dados para um aplicativo para o Android é um pouco diferente que o de um programa para uma plataforma PC, por exemplo, pois todas as ações no Android estão associadas a algum evento, como o usuário clicar na tela.

4.1 Lógica de Funcionamento do Aplicativo

Uma aplicação Android é composta por uma serie de componentes lógicos que interagem entre si, os principais componentes são:

- - Atividades (*Activity*);
- - *Intents*;
- - Interface gráfica (*View*);
- - Banco de dados (SQLite).

As *Activities*, ou atividades, são componentes capazes de apresentar uma tela para interagir com os usuários, através delas que ocorrem as vontades, interações do usuário com o aplicativo.

Através das *Activities* as *Intents* ganham vida, as *Intents* são criadas a partir de ações do usuário e representam a intenção de se realizar algo.

A interface gráfica (*Views*), a plataforma Android possui uma grande fonte de elementos gráficos, é possível encontrar aplicativos para jogos, GPS, os mais diversos; os elementos gráficos mais utilizados são:

- Ferramentas de Layout;
- -DatePicker;
- -Spinner;
- -ListView;
- -AlertDialog.

Para a persistência dos dados, está disponível para a plataforma Android o *SQLite*, que não necessita de um processo servidor. O *SQLite* armazena as tabelas, *views*, índices e *triggers* em apenas um arquivo em disco, no qual são realizadas as operações de leitura e escrita.

O aplicativo sendo composto por *activities*, sendo cada vez que o usuário muda de tela ou acessa outro conteúdo, ele está acessando diferentes atividades, por isso as *activities* possuem um ciclo de vida, como pode ser visto na figura 5.

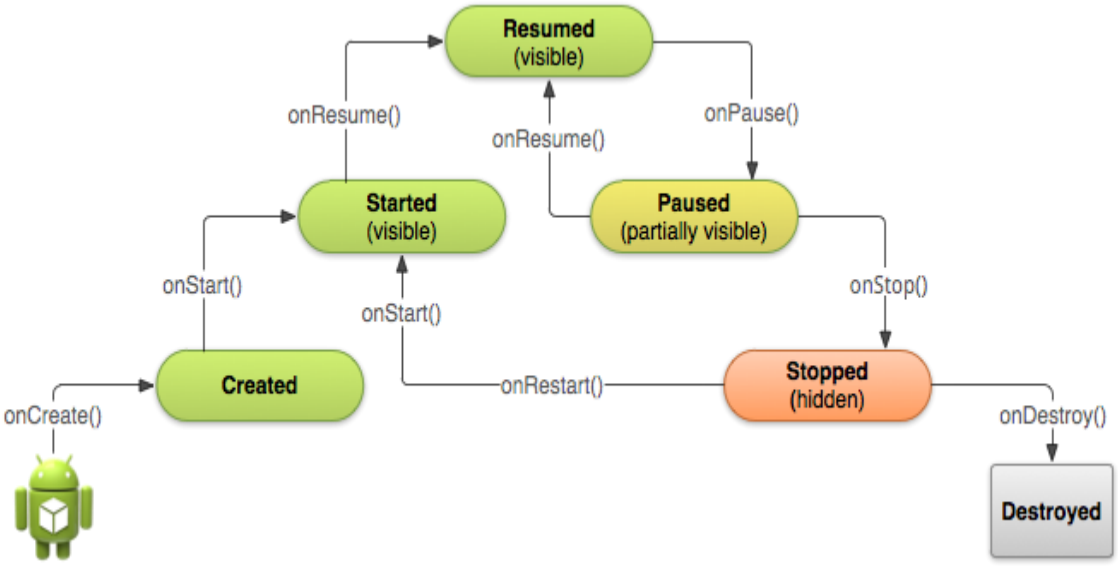

Figura 6 – Ciclo de vida de uma Activity

https://dariomungoi.wordpress.com/2015/01/27/desenvolvendo-para-android-ciclo-de-vida-de-uma-activity/. - Acessado dia 25/06/16.

Sempre que uma *Acitivity* muda de estado, o Android aciona um método *call-back* correspondente. Assim que o usuário inicia uma aplicação, o Android cria atividade principal

que está declarada no *AndroidManifest.xml* e invoca seu método *onCreate*. Em seguida, o Android invoca os métodos onStart e logo após o onResume. A *Activity* torna- se visível para o usuário no estado *Started* e assim permanece até os métodos onPause (visível parcialmente) ou *onDestroy* serem chamados. Quando a *Acitivity* está no estado *Resumed* dizemos que ela está no *foreground* e pode realizar interação com o usuário.

A *Acitivity* muda para o estado *Paused* quando for parcialmente coberta por outra *Activity*, que pode não ocupar toda a tela ou ser transparente. Se o usuário sair da aplicação ou iniciar outra atividade que encubra totalmente a que está sendo executada, então o método *onStop* é invocado e a *Acitivity* vai para *background*.

