UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ARTHUR LUIS VIÑAS SANTOS

Economia de combustível com o uso de telemetria para veículos de passeio

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Alexandre Carissimi

Porto Alegre
2016

“O sucesso geralmente vem para aqueles
que estão muito ocupados para estarem procurando por ele.”
— Sir Henry David Thoreau

AGRADECIMENTOS

Agradeço, acima de tudo, a Deus e aos meus guias espirituais, que me trouxeram equilíbrio, sabedoria, paciência e determinação para superar os desafios que me foram apresentados durante a vida, em especial, durante a graduação.

Ao meu orientador, Alexandre Carissimi, por todas as dicas e conselhos, pela disponibilidade e por todo auxílio prestado, obrigado.

Ao meu pai, João Santos, pelo exemplo de honestidade e autenticidade que tem sido para mim, desde minha infância, obrigado.

À minha mãe, Iara Resena, pelos princípios e valores que me ensinou, e quem, de muitas formas, me deu apoio e incentivo para alcançar meus objetivos, mesmo nos momentos de dificuldade.

À minha irmã, Sue Ellen Lia, pelas constantes brigas na infância, mas que resultaram em uma amizade eternamente verdadeira, e com quem sempre contei e poderei contar, obrigado.

À minha esposa, Simone Santos, que me acompanhou e me apoiou, pacientemente, em cada etapa da graduação, e que me deu a principal razão para concluí-la, nosso amado filho, Erick Santos, obrigado.

Aos meus avós, paternos e maternos, que amo muito, e que direcionaram suas orações a mim, muito obrigado.

Agradeço aqui todas as pessoas que, de uma forma ou outra, contribuíram na minha formação como pessoa, como profissional e no sucesso deste trabalho, em particular, Ricardo Tonding, Jairo Mello, Claudio Comunelo, Rafael Godoy, César Heitor e Rodrigo Borges.

RESUMO

As tecnologias intraveiculares atuais permitem obter informações em tempo real sobre diferentes aspectos do veículo, como velocidade, rotações por minuto, temperatura do motor, posição do acelerador, entre outros. No entanto, ainda é difícil mensurar a relação entre maus hábitos dos motoristas e o consumo excessivo de combustíveis. O intuito deste trabalho é propor uma aplicação para *smartphones* com sistema operacional Android que, integrada ao barramento de diagnóstico do veículo, forneça ao condutor meios para observar e aprimorar seu comportamento ao volante. Com isso, pretende-se minimizar os custos com combustíveis e, por consequência, a emissão de gases poluentes na atmosfera.

Palavras-chave: Telemetria, OBD, economia de combustível,veículos inteligentes.

Fuel economy with the use of telemetry for passenger cars

ABSTRACT

Current in-vehicle technology allows to obtain real-time information on different aspects of the vehicle, as current speed, RPM, engine temperature, throttle position and others. However, it is still difficult to measure the relationship between bad habits of drivers and excessive fuel consumption. The purpose of this work is to propose a mobile application, for smarphones with Android operational system, that provides ways to the driver observe and enhance your driving behavior. In this cenario, it is intended to minimize fuel cost and pollutants emission into the atmosphere.

Keywords: Telemetry, OBD, fuel economy, intelligent vehicles.

LISTA DE ABREVIATURAS E SIGLAS

ABS	*Anti-lock Breaking System*
ANP	Agência Nacional de Petróleo
API	*Application Programming Interface*
ARB	*Air Resources Board*
CAN	*Controller Area Network*
CCC	*Car Connectivity Consortium*
CRC	*Cyclic Redundancy Check*
CSMA-CR	*Carrier Sense Multiple Access/Collision Resolution*
CSMA	*Carrier Sense Multiple Access*
DDL	*Data Definition Language*
DLC	*Data Link Connector*
DML	*Data Manipulation Language*
DTC	*Diagnostic Trouble Codes*
ECU	*Engine Control Unit*
IDL	*Interactive Data Language*
IOT	*Internet of Things*
ISO	*International Organization for Standardization*
ITS	*Inteligent Transportation Systems*
IVI	*In-Vehicle Infotainment*
MIL	*Malfunction Indicator Light*
NRZ	*Non Return to Zero*
OAA	*Open Automotive Alliance*
OBD	*On-board Diagnostics*
OSI	*Open Systems Interconnection*

PDM	*Pulse Duration Modulation*
PIC	*Programmable Interface Controllers*
PID	*Parameter Identification*
PWM	*Pulse Width Modulation*
RPM	Rotações por Minuto
SD	*Secure Digital*
SDK	*Software Development Kit*
SMS	*Short Message Service*
SQL	*Structured Query Language*
UART	*Universal Asynchronous Receiver/Transmitter*
VIN	*Vehicle Identification Number*
VPW	*Variable Pulse Width*
XML	*EXtensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo a ANP (Agência Nacional de Petróleo), o consumo de petróleo em 2014 cresceu 5,95% no Brasil, totalizando 3.228.823 barris (ANP, 2015). Em 2015, em resposta ao clima de retração econômica do país, o consumo sofreu queda de 1,9% em relação ao ano anterior, no entanto, ainda representa a queima de 3.167.475 barris de combustíveis. Isso significa que uma pequena queda no consumo médio de combustível do país poderia representar uma redução na queima de dezenas de milhões de litros ao ano. Além do aspecto financeiro, que está diretamente relacionado a esses números, nos dias atuais, em que cientistas alertam para os males do aquecimento global para o meio ambiente e para os seres humanos, é de extrema relevância qualquer medida que permita reduzir o consumo até que as tecnologias híbridas e renováveis sejam largamente adotadas.

As chamadas *Smart Cities*, ou cidades inteligentes, são um conceito baseado na oferta de serviços tecnológicos para solucionar problemas dos cidadãos, em diferentes nichos do ecossistema em que estão inseridos, proporcionando um modo de vida mais sustentável ambiental, social e economicamente. Uma *Smart City* necessita de soluções para vários problemas, como: trânsito, vigilância, atendimento de emergência, monitoramento e preservação ambiental, saúde, educação e inclusão digital. Por meio de uma infraestrutura de comunicação bem planejada é possível coletar diversos dados e indicadores urbanos que subsidiarão o fornecimento dessas soluções. Do ponto de vista de mobilidade, os esforços se concentram na criação de um sistema de transporte mais eficiente, que utilize os recursos disponíveis da melhor maneira possível, tais como, veículos, vias e combustíveis.

Segundo Luby (LUBY, 2014), o desenvolvimento de tecnologias para carros inteligentes tem sido cada vez mais visado pelas gigantes de tecnologia. É possível perceber um grande interesse em tornar os veículos mais próximos de seus usuários, proporcionando maior comodidade, segurança e economia. As empresas Google, Apple e Microsoft investem fortemente em interatividade ao volante, trazendo a integração de aplicativos móveis à central multimídia, além do reconhecimento de fala para uma condução mais segura. Mais ousado que seus concorrentes, o Google já realiza testes em veículos completamente autônomos (GOOGLE, 2015a), capazes de levar os passageiros a qualquer lugar com o simples apertar de um botão. Iniciativas como essas permitem projetar um futuro onde haverá mais acessibilidade, menos acidentes no trânsito, além da redução de congestionamentos, economia de tempo, de dinheiro e de combustível.

