

**UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

EMERSON LEONARDO ZOCK ALVES

**FERRAMENTA ANDROID PARA O AUXÍLIO TÉCNICO AUTOMOBILÍSTICO NO
MONITORAMENTO DE VEÍCULOS COM O ESCANEAMENTO OBD2**

CRICIÚMA

2020

EMERSON LEONARDO ZOCK ALVES

**FERRAMENTA ANDROID PARA O AUXÍLIO TÉCNICO AUTOMOBILÍSTICO NO
MONITORAMENTO DE VEÍCULOS COM O ESCANEAMENTO OBD2**

Trabalho de Conclusão de Curso, para
obtenção do grau de Bacharel no curso de
Ciência da Computação da Universidade do
Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Matheus Leandro
Ferreira

CRICIÚMA

2020

EMERSON LEONARDO ZOCK ALVES

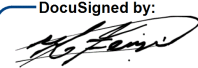
**FERRAMENTA ANDROID PARA O AUXÍLIO TÉCNICO AUTOMOBILÍSTICO NO
MONITORAMENTO DE VEÍCULOS COM O ESCANEAMENTO OBD2**

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Desenvolvimento Móvel

Criciúma, 03 de agosto de 2020.

BANCA EXAMINADORA

Prof. Matheus Leandro Ferreira - Especialista - (UNESC) – Orientador

DocuSigned by:

18698336A82D41E...

Prof. Gustavo Bisognin - Mestre - (UNESC)

DocuSigned by:

85642183928E460...

Prof. Marcel Campos Inocêncio - Especialista – (UNESC)

DocuSigned by:

2122C672EA4A42E...

**A todos que conheci ao longo desta jornada
e que de alguma forma contribuíram para
minha formação.**

AGRADECIMENTOS

Agradeço a instituição UNESCO, toda a direção, administração e seu corpo docente, que me proporcionaram todo o conhecimento necessário no processo de formação profissional. Agradeço de forma especial ao orientador Prof. Matheus Leandro Ferreira, o qual ao longo desta jornada esteve sempre presente, de forma competente e efetiva na solução dos mais diversos problemas, me direcionando no caminho certo durante o desenvolvimento do projeto.

Agradeço a minha família, por cada valor ensinado e por toda a compreensão e apoio recebido para que eu concluísse o curso de ciência da computação.

Agradeço aos amigos que fiz ao longo desta jornada universitária e que de alguma forma, direta ou indiretamente contribuíram para a minha formação.

Agradeço em especial ao meu amigo Victor Pavei Goes, que sempre permanecerá em minha memória como um grande acadêmico e colega universitário.

A todos que de alguma forma contribuíram para que eu chegasse até aqui, o meu muito obrigado.

“Falar é fácil, me mostre o código”

Linus Torvalds

RESUMO

A mecânica ligada a informática proporciona a criação de equipamentos, softwares e técnicas, possibilitando aos profissionais da área utilizarem para diagnósticos veiculares. Desde o surgimento da interface de sistema de autodiagnóstico os mecânicos têm-se usado para encontrar rapidamente as falhas de um determinado veículo. Porém, os leitores mecânicos possuem um preço não acessível para pequenas mecânicas e também para os proprietários dos veículos, que para aumentar o “tempo de uso” de seus carros, é preciso fazer manutenção regularmente, principalmente para que não ocorra um problema enquanto estiver na estrada, podendo gerar um sinistro ou até a morte. Levando isso em consideração, este projeto de conclusão de curso visa a compreensão da interface de sistema de autodiagnóstico na construção de um protótipo que auxilie os donos de veículos a manterem seu automóvel em estado funcional. O protótipo foi desenvolvido levando em consideração as funcionalidades dos dois principais softwares pagos na *Google Play*, e de suas mais frequentes reclamações feitas por brasileiros. Após a escolha destes softwares foi desenvolvida uma solução e testada em veículos reais, sendo capaz de analisar as falhas existentes junto de um escâner de sistema de autodiagnóstico em que os resultados foram satisfatórios mesmo não implementado todas as funcionalidades dos softwares pagos, todas as principais reclamações foram atendidas, e validadas em veículos reais.

Palavras-chave: OBD2. Comunicação de dados. Desenvolvimento para Android. Internet das Coisas.

ABSTRACT

Computer-related mechanics provide the creation of equipment, software and techniques, enabling professionals in the field to use for vehicle diagnostics. Since the appearance of the self-diagnosis system interface, mechanics have been used to quickly find faults in a given vehicle. However, mechanical readers have an inaccessible price for small mechanics and also for vehicle owners, that in order to increase the “usage time” of their cars, it is necessary to do maintenance regularly, mainly so that there is no problem while on the road. road, which can lead to an accident or even death. Taking this into account, this course completion project aims to understand the self-diagnosis system interface in the construction of a prototype that helps vehicle owners to keep their car in a functional state. The prototype was developed taking into account the features of the two main paid software on Google Play, and their most frequent complaints made by Brazilians. After choosing these software, a solution was developed and tested in real vehicles, being able to analyze the existing failures with a self-diagnosis system scanner in which the results were satisfactory even if not all the functionalities of the paid software were implemented, all the main complaints were met, and validity in real vehicles.

Keywords: OBD2. Data Communication. Android Development. Internet of Things.

LISTA DE ILUSTRAÇÕES

Figura 1. Estrutura de arquivos do protótipo móvel.....	37
Figura 2. Tela de login.....	38
Figura 3. Tela de monitoramento antes de se conectar ao OBD2.....	39
Figura 4. Manifesto com permissão de acesso ao Bluetooth.....	40
Figura 5. Gerenciamento da conexão e envio de dados para o escâner.....	40
Figura 6. Função que espera o retorno da conexão serial.....	41
Figura 7. Tela de monitoramento após se conectar ao OBD2.....	42
Figura 8. Tela de monitoramento após buscar falhas.....	44

LISTA DE TABELAS

Tabela 1. Valores a serem substituídos no retorno do OBD.....	18
Tabela 2. Tabela de comandos utilizados no aplicativo.....	36

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DTC	Diagnostic Trouble Codes
EUA	Estados Unidos da América
IDE	Integrated Development Environment
OBD	<i>On-Board Diagnosis</i>
OBD2	<i>On-Board Diagnosis 2</i>
PAE	<i>Protection Agency Environmental</i>
PMC	<i>Prefeitura Municipal de Criciúma</i>
REST	<i>Representational State Transfer</i>
RGE	<i>Recirculação do Gás de Escape</i>
SAE	Sociedade de Engenheiros Automotivos
SMS	Short Message Service
WSDL	Web Services Description Language
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO.....	13
1.2 OBJETIVO GERAL.....	14
1.3 OBJETIVOS ESPECÍFICOS.....	14
1.4 JUSTIFICATIVA.....	14
2 INTRODUÇÃO AO OBD2.....	16
2.1 HISTÓRIA DO OBD2.....	16
2.2 MONITORAMENTO DE SISTEMA E DETECÇÃO DE FALHAS.....	17
2.2.1 Monitor da eficiência do catalisador.....	19
2.2.2 Monitor de falhas de ignição.....	19
2.2.3 Monitor do sistema de combustível.....	20
2.2.4 Monitor dos sensores de oxigênio aquecido.....	20
2.2.5 Monitor do sistema de controle de evaporação de combustível.....	20
2.3 REDES DE COMUNICAÇÃO DE DADOS DIAGNÓSTICOS.....	21
2.3.1 Serviços de comunicação.....	21
3 PRINCIPAIS SOFTWARES PAGOS NA GOOGLE PLAY QUE UTILIZAM OBD2	23
3.1 PRINCIPAIS FUNÇÕES DO SOFTWARE.....	23
3.1.1 Diferenciais dos principais softwares.....	23
3.1.1 Dificuldades dos principais softwares.....	24
4 APLICAÇÃO PARA DISPOSITIVOS MÓVEIS ANDROID.....	25
4.1 METODOLOGIA DE DESENVOLVIMENTO.....	25
4.2 MODELAGEM DO BANCO DE DADOS.....	25
4.3 LINGUAGEM DE PROGRAMAÇÃO.....	26
4.3.1 Java.....	26
4.3.2 Flutter.....	26
4.3.3 Play framework.....	26

4.1.4 Vuejs	27
4.1.5 Mariadb	27
4.5 COMUNICAÇÃO WEB SERVICE.....	27
5 TRABALHOS CORRELATOS	29
5.1 UM SISTEMA DE ASSISTÊNCIA AO ECODRIVING EM ANDROID PARA MELHORAR A SEGURANÇA E EFICIÊNCIA DOS CARROS COM MOTOR DE COMBUSTÃO INTERNA.....	29
5.2 SISTEMA DE COMUNICAÇÃO VEICULAR UTILIZANDO SMARTPHONE.....	30
5.3 USO DE DIÁRIOS DE BORDO PARA MITIGAR O PERIGO EM ROTAS E O COMPORTAMENTO ARRISCADO DOS MOTORISTAS IDOSOS.....	30
5.4 SISTEMA DE AQUISIÇÃO DE DADOS BASEADOS EM STM32 PARA INTERNET DE VEÍCULOS.....	31
5.5 SISTEMA DE MONITORAMENTO REMOTO DE SAÚDE E MANUTENÇÃO PROGNÓSTICA DE VEÍCULOS.....	31
5.6 SCANNER OBD-II EM PLATAFORMA LABVIEW.....	32
5.7 SISTEMA DE MONITORAMENTO AUTOMOTIVO REMOTO.....	32
6 FERRAMENTA ANDROID PARA O AUXÍLIO TÉCNICO AUTOMOBILÍSTICO ...	33
6.1 METODOLOGIA.....	33
6.2 RECURSOS NECESSÁRIOS.....	34
6.3 FUNCIONAMENTO DA APLICAÇÃO.....	35
6.4 DESENVOLVIMENTO DO PROTÓTIPO.....	36
6.5 RESULTADOS E DISCUSSÃO DOS RESULTADOS.....	44
7 CONCLUSÃO	46
REFERÊNCIAS	47

1 INTRODUÇÃO

Pesquisas realizadas pelo Observatório Nacional de Segurança Viária (2020) mostram que ocorreu um aumento de 48,7% no número de pessoas mortas por acidente de veículos entre o ano de 2001 a 2012, em que um dos principais motivos são o aumento de 139% na frota de veículos neste período, em 2001 com 31,91 milhões de veículos a 2012 com 76,14 milhões e como esses automóveis são máquinas, é necessário fazer verificações periódicas em oficinas para manter tudo em ordem.

Comentado por Machado e Oliveira (2007), a muitos anos os automóveis começaram a utilizar módulos eletrônicos, sendo que os primeiros faziam um pouco mais que controlar a injeção eletrônica comparado com os modelos de 2007. Os módulos eletrônicos recebem dados de uma variedade de sensores, que na detecção de alguma anomalia os próprios módulos ajustam os parâmetros como por exemplo quantidade de oxigênio enviado ao motor para tentar corrigir as falhas.

A mecânica ligada à informática, quando voltada à evolução da tecnologia, proporcionam o desenvolvimento de equipamentos e técnicas para profissionais da área que são utilizados para diagnósticos de automóveis. Atualmente existem projetos desenvolvidos que padronizam o diagnóstico possibilitando identificar suas falhas e talvez até mesmo corrigi-las. Estes equipamentos, denominados *On-Board Diagnosis* (OBD) em português Sistema de autodiagnósticos), foram criados pela Sociedade de Engenheiros Automotivos (SAE) com objetivo de atingir estes propósitos. Entre as vantagens de se utilizar a interface OBD pode-se destacar:

[...]o sistema OBD também favorece o próprio cliente, pois permite que o veículo funcione conforme foi projectado, economizando combustível e consequentemente dinheiro. [...] Outro das vantagens do sistema OBD, é dar a conhecer ao condutor, através da luz indicadora no painel de instrumentos, de um qualquer problema menor, podendo assim prevenir o efeito “bola de neve” do problema, que levaria a uma reparação mais dispendiosa.[..] (MACHADO; OLIVEIRA, 2007, p.4)

De acordo com o guia de introdução ao Arduino (2018, tradução nossa), ele é uma plataforma de *hardware* e *software* fácil de usar e de código aberto onde é possível informar a placa o que fazer enviando instruções ao microcontrolador por meio da linguagem de programação baseada em *Wiring*.

De acordo com o Oracle (2020) o Java é uma linguagem de programação numero 1 do mercado, com milhões de desenvolvedores ativos e mais de 45 milhões de maquinas virtuais sendo executado pelo mundo, com ele é possível implementar um código para diferente plataformas como por exemplo: *Linux, Mac, Windows e Smartphone*.

Levando em consideração as situações listadas anteriormente e compreendendo que os veículos possuem sensores cujo responsabilidade é coletar dados de sua saúde, a presente proposta visa desenvolver uma ferramenta capaz de aproveitar os recursos do escaneamento *On-Board Diagnosis 2 (OBD2)* com as principais funções extraídas de *softwares* pagos da *Google Play*, transmitindo-os de forma transparente para o proprietário do veículo via *smartphone Android* e mantendo-o informado sobre possíveis problemas para poder ter percepção na tomada de decisão desta forma podendo diminuir as chances de acidente.