Quando uma *Acitivty* está nos estados *Paused* ou *Stopped*, o sistema operacional pode removê-la da memória, invocando seu método *finish* ou encerrando arbitrariamente o seu processo. Nestas condições, o método *onDestroy* é disparado. Segundo [Monteiro, 2013]

Quando estendemos a aplicação para o sistema Android Auto, ela retoma outro fluxo de dados, sabendo que a extensão só permite “notificações / mensagens e media player”, conforme apresentado na figura abaixo, notamos que a reprodução do Android Auto *app* acontece da interação do Android Media App e o Android Auto App, quando executamos uma ação, seja na central multimídia ou no DHU, existe todo um caminho a ser percorrido.

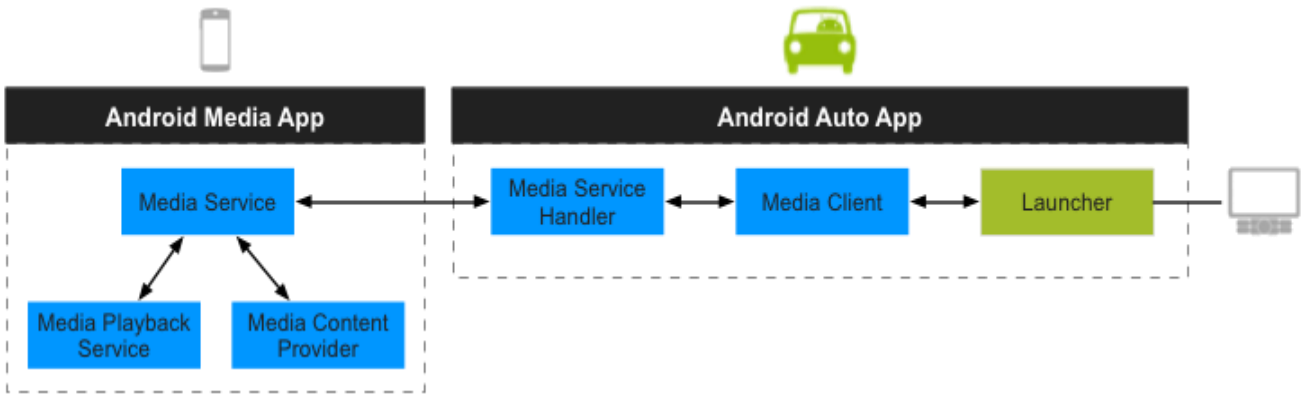

Figura 7 - Representa a lógica do sistema com a expansão Android Auto.

O aplicativo em Android integra duas linguagens de programação JAVA e XML, através do JAVA criamos as "*activity",* fazendo parte do código fonte e os *arquivos.xml* que fazem parte da programação gráfica. Com essa breve introdução, começo a explicar os códigos do aplicativo.

5 RESULTADOS E CONSIDERAÇÕES FINAIS

A pesquisa e escrita técnica do trabalho possui nível de dificuldade igual ao desenvolvimento da parte pratica, exigindo muitas revisões e tempo.

5.1 Propostas Para Melhorias Futuras

Propostas para melhoria do 4CAR:

- Extrair dados da rede *CAN* e transmitir para o *smartphone*;
- Agenda no aplicativo para criar eventos, integrado com a quilometragem, como troca de óleo, revisões, dentre outras, para que o aplicativo lembre o usuário, fazendo com que tenha uma manutenção preventiva;
- Através do código de erro do sistema *OBD* é possível avisar o usuário sobre possíveis falhas;

Através desse trabalho é possível criar outras aplicações para outros determinados usos.

6 REFERÊNCIAS

MONTEIRO, João Bosco, *Google Android*, Casa do código, São Paulo, (2013).

GUIMARÃES, Alexandre de Almeida, *Eletrônica Embarcada*, Érica, São Paulo, (2007).

CAELUM, Ensino e Inovação, *FJ-11 Java e Orientação a Objetos*.

LECHETA, Ricardo R., *Google Android, aprenda a criar aplicações para dispositivos móveis com o Android SDK*, Novatec, São Paulo, (2010).

SIERRA e BATES, Kathy e Bert, *Use a Cabeça! Java*, Alta Books & O'Reilly, Rio de Janeiro, (2005).

SILVA, Luciano Alves, *Apostila de Android passo a passo*, 3°edição (2008), (lucianopascal@yahoo.com.br).

Notas de aula do Prof. Msc. Murilo Zanini de Carvalho.

A história do Android. Veja como surgiu e como se tornou o sistema operacional mobile mais utilizado do mundo, e se você acha que ele é usado só em *smartphone*s, saiba que ele é usado em muito mais, desde tv's até geladeiras. [Internet]. Maximiliano Meyer. [atualizado em Mai 05 2016; visitado em Out 27 2015]. Disponível em: https://www.oficinadanet.com.br/post/13939-a-historia-do-android

Android nasceu 5 anos antes do primeiro *smartphone*; em 2003. [Internet]. FILIPE GARRETT. [Atualizado em 2015 Dez 08; visitado em 2015 Out 27]. Disponível em: http://www.techtudo.com.br/artigos/noticia/2011/12/android-nasceu-5-anos-antes-do-primeiro-*smartphone*-em-2003.html

O que é Kernel? [Internet]. FABIO EDUARDO AMARALEM. [Atualizado em FEV 26 2009; visitado em OUT 27 2015]. Disponível em: http://www.tecmundo.com.br/mac-os-x/1636-o-que-e-kernel-.htm