No entanto, são inúmeros os desafios para que essas inovações sejam colocadas em prá-

tica, e não se limitam apenas a problemas técnicos. É preciso haver consenso e aceitação por parte da população, além de uma considerável adequação nas leis e na infraestrutura das cidades. Somente assim, os sistemas autônomos poderão ser amplamente adotados. As chamadas plataformas IVI, ou *In-Vehicle Infotainment Platforms*, são propostas aplicáveis no curto prazo, e consistem em sistemas informatizados presentes nos automóveis com a finalidade de entregar entretenimento e informação de forma simples e segura aos seus usuários. Geralmente, esses sistemas fazem o uso de tecnologias como *Bluetooth* e comandos de voz de forma integrada aos *smartphones* para facilitar o controle de suas funções pelo motorista. As aplicações mais comuns envolvem facilidades como ouvir e enviar de SMS por comandos de voz, realizar chamadas sem tirar as mãos do volante, controlar a *playlist*, ou mesmo o acessar à Internet para obter informações como pontos turísticos, congestionamentos e previsão do tempo.

Assim como nos carros de Fórmula-1, a telemetria também pode ser empregada em veículos de passeio. Desde 1968, quando os antigos sistemas de carburação estavam sendo substituídos pelo moderno sistema de injeção eletrônica, os barramentos de sensoriamento e diagnóstico estão presentes nos automóveis (WIKIPEDIA, 2014) gerando grandes quantidades de dados a todo momento. Esses dados, no entanto, nem sempre estiveram disponíveis aos usuários. Atualmente, existem dispositivos que permitem extrair tais informações em tempo real, de modo que, elas possam auxiliar o condutor a identificar comportamentos indevidos ou falhas no sistema. Diferente dos carros de Fórmula-1, em que a telemetria tem o objetivo de entender o comportamento do piloto, ou do veículo, para maior desempenho, o propósito do uso desses dados nos veículos de passeio é proporcionar maior economia e segurança na condução.

Atualmente, existem padrões definidos para a leitura de dados de diagnóstico da maioria dos veículos em circulação, o que possibilita seu fácil acesso em tempo real por meio de *In-Vehicle Infotainment Platforms* e de *Smartphones*. Com base nessas tecnologias, é possível criar ferramentas que relacionem dados de diagnóstico com o gasto de combustível por quilômetro rodado. Essas soluções inteligentes podem auxiliar as pessoas a perceberem que seu comportamento ao volante pode estar diretamente relacionado às suas despesas e, então, impulsionar certa conscientização a respeito do assunto. Essa percepção possui influência ambiental bastante positiva, mesmo que a motivação principal seja financeira. É nesse contexto que o presente trabalho se insere.

1.1 Objetivos do Trabalho

Em meio às diferentes iniciativas e tecnologias voltadas à mobilidade urbana nas *Smart Cities*, o intuito deste trabalho é propor uma aplicação para *smartphones* com sistema operacional Android que, com o uso de telemetria, auxilie o condutor na redução dos seus custos com combustíveis. A aplicação deverá ser capaz de ler dados de diagnóstico do computador central do veículo e, então, auxiliar as pessoas a monitorarem e aprimorarem o seu perfil de condução para a obtenção de melhores resultados.

Serão adotadas duas estratégias distintas para a interação com o usuário. A primeira delas propõe o disparo de notificações em tempo real durante a condução. Essas notificações operarão como um estímulo imediato às atitudes que acarretem maior consumo de combustível possibilitando que, inconscientemente, o usuário passe a evitá-las. A segunda estratégia diz respeito às análises posteriores à realização de um percurso. Essas análises permitirão uma observação crítica e detalhada sobre o estilo de condução de determinado percurso, como tempo em que o veículo permaneceu parado, número de excessos de velocidade e número de aceleradas bruscas, correlacionando esses dados com o consumo de combustível praticado.

1.2 Organização do Texto

Este trabalho está organizado em 6 capítulos considerando a presente introdução. No capítulo 2 serão abordados os principais conceitos envolvidos na telemetria de automóveis, seus protocolos e os principais padrões existentes. Além disso, serão detalhadas as tecnologias de interatividade para veículos existentes no mercado e seu papel na economia de combustíveis. O capítulo 3 fornece a especificação da solução proposta, nele estará ilustrada a fase de concepção da aplicação, bem como os requisitos e protótipos iniciais. No capítulo 4 são discutidos os detalhes da implementação, as tecnologias de hardware e software utilizadas e as dificuldades enfrentadas. O capítulo 5 apresenta uma avaliação do protótipo desenvolvido e contempla a verificação de seu funcionamento, a análise de um subconjunto dos dados coletados e a estimativa de consumo de recursos do *smartphone*. Por fim, o capítulo 6, conclusão, ressalta os principais desafios e contribuições deste trabalho, assim como trabalhos futuros que poderão sucedê-lo.

2 ESTADO DA ARTE

A crescente elevação da temperatura global, associada ao derretimento de geleiras e à elevação do nível dos oceanos impulsionou movimentos científicos ao redor do mundo com a finalidade de reduzir as emissões de gases causadores do efeito estufa na atmosfera, como o gás carbônico. Em 1999, 55 países, que juntos produzem 55% das emissões de poluentes, ratificaram um tratado chamado Protocolo de Kyoto. Esse tratado estipularia um calendário de metas de redução nas emissões de gases do efeito estufa, principalmente para países desenvolvidos. O tratado prevê diferentes medidas para atender as metas estipuladas como: reformar os setores de energia e transportes, promover o uso de fontes de energia renováveis, proteger florestas e outros sumidouros de carbono e limitar as emissões de metano no gerenciamento de resíduos e dos sistemas energéticos .

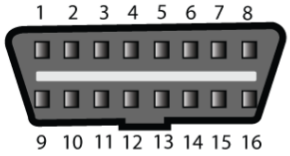
Contudo, a preocupação com a emissão de poluentes já era considerada um assunto importante mesmo antes do Protocolo de Kyoto. Por volta de 1982, o Conselho de Recursos Aéreos da Califórnia (ARB) começou a desenvolver regulamentações que exigiriam que todos os veículos fabricados fossem equipados com um dispositivo de diagnóstico capaz de detectar a emissão excessiva de poluentes. Dessa forma seria possível fiscalizar de forma mais efetiva veículos altamente poluentes. Desde então, as soluções de diagnóstico evoluíram e agregaram cada vez mais informações. Nos dias de hoje, essas ferramentas são capazes de entregar informações em tempo real ao usuário por meio de seus celulares ou mesmo dos sistemas interativos presentes em seus veículos. Com esse avanço, passa a ser possível o acesso a dados de diagnóstico que até então eram obtidos apenas pelas concessionárias, o que cria possibilidades para o desenvolvimento de aplicações mais sofisticadas para grandes frotas de carros e caminhões ou para simples veículos de passeio.

2.1 *On-board diagnostics* e protocolos

A sigla OBD (do inglês *On-Board Diagnostic*) designa um sistema de autodiagnóstico disponível na maioria dos veículos automotores que circulam atualmente. A ligação ao sistema ocorre por meio de um conector padronizado que foi sancionado como obrigatório na Europa e nos Estados Unidos para todos os veículos produzidos a partir de 1996, e, no Brasil, a partir de 2010 com o padrão de segunda geração OBD-II. A medida tem a finalidade de fiscalizar a emissão de gases poluentes na atmosfera, dado que, alguns países possuem acordos mundiais em que se comprometem com a preservação ambiental, como o protocolo de Kyoto.