1.2 OBJETIVO GERAL

Realizar a prototipagem de um software contendo as principais funções dos aplicativos já existentes e não gratuitos na *Google Play* tornando-o acessível a qualquer mecânica de pequeno porte e proprietários de veículos automotivos.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho consistem em compreender o funcionamento da interface OBD2, compreender o sistema de monitoramento de automóveis, analisar *softwares* pagos da *Google Play* que utilizam o *OBD2*, desenvolver um protótipo amigável agregando as funções mais utilizadas dos softwares pagos escolhidos, analisar os resultados e validar o protótipo.

1.4 JUSTIFICATIVA

Segundo o censo do Detran (2018) referente ao ano de 2001, ocorreram 9.373 acidentes apenas em Santa Catarina causando 492 fatalidades, e o último censo do Departamento Estadual de Trânsito de Santa Catarina feito em 2005

ocorreram 12.727 acidentes e 543 fatalidades registrando um aumento de 35.78% na quantidade de acidentes em 4 anos.

Com a quantidade de veículos em circulação aumentando, os proprietários precisam fazer verificações periódicas para evitar falhas mecânicas e diminuir os acidentes e o risco de vida na estrada. O valor gasto para consertar o dano causado pelo sinistro ou uma multa, pode ser maior do que o valor de uma manutenção preventiva, as falhas mecânicas muitas vezes são identificadas apenas enquanto o veículo está em movimento.

Uma solução para identificar antecipadamente problemas no automóvel e minimizar o risco de acidentes seria utilizar o escaneamento OBD2 com um sistema embarcado para mostrar o estado do automóvel sem precisar estar no veículo, possibilitando ao proprietário analisar as informações recebida pelos sensores do veículo em seu celular após o uso do automóvel.

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa consiste em sete capítulos, onde o capítulo um aborda o contexto e os objetivos a serem alcançados em sua pesquisa. O capítulo dois a introdução a interface *OBD2*, sua história, os sensores que são reconhecidos por ele assim como a comunicação de dados entre eles. No capítulo três é onde estão as definições dos principais softwares pagos que utilizam a interface *OBD2* na *Google Play*, assim como seus diferenciais e as maiores dificuldades reclamadas pelos usuários nos comentários. No capítulo quatro está definido o levantamento bibliográfico sobre as tecnologia utilizadas para desenvolver a aplicação. No capítulo cinco os trabalhos correlatos, no capítulo seis as etapas do desenvolvimento do protótipo, sua metodologia e os recursos necessários, por fim no capítulo sete a conclusão.

2 INTRODUÇÃO AO OBD2

De acordo com Machado e Oliveira (2007) os primeiros módulos eletrônicos faziam utilizados por veículos faziam pouco mais que controlar a injeção eletrônica compara com os modelos de 2007, Entre as vantagens de se utilizar uma interface que se comunica com estes módulos está em conseguir informações se o veiculo esta funcionando conforme o projetado, economizando combustível ou dinheiro avisando ao condutor ou proprietário caso exista algum problema mecânico ou elétrico.

Ribeiro (2015) aponta que existem cinco protocolos que podem ser usados na interface *OBD2*, são eles *SAE J1850 PWM*, *SAE J1850 VPW*, *ISO 14230 KWP2000*, *ISO 15765 CAN*, sendo que a grande maioria dos fabricantes utilizem o primeiro mencionado e todos podem ser identificados facilmente apenas analisando quais pinos estão presentes no conector. O *padrão OBD2* especifica os protocolos de sinalização assim como os sinais que podem ser trocados e quais sensores monitorados pelo diagnóstico.

2.1 HISTÓRIA DO OBD2

O sistema de autodiagnostico foi desenvolvido pelo governo da Califórnia para ter um controle maior sobre a emissão de gases emitidos pelo combustível fóssil, pois na época em meados dos anos 50 existia muito nevoeiro (*smog*) gerado pela frota que estava alarmando o estado da Califórnia segundo Machado e Oliveira (2007).

Desenvolvido em 1970 nos Estados Unidos pela Agência de Proteção Ambiental (PAE) que até os dias atuais controla o que se diz respeito às normas que os fabricantes devem respeitar na comunicação com a interface *OBD*, *porem os proprietários de veículos criavam seus próprios padrões de conexão e não existia um padrão universal para todos. Foi então que a SAE emitiu um regulamento para a uniformidade dos conectores, o que levou na criação do OBD2 e uma homogeneidade no dispositivo que obtém informações de sensores instalados no veículo. Onde OBD2 começou a ser adotado como o padrão para avaliar o estado dos veículos nos Estados Unidos (MACHADO; OLIVEIRA, 2007).*

Foi com esse regulamento nos Estados Unidos da América que possibilitou implantar este sistema OBD em veículos destinados á Europa nos anos 90, e a partir de 2010 na frota brasileira com OBD2, sendo o *padrão* do OBD2 o protocolo *SAE J1962* no Brasil, consistindo em uma interface com dezesseis pinos formado por duas linhas de oito pinos que deve estar instalado junto a cabine do motorista.

2.2 MONITORAMENTO DE SISTEMA E DETECÇÃO DE FALHAS

Nos anos 90, Silva (2017) comenta que a *SAE* e a International Standardization Organization (ISO) emitiram informações para os fabricantes de veículos para utilizarem no conector o protocolo *SAE J1962* nos automóveis compatíveis ao OBD2. Segundo Machado e Oliveira (2007) o sistema OBD possuía sensores para monitoramento de oxigênio, recirculação do gás de escape, componentes eletrônicos, sistemas eletrônicos, informações de diagnósticos e códigos de erros para facilitar a identificação dos porque os veículos estavam emitindo poluição.

Ribeiro (2015) comenta que existem cinco protocolos que podem ser utilizados na interface OBD2, *SAE J1850 PWM*, *SAE J1850 VPW*, *ISO 9141-2*, *ISO 14230 KWP2000*, *ISO 15765 CAN*, todos estes podem ser facilmente diferenciados olhando quais pinos estão presentes.

Ribeiro (2015) complementa que o padrão *SAE J1979* define o monitoramento de sistema e detecção de falhas, que é o método *Diagnostic Trouble Codes* (DTC), são mensagens de requisição a rede de diagnóstico, que permite fazer consultas sobre o estado dos componentes do veículo, podendo receber notificações sobre os sensores além das já enviadas por defeito e dos alertas.

Os DTC são mensagens de quatro dígitos se iniciam em uma letra que serve para identificar qual o grupo de componentes a falha pertence, entre estas letras e grupos possíveis são *PowerTrain (P)* que indica uma requisição acerca da transmissão e/ou motor, *Body (B)* indica sensores da cabine, que pode corresponder ao estado do autorrádio, luzes, ar-condicionad, existe também a letra correspondente ao Chassis (C) que indica as requisições sobre de chassis, como travões, velocidade das rodas, suspensão, e pôr fim a letra que representa o *Network (U)* que precede pedidos acerca do estado da rede de diagnóstico.

O sistema OBD2, segundo Belo (2003), utiliza uma estratégia denominada “Monitores”, para testar as operações do sistema, componentes ou funções específicas do veículo, que estabelece um controle muito rígido sobre os parâmetros operacionais dos componentes do sistema, principalmente os relacionados aos poluentes.

Para se comunicar com o veículo, é necessário enviar comandos para o escâner, entre os comandos existentes um se destaca, o 03 que analisa as falhas detectadas pelo DTC, no retorno deste comando pode-se receber o valor 43 01 33 por exemplo, em que o valores 43 indicam que foi solicitado o comando 03, e os outros valores o código de erro, sendo necessário substituir o primeiro valor para encontrar a letra identificadora de acordo com a tabela 1. (ELM Eletronics, 2008, tradução nossa).

Tabela 1. Valores a serem substituídos no retorno do OBD.

0	P0
1	P1
2	P2
3	P3
4	C0
5	C1
6	C2
7	C3
8	B0
9	B1
A	B2
B	B3
C	U0
D	U1
E	U2
F	U3

Fonte: ELM Eletronics, 2008, tradução nossa.

Os números retornados *0133*, o primeiro valor da esquerda para direita é *0* então deve ser substituído para *P0* como mostra na tabela 1, para concatenar o resto dos números nos dando o código *P0133*, que significa que a resposta do sensor de oxigênio está baixa e este mesmo processo deve ser efetuado com todos os códigos de falhas retornados pelo comando *03*.

2.2.1 Monitor da eficiência do catalisador

Os catalisadores de acordo com Boldt e Silva (2014) são responsáveis pela diminuição dos principais poluentes da queima de combustíveis fósseis que causam graves ações prejudiciais à saúde humana. Sendo um dos mais importantes monitores do automóvel em relação a poluição, segundo Belo (2003) um catalisador trabalhando de forma eficiente consegue reduzir os poluentes produzidos por motores a álcool e gasolina em 70%.

Tal monitoramento segundo o mesmo autor é efetuado usando a medida de dois sensores de oxigênio aquecido, que se encontram nas saídas do catalisador, monitorando o nível de oxigênio que entra no catalisador comparando com o nível de oxigênio que sai no outro lado do catalisador, onde a maioria das moléculas de oxigênio lidas pelo primeiro sensor deveriam ser gastas durante o processo de oxidação efetuado pelo conversor do catalisador. Caso os valores reconhecidos pelos monitores do catalisador estejam em níveis fora do padrão estabelecido para o automóvel, é realizado o armazenamento da informação e iluminado a luz de avaria que normalmente se encontra no painel do automóvel.

2.2.2 Monitor de falhas de ignição

Outro fator que consegue provocar um aumento considerável de poluentes pelo automóvel é a falha de ignição que pode causar falhas de combustão no motor, segundo Belo (2003) consiste na falta de combustível no cilindro do automóvel causada pelo desgaste dos componentes da ignição ou por falhas do sistema elétrico ou por uma mistura de pouco combustível, ou seja com muito ar e pouco combustível.

Estas falhas podem ser monitoradas como comenta Salamani (2018) observando a amplitude de duas frequências, sendo a primeira a frequência de

ignição e de explosão dentro da câmara de combustão do motor, e a segunda é com a do componente da velocidade do motor em um período de 720 graus para um motor de quatro cilindros, já que o cilindro é impulsionado a cada 720 graus de rotação do virabrequim.

2.2.3 Monitor do sistema de combustível

O sistema de combustível é monitorado constantemente em cada rotação do motor, caso ocorra uma falha o sistema de controle adiciona fatores de correção o que mantêm o carro funcionando só que aumenta o nível de poluição gerado pelo motor, como comenta Belo (2003) o controle do sistema de injeção eletrônica de combustível tem como estratégia manter a relação correta de ar e combustível, exceto em partidas a frio, os parâmetros e componentes envolvidos no controle são o sistema de medida de combustível, bomba de combustível, temporização da ignição, injetores de combustível, largura do pulso de injeção e controle da mistura estequiométrica, onde a largura do pulso de injeção necessário para manter a mistura correta entre ar combustível é determinado por uma faixa de valores próxima desta largura.

2.2.4 Monitor dos sensores de oxigênio aquecido

Como os sensores de oxigênio são utilizados para o monitoramento do catalisador, no controle da mistura de ar e combustível para o sistema de combustível, os sensores de oxigênio são de suma importância para controlar a emissão de poluentes do automóvel. Estes sensores como diz Belo (2003) realiza alguns testes, como o teste de continuidade de circuito, o teste dos aquecedores e o teste de operação com motor desligado e funcionando, caso algum destes testes falhe é armazenado o código de erro pelo sistema OBD2 e iluminado a luz de avaria.

2.2.5 Monitor do sistema de controle de evaporação de combustível

A evaporação de combustível causada principalmente pelo aumento de temperatura no ambiente ou pela vazão de combustível não utilizada pelo motor é uma das principais fontes de emissão de hidrocarboneto, sendo que a técnica

utilizada para controlar a evaporação do combustível consiste em uma ventilação que sai do tanque que leva o vapor para um elemento de carvão que consegue absorver os poluentes e libera somente o ar para a atmosfera. Assim Belo (2003) comenta que este monitor consegue descobrir vazamentos em qualquer ponto do sistema, graças ao uso de sensores de pressão modificados no tanque de combustível, e caso qualquer falha é iluminado a luz de avaria.

2.3 REDES DE COMUNICAÇÃO DE DADOS DIAGNÓSTICOS

A arquitetura de rede do sistema OBD2 é baseado no modelo OSI que como comenta Pereira e Yamada (2018), é um modelo de rede que divide as funções da rede em camadas, cujo objetivo é ser um padrão entre os mais diversos sistemas de rede. Sendo as camadas do modelo OSI: Física onde fica o hardware, enlace de dados que é onde é produzido a estrutura dos dados e verificado se existem erros, a camada de rede onde é realizado o roteamento de funções e também a fragmentação e a remontagem, camada de transporte que é responsável por receber os dados de sessão e segmentá-los para que sejam enviados para a camada de rede e vice-versa caso esteja voltando, camada de sessão que é responsável pela troca de dados e a comunicação entre *hosts*, a camada de apresentação, também chamada de camada de Traduções, que converte o formato do dado recebido pela camada de aplicação para um formato que possa ser entendido pela camada de aplicação, e por último a camada de aplicação, que é a interface entre a unidade de comunicação e a aplicação real.