Relatório da Google aponta uma futura central Android completa em veículos. [Internet]. BRUNO MICALI. [Atualizado em OUT 23 2015; visitado em OUT 27 2015]. Disponível em: http://www.tecmundo.com.br/android-auto/88445-relatorio-google-aponta-futura-central-android-completa-veiculos.htmb

Profiles in Sciense [Internet]. O kernel do Linux (BR). [Visitado em OUT 28 2015]. Disponível em: https://kidlinux.wordpress.com/o-kernel-do-linux/

TIPOS DE MANUTENÇÃO: preventiva, preditiva e corretiva. [Internet]. FERNANDO SIQUEIRA. [Atualizado em OUT 2014; visitado em JUL 18 2016]. Disponível em: http://blog.tribunadonorte.com.br/autosemotores/2014/10/09/tipos-de-manutencao/

Manutenção é 40% mais barata que conserto, diz especialista. [Internet]. LUCIANA MASTROROSA. [Atualizado em OUT 25 2007; visitado em JUL 18 2016]. Disponível em: http://g1.globo.com/Noticias/Carros/0,,MUL155595-9658,00.html

Profiles in Sciense [Internet]. Android Auto. A informação certa para a estrada à frente. [Visitado em ABR 26 2016]. Disponível em: https://www.android.com/auto/

O imenso mercado dos aplicativos. [Internet]. CTAURION (CEZAR CHEDE). [Atualizado em FEV 11 2013; visitado em ABR 05 2016]. Disponível em: https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/o_imenso_mercado_dos_aplicativos8?lang=en

INTRODUCING THE OPEN AUTOMOTIVE ALLIANCE. [Internet]. Open automotive alliance. [Visitado em: ABR 16 2016]. Disponível em: http://www.openautoalliance.net/#about

Como usar o Android Debug Bridge (ADB)… no Windows. [Internet]. Pedro Pinto. [Atualizado em MAR 27 2012; visitado em ABR 06 2016]. Disponível em: http://pplware.sapo.pt/*smartphone*s-tablets/android/como-usar-o-android-debug-bridge-adb/

7 Vantagens do Android Auto que você precisa conhecer. [Internet]. André Luiz. [Atualizado em ABR 14 2015; visitado em ABR 06 2016]. Disponível em: http://www.androidpit.com.br/android-auto-vantagens

App AutoMate transdorma qualquer Android em um Android Auto. [Internet]. André Luiz. [Atualizado em ABR 07 2015; visitado em ABR 06 2016]. Disponível em: http://www.androidpit.com.br/automate-app-android-auto

Android Auto chega ao Brasil no VW Fox! Só falta o Google colaborar. [Internet]. André Luiz. [Atualizado em JUN 03 2015; visitado em JUL 18 2016]. Disponível em: http://www.androidpit.com.br/android-auto-volkswagen-fox

Apple CarPlay. Sua experiência com o iPhone ficou ainda mais móvel. [Internet]. Apple [visitado em JUL 18 2016]. Disponível em: http://www.apple.com/br/ios/carplay/

AutoMade for Android Beta – Comunidade- Google+. [Internet]. Kahtaf Alam (proprietário). [Visitado em ABR 03 2016]. Disponível em: https://plus.google.com/communities/102908788212710276577

Getting Started with Auto. Android Auto extends the Android platform into the car. When users connect their handheld devices running Android 5.0 or higher to a compatible vehicle, the Auto user interface provides a car-optimized Android experience on the vehicle's screen. Users interact with compatible apps and services through voice actions and the vehicle's input controls (like a touchscreen or dashboard buttons). [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/training/auto/start/index.html?hl=pt-br

Android for Work Developer Overview. As an Android for Work developer, you can deploy devices and apps to employees and keep your corporate data secure. With Android for Work, organizations can choose what devices, APIs, and framework they want to use to develop apps. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/work/overview.html

Providing Messaging for Auto. Staying connected through messages is important to many drivers. Chat apps can let users know if a child need to be picked up, or if a dinner location has been changed. The Android framework enables messaging apps to extend their services into car dashboards using a standard user interface that lets drivers keep their eyes on the road. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/training/auto/messaging/index.html

Providing Audio Playback for Auto. Drivers want to access their music and other audio content on the road. Audio books, podcasts, sports commentary, and recorded talks can make a long trip educational, inspirational, and enjoyable. The Android framework allows you to extend your audio app so users can listen to their favorite tunes and audio content using a simple, yet customizable user interface. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/training/auto/audio/index.html

Testing Apps for Auto. Testing your Auto app ensures that users do not encounter unexpected results or have a poor experience when interacting with your apps. Android now provides Desktop Head Unit (DHU), a testing tool for Auto apps that lets you test pre-released versions of your Android Auto apps without having to work from your car. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/training/auto/testing/index.html

Android Debug Bridge. Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/studio/command-line/adb.html#devicestatus

Run Apps on a Hardware Device. When building an Android app, it's important that you always test your application on a real device before releasing it to users. This page describes how to set up your development environment and Android-powered device for testing and debugging on the device. [Internet]. Developers Android (US). [Visitado em SET 20 2015]. Disponível em: https://developer.android.com/studio/run/device.html#developer-device-options