Além do aspecto ambiental, o padrão OBD visa popularizar o serviço de reparo eletrônico, reduzindo drasticamente o custo das oficinas. A partir da leitura dos dados de diagnóstico, é possível identificar diversos tipos de anomalias, desde uma simples lâmpada queimada no painel de instrumentos, até sérios problemas no sistema de injeção eletrônica. A padronização e a abertura dos protocolos de comunicação trouxeram ao mercado equipamentos que podem ser comprados pelos próprios consumidores e que possuem tecnologia *Bluetooth* para sincronização com um computador ou aparelho celular. Isso significa que, com algum conhecimento básico sobre o funcionamento do veículo, o proprietário pode chegar ao mecânico com uma suposição a respeito do problema e, então, evitar que seja enganado. Tipicamente, o conector OBD está localizado na região inferior do painel do motorista ao lado esquerdo do volante, a Figura 2.1 ilustra seu formato, bem como, a numeração de seus pinos.

Figura 2.1 – Conector padrão OBD - padrão SAE J1962



Fonte: (OBDTESTER, 2015)

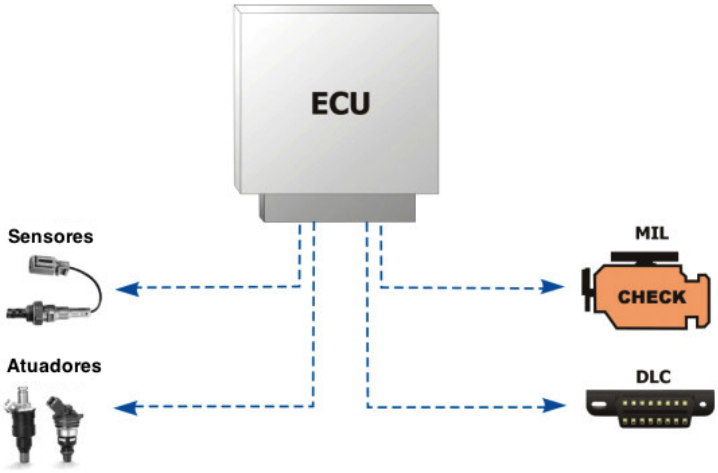
Os sistemas OBD possuem duas versões: OBD-I e OBD-II. A primeira foi desenvolvida nos anos 80 e definia uma série de padrões de comunicação entre as centrais de controle eletrônico dos veículos e as ferramentas de diagnóstico. No entanto, cada fabricante utilizava seus próprios conectores e códigos de erros (DTC, ou *Diagnostic Trouble Codes*). A partir de 1996, os veículos fabricados nos Estados Unidos e na Europa passaram a sair de fábrica equipados com o sistema OBD-II, o qual trouxe alguns avanços para a tecnologia. Dentre eles, a padronização do conector J1962 (SAE, 1992) e dos protocolos suportados. Com essa evolução, um único dispositivo é capaz obter dados de diagnóstico de veículos de diferentes marcas (WIKIPEDIA, 2014).

2.1.1 Arquitetura de sistemas OBD

Um sistema básico OBD consiste de uma ECU (*Engine Control Unit*), que usa a entrada de vários sensores (por exemplo, sensores de oxigênio) para controlar os atuadores (por exemplo, injetores de combustível) para obter o desempenho desejado. O sistema também reporta e registra anomalias, que podem ser identificadas pelo condutor através da luz de *check engine*

no painel de instrumentos, também conhecida como MIL (*Malfunction Indicator Light*). Um veículo moderno pode suportar centenas de parâmetros, que podem ser acessados através do DLC (*Diagnostic Link Connector*) usando ferramentas de varredura, popularmente conhecidas como *Scanners* (OBD SOLUTIONS, 2015). A Figura 2.2 ilustra os componentes básicos de um sistema OBD.

Figura 2.2 – Componentes de um sistema OBD



Fonte: (OBD SOLUTIONS, 2015)

Nos veículos modernos, a ECU é responsável pela leitura dos dados de um grande conjunto de sensores analógicos. Com base nessas informações, ela é capaz de aplicar algoritmos inteligentes para controlar os atuadores de forma a obter níveis adequados de emissão de poluentes e de consumo de combustível, além de permitir maior segurança durante a condução. Uma ECU é responsável, por exemplo, por determinar qual é o momento correto para acionar o sistema antibloqueio das rodas, popularmente conhecido como ABS, permitindo a parada total do veículo em um intervalo de tempo menor, se comparado a não utilização desse sistema.

Tipicamente, as ECUs são baseadas em microcontroladores programáveis de 8 a 32 bits, com poucos megabytes de memória, operando entre 32 e 100 MHz. Apesar da baixa frequência de operação, se comparado aos computadores pessoais, esses controladores são capazes de processar em tempo real, desde funções simples, como a sinalização de abertura de uma porta, até o controle de sistemas críticos como os de frenagem e tração. Para isso, alguns requisitos básicos são essenciais, como: suporte à conversão de sinais analógicos e digitais, geração de sinais, temporizadores, entrada e saída serial e paralela, tolerância a falhas e baixo tempo de resposta. Alguns exemplos de controladores que se destacaram em sistemas desse tipo, são, por exemplo: Infineon Tri-core, Atmel AVR, Microchip PIC, Renesas e o Intel 8051. Diferente de

outras aplicações, como aquelas relacionadas à Internet das Coisas, em que há uma constante preocupação com o baixo consumo de energia, a ECU possui alimentação provida pela bateria do veículo, a qual é carregada pela própria combustão do motor, dessa forma, é possível utilizar dispositivos comuns de mercado e, por consequência, mais baratos.

2.1.2 Modos de Operação e *Parameter IDs*

As informações que podem ser obtidas de um sistema OBD-II estão organizadas em modos de operação e códigos de parâmetros (PIDs). O modo de operação deve ser informado ao realizar uma requisição para a ECU e consiste em um valor hexadecimal de dois dígitos que deve estar entre os valores 0x01 e 0x0A. Cada modo de operação pode ter até 256 PIDs, que são representados também por um número em hexadecimal de dois dígitos. O padrão OBD-II regulamenta um conjunto de PIDs e modos de operação, sendo que alguns deles são de disponibilização obrigatória pelo fabricante, especialmente aqueles voltados à emissão de gases poluentes. Outros parâmetros previstos pelo padrão são de adoção facultativa, ou seja, cabe ao fabricante decidir se eles estarão disponíveis de forma aberta para ferramentas de diagnóstico.

Além dos modos de operação e PIDs padronizados, o próprio fabricante pode definir códigos adicionais cuja leitura só pode ser realizada com o conhecimento das especificações proprietárias. O mercado de certificações para o desenvolvimento dessas ferramentas é bastante restritivo e, geralmente, as montadoras cobram caro por esse tipo de capacitação. Devido a isso, os chamados *scanners*, em geral, não são acessíveis ao consumidor final. O custo de um dispositivo moderno certificado pelo fabricante pode variar de centenas até alguns milhares de reais (MERCADODOMECANICO, 2016), por consequência, uma varredura completa do veículo passa a ser realizada apenas por oficinas mecânicas na ocorrência de anomalias perceptíveis pelo condutor.