Alguns dispositivos utilizam partes do modelo OSI, e a maneira que o OBD2 o utiliza de acordo com Belo (2003) com a camada de aplicação sendo responsável de apresentar os serviços disponíveis para o usuário, construído de acordo com as camadas anteriores e a camada de enlace responsável de receber os dados da camada física transformando os dados em mensagens de detecção de erros durante a comunicação.

2.3.1 Serviços de comunicação

É na camada onde fica o hardware e a de enlace á responsável pela estruturação dos dados, reconhecimento das mensagens, erros e transmissão de

dados pelo meio físico, essas camadas podem possuir diferentes protocolos como *ISO 9141-2*, *ISO 14230*, *SAE J1850 PWM* ou *SAE J1850 VPW*, mesmo assim para manter sua compatibilidade com diferentes protocolos de acordo com Belo (2003) um dos principais requisitos do sistema *OBD2* é a capacidade de interação com todos os sistemas *protocolos*, onde a ferramenta possui implementado todos os protocolos referentes as camadas e um mecanismo que detecta qual protocolo usar em um sistema específico. Sendo os mais comuns utilizados para a comunicação entre celulares e os *Escâneres*, o *Bluetooth* e *IEEE 802.11* (Wi-Fi).

Estes dois protocolos mais utilizados definem uma camada física e uma camada *MAC* para comunicações sem fio de curto alcance e baixo consumo de energia. *Bluetooth* é orientado a conectar a dispositivos próximos servindo como substituto à cabos, enquanto o Wi-Fi é orientado a conexões de computador para computador, como uma extensão ou substituição para cabos de rede de área local (LAN) (FERRO; POTORTI, 2005, tradução nossa).

3 PRINCIPAIS SOFTWARES PAGOS NA GOOGLE PLAY QUE UTILIZAM OBD2

Todos os fabricantes de automóveis da atualidade possuem sistemas que ajudam o motorista a dirigir de forma mais eficiente e consciente do estado de seu veículo, porém sua grande maioria é simples ou somente mostram os dados no pós- viagem para ser analisado, exibindo informações gerais sobre as resoluções do motorista e permitindo que se faça uma comparação da direção com outros motoristas. Entre esses aplicativos existe o *eco: Drive Live*, um aplicativo comercial da FIAT que fornece conselhos em tempo real sobre quatro indicadores, a aceleração, desaceleração, mudança de marcha e a velocidade, no entanto eles não dão conselhos antecipados e também funcionam somente no veículo, não sendo possível utilizá-lo no *smartphone* (ORFILA; PIERRE; MESSIAS, 2015, tradução nossa).

O retorno do aplicativo em tempo real das condições dos veículos é um dos motivos de alguém comprar o dispositivo com a interface OBD2, e foram escolhidos os 2 softwares pagos mais baixados e votados da *Play Store*, a loja oficial utilizada pelo *Android* para se entender quais as principais funções e as dificuldades dos usuários Android. Foi então escolhido o software *Torque Pro (OBD2 / CARRO)* de R\$ 10,99, com mais de 60.000 *downloads* e o software *OBD Fusion (Car Diagnostics)* de R\$ 19,99, com mais de 1.000 *downloads*.

3.1 PRINCIPAIS FUNÇÕES DO SOFTWARE

As principais funções que se pode encontrar nos dois *softwares* são exibir os dados de falha do veículo, como também apresentando informações do torque do motor, a potência, leitura de emissão CO₂, cálculo de economia de combustível e um banco de dados com os códigos de falhas para pesquisa local do usuário.

3.1.1 Diferenciais dos principais softwares

As funções consideradas diferenciais foram definidas analisando os requisitos que os dois aplicativos diferenciam entre si, que são limpar o código de problema de diagnóstico, exportar os dados coletados do veículo em um arquivo CSV para leitura em planilha posteriormente, log do trajeto do veículo e seus dados

como uma caixa preta, rastreamento por GPS e informação da voltagem da bateria do veículo.

3.1.1 Dificuldades dos principais softwares

As principais dificuldades dos aplicativos escolhidos foram definidas através de reclamações dos usuários na página de comentários da *Google Play*, entre elas está a dificuldade de acessar após a compra, travamento do aplicativo, código de falha não encontrado, dificuldade ao se conectar ao *OBD2*, não possui nenhuma documentação ou guia de como deve-se iniciar a utilização do aplicativo, atraso entre a informação que aparece no painel do veículo e no celular, e não está em português.

4 APLICAÇÃO PARA DISPOSITIVOS MÓVEIS ANDROID

Android é o sistema operacional *open source* desenvolvido pela *Google* para dispositivo móvel mais utilizado no mundo, de acordo com o Netmarketshare (2019) desde janeiro de 2018 até outubro de 2019 o Android manteve-se com mais de 60% do mercado de usuários entre os dispositivos móveis.

Entre os dispositivos que rodam Android, a Google disponibiliza um painel sobre a distribuição das versões do Android mais utilizadas pelos usuários que possuem o sistema, sendo a versão *Oreo*, *Nougat* e *Marshmallow* os mais utilizados em dados coletados até dia 7 de maio de 2019. É importante saber as versões mais utilizadas para se utilizar as mais novas tecnologias da API do Android e saber até qual versão seria mais vantajoso lançar o aplicativo.

Para o desenvolvimento da aplicação Android e web será utilizado a metodologia ágil *Scrum*, e no gerenciamento do projeto as tecnologias *Flutter*, *Java*, *Play Framework*, *vueJS* e *MariaDB*.

4.1 METODOLOGIA DE DESENVOLVIMENTO

O *Scrum* como comenta Borges (2017) é uma metodologia que utiliza pequenos ciclos de entrega chamado *sprint*, que varia dependendo do tamanho e complexidade da atividade a ser desenvolvida, com isto é possível afirmar que o *Scrum* é uma metodologia adaptável, visto que a cada novo *sprint* pode-se fazer ajuste, criar uma nova atividade com correções, alterações e adições do que precisa ser feito.

4.2 MODELAGEM DO BANCO DE DADOS

A modelagem de dados de acordo com BORGES et al. (2005), procura sistematizar de forma abstrata o entendimento das funções e objetos que representarão as informações salvas em um sistema desenvolvido.

4.3 LINGUAGEM DE PROGRAMAÇÃO

Ferting e Medina (2006) comentam que algoritmo não é um termo restrito da computação, que o termo é utilizado em outras áreas como na engenharia e administração, e que é um procedimento passo a passo para a solução de um problema ou uma sequência detalhada de ações a serem executadas para a solução de um problema. As linguagens de programações a serem utilizadas nesta pesquisa vão ser *Dart* com *Flutter*, *Java* para web com o *Play Framework* para o *backend* junto de *Mysql* e *Vuejs* para o *frontend*.

4.3.1 JAVA

O Java é uma linguagem de programação reflexiva e orientada a objetos, desenvolvida inicialmente por James Gosling e colegas da Sun Microsystems, que pretendia substituir o C++, embora o conjunto de recursos se assemelhe melhor ao Objective-C. As especificações da linguagem Java, da Java Virtual Machine (JVM) e da API Java são mantidas pela comunidade por meio do Java Community Process gerenciado pela Sun (ORACLE, 2019, tradução nossa).

4.3.2 Flutter

Flutter é um kit de ferramentas criado pela Google para a criação de interfaces e aplicativos compilados nativamente para dispositivos móveis, Web e Computadores a partir de uma única base de código, o Dart (FLUTTER-DEV, 2019, tradução nossa), uma linguagem criada pela mesma empresa com foco em produtividade (GOOGLE, 2019, tradução nossa).

4.3.3 Play framework

O *Play framework* teve sua primeira versão lançada em 2007 e deste então de acordo com a documentação do *Play 2.7* é construído para aumentar a produtividade em aplicações desenvolvidas em Java e *Scala* para web, que integra componentes e API para o desenvolvimento de modernas aplicações web. Podendo usar como arquitetura padrão do tipo Modelo, Visualização, Controlador (MVC), que

é uma arquitetura muito usada no desenvolvimento web (LIGHTBEND, 2019, tradução nossa).

4.1.4 Vuejs

Assim como o *Play Framework*, o Vuejs é um framework de código fonte aberto criado para aumentar a produtividade no desenvolvimento de aplicações, porém de acordo com a documentação do Vuejs, ele foi projetado desde sua concepção para um desenvolvimento escalável, focando sua biblioteca principal na camada visual, sendo capaz de fazer aplicações modernas em *Single-Page* (do português Página única), *VueJS* tem um foco no *frontend* (EVAN YOU, 2019, tradução nossa).

4.1.5 Mariadb

De acordo com a documentação oficial do *MariaDB*, ele é um dos bancos de dados relacionais mais populares no mundo, foi criado pelo desenvolvedor original do MySQL, e com garantias de ficar com o código fonte aberto. Empresas notáveis que o utilizam são *Wikipedia*, *WordPress* e *Google* (MARIADB FOUNDATION, 2019, tradução nossa).

4.5 COMUNICAÇÃO WEB SERVICE

Com a utilização de serviços web ou em inglês *web services* é possível realizar a comunicação entre diferentes sistemas escritos em diferentes linguagens, que podem ser definidas como objetos de softwares que podem ser montados pela internet se comunicando por algum protocolo padrão para executar função ou processos de negócios, criando assim a reutilização de componentes de softwares fracamente acoplado (FENSEL; BUSSLER, 2002, tradução nossa).

Sendo assim possível ter um sistema com diversos serviços altamente escalável que permita a atualização de seus serviços sem precisar mexer no núcleo do sistema que os utiliza, e a implementação destas aplicações são rapidamente implementáveis, oferecem aos desenvolvedores a possibilidade de reutilização e aos

usuários o acesso contínuo a uma variedade de serviços complexos (MILANOVIĆ; MALEK, 2004, tradução nossa).

Entre os protocolos utilizados para o consumo de *web services* entre *smartphones* e web, se destaca Transferência Representacional de Estado (*REST*), pois são mais fáceis de consumir porque o cliente e o servidor utilizam um protocolo simples de chamada e resposta, eliminando o requisito de análise de metadados, como o Web Service Description Language (*WSDL*) com *Extensible Markup Language (XML)* (CHRISTENSEN, 2009, tradução nossa).

5 TRABALHOS CORRELATOS

Para a realização desta pesquisa foram encontrados trabalhos relacionados ao tema para servir de exemplo na elaboração e decisão dos capítulos, em que estes trabalhos não são iguais, mas próximos do tema, que seria a pesquisa do protocolo OBD2 e a sua utilização na prototipagem de um aplicativo para celular, que visa auxiliar o motorista na manutenção de seu veículo.

5.1 UM SISTEMA DE ASSISTÊNCIA AO ECODRIVING EM ANDROID PARA MELHORAR A SEGURANÇA E EFICIÊNCIA DOS CARROS COM MOTOR DE COMBUSTÃO INTERNA

No artigo é apresentada a elaboração completa de uma aplicação *ecodriving* para celular descrevendo todas as principais funções que já existem, como a estimativa de nível ecológico, a modelagem de combustível, indicador de mudança de marcha, então completando com a adição de funções que dão retorno ao motorista em tempo real, como otimizações de velocidade, e notificações antecipadas.

Este artigo foi desenvolvido por Olivier Orfilia, Guillaume Saint Pierre e Mickael Messias hospedado na base de dados da ScienceDirect em 2015 onde é comentado que existem diversas aplicações para monitoramento e auxílio ao *ecodriving*, uma prática de direção onde é minimizado o consumo de combustível e emissão de dióxido de carbono. Porém estas aplicações não são destinadas a aparelhos móveis como um celular e não possuem nenhuma resposta de retorno em tempo real para o motorista, elas são utilizadas para que o motorista saiba se está dirigindo de maneira econômica e permite comparar com outros motoristas em um tipo de ranking.

Na conclusão os autores estavam convencidos que ferramentas em dispositivos móveis podem melhorar a segurança no trânsito, eles comentam que foram realizadas várias experiências em diferentes países europeus dirigindo com e sem o aplicativo, porém não foram disponibilizados os resultados até a publicação do artigo. É apresentado também como sendo possível enriquecer o aplicativo dinamizando as notificações dadas ao motorista com estatísticas de dados coletados

da condução do próprio motorista (ORFILA; PIERRE; MESSIAS, 2015, tradução nossa).

5.2 SISTEMA DE COMUNICAÇÃO VEICULAR UTILIZANDO SMARTPHONE

Neste artigo escrito por Alexandra Elisabeta Lorincz, Marius Cucaila e Charles Rostand Mvongo Mvodo, é apresentado o funcionamento do protocolo OBD2 para o envio de informações para o smartphone, que este protocolo possui cinco tipos de comunicação para descriptografar as mensagens, as vantagens e desvantagens de cada um, sendo *Control Area Network* (CAN), o protocolo mais utilizado.

Na conclusão desta pesquisa foi apresentado que o CAN é o mais moderno entre eles, comparado com os outros, este possui uma velocidade muito alta para a transferência de dados e o cabeamento tem custos muito baixos, podendo até funcionar em más condições. Mas também que os protocolos mais antigos possuem a vantagem do tempo, por serem antigos é mais fácil encontrar conteúdo sobre a compreensão dele. Ao final os autores afirmam que a comunicação com seu próprio carro ao celular em tempo real é uma vantagem. (LÖRINCZ; CUCĂILĂ; MVODO, 2016, tradução nossa).