Quem vai construir o carro do futuro? Para fazer veículos sem motorista, montadoras e gigantes do setor de tecnologia trabalham às vezes como parceiros, às vezes como concorrentes. Esse é o grande tema do Salão do Automóvel de Frankfurt deste ano. [Internet]. Henrik Böhme (ca)[Deutsche Welle (DW) Made For Minds]. [Atualizado em SET 16 2015; visitado em SET 17 2015]. Disponível em: http://www.dw.com/pt/quem-vai-construir-o-carro-do-futuro/a-18718532

MediaBrowserServiceCompat and the modern media playback app. Media apps, more so than most, benefit strongly from working with the Android system and other apps. Some things, like handling interruptions with audio focus, have been a constant since nearly the beginning of Android and are just as important now as ever. While Media Playback with MediaSessionCompat is relatively new, it provides a consistent way of talking to the system across all API 4+ API levels. [Internet]. Ian Lake (US). [Atualizado em MAR 02 2016; visitado em ABR 30 2016]. Disponível em: https://medium.com/google-developers/mediabrowserservicecompat-and-the-modern-media-playback-app-7959a5196d90#.gt3q1yfnb

7 APÊNDICES

Apêndice A: Desk-Head-Unit (DHU)

Passo a passo para instalação do Desk-Head-Unit (DHU)

1- Abra o Android Studio ou procure na pasta Android do seu disco o programa SDK Manager;
2- Através do SDK Manager é possível realizar todos os complementos, o arquivo executável do DHU estará dentro da pasta “Auto” podendo ser localizada na pasta “Extras”, realize o download dela;

Figura 8– Imagem do SDK Manager

Passo a passo para conexão do Desk-Head-Unit (DHU) com o App Android Auto

1- Manter o celular conectado com sua maquina de desenvolvimento via USB(no caso estou usando Windows);
2- Abrindo o App do Android Auto no celular, toque 10 vezes no Titulo “Android Auto” para ativar o modo desenvolvedor do App;
3- Na opção de configurações do App escolha a opção “*Start Head Unit Server*”;

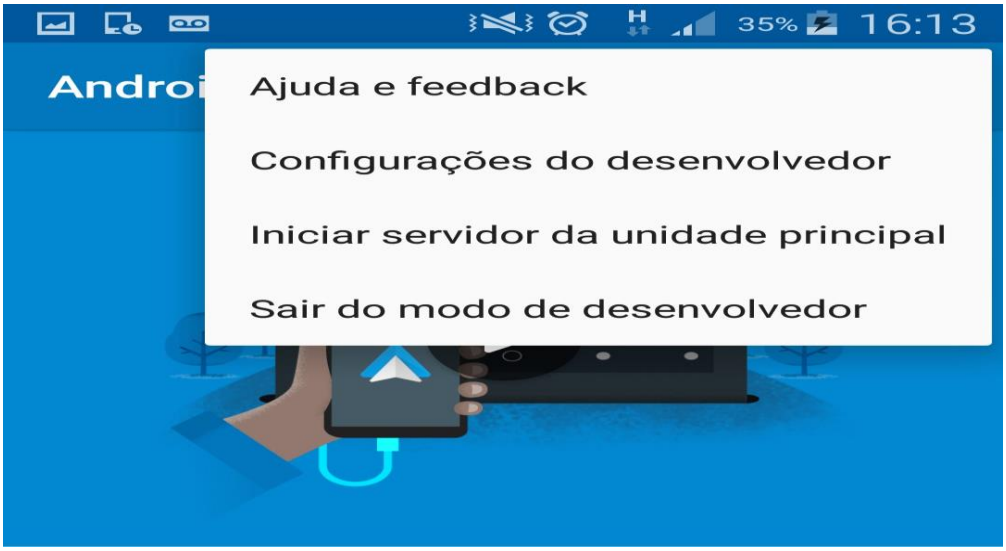

Figura 9– Tela de iniciar o modo de desenvolvedor

4- Para que o dispositivo móvel e sua maquina de desenvolvimento tenham o mesmo canal de comunicação é preciso enviar um comando via ADB para estabelecer o canal de comunicação, "$ adb forward tcp:5277 tcp:5277";