O código de leitura 0x0000 (modo de operação e PID) é reservado para uma informação bastante relevante, que diz respeito à aderência do veículo ao padrão OBD. A resposta de uma requisição por esse código é um vetor de bits em que cada posição indica se o PID correspondente a essa posição é suportado pelo veículo ou não. A seguir são descritos os outros modos de operação do padrão OBD-II, segundo o *Website The Best OBD-II Scanners* (SCANNERS, 2016).

O modo 0x01 é destinado para dados de diagnóstico atuais e corresponde à primeira interação entre o veículo e as ferramentas de escaneamento. Esse modo permite obter informações sobre o estado atual do veículo como o tipo de motorização, a presença ou não de falhas no sistema, a intensidade da emissão de poluentes, algumas informações em tempo real, como

a velocidade instantânea, a frequência de rotação do motor (em RPM), a temperatura do fluído de arrefecimento, a posição do pedal do acelerador e a proporção entre etanol e gasolina da mistura de combustível presente no tanque. Apesar de ser apenas a fase inicial do processo de diagnóstico, o modo 0x01 é de extrema importância, pois facilita a identificação da causa raiz do problema em vez de apenas direcionar para o desligamento da luz de indicação de falhas no painel de instrumentos.

O modo 0x02, dados de diagnóstico armazenados, permite consultar as mesmas informações do modo 0x01, porém, em uma visão armazenada em um determinado instante de tempo. O modo 0x02 pode ser considerado como um *snapshot* da situação do veículo no momento da falha, o que facilita o trabalho de reparo quando não é possível reproduzir a situação relatada pelo condutor. Alguns veículos tem a capacidade de armazenar dados de diagnóstico para múltiplas falhas, outros, no entanto, armazenam apenas da última ocorrência, mesmo assim, trata-se de uma informação muito útil para nortear a solução do problema.

O modo 0x03, códigos de erros armazenados, é considerado pelos mecânicos como o modo mais importante dos sistemas OBD. Esse modo permite consultar a relação de falhas ocorridas no sistema desde a última limpeza dos dados. Esses códigos possuem cinco dígitos cujas combinações representam erros bastante específicos, por exemplo, nos sistemas de transmissão, combustão, exaustão e também na rede de comunicação entre os componentes. Algumas ferramentas de diagnóstico podem, inclusive, sinalizar soluções para cada código de erro que estiver sendo reportado. Nos veículos modernos, em que existe um grande conjunto de componentes, seria impraticável a manutenção artesanal sem o auxílio dessas ferramentas.

O modo de operação 0x04 possibilita a limpeza dos registros de erros. Por meio desse modo é possível desligar a luz de indicação de problemas do painel de instrumentos (*Malfunction Indicator Light - MIL*) após a solução da anomalia. Seja em ferramentas altamente complexas e caras, ou em equipamentos de diagnóstico simples e baratos, o modo 0x04 sempre será utilizado, pois restaura o estado do sistema e ajuda a monitorar sua reincidência.

Em veículos movidos por combustíveis fósseis, o modo 0x05, permite consultar os resultados de testes do sensor de oxigênio, cujas leituras são realizadas pela sonda lambda [1]. Em ECUs mais modernas, esse modo de operação não é mais suportado, tendo sido suas funções substituídas pelo modo de operação 0x06.

O modo 0x06 está relacionado aos testes de monitoradores não contínuos, ou seja, que atuam sobre sistemas sem monitoramento constante. Trata-se de um modo avançado que é

[1] A sonda lambda, ou sensor de oxigênio, é um dispositivo que envia um sinal elétrico para a ECU do veículo, que por sua vez controla a injeção eletrônica de acordo com a presença de oxigênio medida nos gases de escape. Dessa forma, é possível controlar a quantidade de combustível a enviar para o motor.

executado esporadicamente, por exemplo, ao dar partida no motor. Após isso, o mecanismo é adormecido até a realização do próximo teste, diferente dos sistemas de combustão e ignição, que estão continuamente sendo averiguados. Um exemplo de informação desse tipo é o monitoramento do sistema catalisador, o qual é responsável por transformar gases tóxicos em gases menos poluentes.

O modo 0x07, informação de testes de monitores contínuos, geralmente está presente em ferramentas mais avançadas de diagnóstico. Esse modo permite consultar dados referentes a um determinado ciclo de condução. Essa informação é útil para que o mecânico possa verificar se o reparo realizado de fato resolveu o problema. Os dados armazenados, se referem a sistemas que estão constantemente sendo diagnosticados, por exemplo, análise do combustível utilizado, falhas de ignição e emissão de gases. Esse modo de operação, assim como o modo 0x06, exige maior experiência para sua correta interpretação, além disso, está disponível apenas em ferramentas mais caras.

Diferente dos demais, o modo 0x08, controle do sistema de bordo ou teste de componente, permite uma comunicação bidirecional entre a ferramenta de escaneamento e a ECU do veículo. Ou seja, com ele é possível alterar parâmetros de operação e, portanto, testar diferentes hipóteses até identificar a causa do problema. Esse modo deve ser utilizado com cuidado, pois se utilizado sem o conhecimento necessário pode causar danos ao veículo. A chamada *chipagem*, técnica que se tornou bastante popular nos últimos anos, permite aumentar a potência do motor em alguns de cavalos de força apenas por meio da reprogramação da ECU do veículo. Existem produtos de mercado, como o Nitro OBD 2 Tunning e o ProRacingX OBD 2, que prometem um ganho de até 35% em potência e de até 25% em torque apenas com o uso de um dispositivo OBD-II. Apesar disso, essa técnica não é recomendada pelos mecânicos e fabricantes, pois expõe o veículo à situações de operação para as quais não foi projetado. Por outro lado, existem adaptadores que propõem uma abordagem mais ecológica, como o ECO OBD2. Esse produto remapeia o sistema de injeção eletrônica de acordo com os hábitos do motorista para obter até 15% de redução no consumo de combustível.

O modo de operação 0x09 permite obter informações sobre o veículo e sobre o software instalado na ECU. Seu uso mais comum é consultar o chamado VIN (*Vehicle Identification Number*) que consiste em um código identificador único para carros e caminhões. Outro uso típico é o acesso à identificação e versão do software instalado. Esse modo também permite obter relatórios envolvendo o sistema catalisador, o sensor de oxigênio, vazamentos no sistema de evaporação, controle de pressão, e contadores de eventos como o número de ignições e de testes de autodiagnóstico realizados.

Por fim, o modo 0x0A, códigos de erros permanentes, consiste em uma lista de erros armazenados toda vez que a luz de indicação de falhas é ativada. Diferente do modo 0x03, essa lista não pode ser apagada, nem mesmo desconectando a alimentação da ECU, por isso, funciona como um histórico permanente das anomalias do veículo. A motivação principal para a criação desse modo de operação foi evitar que o condutor apagasse a relação de erros contida sistema antipoluição imediatamente antes dos testes realizados por órgãos fiscalizadores. Em algumas localidades dos Estados Unidos, que são altamente populadas, como Los Angeles, Dallas e Boston, esse modo de operação é obrigatório, pois possui grande importância para a redução efetiva na emissão de poluentes.