5.3 USO DE DIÁRIOS DE BORDO PARA MITIGAR O PERIGO EM ROTAS E O COMPORTAMENTO ARRISCADO DOS MOTORISTAS IDOSOS

Este artigo publicado por Payyanadan et al. (2017) que fizeram uma pesquisa com 33 motoristas com 65 anos ou mais em veículos com OBD2, na utilização de um aplicativo que retornava informações de diário de bordo para os motoristas mais idosos, de acordo com os autores os motoristas idosos regulam seu comportamento enquanto estão dirigindo, na escolha de rotas menos desafiadoras para as capacidades de uma pessoa mais velha.

Foi criado um aplicativo web que fornecia informações pós viagem das rotas percorridas, fazendo um teste de quatro meses com resultados a favor do aplicativo web, pois reduziu o risco de acidentes estimado dos motoristas mais idosos para 2,9% e sua frequência de velocidade média em 0,9% por semana, no geral, o aplicativo conseguiu reduzir a taxa de acidentes esperada de 1 em 6172

viagens para 1 em 7173 viagens, e a frequência de velocidade esperada de 46% a 39%, concluindo-se que oferecendo aos motoristas mais idosos um retorno de seu comportamento ao dirigir e dos riscos de acidente, podem ajudá-lo a se autorregular mais efetivamente seu comportamento e dirigir diminuir as chances de colisão (PAYYANADAN et al., 2017, tradução nossa).

5.4 SISTEMA DE AQUISIÇÃO DE DADOS BASEADOS EM STM32 PARA INTERNET DE VEÍCULOS

Este artigo escrito por Xie et al. (2017) publicado na base de dados da IEEE é sobre o desenvolvimento de um protótipo de um dispositivo baseado em STM32 que recebe as informações através da interface ODB2 e em seguida apresenta o retorno que recebeu pela interface em uma tela de LED, para saber a viabilidade e segurança dos dados que são enviados pela internet que permitiria a realização de diagnóstico online em tempo real.

No final desta pesquisa foi concluído pelos autores que é possível adquirir dados pela interface OBD2 em tempo real, garantindo diagnóstico online e possíveis detecções de ameaças à segurança dos motorista e seus passageiros, os autores comentam também que como trabalho futuro planejam testar enviar esses dados em tempo real por meio de Wi-Fi baseado em 4G/IPv6 (XIE et al., 2017, tradução nossa).

5.5 SISTEMA DE MONITORAMENTO REMOTO DE SAÚDE E MANUTENÇÃO PROGNÓSTICA DE VEÍCULOS

Este artigo apresenta uma abordagem de previsão de falhas em quatro principais subsistemas do veículo, sistema de combustível, sistema de ignição, sistema de escapamento e sistema de refrigeração. Utilizando a interface OBD2 coletando dados dos sensores quando o veículo está em movimento, tanto quando o veículo está com problema enviando os dados para um servidor que irá fazer a análise e utilizar métodos de aprendizado de máquina para encontrar padrões e detectar futuras falhas nos veículos que demonstrarem os mesmos padrões. Tendo como objetivo aumentar o tempo de atividade dos veículos como foi demonstrado em 70 veículos da Toyota do tipo Corola, os autores propõem como trabalho futuro

fazer o mesmo teste em veículos de outras marcas e declaram que não houve conflito de interesse na publicação de seu artigo (SHAFI et al., 2018, tradução nossa).

5.6 SCANNER OBD-II EM PLATAFORMA LABVIEW

Esta monografia realizada na Faculdade de Tecnologia de Santo André apresenta os principais conceitos sobre diagnósticos das normas *OBD*, criando seu próprio *escâner* automotivo que utiliza a interface *OBD*, que é gerenciado por uma *interface* gráfica criada pela plataforma *LabView* em um computador. Neste artigo observou-se que o sistema de diagnóstico OBD2 é mais eficiente na detecção de falhas em relação a sua primeira versão. Visto que o *OBD* permite que o veículo fique operando por mais tempo em condições que favoreciam o aumento de emissão de poluentes (ALMEIDA; FARIA, 2013).

Os autores deste trabalho correlato ao final apresentaram que tiveram êxito na criação de um hardware que possui um circuito integrado ELM327, e que atende as especificações da interface OBD2, visando auxiliar a *interface* homem máquina (ALMEIDA; FARIA, 2013).

5.7 SISTEMA DE MONITORAMENTO AUTOMOTIVO REMOTO

Este projeto de pesquisa da Faculdade de Tecnologia de Santo André em São Paulo foi desenvolvido um protótipo que efetua a leitura pela interface *OBD*, utilizando um microcontrolador para fazer o tratamento de dados e então enviar para um servidor web para ser apresentado através de um navegador na internet, permitindo que o usuário consiga monitorar a situação do seu veículo pela web a qualquer momento. Em sua conclusão os autores conseguiram efetuar a leitura, porém não conseguiram enviar os dados para um servidor e ser monitorado de qualquer local por falta de conhecimento sobre tecnologias web, os autores desta monografia fizeram a transmissão de dados lidos via Serviço de Mensagens Curtas (*SMS*), obtendo sucesso (FAGUNDES; DA SILVA; ASSIS, 2015).

6 FERRAMENTA ANDROID PARA O AUXÍLIO TÉCNICO AUTOMOBILÍSTICO

Visando auxiliar a ter menos acidentes nas estradas e manter os motoristas alertas com a manutenção de seus veículos, apresentou-se um protótipo de aplicativo com as principais funcionalidades de softwares pagos que utilizam a interface OBD2 para auxiliar os motoristas proprietários de veículos a manter seu veículo em melhores condições antes de saírem de casa. Para isto o processo metodológico da pesquisa foi dividido em sete etapas, sendo elas: levantamento de requisitos, modelagem de dados, desenho das telas a serem desenvolvidas, implementação da aplicação *mobile*, testes de conexão *Bluetooth* para enviar e receber informações via serial, testes em um veículo real com auxílio de um mecânico, e por fim, a análise dos resultados obtidos por meio da aplicação *mobile*

Para a criação do aplicativo com intuito de suprir as principais dificuldades relatadas pelos usuários que utilizaram os aplicativos pagos estudados na pesquisa bibliográfica, foi utilizado tecnologias modernas altamente escaláveis da própria Google como flutter e a *IDE Android Studio*, juntamente, para gerenciar a conexão *Bluetooth* e nos permitir conectar ao *escâner* será utilizada a biblioteca *open source flutter_bluetooth_serial* criada pelo desenvolvedor Eduardo Folly.

O protótipo oferece para os proprietários de veículo um meio de obter informações relevantes sobre o estado se deu veículo, utilizando o próprio celular junto de um equipamento barato, estas informações podem evitar com que o veículo parem de funcionar no meio de uma viagem ou até cause um acidente.

6.1 METODOLOGIA

Esta pesquisa se iniciou como uma proposta e o levantamento bibliográfico para determinar sua viabilidade para aprovação, como a interface OBD2 funciona, suas respostas, sensores reconhecidos que podem ser lido pelos protocolos, como se conectar, tecnologias a se utilizar, quais os softwares pagos mais utilizados na loja principal do Android, no caso tudo que constitui o levantamento bibliográfico e referencial teórico.

Por meio do levantamento bibliográfico, descobriu-se a biblioteca *open source flutter_blue* para realizar a conexão *Bluetooth* por serial e trocar informações, porem observou-se que a biblioteca em questão possuía deficiências de conexão

com *Android*, por isso, entre as bibliotecas testadas escolheu-se a biblioteca open source *flutter_bluetooth_serial*, que conseguia suprir todas as necessidades, e foi criada justamente por alguém que precisava utilizar a conexão *Bluetooth* para aplicações destinada a internet das coisas no flutter e o *flutter_blue* não atendia a sua necessidade.

O desenvolvimento do protótipo foi dividido em cinco etapas, sendo a primeira o desenho das telas do aplicativo *Android*, então com base nas telas foi iniciado a segunda etapa, a criação de um modelo relacional.

Consequentemente na terceira etapa se realizou o desenvolvimento da aplicação *Android* com o *Android Studio* e seus emuladores. Com os desenvolvimento das telas foi iniciado a quarta etapa onde será necessário buscar um *hardware* para testar a conexão *Bluetooth* sem precisar de um veículo real, nesta etapa foi utilizado um *Arduino Mega 2560*, e um módulo *Bluetooth HC-05*. Para que então se inicie os testes com serial *Bluetooth* com um dispositivo *Android* real e um *Arduino*, visto que os emuladores disponibilizados pela *Google* no *Android Studio* não possuem função de *Bluetooth*. Na conclusão desta atividade, inicia-se a quinta etapa, a realização dos testes de leitura em um veículo real, sendo assim necessário um escâner OBD2, especificamente o *ELM327* e um veículo do ano 2010 ou superior, nos testes foram utilizados um celta 2010 e um x60 2016.

Planejado então o momento de realização de cada etapa por atividades em *Sprints* seguindo a metodologia ágil *Scrum*, que a ultima atividade foi realizada outro teste gravado em video para então feito o relatório de conclusão do projeto.

6.2 RECURSOS NECESSÁRIOS

Para a realização do desenvolvimento foi necessário comprar os softwares de escâner OBD2 mais utilizados na *Play Store* com isso foi possível validar o que está desenvolvido com o que os aplicativos já fazem, assim como comprar um *Arduino mega*, módulo *Bluetooth* e escâner *ODB2 ELM327* para ser utilizado junto de um *smartphone* *Android*.

Também sobre as ferramentas de desenvolvimento do protótipo, foi definida a linguagem *open source* da *Google*, *Dart* que é utilizada pelo flutter por meio da *IDE* gratuita *Android Studio 4.0*. E para o desenvolvimento do emulador *Bluetooth* com *Arduino* e o módulo *Bluetooth* foi necessário a utilização da *IDE*

gratuita *arduino-ide* feita em java, sendo a implementação utilizada para o *Arduino* a linguagem do *Arduino* que se assemelha com C apresentada nos primeiros semestres da instituição.

O *LogCat* é uma ferramenta do *Android Studio* disponibilizada pelo Kit de Desenvolvimento de Software e tem funcionalidades de logs para a aplicação. Esta ferramenta auxiliou na depuração da aplicação mobile para encontrar processos falhos no desenvolvimento do protótipo.

Precisa-se de um computador com no mínimo oito gigabyte de RAM para o desenvolvimento da aplicação junto de seus emuladores. A implementação será feita por escolha do autor e já estarem a disposição quando foi realizada a pesquisa, no Arch Linux com o *Kernel* 5.7.6 e *Debian* 10 com *Kernel* 4.19.0, ambos com GNOME, que são todos *softwares* gratuitos e livres.

Para finalizar utilizou-se o *software Figma*, para a criação dos *mockups* e desenhos das telas do aplicativo mobile.

6.3 FUNCIONAMENTO DA APLICAÇÃO

Após a definição dos recursos foi necessário um estudo para entender o funcionamento da detecção de falhas pela unidade de controle dos veículos, quando ele detecta, como causar uma falha, como capturar a falha e como limpar as falhas da unidade de controle, seus comandos e retornos podem ser vistos na tabela 2.

Tabela 2. Tabela de comandos utilizados no aplicativo

Comando	Resposta	Comentário
ATZ	ELM327 V2.1 >	Resetar – versão do software
ATSP0	OK >	Configura protocolo para automático
ATDP	AUTO >	Retorna o protocolo setado
0101	41 01 81 04 00 00	Retorna quantas falhas detectadas desde a última vez limpado, incluindo se a luz de avaria está acessa, e os testes realizados
03	43 00	Retorna a quantidade de falhas, 43 00 significa sem falhas

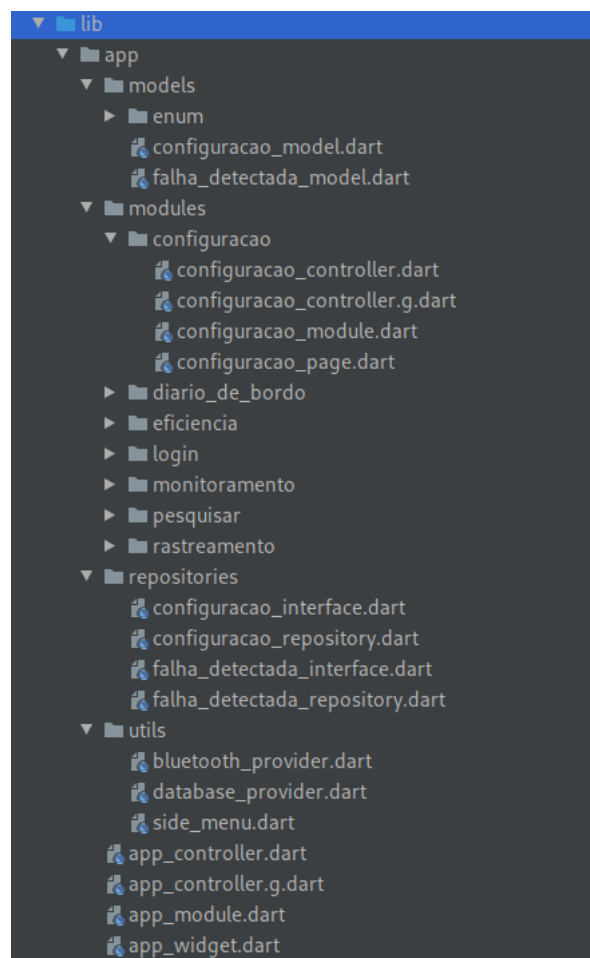
Fonte: Do autor

Os valores retornados pela OBD2, principalmente o comando para identificar se a luz de avaria e quantos erros estão acessos, estão codificados utilizando a numeração de bits, especificamente o cálculo de byte mais importante, neste caso para conseguir entender o que estamos recebendo, precisa-se decodificar os valores hexadecimais para binário e identificar o que cada posição dos bits significam e retornar para o proprietário do veículo uma informação em português sobre a situação do veículo.