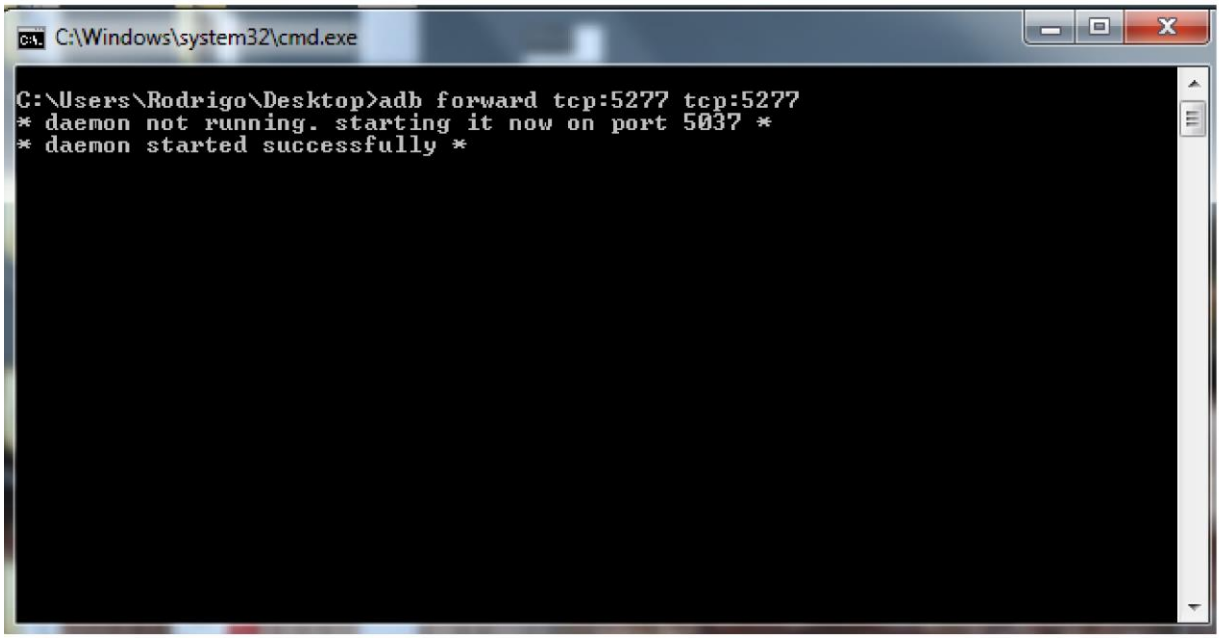

Figura 10– Configurando as portas para comunicação.

5- Pronto, inicie o DHU e veja se o mesmo funciona corretamente, no momento de iniciar pode ser encontrado algum problema, se ocorrer, inicie o DHU pelo propt de comando do Windows.

Apêndice B: Android Studio

Passo a passo para instalação do Android Studio e criando uma App para Android Auto

1- Abrindo uma guia da internet e pesquisando pelo software Android Studio logo encontrara uma pagina do “developers.android” que oferece o serviço de download;
2- Siga o processo de instalação do software;
3- Abrindo o Android Studio, na aba “File” selecione “New Project”;
4- Logo abrira uma janela, Escolha um nome para o projeto e clique em “Next”;
5- Selecione a opção em que você queira que o seu aplicativo funcione, no nosso caso, “Phone and Tablet” e “Android Auto”, ambas com a API 21 ou maiores;
6- Na próxima tela, terá opções de activity, podendo escolher desde “Blank Activity” a “Google Maps Acitivity”;
7- Depois ele te dará opção de activity para o Android Auto, que consiste em “Messaging servisse” e “Media servisse”, selecione uma e finalize;

Pronto, mesmo sem escrever nada o Android Studio gera um aplicativo que porem não faz nada.

Apêndice C: Android Debug Bridge (ADB)
Passo a passo para instalação e utilização do ADB

1- Conecte o seu dispositivo móvel no computador;
2- No celular habilite a opção do modo de desenvolvedor e depois habilite a depuração via USB;
3- Para utilizar o ADB é preciso instalar um driver de comunicação via USB da marca de seu aparelho, no site de desenvolvedor do Google é possível localizar, instale logo em seguida;
4- Pronto agora você consegue enviar comandos via ADB, no site de desenvolvedor do Google é possível obter uma lista de códigos de comandos ADB, para começar e saber se o seu celular esta comunicando pode executar o comando “# adb devices”.

Apêndice D: Explicando o Programa

Para teste da comunicação e funcionamento do aplicativo foi elaborado um programa com a finalidade de verificar o recebimento das notificações.

O primeiro código remete ao arquivo *Androidmanifest.xml*:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.messagingservice">

    <application android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@drawable/ic_launcher"
        android:theme="@style/AppTheme">

        <meta-data android:name="com.google.android.gms.car.application"
                   android:resource="@xml/automotive_app_desc"/>

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <service android:name=".MessagingService">
        </service>

        <receiver
            android:name=".MessageReadReceiver"
            android:exported="false">
            <intent-filter>
                <action
android:name="com.example.android.messagingservice.ACTION_MESSAGE_READ"/>
            </intent-filter>
        </receiver>

        <receiver
            android:name=".MessageReplyReceiver"
            android:exported="false">
            <intent-filter>
                <action
android:name="com.example.android.messagingservice.ACTION_MESSAGE_REPLY"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

O application descreve o nome e o icone do aplicativo.

O meta-data habilita o aplicativo para trabalhar com o Android Auto.

O Service habilita o aplicativo para serviço de mensagens, habilitando o serviço de leitura e resposta da mensagem.