2.1.3 Protocolos OBD

Segundo Nagel (NAGEL, 2011), o padrão OBD utiliza apenas três camadas do modelo de referência OSI: as camadas um, dois e sete. A primeira refere-se ao nível físico e define a interface do barramento, o conector padrão, o meio de transmissão e a codificação de linha. A segunda define o controle de acesso ao meio e o controle lógico de enlace. Além disso, oferece detecção de erros e encapsulamento de mensagens do nível superior. A camada sete corresponde à aplicação, onde se encontra o padrão OBD (J1979/ISO 15031-5). Uma série de padrões foram definidos para as arquiteturas de comunicação intraveiculares, muitos deles normatizados pelo ISO (*International Organization for Standardization*) os quais estão ilustrados na Tabela 2.1.

Tabela 2.1 – Modelo OSI e Protocolos OBD

Aplicabilidade	**Camada OSI**	**Protocolos de Diagnóstico**			
Sete Camadas de acordo com ISO/IEC 7498 e ISO/IEC 10731	Física (nível 1)	ISO 9141-2	ISO 14230-1	SAE J1850	ISO 11898 / ISO 15765-4
	Enlace (nível 2)	ISO 9141-2	ISO 14230-2	SAE J1850	ISO 11898 / ISO 15765-4
	Rede (nível 3)				ISO 15765-2 / ISO 15765-4
	Transporte (nível 4)				
	Sessão (nível 5)				ISO 15765-4
	Apresentação (nível 6)				
	Aplicação (nível 7)	SAE J1979 / ISO 15031-5	SAE J1979 / ISO 15031-5	SAE J1979 / ISO 15031-5	SAE J1979 / ISO 15031-5

Fonte: (SAE, 1991)

Em relação aos níveis físico e de enlace, o padrão OBD-II prevê cinco protocolos diferentes, no entanto, a maioria dos veículos implementa apenas um deles. O padrão E-OBD (*European On-board Diagnostics*) ainda prevê dois protocolos adicionais: ISO 13400-3 e J1939-21. No entanto, o foco do presente trabalho estará sobre o padrão americano, o qual é largamente adotado em diferentes partes do mundo, inclusive no Brasil. Em geral, é possível identificar o protocolo utilizado pelo veículo por meio dos pinos presentes no conector SAE J1962. A Tabela 2.2 detalha as características dos protocolos de sinalização suportados.

Tabela 2.2 – Características dos protocolos permitidos pelo padrão OBD-II

Protocolo	Taxa	Mensagem	Tensão	Observações
SAEJ1850 PWM	41,6 kbit/s	12 bytes	+5V	Comunicação diferencial de duas vias, controle de acesso ao meio CSMA, Modulação por Largura de Pulso
SAEJ1850 VPW	10,4 kbit/s ou 41,6 kbit/s	12 bytes	+7V	Comunicação em uma via, controle de acesso ao meio CSMA, Largura de Pulso Variável
ISO9141-2	10,4 kbit/s	Até 12 bytes	+12V	Comunicação serial assíncrona
ISO14230-2 KWP2000	Até 10,4 kbit/s	Até 255 bytes de dados	+12V	Comunicação serial assíncrona
ISO15765-4 CAN	250 kbit/s ou 500 kbit/s	Até 8 bytes de dados	+3.5V	Comunicação diferencial e balanceada (correntes em sentido inverso em cada sinal)

Fonte: Elaborada pelo autor

Independente do protocolo de sinalização utilizado, os comandos esperados pelas ECUs são os mesmos, o padrão J1979 (SAE, 1991) define a lista de códigos identificadores (PIDs) que permitem consultar os parâmetros de diagnóstico, como velocidade, RPM, temperatura do motor, entre outros. A identificação do protocolo de sinalização utilizado pelo veículo pode ser realizada por meio dos pinos presentes no conector OBD padrão, conforme a Tabela 2.3.

Tabela 2.3 – Identificação do protocolo de sinalização baseada nos pinos do conector J1979

Protocolo	Pino 2	Pino 6	Pino 7	Pino 10	Pino 14	Pino 15
SAEJ1850 PWM	necessário	-	-	necessário	-	-
SAEJ1850 VPW	necessário	-	-	-	-	-
ISO9141-2	-	-	necessário	-	-	opcional
ISO14230-2 KWP2000	-	-	necessário	-	-	opcional
ISO15765-4 CAN	-	necessário	-	-	necessário	-

Fonte: (OBDTESTER, 2015)

O protocolo ISO 15765-4 (ISO, 2005) é o mais recente e é obrigatório para todos os veículos fabricados a partir de 2008 nos Estados Unidos. Trata-se de um padrão internacional de envio de pacotes de dados através de um barramento CAN (*Controller Area Network*). Um barramento CAN baseia-se no uso de mensagens geradas por *broadcast* contendo um dispositivo central controlador de mensagens, no caso específico dos sistemas OBD, a ECU do veículo. As CANs trabalham com uma topologia de rede física em estrela e lógica em barramento, ou seja, os dispositivos periféricos não estão fisicamente conectados entre sí, mas ao dispositivo central que, por sua vez, está conectado aos demais. As mensagens são enviadas em modo *broadcast*, portanto, todos os nós podem ouvir a todas as mensagens que são transmitidas no barramento. Como baseiam-se em um meio físico compartilhado, as redes CAN necessitam de um rigoroso controle de acesso que é orquestrado pelo dispositivo central.

O padrão ISO 14230-1 (ISO, 2000), também conhecido como *Keyword Protocol 2000*, é muito comum em veículos fabricados a partir de 2003. Tendo sido oficialmente publicado em junho do ano 2000, o protocolo possui comunicação serial bidirecional com codificação de linha NRZ (*Non Return to Zero*) e sua taxa de símbolos está entre 1,2 e 10,4 Kbaud. A especificação ISO 14230-2 diz respeito à camada de enlace, a qual possui pacotes de até 255 bytes de dados com detecção de erros por meio de um *checksum* de oito bits. O protocolo prevê a operação de múltiplas ECUs sobre o mesmo barramento, portanto, o cabeçalho dos pacotes de nível de enlace possui endereço de origem e de destino da mensagem.

Inicialmente utilizado pelas fabricantes Chrysler, European e Asia Vehicles, o protocolo ISO 9141-2 baseia-se em comunicação serial assíncrona semelhante ao protocolo RS-232 (WIKIPEDIA, 2014), no entanto, com níveis de sinal diferentes e a comunicação ocorrendo em uma única linha digital, bidirecional com sinalização UART (*Universal Asynchronous Receiver/Transmitter*). Um transmissor UART é um dispositivo de hardware que converte dados paralelos para a forma serial de modo a realizar o envio por meio de um canal digital. O receptor UART, por sua vez, é capaz de converter o sinal serial para a sua forma original paralela. A designação universal indica que as velocidades de transmissão de dados são configuráveis. Os pacotes podem conter até 255 bytes de dados e possuem marcadores de início e término de pacote, além de bits de paridade para detecção de erros (SINGH, 2014).