6.4 DESENVOLVIMENTO DO PROTÓTIPO

Para o desenvolvimento do protótipo foi pensado um uso para a aplicação que facilitasse bastante no escaneamento efetuado pelo proprietário do veículo, a estrutura do projeto foi gerenciado pelo slidy, uma ferramenta de linha de comando para gerenciar o projeto, na criação do projeto foi escolhido a utilização de flutter modular, que significa que o projeto será separado em módulos, junto do mobx, que é um sistema de gerencia de estados moderno e pratico, criando uma estrutura como apresentado na figura 1.

Figura 1. Estrutura de arquivos do protótipo móvel



Fonte: Do autor

Em que exista um diretório de modelos chamado *models*, no diretório *utils* as classes de *provider*, e objetos utilizados em diversos módulos como a conexão de Bluetooth, menu lateral e conexão do banco. O diretório *modules* é o mais importante, pois ele está dividido em recursos e cada um possui *View*, *Controller* e um objeto de injeção de dependências representando o módulo do recurso, que recebe o nome do recurso mais *module* ponto dart.

A primeira tela vista ao acessar o protótipo é a tela de login que poderá ser autenticada pela própria conta do Google, colocando o e-mail de usuário e sua senha, como mostra na figura 2.

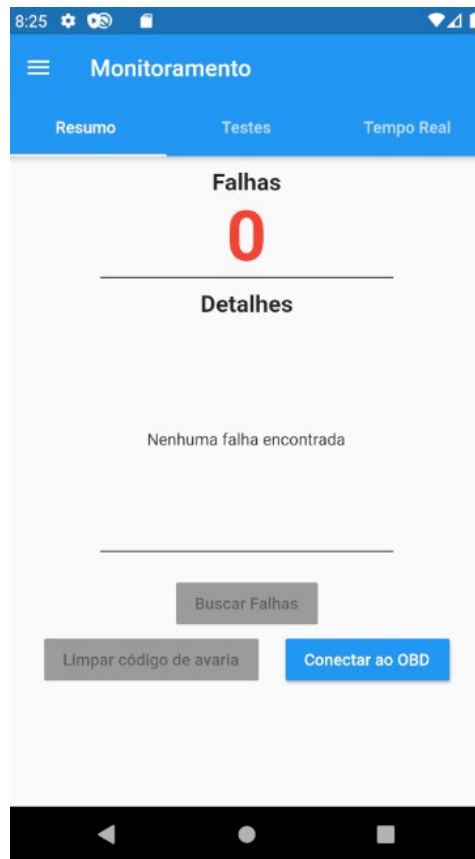
Figura 2. Tela de login



Fonte: Do autor

Após isso será direcionado para a tela de monitoramento, onde estarão as funcionalidades principais do aplicativo, entre elas se conectar ao *escâner* OBD2 por *Bluetooth*, pelo botão “Conectar ao OBD” como é possível observar na figura 3, que pode se observar a tela de monitoramento.

Figura 3. Tela de monitoramento antes de se conectar ao OBD2



Fonte: Do autor

Entretanto, para que o aplicativo possa utilizar as funções de *Bluetooth* do Android, é necessário ficar atento a questões de segurança adotadas pela Google, onde é necessário pedir a permissão do usuário do dispositivo para que o aplicativo utilize o módulo de *Bluetooth* do celular. No *Flutter*, foi só necessário adicionar a exigência do aplicativo no manifesto do Android (figura 4), fazendo assim com que quando o aplicativo requisitasse a utilização do *Bluetooth*, aparecesse uma janela automática perguntando ao usuário Android se ele deseja permitir o protótipo utilizar o seu módulo *Bluetooth*.

Figura 4. Manifesto com permissão de acesso ao Bluetooth

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="com.example.tcc_mobile">
3
4   <uses-permission android:name="android.permission.BLUETOOTH" />
5   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
6   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
7
8   <application
```

Fonte: Do autor

É possível acessar as informações do módulo *Bluetooth*, foi criado utilizando um padrão de projeto chamado *Singleton* para gerenciar a conexão *Bluetooth* e não permitir existir mais de um objeto gerenciando a conexão ao mesmo tempo, neste objeto existem as funções de buscar dispositivos *Bluetooth*, se conectar ao OBD2, se desconectar ao OBD2, um socket que fica retornando as mensagens do OBD2 conectado, e a função para enviar informações ao OBD2 conectado (figura 5).

Figura 5. Gerenciamento da conexão e envio de dados para o escâner

```
1 BluetoothProvider._privateConstructor();
2
3 static final BluetoothProvider instance = BluetoothProvider._privateConstructor();
4 static FlutterBluetoothSerial _flutterBlueS;
5 BluetoothConnection connection;
6 BluetoothDiscoveryResult _resultDeviceConnected;
7 StreamSubscription<BluetoothDiscoveryResult> _streamSubscription;
8
9 Future<FlutterBluetoothSerial> get getFlutterBlueS async {
10     if (_flutterBlueS != null) return _flutterBlueS;
11     _flutterBlueS = FlutterBluetoothSerial.instance;
12     return _flutterBlueS;
13 }
14
15 sendCommand(String data) async {
16     if(connection != null) {
17         connection.output.add(utf8.encode(data + "\r\n"));
18         await connection.output.allSent;
19     }
20 }
```

Fonte: Do autor

Como pode ser observado na figura 5, no *provider BluetoothProvider*, sua instancia é *final e estática*, e quando é chamado a função *getFlutterBlueS* é feito uma verificação se a variável estática *_flutterBlueS* já está instanciada, caso sim ele retornara a existente, caso não, será retornada uma instancia nova do objeto da classe *FlutterBluetoothSerial*, que ficara já com seu endereço de memória salvo na atual instancia *Singleton* de *BluetoothProvider*. Neste mesmo *provider* apresentado na figura 5 pode ser observado como é enviada informação por *Bluetooth para o OBD2*, onde primeiro é verificado se existe alguma conexão ativa. No comando enviado é necessário enviar junto dos dados a informação *\r\n* que representa a quebra de linha, pois sem estes dados no final do comando o OBD2 não sabe o que

retornar e ficaria retornando `>comando ?`, isto não estava documentado em lugar algum na metodologia, e atrasou bastante no momento dos testes em veículo real.

Ainda sobre o envio de comando, a variável de conexão que é verificada é criada no momento que é encontrado um dispositivo para ser conectado conectado, e junto desta conexão é criado um *listener* (ouvinte) com a função de retorno (*callback*) (figura 6), em que o *callback* recebe os dados da conexão serial depois de tratado o *buffer* e já se tenha retornado do escâner todos os dados esperado, porque os dados da conexão serial podem vir em partes.

Figura 6. Função que espera o retorno da conexão serial.