Código *Conversations.java*

```
public class Conversations {

    /**
     * Set of strings used as messages.
     */
    private static final String[] MESSAGES = new String[]{
            "DTC: P0333?",
            "Possivel falha no sensor de combustao",
            "Velocidade maxima ultrapassada",
            "DTC: C0123",
            "Falha no farol de milha",
            "Acidente mais a frente"
    };

    /**
     * Senders of the said messages.
     */
    private static final String[] PARTICIPANTS = new String[]{

            "4CAR"
    };

    static class Conversation {

        private final int conversationId;

        private final String participantName;

        /**
         * A given conversation can have a single or multiple messages.
         * Note that the messages are sorted from *newest* to *oldest*
         */
        private final List<String> messages;

        private final long timestamp;

        public Conversation(int conversationId, String participantName,
                            List<String> messages) {
            this.conversationId = conversationId;
            this.participantName = participantName;
            this.messages = messages == null ?
Collections.<String>emptyList() : messages;
            this.timestamp = System.currentTimeMillis();
        }
```

```
        public int getConversationId() {
            return conversationId;
        }

        public String getParticipantName() {
            return participantName;
        }

        public List<String> getMessages() {
            return messages;
        }

        public long getTimestamp() {
            return timestamp;
        }

        public String toString() {
            return "[Conversation: conversationId=" + conversationId +
                    ", participantName=" + participantName +
                    ", messages=" + messages +
                    ", timestamp=" + timestamp + "]";
        }
    }

    private Conversations() {
    }

    public static Conversation[] getUnreadConversations(int
howManyConversations,
                                                        int
messagesPerConversation) {
        Conversation[] conversations = new
Conversation[howManyConversations];
        for (int i = 0; i < howManyConversations; i++) {
            conversations[i] = new Conversation(
                    ThreadLocalRandom.current().nextInt(),
                    name(), makeMessages(messagesPerConversation));
        }
        return conversations;
    }

    private static List<String> makeMessages(int messagesPerConversation) {
        int maxLen = MESSAGES.length;
        List<String> messages = new ArrayList<>(messagesPerConversation);
        for (int i = 0; i < messagesPerConversation; i++) {
            messages.add(MESSAGES[ThreadLocalRandom.current().nextInt(0,
maxLen)]);
        }
        return messages;
    }

    private static String name() {
        return PARTICIPANTS[ThreadLocalRandom.current().nextInt(0,
PARTICIPANTS.length)];
    }
}.
```

Através desse código, são definidos os participantes, nesse caso o “4CAR” que mandará uma mensagem para o usuário, e as conversas, nesse exemplo estão salvas algumas conversas pré-configuradas que serão enviadas para o usuário.

Código *MessagingService.java*:

```
public class MessagingService extends Service {
    private static final String TAG =
MessagingService.class.getSimpleName();
    private static final String EOL = "\n";
    private static final String READ_ACTION =
            "com.example.android.messagingservice.ACTION_MESSAGE_READ";
    public static final String REPLY_ACTION =
            "com.example.android.messagingservice.ACTION_MESSAGE_REPLY";
    public static final String CONVERSATION_ID = "conversation_id";
    public static final String EXTRA_REMOTE_REPLY = "extra_remote_reply";
    public static final int MSG_SEND_NOTIFICATION = 1;

    private NotificationManagerCompat mNotificationManager;

    private final Messenger mMessenger = new Messenger(new
IncomingHandler(this));

    @Override
    public void onCreate() {
        Log.d(TAG, "onCreate");
        mNotificationManager =
NotificationManagerCompat.from(getApplicationContext());
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.d(TAG, "onBind");
        return mMessenger.getBinder();
    }

    // Creates an intent that will be triggered when a message is marked as
read.
    private Intent getMessageReadIntent(int id) {
        return new Intent()
                .addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES)
                .setAction(READ_ACTION)
                .putExtra(CONVERSATION_ID, id);
    }

    // Creates an Intent that will be triggered when a voice reply is
received.
    private Intent getMessageReplyIntent(int conversationId) {
        return new Intent()
                .addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES)
                .setAction(REPLY_ACTION)
                .putExtra(CONVERSATION_ID, conversationId);
    }

    private void sendNotification(int howManyConversations, int
messagesPerConversation) {
```

```
        Conversations.Conversation[] conversations =
Conversations.getUnreadConversations(
                howManyConversations, messagesPerConversation);
        for (Conversations.Conversation conv : conversations) {
            sendNotificationForConversation(conv);
        }
    }

    private void sendNotificationForConversation(Conversations.Conversation
conversation) {
        // A pending Intent for reads
        PendingIntent readPendingIntent =
PendingIntent.getBroadcast(getApplicationContext(),
                conversation.getConversationId(),
                getMessageReadIntent(conversation.getConversationId()),
                PendingIntent.FLAG_UPDATE_CURRENT);

        // Build a RemoteInput for receiving voice input in a Car
Notification or text input on
        // devices that support text input (like devices on Android N and
above).
        RemoteInput remoteInput = new
RemoteInput.Builder(EXTRA_REMOTE_REPLY)
                .setLabel(getString(R.string.reply))
                .build();

        // Building a Pending Intent for the reply action to trigger
        PendingIntent replyIntent =
PendingIntent.getBroadcast(getApplicationContext(),
                conversation.getConversationId(),
                getMessageReplyIntent(conversation.getConversationId()),
                PendingIntent.FLAG_UPDATE_CURRENT);

        // Build an Android N compatible Remote Input enabled action.
        NotificationCompat.Action actionReplyByRemoteInput = new
NotificationCompat.Action.Builder(
                R.drawable.notification_icon, getString(R.string.reply),
replyIntent)
                .addRemoteInput(remoteInput)
                .build();

        // Create the UnreadConversation and populate it with the
participant name,
        // read and reply intents.
        UnreadConversation.Builder unreadConvBuilder =
                new
UnreadConversation.Builder(conversation.getParticipantName())
                .setLatestTimestamp(conversation.getTimestamp())
                .setReadPendingIntent(readPendingIntent)
                .setReplyAction(replyIntent, remoteInput);

        // Note: Add messages from oldest to newest to the
UnreadConversation.Builder
        StringBuilder messageForNotification = new StringBuilder();
        for (Iterator<String> messages =
conversation.getMessages().iterator();
             messages.hasNext(); ) {
            String message = messages.next();
            unreadConvBuilder.addMessage(message);
            messageForNotification.append(message);
            if (messages.hasNext()) {
```

```
                messageForNotification.append(EOL);
            }
        }

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(getApplicationContext())
                .setSmallIcon(R.drawable.notification_icon)
                .setLargeIcon(BitmapFactory.decodeResource(
                        getApplicationContext().getResources(),
R.drawable.android_contact))
                .setContentText(messageForNotification.toString())
                .setWhen(conversation.getTimestamp())
                .setContentTitle(conversation.getParticipantName())
                .setContentIntent(readPendingIntent)
                .extend(new CarExtender()
                        .setUnreadConversation(unreadConvBuilder.build())
                        .setColor(getApplicationContext().getResources()
                                .getColor(R.color.default color light)))
                .addAction(actionReplyByRemoteInput);

        MessageLogger.logMessage(getApplicationContext(), "Sending
notification "
                + conversation.getConversationId() + " conversation: " +
conversation);

        mNotificationManager.notify(conversation.getConversationId(),
builder.build());
    }

    /**
     * Handler for incoming messages from clients.
     */
    private static class IncomingHandler extends Handler {
        private final WeakReference<MessagingService> mReference;

        IncomingHandler(MessagingService service) {
            mReference = new WeakReference<>(service);
        }

        @Override
        public void handleMessage(Message msg) {
            MessagingService service = mReference.get();
            switch (msg.what) {
                case MSG_SEND_NOTIFICATION:
                    int howManyConversations = msg.arg1 <= 0 ? 1 :
msg.arg1;
                    int messagesPerConversation = msg.arg2 <= 0 ? 1 :
msg.arg2;
                    if (service != null) {
                        service.sendNotification(howManyConversations,
messagesPerConversation);
                    }
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }
}.
```