A especificação SAE J1850 faz referência a duas versões do protocolo (SPARKS, 2004). A versão PWM possui taxa de transmissão de 41,6 Kbps, em duas vias diferenciais e baseia-se em modulação por largura de pulso (*pulse width modulation*), enquanto que a versão VPW possui taxa de transmissão de 10,4 Kbps e modulação de linha baseada em Largura de Pulso Variável (*variable pulse width modulation*) em uma única via. Apesar de ambas as modulações

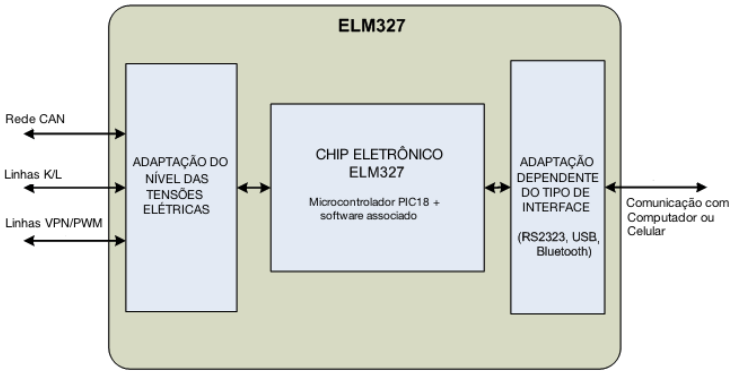
serem um tipo de PDM (*pulse duration modulation*), são ligeiramente diferentes. A principal diferença entre as duas modulações é que, na versão PWM, apesar de a largura do pulso ser variável, o período de onda é constante, ou seja, o intervalo entre os pulsos é o mesmo. Já na versão VPW, utilizada pelas empresas General Motors e Chrysler, o tempo em que o barramento permanece em um determinado potencial é variável, ou seja, o período entre os pulsos não é importante. Ambas as versões utilizam detecção de erros por meio de um código CRC e controle de acesso ao meio CSMA/CR, onde, em caso de conflitos, o protocolo determina qual dispositivo possui maior prioridade.

Em virtude da variedade de protocolos que fazem parte do ecossistema OBD, a empresa *ELM Electronics* foi precursora na criação do protocolo ELM-327, o qual é capaz de abstrair as complexidades de comunicação com os diferentes modelos de veículo existentes. Seu protocolo consiste em um conjunto de comandos simples e permite, inclusive, a identificação automática dos protocolos de nível mais baixo. Dessa forma, o programador não precisa se prender a detalhes de comunicação com o meio físico. A interface original de comunicação com o ELM 327 é o protocolo serial RS-232, um padrão largamente conhecido e adotado em aplicações para computadores pessoais. Com esse avanço, a comunicação com a ECU do veículo se tornou relativamente simples, porém exige a necessidade de aquisição de dispositivos de hardware que implementem esse protocolo.

2.1.4 Adaptadores de Hardware

Como o funcionamento do protocolo ELM 327 é de domínio público, existem inúmeros dispositivos no mercado que o implementam. Esses dispositivos possuem interface RS 232, *Bluetooth*, USB e até mesmo WiFi, o que possibilita sua integração com aplicativos para *smartphones* ou computadores pessoais. Apesar disso, nem todos os veículos respondem a todos PIDs solicitados, o que dificulta a criação de aplicações com vasta portabilidade. Esse é um dos principais desafios para que soluções utilizando dados de diagnóstico sejam desenvolvidas e adotadas pelo público geral. Como a regulamentação ainda é fraca em relação a esse assunto, obrigando apenas a disponibilização de dados relacionados à emissão de poluentes, a maioria dos fabricantes de veículos não oferecem todos os dados. A Figura 2.3 ilustra a arquitetura de um adaptador ELM 327.

Figura 2.3 – Arquitetura de um adaptador ELM 327



Fonte: (OUTILS OBD FACILE, 2010)

Um adaptador ELM 327 é relativamente barato e pode ser adquirido por qualquer pessoa por algumas dezenas de reais. Em geral, esses dispositivos são implementados sobre o microcontrolador PIC da empresa Microchip Technology, ou assemelhados, tal qual a própria ECU do veículo. Em geral, os adaptadores que disponibilizam interface WiFi são mais caros e gastam mais energia, porém, oferecem maior segurança se comparado aos dispositivos Bluetooth. Além disso, os adaptadores WiFi são os únicos que podem se comunicar com *smartphones* e *tablets* da empresa Apple. A Figura 2.4 mostra alguns exemplos de adaptadores existentes no mercado.

Figura 2.4 – Diferentes adaptadores ELM 327

Fonte: (OUTILS OBD FACILE, 2010)

Também existem dispositivos independentes do protocolo ELM 327, dentre eles pode-se citar: OpenXC, FleetCarma, Garmin EcoRoute, GoPoint BT1 e Actron U-Scan. Esses dispositivos expõem uma interface que não é compatível com o padrão ELM 327, portanto, exigem software proprietário para a leitura dos parâmetros de diagnóstico. A plataforma OpenXC oferece um conjunto de bibliotecas que facilitam sua integração com aplicativos Android. Além disso, o projeto de hardware também é *Open Source*, portanto, o próprio usuário pode fabricar seu adaptador. A empresa FleetCarma, anteriormente conhecida como Crosscharm, oferece uma solução baseada na plataforma OpenXC que acompanha uma solução de computação em nuvem para o acompanhamento do consumo de combustível de veículos de passeio e de frotas. A empresa cobra pelo serviço e também oferece suporte a veículos elétricos.

2.1.5 Aspectos de Segurança

A partir do momento em que informações sensíveis dos usuários, como sua localização geográfica, modelo de seu veículo e seus hábitos diários estão disponíveis em uma interface de comunicação sem fio, torna-se necessária a preocupação com a segurança. Infelizmente, sabemos que, no mundo virtual, sempre há quem queira utilizar informações a seu próprio benefício ou para prejudicar outras pessoas. Evidência disso são os inúmeros mecanismos de criptografia e segurança que são necessários para tornar a Internet menos perigosa aos usuários.

Sabendo que o padrão OBD permite a modificação de parâmetros da ECU do veículo, o acesso à sua interface pode ser bastante perigoso se estiver em mãos erradas. Alguns adaptadores, como o *OBDII Car Auto Door Lock/Unlock*, permitem, por exemplo, bloquear as portas do veículo quando este estiver a mais de 10 Km/h, ou então, destravá-las quando o veículo for estacionado. Considerando que esse tipo de interação é possível, pessoas mal intencionadas poderiam, por exemplo, obter acesso ilícito à essa interface e roubar um veículo que estivesse estacionado sem a necessidade de abri-lo à força. Nas redes sociais de compartilhamento de vídeos é fácil encontrar simulações controladas sobre o que é possível fazer a partir do acesso ao barramento CAN. Em alguns desses vídeos, o programador permanece sentado ao lado do motorista, apenas com conexão à Internet, sem ao menos estar conectado diretamente ao veículo e consegue interferir na condução de forma bastante perigosa. São exemplos de intervenções que podem ser realizadas: desligamento total do motor, controle do volante, obtenção das coordenadas geográficas do veículo, acionamento da buzina, dos faróis e dos freios. Como em todas as áreas de tecnologia, as inovações trazem grandes avanços e melhoram a vida das pessoas, porém está sempre acompanhada de alguns riscos. Com todos esses recursos em mãos, um

atacante poderia controlar completamente o veículo de terceiros e, inclusive, colocando a vida de pessoas em risco.