```
102 String dataCompleto = "";
103 void _onDataReceived(uint8List data, callback) {
104     // Allocate buffer for parsed data
105     int backspacesCounter = 0;
106     data.forEach((byte) {
107         if (byte == 8 || byte == 127) {
108             backspacesCounter++;
109         }
110     });
111     uint8List buffer = uint8List(data.length - backspacesCounter);
112     int bufferIndex = buffer.length;
113
114     // Apply backspace control character
115     backspacesCounter = 0;
116     for (int i = data.length - 1; i >= 0; i--) {
117         if (data[i] == 8 || data[i] == 127) {
118             backspacesCounter++;
119         } else {
120             if (backspacesCounter > 0) {
121                 backspacesCounter--;
122             } else {
123                 buffer[--bufferIndex] = data[i];
124             }
125         }
126     }
127
128     // Create message if there is new line character
129     String dataString = String.fromCharCode(buffer);
130     dataCompleto += dataString;
131     if(dataString.contains(">")) {
132         callback(dataCompleto);
133         dataCompleto = "";
134     }
}
```

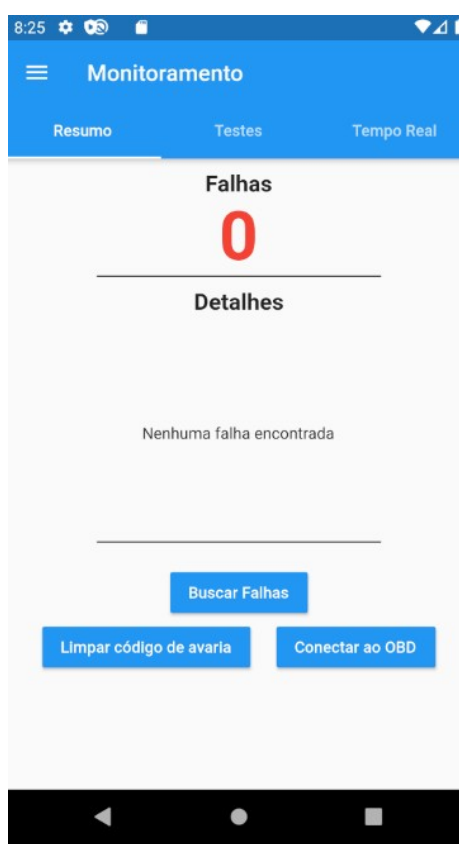
Fonte: Do autor.

Observe que na linha 102 apresentado na figura 6 existe um objeto do tipo *String* vazio chamado *dataCompleto*, esta variável serve para concatenar cada resposta que o escâner retorna e enviar para o *callback* no final só quando retorno estiver completo, pois o *listener* ouvindo a conexão serial será chamado sempre que

o OBD2 responder algo, e quando o valor retornado possui um *buffer* grande, ele virá em partes, por padrão quando o escâner nos retorna >, significa que seu retorno chegou ao fim, e então chamamos a função de *callback* enviando a variável *dataCompleto*, que será tratada pelo *controller* da aplicação.

Na *View* do protótipo, após se conectar a um dispositivo, como ilustrado na figura 7 os botões “Buscar falhas” e “Limpar código de avaria” ficam disponíveis para serem tocados.

Figura 7. Tela de monitoramento após se conectar ao OBD2



Fonte: Do autor

No protótipo é criado um objeto de falhas contendo o código e descrição da falha, junto é criado campos que consistem por constantes enumeradas para o aplicativo consultar as falhas existentes de acordo com a documentação do OBD2, porem algumas fabricantes podem criar códigos próprios de falhas, e para abrangermos estas falhas e mostrar para o proprietário do veículo, no momento que pegamos falha do veículo e consultarmos no *enum* de falhas, caso o código não esteja definido na aplicação, será retornado o código detectado com a descrição

“Código Desconhecido”, garantindo assim que o proprietário saiba que existe uma falha e tenha um código para assim que descobrir o problema conseguir identificar em uma busca de falhas futura.

Quando realizado a busca de falhas pelo aplicativo, primeiro é automaticamente executado o comando para reconfigurar o *escâner* OBD2 para o seu estado inicial e então aplicado para usar o protocolo automático, desta forma não será necessário controlar manualmente o tipo de protocolo utilizado, deixando esta parte mais transparente para o usuário. Após feita esta configuração enviado pela conexão serial *Bluetooth*, o comando 0101 que retornara quais testes o veículo fez, se a luz de avaria está acesa e quantos erros foram detectados. Caso existam erros detectados o aplicativo envia após este comando, o comando 03 para capturar os erros armazenados na unidade de controle do veículo, com este comando pode-se pegar todos os códigos de erros do veículo, estes erros são armazenados em uma lista de erros e apresentados para o usuário conforme é apresentado na figura 8.

Figura 8. Tela de monitoramento após buscar falhas



Fonte: Do autor

Com isso o proprietário consegue saber o motivo de sua luz de avaria estar acessa e quais problemas ele possui, podendo ele mesmo solucionar o problema e limpar o código de avaria, pois dependendo do veículo a luz de avaria não se apagará, para isto deve-se apenas tocar no botão de “Limpar código de avaria”. Vale ressaltar que caso a falha não tenha sido corrigida e for executado a limpeza de avaria, após ligar o veículo e deixar o motor um pouco ligado, a luz de avaria retornará.

6.5 RESULTADOS E DISCUSSÃO DOS RESULTADOS

Apresentar os dados retornados pela interface OBD2 para o usuário comum de maneira leiga é um grande desafio pois é necessário converter informação técnica codificada em informação simples e em português na tela. Para melhor segurança na leitura dos dados foi mantido o protocolo no OBD2 como automático, fazendo assim o *DTC* definir ele mesmo o melhor protocolo a ser usado para cada comando executado, mantendo esta parte transparente para o aplicativo e não precisando se preocupar com isto. Dito isto não precisaríamos configurar para ser utilizado o protocolo CAN, que de acordo com Löricz, Cualãilã e Mvodo (2016) é o mais eficiente e seguro a ser utilizado na leitura de dados por OBD2, já que o DTC decidirá pelo aplicativo.

Durante o desenvolvimento da aplicação foi-se utilizado o *guideline* do Material Design que foi criado seguindo vários dados de usabilidade coletados por usuários ao redor do mundo que utilizam produtos Google. Levando em consideração os aplicativos pagos aos quais foram comparados, o protótipo apresentado, mostrava inicialmente a tela de monitoramento e apresentava seguir mais fielmente o padrão de interface gráfica da Google, graças ao Flutter respeitar a *guideline* da plataforma para qual esteja sendo desenvolvida por padrão, no caso o Android com *Material*.

Sobre os problemas relatados nos aplicativos pagos, as conexões por *Bluetooth* se mostraram estáveis, uma das dificuldades relatadas por Fagundes, Silva e Assis (2019) era a troca de dados com o *escâner*, que utilizavam um protocolo antigo e facilmente corrompível. O autor Almeida e Faria tem como proposta futura um software capaz de auxiliar o operador a interpretar os códigos de falhas com um software de pouca complexidade, que é o que foi apresentado neste

protótipo, com isso tem-se algo similar ao apresentado pelo Shafi et al. (2018), onde é possível aumentar o tempo de vida do veículo com monitoramento diário, na pesquisa feita pela Toyota.

Todas as mensagens que o usuário recebe no protótipo são em português e claras, que é o principal problema relatado na utilização dos aplicativos pagos. Não foi possível criar a aplicação web, para monitorar o veículo em tempo real durante a viagem assim como suas rotas de maneira parecida ao que Payyanadan et al. (2017, tradução nossa) apresentaram em sua pesquisa.

7 CONCLUSÃO

Criar um protótipo de aplicação móvel integrado ao escâner *OBD* requer uma grande variedade de ferramentas e conhecimento como observado durante o desenvolvimento do trabalho. Por meio dos assuntos estudados, como *bluetooth*, interface *OBD* e tecnologias para desenvolvimento móvel híbrido, foi possível alcançar os principais objetivos propostos desde o início. A compreensão e funcionamento da interface *OBD2* no desenvolvimento de um protótipo amigável, agregando as funções mais utilizadas de softwares não gratuitos, foi parcialmente realizado. Pois seguindo o *guideline* do material, foi possível criar uma conexão estável e mostrar de forma transparente as informações (em português) de forma coerente, porém existem outras funcionalidades que eram utilizadas e não foi possível adicionar ao protótipo. Como por exemplo, informações em tempo real do torque e um diário de bordo do percurso percorrido pelo veículo.

Uma das maiores dificuldades foi compreender a decodificação do *OBD2*, visto que são utilizadas decodificações diferentes e precisam ser tratadas de forma individual. Porém, vale ressaltar, que após sua compreensão foi fácil decodificar todo o resto dos retornos da interface.

Por fim, após a conclusão, pode-se observar algumas melhorias para dar continuidade ao projeto para novas pesquisas na área, conforme segue:

- a) implementação de um sistema web contendo as informações em tempo real sobre o veículo;
- b) implementação do diário de bordo sincronizado com a aplicação web;
- c) implementação das funções de economia de combustível.

Mesmo tendo trabalhos futuros a serem desenvolvidos, pode-se concluir que o trabalho cumpriu com os itens propostos e possui um resultado satisfatório.

REFERÊNCIAS

ALMEIDA, Eduardo Luciano; FARIA, Felipe Freitas. **SCANNER OBD-II EM PLATAFORMA LabVIEW**. 2013. 139 f. TCC (Graduação) - Curso de Tecnologia Eletrônica Automotiva, Centro Tecnológico, FATEC, São Paulo, 2013. Disponível em: <<http://fatecsantoandre.edu.br/arquivos/TCC232.pdf>>. Acesso em: 06 julho 2020.

BELO, Valdeci Pereira. **Sistema para Diagnóstico Automático de Falhas em Veículos Automotores OBD-2**. 2003. 110 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro Tecnológico, Universidade Federal de Minas Gerais, Belo Horizonte, 2003. Cap. 1.

BOLDT, Thiago; SILVA, Adriano Willian da. **CARACTERIZAÇÃO DOS CATALISADORES AUTOMOTIVOS: COMPOSIÇÃO, CLASSIFICAÇÃO, VARIÁVEIS FÍSICO-QUÍMICAS E EFICIÊNCIA CATALÍTICA**. 2014. 6 f. TCC (Graduação) - Curso de Curso Técnico Integrado em Contabilidade do Ifpr, Instituto Federal do Paraná- Campus Curitiba, Curitiba, 2014. Disponível em: <<http://eventos.ifc.edu.br/wp-content/uploads/sites/5/2014/09/CET-54.pdf>>. Acesso em: 06 maio 2019.

BORGES, Bruno Rodrigues. **UNIVERSIDADE FEDERAL DE UBERLÂNDIA Bruno Rodrigues Borges Desenvolvimento de Aplicação Mobile Utilizando Metodologia Ágil SCRUM**. 2017. 64 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Tecnológico, Universidade Federal de Uberlândia, Uberlândia, 2017. Cap. 1. Disponível em: <<https://repositorio.ufu.br/bitstream/123456789/20098/1/DesenvolvimentoAplicacaoMobile.pdf>>. Acesso em: 25 nov. 2019.

BORGES, Karla AV; DAVIS JR, Clodoveu A.; LAENDER, Alberto HF. Modelagem conceitual de dados geográficos. **CASANOVA, et. al. Banco de Dados Geográfico. MundoGEO: Curitiba**, p. 83-136, 2005.

CHRISTENSEN, Jason H.. Using RESTful web-services and cloud computing to create next generation mobile applications. **Proceeding Of The 24th Acm Sigplan Conference Companion On Object Oriented Programming Systems Languages And Applications - Oopsla '09**, [s.l.], p.627-633, 2009. ACM Press. <http://dx.doi.org/10.1145/1639950.1639958>. Disponível em: <<https://dl.acm.org/citation.cfm?id=1639958>>. Acesso em: 25 nov. 2019.

DEITEL, Paul; DEITEL, Harvey; DEITEL, Abbey. **Android para programadores: Uma abordagem baseada em aplicativos**. 2. ed. Porto Alegre: Bookman, 2015. Tradução de: João Eduardo Nóbrega Tortello.

ELM Eletronics. **ELM327 OBD to RS232 Interpreter**. 2008. Disponível em: <<https://www.elmelectronics.com/ic/elm327/>>. Acesso em: 7 julho 2019.

ESTATISTICA onsv. 2016. Disponível em: <<http://iris.onsv.org.br/iris-beta/#/stats/profiles/42/fleet>>. Acesso em: 14 maio 2018.

EVAN YOU. **What is Vue.js?** 2019. Disponível em: <<https://vuejs.org/v2/guide/>>. Acesso em: 18 nov. 2019.]

FAGUNDES, Felipe Augusto Vieira; SILVA, Gustavo Luiz; ASSIS, Marco Aurélio Scomparim. **Sistema de monitoramento automotivo remoto**. 2015. 86 f. TCC (Graduação) - Curso de Tecnologia Eletrônica Automotiva, Centro Tecnológico, FATEC, São Paulo, 2015. Disponível em: <<http://fatecsantoandre.edu.br/arquivos/TCC342.pdf>>. Acesso em: 06 julho 2020.

FENSEL, Dieter; BUSSLER, Christoph. The Web Service Modeling Framework WSMF. **Electronic Commerce Research And Applications**, [s.l.], v. 1, n. 2, p.113-137, jun. 2002. Elsevier BV. [http://dx.doi.org/10.1016/s1567-4223\(02\)00015-7](http://dx.doi.org/10.1016/s1567-4223(02)00015-7). Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1567422302000157>>. Acesso em: 25 nov. 2019.

FERRO, E.; POTORTI, F.. Bluetooth and wi-fi wireless protocols: a survey and a comparison. **Ieee Wireless Communications**, [s.l.], v. 12, n. 1, p.12-26, fev. 2005. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/mwc.2005.1404569>.

FERTING, Cristina; MEDINA, Marco. **Algoritmos e Programação: Teoria e Prática**. São Paulo: Novatec, 2006. 384 p.

FLUTTER-DEV. **Flutter**. 2019. Disponível em: <<https://flutter.dev/>>. Acesso em: 18 nov. 2019.

GOOGLE. **Productive development**. 2019. Disponível em: <<https://dart.dev/>>. Acesso em: 18 nov. 2019.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ISO/DIS 14230-1 - **Road Vehicles – Diagnostic systems – Keyword Protocol 2000 – Part 1: Physical Layer**. ISO, 1995.

Java programming language. *Database and Network Journal 2018*. Disponível em: <https://link.gale.com/apps/doc/A163332676/AONEu=capes&sid=AONE&xid=8d8e84d1>. Acesso em: 4 Nov. 2019.

JUCIANO MARTINS RODRIGUES (Rio de Janeiro). Coordenação Nacional do Observatório das Metrôpoles. **EVOLUÇÃO DA FROTA DE AUTOMÓVEIS E MOTOS NO BRASIL 2001 – 2012**. Rio de Janeiro: Instituto Nacional de Ciência e Tecnologia, 2013. 40 p. Disponível em: <http://www.observatoriodasmetrololes.net/download/auto_motors2013.pdf>. Acesso em: 14 maio 2018.

LEAL, Nelson Glauber de Vasconcelos. **Dominando o Android: Do básico ao avançado**. São Paulo: Novatec, 2015. 789 p.

LECHETA, Ricardo R. **Android Essencial**. 5. ed. São Paulo: Novatec, 2016. 384 p.

LECHETA, Ricardo R. (2015). **Google Android: Aprenda a criar aplicações para dispositivos móveis com Android SDK**. 4th ed. São Paulo: Novatec.

LIGHTBEND. **What is Play?** Disponível em:

<<https://www.playframework.com/documentation/2.7.x/Introduction>>. Acesso em: 18 nov. 2019.

LÖRINCZ, ALEXANDRA ELISABETA; CUCĂILĂ, MARIUS; MVODO, CHARLES ROSTAND MVONGO. VEHICLE COMMUNICATION SYSTEM USING SMARTPHONE. **ELECTRICAL ENGINEERING VOL. 18 (XLV)**, p. 47, 2016.

LUCKOW, Décio Heinzemann; MELO, Alexandre Altair de. **Programação Java para web**. 2. ed. São Paulo: Novatec., 2015. 675 p. (2).