Esse código mostra a configuração do serviço de mensagens, através dele são criados os eventos que sinalizam mensagens como lidas, mensagens para responder, organiza as mensagens colocando as mais novas antes.

Código *frament_message_me.xml*:

```
LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center_horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <Button
        android:id="@+id/send_1_conversation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_1_conversation"/>

    <Button
        android:id="@+id/send_2_conversations"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_2_conversations"/>

    <Button
        android:id="@+id/send_1_conversation_3_messages"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_1_conv_3_messages"/>

    <TextView
        android:id="@+id/data_port"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:scrollbars="vertical"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/clear"
        android:text="@string/clear_log"/>

</LinearLayout>.
```

No Android Auto as mensagens não podem ocupar a tela inteira, então para isso, o programa utiliza de um elemento gráfico, conhecido como *fragments*. Esse código faz com que se gerem botões para iniciar uma mensagem para o usuário verificar. Além de criar é preciso colocar identificação (*ID's*) nos botões para a parte lógica funcionar.

Resultados da notificação gerada

Depois de toda implementação por parte dos códigos é gerada uma notificação, lembrando que notificações quando estão no modo Android Auto não aparece seu conteúdo escrito e sim através de comandos de voz.

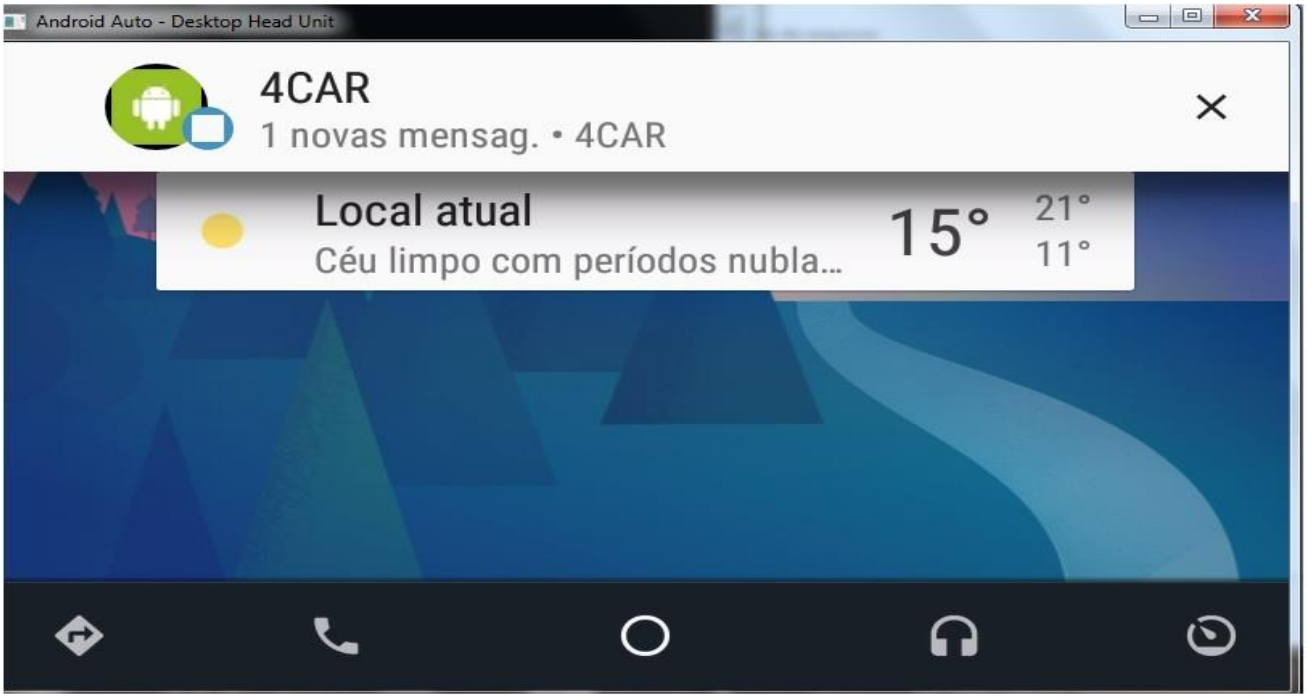

Figura 11 – Recebimento de Notificação.

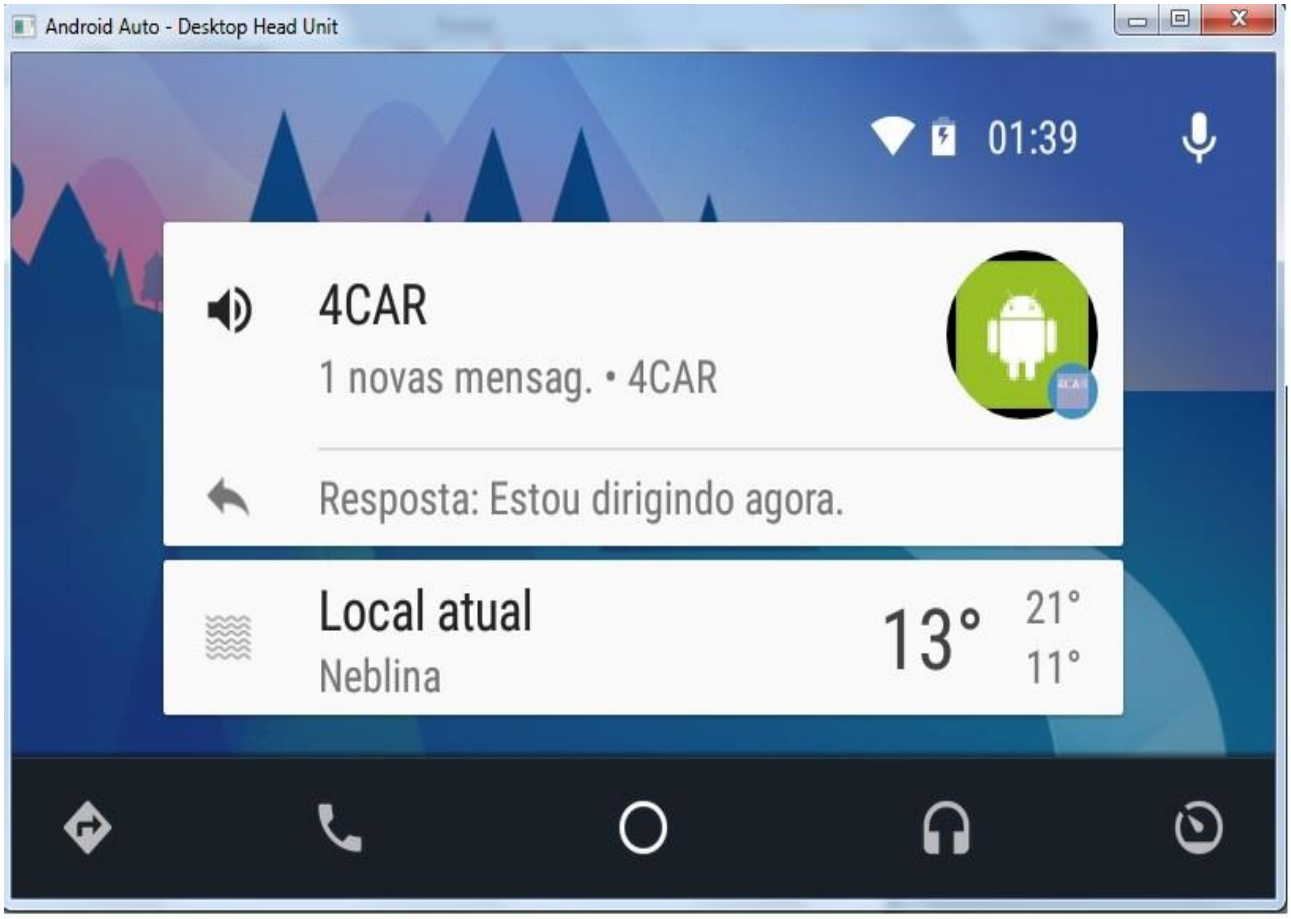

Figura 12 – Depois que a notificação foi recebida, ela fica aguardando o sinal para reproduzir.