Por exemplo, em uma notícia publicada pelo *Website* Wired (ZETTER, 2015), foi divulgada a possibilidade de invasão do sistema a bordo de um veículo Model S da empresa Tesla Motors. A vulnerabilidade foi apontada por Kevin Mahaffey e Marc Rogers, dois pesquisadores que perceberam serem capazes de conectar seus *laptops* em um cabo de rede presente no veículo e, então, dar partida e controlá-lo por meio de comandos de computador. A descoberta foi realizada após dois anos de estudos e testes sobre a arquitetura do veículo Tesla Model S e foi discutida durante o evento *Def Con hacker conference* em Las Vegas. A vulnerabilidade estava em uma versão desatualizada do sistema de entretenimento a bordo, mais especificamente em uma versão da *engine* para *web browsers* chamada Webkit. Teoricamente, um *hacker* poderia criar uma página maliciosa na Internet que, se acessada a partir de um Tesla Model S, poderia fornecer controle total do veículo para o atacante. Algumas semanas depois, a Tesla Motors, em trabalho conjunto com os dois pesquisadores, lançou um pacote de atualização que corrigiu essa e outras cinco vulnerabilidades identificadas no veículo. Os pesquisadores planejam continuar trabalhando com a Tesla na segurança de seus veículos. A empresa de automóveis também recentemente contratou Chris Evans, engenheiro de segurança altamente respeitado, que costumava liderar equipes de segurança do Google Chrome, para dirigir a sua própria equipe de segurança.

2.2 *In-Vehicle Infotainment Systems*

Os chamados *In-Vehicle Infotainment Systems*, ou IVI, consistem em sistemas de hardware e software integrados à arquitetura dos veículos automotores fabricados atualmente e que tem por objetivo entregar informação e entretenimento de forma inteligente aos seus passageiros. Esses sistemas utilizam, frequentemente, tecnologias como *Bluetooth* para a integração com *smartphones* de modo a potencializar as facilidades oferecidas. Dentre essas, estão o acesso a funções do celular, o controle do sistema multimídia por comandos de voz, o acesso a informações relevantes durante a condução, como condições do tráfego e do tempo, o acesso e o acompanhamento de redes sociais e a navegação por georreferenciamento. Entretanto, os sistemas IVI têm o papel de oferecer esses recursos ao condutor de modo que isso não comprometa a segurança durante a condução.

2.2.1 O papel de *Infotainment Systems* na economia de combustível

É de conhecimento das pessoas que existem diversos fatores que contribuem para o consumo excessivo de combustíveis: veículos cada vez mais potentes, engarrafamentos frequentes, uso desnecessário do automóvel em trajetos curtos, transporte de cargas pesadas, manutenções atrasadas, aceleradas bruscas durante o percurso e, sobretudo, pouco incentivo à adoção de veículos híbridos ou puramente elétricos. A maioria desses fatores não estão ao controle das pessoas que utilizam esse meio de transporte. No entanto, por estarem tão perto dos passageiros, os sistemas IVI tem o potencial de agir sobre uma das causas que dependem exclusivamente das atitudes ao volante: o estilo de condução. Apesar de todas as padronizações de protocolos e tecnologias que se aplicam aos sistemas de diagnóstico a bordo, ainda existe certa separação desses em relação aos sistemas IVI. A integração entre ambos poderia, de certa forma, ser melhor aproveitada de modo a agregar valor econômico, para o condutor, e ambiental para o planeta.

2.2.2 Tecnologias de *Infotainment Systems*

Os sistemas IVI permitem integrar diversos sensores e tecnologias para oferecer uma experiência mais interativa aos passageiros como GPS (*Global Position System*) para uma navegação facilitada por meio de mapas georreferenciados; redes sem fio para a integração entre os componentes do sistema e para a comunicação via Internet; sistemas multimídia para o controle de reprodução de músicas; reconhecimento de fala para uma condução mais segura permitindo o uso do sistema sem tirar as mãos do volante; Bluetooth para a comunicação com dispositivos celulares; aplicativos de trocas de mensagens integrados ao reconhecimento de fala; câmeras auxiliares para a substituição dos espelhos retrovisores, ou mesmo para a assistência ao estacionamento; WiFi para o compartilhamento do acesso à Internet com os passageiros do veículo e *smart watches*, que são relógios de pulso interativos, que podem ser integrados aos veículos para, por exemplo, controlar a temperatura do ar condicionado.

Apesar da presença de tantas tecnologias em um simples veículo de passeio, ainda são poucos os fabricantes que disponibilizam a integração entre os sistemas IVI e o sistema de diagnóstico a bordo. Uma experiência ainda mais personalizada e diferenciada poderia ser proporcionada se estivessem disponíveis informações como: custo associado a uma determinada viagem, média de consumo praticada em uma avenida no instante em que o condutor estiver saindo de casa, velocidades em que o veículo se torna mais econômico, horários em que o

trajeto para o trabalho é mais barato, média de consumo de outros condutores que possuem o mesmo veículo e seu histórico de comportamentos de condução. A tecnologia existente já é suficiente para que esse tipo de recurso esteja disponível, a dependência, no entanto, está no mercado que ainda se consolida nas funcionalidades mais usuais e populares dos sistemas IVI, como a integração com os *smartphones*, a realização de ligações telefônicas, a interface de comandos de voz e a navegação GPS.

2.3 Mercado atual de *Infotainment Systems*

Segundo o Gartner, os chamados *In-Vehicle Infotainment Systems*, ou IVI, possuem um forte potencial para o mercado de semicondutores podendo movimentar cerca de três bilhões de dólares em 2017 (HINES, 2014). No entanto, a indústria automotiva deve abordar alguns problemas de usabilidade e obsolescência tecnológica para impulsionar a adoção generalizada pelo público. Em outras palavras, ainda existem aspectos a aprimorar para que esse tipo de tecnologia possa ser largamente adotado, exemplo disso é a infraestrutura de telefonia das cidades brasileiras que não acompanhou o grande crescimento na demanda por serviços de dados. Por outro lado, as relações entre fabricantes de veículos e empresas de tecnologia ainda são recentes, e que alguns produtos ainda precisam se adequar aos padrões de usabilidade e exigência do usuário moderno.

2.3.1 GENIVI Alliance

A GENIVI Alliance (GENIVI ALLIANCE, 2015) é uma das iniciativas pioneiras no que diz respeito à introdução de software livre dentro dos carros. Trata-se de uma organização sem fins lucrativos que visa padronizar e fornecer arquiteturas de qualidade, baseadas em software livre, para impulsionar a adoção de novas tecnologias intraveiculares pelo mercado. Existem diversos programas de *compliance* a partir dos quais, empresas podem receber insumos produzidos pela GENIVI para desfrutar de uma drástica redução nos custos de desenvolvimento e no *time to market*.

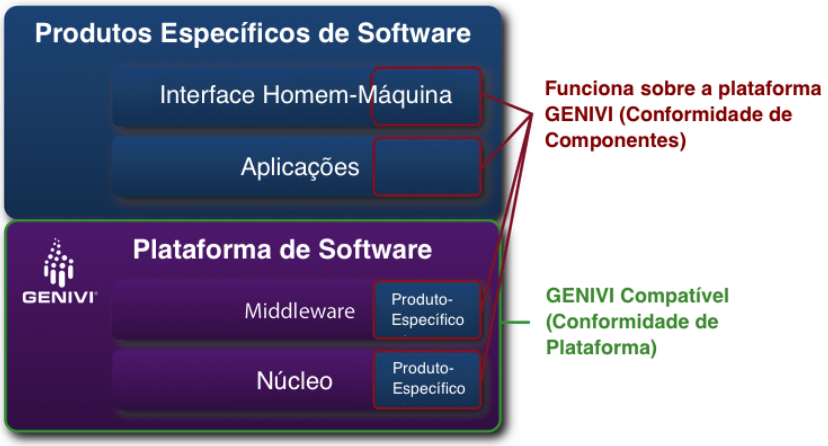
É possível destacar duas grandes frentes de atuação propostas pela GENIVI Alliance: projetos de código aberto para a comunidade geral e a aliança exclusiva para membros. A GENIVI fornece um conjunto de entregáveis, dentre eles, projetos de software livre que podem ser utilizados por membros e não membros no desenvolvimento de aplicações automotivas.