MACHADO, António Sérgio Leite; OLIVEIRA, Bruno Rafael Resende. **O Sistema OBD (On-Board Diagnosis)**. 2007. 1 f. Dissertação (Mestrado) - Curso de Automação e Sistemas, Instituto Superior de Engenharia do Porto, Porto, 2008. Disponível em: <http://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos/2007-08/Trabalhos/SIAUT_OBD.pdf>. Acesso em: 14 maio 2018.

MARIADB FOUNDATION. **About MariaDB Server**. 2019. Disponível em: <<https://mariadb.org/about>>. Acesso em: 18 nov. 2019.

MILANOVIC, N.; MALEK, M.. Current solutions for Web service composition. **Ieee Internet Computing**, [s.l.], v. 8, n. 6, p.51-59, nov. 2004. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/mic.2004.58>. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/1355922>>. Acesso em: 25 nov. 2019.

OBSERVATÓRIO NACIONAL DE SEGURANÇA VIÁRIA. **90% dos acidentes são causados por falhas humanas, alerta OBSERVATÓRIO**. Disponível em: <<https://www.onsv.org.br/90-dos-acidentes-sao-causados-por-falhas-humanas-alerta-observatorio/>>. Acesso em: 14 maio 2018.

OBSERVATÓRIO NACIONAL DE SEGURANÇA VIÁRIA. **Número de vítimas de acidentes de trânsito cresce 48% em 12 anos no Brasil**. Disponível em: <<https://www.onsv.org.br/numero-de-vitimas-de-acidentes-de-transito-cresce-48-em-12-anos-no-brasil/>>. Acesso em: 11 ago. 2020

OBTENHA Informações sobre a Tecnologia Java. 2018. Disponível em: <https://www.java.com/pt_BR/about/>. Acesso em: 14 maio 2018.

ORACLE. **Java™ Programming Language**. 2019. Disponível em: <<https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html>>. Acesso em: 18 nov. 2019.

ORACLE. **Oracle Java**. 2020. Disponível em: <<https://www.oracle.com/br/java/>>. Acesso em: 12 ago. 2020.

ORFILA, Olivier; PIERRE, Guillaume Saint; MESSIAS, Mickaël. An android based ecodriving assistance system to improve safety and efficiency of internal combustion engine passenger cars. **Transportation Research Part C: Emerging Technologies**, [s.l.], v. 58, p.772-782, set. 2015. Elsevier BV. <http://dx.doi.org/10.1016/j.trc.2015.04.026>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0968090X15001679>>. Acesso em: 18 nov. 2019.

PAYYANADAN, Rashmi P. et al. Using trip diaries to mitigate route risk and risky driving behavior among older drivers. **Accident Analysis & Prevention**, [s.l.], v. 106, p.480-491, set. 2017. Elsevier BV. <http://dx.doi.org/10.1016/j.aap.2016.09.023>. Disponível em: <<https://www-sciencedirect.ez318.periodicos.capes.gov.br/science/article/pii/S0001457516303530?via%3Dihub>>. Acesso em: 04 nov. 2019.

PEREIRA, Eduardo da Fonseca; YAMADA, William Jun. **IMPLEMENTAÇÃO DO PROTOCOLO INDUSTRIAL DE CAMPO DEVICENET EM PLATAFORMA RASPBERRY PI 3 PARA AUTOMAÇÃO DE MAQUETE FERROVIÁRIA**. 2018. 113 f. TCC (Graduação) - Curso de Engenharia Mecatrônica, Centro Tecnológico, Universidade de Brasília, Brasília, 2018. Cap. 5. Disponível em: <<http://bdm.unb.br/handle/10483/21320>>. Acesso em: 13 maio 2019.

RIBEIRO, Hugo Miguel Gomes. **Dados CAN / OBD2 em tempo real de viaturas auto**. 2015. 70 f. Dissertação (Mestrado) - Curso de Engenharia Eletrotécnica, Tecnológico, Faculdade de Engenharia da Universidade do Porto, Porto, 2015. Cap. 1. Disponível em: <<https://repositorio-aberto.up.pt/bitstream/10216/90117/2/35794.pdf>>. Acesso em: 15 set. 2018.

SALAMANI, João Carlos. **Análise de risco e aumento da confiabilidade de motores Otto com relação às falhas esporádicas e de difícil diagnóstico**. 2018. 197 f. Tese (Doutorado) - Curso de Ciências, Engenharia Mecatrônica, Escola Politécnica da Universidade de São Paulo, São Paulo, 2018.

SANTA CATARINA. DETRAN/SC. . **Acidentes com vítimas nas rodovias federais (1995 a 2005)**. Disponível em: <<http://www.detran.sc.gov.br/arquivos-detran/estatistica/acidentes>>. Acesso em: 14 maio 2018.

SHAFI, Uferah et al. Vehicle Remote Health Monitoring and Prognostic Maintenance System. **Journal Of Advanced Transportation**, [s.l.], v. 2018, n. 1, p.1-10, 18 jan. 2018. Hindawi Limited. <http://dx.doi.org/10.1155/2018/8061514>. Disponível em: <<https://www.hindawi.com/journals/jat/2018/8061514/>>. Acesso em: 03 nov. 2019.

SILVA, Maicon Machado Gerardi da. **APLICAÇÃO PARA MONITORAMENTO VEICULAR EM TEMPO REAL**. 2017. 109 f. TCC (Graduação) - Curso de Ciência da Computação, Centro Tecnológico, Universidade Regional de Blumenau, Blumenau, 2017. Cap. 1. Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2017_2_maicon-machado-gerardi-da-silva_monografia.p>. Acesso em: 21 set. 2018.

ARDUINO.**WHAT is Arduino?** 2018. Disponível em:
<<https://www.arduino.cc/en/Guide/Introduction#>>. Acesso em: 14 maio 2018.

XIE, Yong et al. STM32-based vehicle data acquisition system for Internet-of-Vehicles. **2017 IEEE/ACIS 16th International Conference On Computer And Information Science (ICIS)**, [s.l.], p.895-898, 26 maio 2017. IEEE.
<http://dx.doi.org/10.1109/icis.2017.7960119>. Disponível em: <<https://ieeexplore-ieee.org.ez318.periodicos.capes.gov.br/document/7960119>>. Acesso em: 03 nov. 2019.

APÊNDICE(S)

APÊNDICE A – ARTIGO CIENTÍFICO

Ferramenta Android para o Auxílio Técnico Automobilístico no Monitoramento de Veículos com o Escaneamento OBD2

Emerson Leonardo Zock Alves¹, Matheus Leandro Ferreira²

¹Acadêmico do Curso de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC) - 88.806-000 – Criciúma – SC – Brazil

²Professor Especialista do Curso de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC) - 88.806-000 – Criciúma – SC – Brazil

leonardo_2707@hotmail.com, mlf@unesc.net

***Abstract.** Computer-related mechanics provide the creation of equipment, software and techniques, enabling professionals in the field to use for vehicle diagnostics. Since the appearance of the self-diagnosis system interface, mechanics have been used to quickly find faults in a given vehicle. However, mechanical readers have an inaccessible price for small mechanics and also for vehicle owners, that in order to increase the “usage time” of their cars, it is necessary to do maintenance regularly, mainly so that there is no problem while on the road. road, which can lead to an accident or even death. Taking this into account, this course completion project aims to understand the self-diagnosis system interface in the construction of a prototype that helps vehicle owners to keep their car in a functional state. The prototype was developed taking into account the features of the two main paid software on Google Play, and their most frequent complaints made by Brazilians. After choosing these software, a solution was developed and tested in real vehicles, being able to analyze the existing failures with a self-diagnosis system scanner in which the results were satisfactory even if not all the functionalities of the paid software were implemented, all the main complaints were met, and validity in real vehicles.*

***Resumo.** A mecânica ligada a informática proporciona a criação de equipamentos, softwares e técnicas, possibilitando aos profissionais da área utilizarem para diagnósticos veiculares. Desde o surgimento da interface de sistema de autodiagnóstico os mecânicos têm-se usado para encontrar rapidamente as falhas*

de um determinado veículo. Porém, os leitores mecânicos possuem um preço não acessível para pequenas mecânicas e também para os proprietários dos veículos, que para aumentar o “tempo de uso” de seus carros, é preciso fazer manutenção regularmente, principalmente para que não ocorra um problema enquanto estiver na estrada, podendo gerar um sinistro ou até a morte. Levando isso em consideração, este projeto de conclusão de curso visa a compreensão da interface de sistema de autodiagnóstico na construção de um protótipo que auxilie os donos de veículos a manterem seu automóvel em estado funcional. O protótipo foi desenvolvido levando em consideração as funcionalidades dos dois principais softwares pagos na Google Play, e de suas mais frequentes reclamações feitas por brasileiros. Após a escolha destes softwares foi desenvolvida uma solução e testada em veículos reais, sendo capaz de analisar as falhas existentes junto de um escâner de sistema de autodiagnóstico em que os resultados foram satisfatórios mesmo não implementado todas as funcionalidades dos softwares pagos, todas as principais reclamações foram atendidas, e validadas em veículos reais.

1. INTRODUÇÃO

Pesquisas realizadas pelo Observatório Nacional de Segurança Viária (2020) mostram que ocorreu um aumento de 48,7% no número de pessoas mortas por acidente de veículos entre o ano de 2001 a 2012, em que um dos principais motivos são o aumento de 139% na frota de veículos neste período, em 2001 com 31,91 milhões de veículos a 2012 com 76,14 milhões e como esses automóveis são máquinas, é necessário fazer verificações periódicas em oficinas para manter tudo em ordem.

Comentado por Machado e Oliveira (2007), a muitos anos os automóveis começaram a utilizar módulos eletrônicos, sendo que os primeiros faziam um pouco mais que controlar a injeção eletrônica comparado com os modelos de 2007. Os módulos eletrônicos recebem dados de uma variedade de sensores, que na detecção de alguma anomalia os próprios módulos ajustam os parâmetros como por exemplo quantidade de oxigênio enviado ao motor para tentar corrigir as falhas.

De acordo com o guia de introdução ao Arduino (2018, tradução nossa), ele é uma plataforma de *hardware* e *software* fácil de usar e de código aberto onde é possível informar a placa o que fazer enviando instruções ao microcontrolador por meio da linguagem de programação baseada em *Wiring*.

De acordo com o Oracle (2020) o Java é uma linguagem de programação número 1 do mercado, com milhões de desenvolvedores ativos e mais de 45 milhões de máquinas virtuais sendo executado pelo mundo, com ele é possível implementar um código para diferentes plataformas como por exemplo: *Linux, Mac, Windows e Smartphone*.

Levando em consideração as situações listadas anteriormente e compreendendo que os veículos possuem sensores cuja responsabilidade é coletar dados de sua saúde, a presente proposta visa desenvolver uma ferramenta capaz de aproveitar os recursos do escaneamento *On-Board Diagnosis 2 (OBD2)* com as principais funções extraídas de *softwares* pagos da *Google Play*, transmitindo-os de forma transparente para o proprietário do veículo via *smartphone Android* e mantendo-o informado sobre possíveis problemas para poder ter percepção na tomada de decisão desta forma podendo diminuir as chances de acidente.

1.1. OBJETIVO GERAL

Realizar a prototipagem de um software contendo as principais funções dos aplicativos já existentes e não gratuitos na *Google Play* tornando-o acessível a qualquer mecânica de pequeno porte e proprietários de veículos automotivos.

1.2. OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho consistem em compreender o funcionamento da interface OBD2, compreender o sistema de monitoramento de automóveis, analisar *softwares* pagos da *Google Play* que utilizam o OBD2, desenvolver um protótipo amigável agregando as funções mais utilizadas dos softwares pagos escolhidos, analisar os resultados e validar o protótipo.

2. ESCÂNER OBD2

De acordo com Machado e Oliveira (2007) os primeiros módulos eletrônicos faziam utilizados por veículos faziam pouco mais que controlar a injeção eletrônica compara com os modelos de 2007, Entre as vantagens de se utilizar uma interface que se comunica com estes módulos está em conseguir informações se o veículo está funcionando conforme o projetado, economizando combustível ou dinheiro avisando ao condutor ou proprietário caso exista algum problema mecânico ou elétrico.

Ribeiro (2015) aponta que existem cinco protocolos que podem ser usados na interface OBD2, são eles *SAE J1850 PWM, SAE J1850 VPW, ISO 14230 KWP2000, ISO 15765 CAN*, sendo que a grande maioria dos fabricantes utilizem o primeiro mencionado e todos podem ser identificados facilmente apenas analisando quais pinos estão presentes no

conector. O *padrão* OBD2 especifica os protocolos de sinalização assim como os sinais que podem ser trocados e quais sensores monitorados pelo diagnóstico.

Desenvolvido em 1970 nos Estados Unidos pela Agência de Proteção Ambiental (PAE) que até os dias atuais controla o que se diz respeito às normas que os fabricantes devem respeitar na comunicação com a interface *OBD*, *porem os proprietários de veículos criavam seus próprios padrões de conexão e não existia um padrão universal para todos. Foi então que a SAE* emitiu um regulamento para a uniformidade dos conectores, o que levou na criação do OBD2 e uma homogeneidade no dispositivo que obtém informações de sensores instalados no veículo. Onde OBD2 começou a ser *adotado como o padrão* para avaliar o estado dos veículos nos Estados Unidos (MACHADO; OILIVEIRA, 2007).

Foi com esse regulamento nos Estados Unidos da América que possibilitou implantar este sistema OBD em veículos destinados á Europa nos anos 90, e a partir de 2010 na frota brasileira com OBD2, sendo o *padrão* do *OBD2* o protocolo *SAE J1962* no Brasil, consistindo em uma interface com dezesseis pinos formado por duas linhas de oito pinos que deve estar instalado junto a cabine do motorista.

2.1 MONITORAMENTO COM OBD2

O sistema OBD2, segundo Belo (2003), utiliza uma estratégia denominada “Monitores”, para testar as operações do sistema, componentes ou funções específicas do veículo, que estabelece um controle muito rígido sobre os parâmetros operacionais dos componentes do sistema, principalmente os relacionados aos poluentes.

Para se comunicar com o veículo, é necessário enviar comandos para o escâner , entre os comandos existentes um se destaca, o 03 que analisa as falhas detectadas pelo DTC, no retorno deste comando pode-se receber o valor 43 01 33 por exemplo, em que o valores 43 indicam que foi solicitado o comando 03, e os outros valores o código de erro, sendo necessário substituir o primeiro valor para encontrar a letra identificadora de acordo com a tabela 1. (ELM Eletronics, 2008, tradução nossa).

0	P0
1	P1
2	P2
3	P3
4	C0
5	C1
6	C2
7	C3
8	B0
9	B1
A	B2
B	B3
C	U0
D	U1
E	U2
F	U3

Os números retornados *0133*, o primeiro valor da esquerda para direita é 0 então deve ser substituído para *P0* como mostra na tabela 1, para concatenar o resto dos números nos dando o código *P0133*, que significa que a resposta do sensor de oxigênio está baixa e este mesmo processo deve ser efetuado com todos os códigos de falhas retornados pelo comando *03*.

3. PRINCIPAIS SOFTWARES PAGOS

Todos os fabricantes de automóveis da atualidade possuem sistemas que ajudam o motorista a dirigir de forma mais eficiente e consciente do estado de seu veículo, porém sua grande maioria é simples ou somente mostram os dados no pós-viagem para ser analisado, exibindo informações gerais sobre as resoluções do motorista e permitindo que se faça uma comparação da direção com outros motoristas. Entre esses aplicativos existe o *eco: Drive Live*, um aplicativo comercial da FIAT que fornece conselhos em tempo real sobre quatro indicadores, a aceleração, desaceleração, mudança de marcha e a velocidade, no entanto eles não dão conselhos antecipados e também funcionam somente no veículo, não sendo possível utilizá-lo no *smartphone* (ORFILA; PIERRE; MESSIAS, 2015, tradução nossa).

O retorno do aplicativo em tempo real das condições dos veículos é um dos motivos de alguém comprar o dispositivo com a interface OBD2, e foram escolhidos os 2

softwares pagos mais baixados e votados da *Play Store*, a loja oficial utilizada pelo *Android* para se entender quais as principais funções e as dificuldades dos usuários *Android*. Foi então escolhido o software *Torque Pro (OBD2 / CARRO)* de R\$ 10,99, com mais de 60.000 *downloads* e o software *OBD Fusion (Car Diagnostics)* de R\$ 19,99, com mais de 1.000 *downloads*.

As principais funções que se pode encontrar nos dois *softwares* são exibir os dados de falha do veículo, como também apresentando informações do torque do motor, a potência, leitura de emissão CO₂, cálculo de economia de combustível e um banco de dados com os códigos de falhas para pesquisa local do usuário.

Sobre as dificuldades dos aplicativos escolhidos, foram definidas através de reclamações dos usuários na página de comentários da *Google Play*, entre elas está a dificuldade de acessar após a compra, travamento do aplicativo, código de falha não encontrado, dificuldade ao se conectar ao *OBD2*, não possui nenhuma documentação ou guia de como deve-se iniciar a utilização do aplicativo, atraso entre a informação que aparece no painel do veículo e no celular, e não está em português.

4. O PROTÓTIPO

Visando auxiliar a ter menos acidentes nas estradas e manter os motoristas alertas com a manutenção de seus veículos, apresentou-se um protótipo de aplicativo com as principais funcionalidades de softwares pagos que utilizam a interface *OBD2* para auxiliar os motoristas proprietários de veículos a manter seu veículo em melhores condições antes de saírem de casa. Para isto o processo metodológico da pesquisa foi dividido em sete etapas, sendo elas: levantamento de requisitos, modelagem de dados, desenho das telas a serem desenvolvidas, implementação da aplicação *mobile*, testes de conexão *Bluetooth* para enviar e receber informações via serial, testes em um veículo real com auxílio de um mecânico, e por fim, a análise dos resultados obtidos por meio da aplicação *mobile*

Para a criação do aplicativo com intuito de suprir as principais dificuldades relatadas pelos usuários que utilizaram os aplicativo pagos estudados na pesquisa bibliográfica, foi utilizado tecnologias modernas altamente escaláveis da própria Google como flutter e a *IDE Android Studio*, juntamente, para gerenciar a conexão *Bluetooth* e nos permitir conectar ao *escâner* será utilizada a biblioteca *open source flutter_bluetooth_serial* criada pelo desenvolvedor Eduardo Folly.

O protótipo oferece para os proprietários de veículo um meio de obter informações relevantes sobre o estado se deu veículo, utilizando o próprio celular junto de um

equipamento barato, estas informações podem evitar com que o veículo parem de funcionar no meio de uma viagem ou até cause um acidente.

No protótipo é criado um objeto de falhas contendo o código e descrição da falha, junto é criado campos que consistem por constantes enumeradas para o aplicativo consultar as falhas existentes de acordo com a documentação do OBD2, porem algumas fabricantes podem criar códigos próprios de falhas, e para abrangermos estas falhas e mostrar para o proprietário do veículo, no momento que pegamos falha do veículo e consultarmos no *enum* de falhas, caso o código não esteja definido na aplicação, será retornado o código detectado com a descrição “Código Desconhecido”, garantindo assim que o proprietário saiba que existe uma falha e tenha um código para assim que descobrir o problema conseguir identificar em uma busca de falhas futura.

Quando realizado a busca de falhas pelo aplicativo, primeiro é automaticamente executado o comando para reconfigurar o *escâner* OBD2 para o seu estado inicial e então aplicado para usar o protocolo automático, desta forma não será necessário controlar manualmente o tipo de protocolo utilizado, deixando esta parte mais transparente para o usuário. Após feita está configuração enviado pela conexão serial *Bluetooth*, o comando 0101 que retornara quais testes o veículo fez, se a luz de avaria está acessa e quantos erros foram detectados. Caso existam erros detectados o aplicativo envia após este comando, o comando 03 para capturar os erros armazenados na unidade de controle do veículo, com este comando pode-se pegar todos os códigos de erros do veículo, estes erros são armazenados em uma lista de erros e apresentados para o usuário.

Com isso o proprietário consegue saber o motivo de sua luz de avaria estar acessa e quais problemas ele possui, podendo ele mesmo solucionar o problema e limpar o código de avaria, pois dependendo do veículo a luz de avaria não se apagará, para isto deve-se apenas tocar no botão de “Limpar código de avaria”. Vale ressaltar que caso a falha não tenha sido corrigida e for executado a limpeza de avaria, após ligar o veículo e deixar o motor um pouco ligado, a luz de avaria retornará.

4.1. RESULTADOS E DISCUSSÃO

Apresentar os dados retornados pela interface OBD2 para o usuário comum de maneira leiga é um grande desafio pois é necessário converter informação técnica codificada em informação simples e em português na tela. Para melhor segurança na leitura dos dados foi mantido o protocolo no OBD2 como automático, fazendo assim o *DTC* definir ele mesmo o melhor protocolo a ser usado para cada comando executado, mantendo esta parte transparente para o aplicativo e não precisando se preocupar com isto. Dito isto não precisaríamos

configurar para ser utilizado o protocolo CAN, que de acordo com Löricz, Cualilã e Mvodo (2016) é o mais eficiente e seguro a ser utilizado na leitura de dados por OBD2, já que o DTC decidirá pelo aplicativo.

Durante o desenvolvimento da aplicação foi-se utilizado o *guideline* do Material Design que foi criado seguindo vários dados de usabilidade coletados por usuários ao redor do mundo que utilizam produtos Google. Levando em consideração os aplicativos pagos aos quais foram comparados, o protótipo apresentado, mostrava inicialmente a tela de monitoramento e apresentava seguir mais fielmente o padrão de interface gráfica da Google, graças ao Flutter respeitar a *guideline* da plataforma para qual esteja sendo desenvolvida por padrão, no caso o Android com *Material*.

Sobre os problemas relatados nos aplicativos pagos, as conexões por *Bluetooth* se mostraram estáveis, uma das dificuldades relatadas por Fagundes, Silva e Assis (2019) era a troca de dados com o *escâner*, que utilizavam um protocolo antigo e facilmente corrompível. O autor Almeida e Faria tem como proposta futura um software capaz de auxiliar o operador a interpretar os códigos de falhas com um software de pouca complexidade, que é o que foi apresentado neste protótipo, com isso tem-se algo similar ao apresentado pelo Shafi et al. (2018), onde é possível aumentar o tempo de vida do veículo com monitoramento diário, na pesquisa feita pela Toyota.

Todas as mensagens que o usuário recebe no protótipo são em português e claras, que é o principal problema relatado na utilização dos aplicativos pagos. Não foi possível criar a aplicação web, para monitorar o veículo em tempo real durante a viagem assim como suas rotas de maneira parecida ao que Payyanadan et al. (2017, tradução nossa) apresentaram em sua pesquisa.

5. CONCLUSÃO

Criar um protótipo de aplicação móvel integrado ao escâner *OBD* requer uma grande variedade de ferramentas e conhecimento como observado durante o desenvolvimento do trabalho. Por meio dos assuntos estudados, como *bluetooth*, interface *OBD* e tecnologias para desenvolvimento móvel híbrido, foi possível alcançar os principais objetivos propostos desde o início. A compreensão e funcionamento da interface OBD2 no desenvolvimento de um protótipo amigável, agregando as funções mais utilizadas de softwares não gratuitos, foi parcialmente realizado. Pois seguindo o *guideline* do material, foi possível criar uma conexão estável e mostrar de forma transparente as informações (em português) de forma coerente, porém existem outras funcionalidades que eram utilizadas e não foi possível adicionar ao

protótipo. Como por exemplo, informações em tempo real do torque e um diário de bordo do percurso percorrido pelo veículo.

Uma das maiores dificuldades foi compreender a decodificação do *OBD2*, visto que são utilizadas decodificações diferentes e precisam ser tratadas de forma individual. Porém, vale ressaltar, que após sua compreensão foi fácil decodificar todo o resto dos retornos da interface.

Por fim, após a conclusão, pode-se observar algumas melhorias para dar continuidade ao projeto para novas pesquisas na área, conforme segue:

- a) implementação de um sistema web contendo as informações em tempo real sobre o veículo;
- b) implementação do diário de bordo sincronizado com a aplicação web;
- c) implementação das funções de economia de combustível.

Mesmo tendo trabalhos futuros a serem desenvolvidos, pode-se concluir que o trabalho cumpriu com os itens propostos e possui um resultado satisfatório.

REFERÊNCIAS

BELO, Valdeci Pereira. **Sistema para Diagnóstico Automático de Falhas em Veículos Automotores OBD-2**. 2003. 110 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro Tecnológico, Universidade Federal de Minas Gerais, Belo Horizonte, 2003. Cap. 1.

ELM Eletronics. **ELM327 OBD to RS232 Interpreter**. 2008. Disponível em: <<https://www.elmelectronics.com/ic/elm327/>>. Acesso em: 7 julho 2019.

FAGUNDES, Felipe Augusto Vieira; SILVA, Gustavo Luiz; ASSIS, Marco Aurélio Scomparim. **Sistema de monitoramento automotivo remoto**. 2015. 86 f. TCC (Graduação) - Curso de Tecnologia Eletrônica Automotiva, Centro Tecnológico, FATEC, São Paulo, 2015. Disponível em: <<http://fatecsantoandre.edu.br/arquivos/TCC342.pdf>>. Acesso em: 06 julho 2020.

LÖRINCZ, ALEXANDRA ELISABETA; CUCĂILĂ, MARIUS; MVODO, CHARLES ROSTAND MVONGO. VEHICLE COMMUNICATION SYSTEM USING SMARTPHONE. **ELECTRICAL ENGINEERING VOL. 18 (XLV)**, p. 47, 2016.

MACHADO, António Sérgio Leite; OLIVEIRA, Bruno Rafael Resende. **O Sistema OBD (On-Board Diagnosis)**. 2007. 1 f. Dissertação (Mestrado) - Curso de Automação e Sistemas, Instituto Superior de Engenharia do Porto, Porto, 2008. Disponível em: <http://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos/2007-08/Trabalhos/SIAUT_OBD.pdf>. Acesso em: 14 maio 2018.

OBSERVATÓRIO NACIONAL DE SEGURANÇA VIÁRIA. **Número de vítimas de acidentes de trânsito cresce 48% em 12 anos no Brasil**. Disponível em: <<https://www.onsv.org.br/numero-de-vitimas-de-acidentes-de-transito-cresce-48-em-12-anos-no-brasil/>>. Acesso em: 11 ago. 2020

ORACLE. **Oracle Java**. 2020. Disponível em: <<https://www.oracle.com/br/java/>>. Acesso em: 12 ago. 2020.

PAYYANADAN, Rashmi P. et al. Using trip diaries to mitigate route risk and risky driving behavior among older drivers. **Accident Analysis & Prevention**, [s.l.], v. 106, p.480-491, set. 2017. Elsevier BV. <http://dx.doi.org/10.1016/j.aap.2016.09.023>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0001457516303530?via%3Dihub>>. Acesso em: 04 nov. 2019.

RIBEIRO, Hugo Miguel Gomes. **Dados CAN / OBD2 em tempo real de viaturas auto**. 2015. 70 f. Dissertação (Mestrado) - Curso de Engenharia Eletrotécnica, Tecnológico, Faculdade de Engenharia da Universidade do Porto, Porto, 2015. Cap. 1. Disponível em: <<https://repositorio-aberto.up.pt/bitstream/10216/90117/2/35794.pdf>>. Acesso em: 15 set. 2018.

SANTA CATARINA. DETRAN/SC. . **Acidentes com vítimas nas rodovias federais (1995 a 2005)**. Disponível em: <<http://www.detransc.gov.br/arquivos-detransc/estatistica/acidentes>>. Acesso em: 14 maio 2018.

SHAFI, Uferah et al. Vehicle Remote Health Monitoring and Prognostic Maintenance System. **Journal Of Advanced Transportation**, [s.l.], v. 2018, n. 1, p.1-10, 18 jan. 2018. Hindawi Limited. <http://dx.doi.org/10.1155/2018/8061514>. Disponível em: <<https://www.hindawi.com/journals/jat/2018/8061514/>>. Acesso em: 03 nov. 2019.

SILVA, Maicon Machado Gerardi da. **APLICAÇÃO PARA MONITORAMENTO VEICULAR EM TEMPO REAL**. 2017. 109 f. TCC (Graduação) - Curso de Ciência da Computação, Centro Tecnológico, Universidade Regional de Blumenau, Blumenau, 2017. Cap. 1. Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2017_2_maicon-machado-gerardi-da-silva_monografia.p>. Acesso em: 21 set. 2018.