Na linha de membros participantes da aliança, existem inúmeras empresas que aderiram ao programa de *compliance* da GENIVI, desde fabricantes de *microchips* até montadoras de veículos. No setor automotivo, são exemplos BMW Group, Hyundai, Jaguar, Land Rover, Nissan, Peugeot / Citroën, Renault e Volvo. Outros membros significativos do grupo são Altera, Xilinx, ARM, Intel, Micron e Nvidia. Essas empresas possuem acesso a diversas especificações técnicas e insumos produzidos pela GENIVI para que possam propor suas próprias arquiteturas de soluções IVI.

As interfaces dos softwares da GENIVI são definidas usando Franca IDL: um *framework* de definição de interfaces criado sobre a plataforma Eclipse em 2011 e que passou a ser adotado como um padrão de mercado para inúmeras finalidades, dentre elas, para as plataformas IVI. Por meio de uma linguagem apropriada, é possível manter sob controle a integração entre softwares de diferentes fabricantes (ECLIPSE, 2013). Contudo, há a possibilidade de estender os recursos, implementando software proprietário para complementá-los.

Existem dois níveis diferentes de *compliance*: o primeiro deles é chamado *Product-Specific Sofware* e é voltado àqueles que desejam desenvolver soluções IVI sobre o *middleware* definido pela GENIVI. Esses parceiros se baseiam nas especificações *Component Compliance*. O segundo nível, *Software Platform*, diz respeito à implementação de componentes internos ao *middleware* e baseia-se nas especificações *Platform Compliance*. A Figura 2.5 ilustra em alto nível a arquitetura GENIVI.

Figura 2.5 – Arquitetura GENIVI



Fonte: (GENIVI ALLIANCE, 2015)

2.3.2 Car Connectivity Consortium - MirrorLink®

Car Connectivity Consortium, ou simplesmente CCC, trata-se de uma organização composta por empresas de tecnologia e fabricantes de veículos que, em conjunto, desenvolvem soluções e padrões abertos para veículos inteligentes. O principal produto do grupo é chamado MirrorLink® (CCC, 2015), uma solução IVI cuja proposta é reduzir distrações ao volante garantindo a maior compatibilidade possível entre veículos e *smartphones*.

O MirrorLink® permite que o condutor execute na central multimídia do carro, ou até mesmo em alguns *DVD players* compatíveis, qualquer aplicativo que tenha sido desenvolvido sobre seu SDK. Dado que não há restrição quanto aos aplicativos compatíveis, é possível entregar ao usuário final o que há de mais recente nas lojas virtuais de aplicativos para celulares, aumentando o valor agregado da ferramenta. O nome da solução deriva do fato de que os aplicativos espelham sua tela na central IVI via conexão Wi-Fi ou USB. Com isso, o usuário pode utilizar a interface *touch-screen* do carro para interagir com o *smartphone* sem a mesma distração necessária para utilizá-lo diretamente. Para garantir esse requisito, os desenvolvedores devem seguir rigorosas *guidelines* de usabilidade definidas pelo Car Connectivity Consortium.

Em um futuro próximo, o CCC pretende integrar a plataforma a diferentes sensores dos veículos para garantir uma experiência ainda mais completa de navegação. Atualmente, o MirrorLink® é suportado por diversos modelos de carros, *smartphones* e reprodutores de mídia. Existem cerca de dez aplicativos compatíveis na loja virtual Google Play, mas ainda não há suporte para celulares com sistema operacional iOS. A interação com o veículo se limita à navegação GPS, ao controle de reprodução de músicas e aos alertas para uma condução segura (CCC, 2014).

2.3.3 Apple - CarPlay

O CarPlay é a solução proposta pela Apple, disponível para iOS9, que pretende tornar o uso do iPhone mais seguro durante a condução. Com ele, é possível utilizar o poderoso mecanismo de reconhecimento de fala - Siri - para controlar o *smartphone* enquanto se dirige. Além disso, a usabilidade do sistema operacional e dos aplicativos foi completamente redesenhada de modo a melhor atender as expectativas de uso na tela do carro. O sistema permite realizar chamadas, ouvir e enviar SMS, controlar a *playlist*, ouvir *podcasts*, livros e rádios por meio de comandos de voz. Com isso, pretende-se proporcionar ao usuário uma navegação mais segura e, ao mesmo tempo, interativa (APPLE INC, 2015).

A criação de aplicativos para o CarPlay ainda é restrito para alguns desenvolvedores selecionados (OLIVER, 2014), portanto, a relação de aplicativos para *download* ainda é pequena. Podcasts, At Bat, Spotify, Sticher, Rdio, Overcast, Audible e Audiobooks.com são alguns exemplos que podem ser encontrados na loja virtual AppStore. Do ponto de vista do mercado automobilístico, o CarPlay é uma das soluções atuais com maior aderência e conta com marcas representativas, tais como: Ferrari, BMW Group, Mitsubish Motors, Chrysler, Porshe, Suzuki, dentre outras.

2.3.4 Google

Em Junho de 2014, a gigante Google firmou novas parcerias com diversas companhias automotivas para alavancar a chamada OAA - *Open Automotive Alliance*. Originalmente fundada pelos membros Audi, GM, Google, Honda, Hyundai e Nvidia, o objetivo dessa iniciativa é trazer os melhores recursos do sistema operacional Android para o interior dos carros (GOOGLE, 2015b). Para os fabricantes de veículos, a aliança é muito vantajosa, dado que os *smartphones* com Android possuem grande penetração no mercado e estão nas mãos de milhões de pessoas. Para os motoristas, a promessa é proporcionar a utilização dos recursos do *smartphone* de maneira mais segura, evitando acidentes e distrações. Além disso, a OAA propõe uma plataforma única de desenvolvimento, a mesma já utilizada para os *smartphones* com Android, reduzindo a curva de aprendizagem e facilitando sua aceitação por fabricantes de software.

O Android Auto é um projeto da OAA que consiste em uma série de recursos para desenvolvedores que permitem integrar os aplicativos Android ao veículo. Dentre os recursos, é possível destacar o avançado sistema de reconhecimento de fala do Google Now, o qual também é capaz de realizar buscas no portal Google, o que o torna ideal para responder perguntas e solucionar as dúvidas do motorista durante a condução (KLEINA, 2014). Além disso, existe a possibilidade de se integrar aos controles do volante, minimizando as distrações para que o condutor possa manter o foco na estrada. A solução já incorpora nativamente alguns aplicativos do próprio Android, como Google Maps, Google Play Music e realização de chamadas de celular.

Para criar aplicativos para Android Auto, os desenvolvedores devem seguir diretivas específicas de usabilidade, as quais são voltadas para uma interface simples e sem distrações. Já existem diversos aplicativos compatíveis na loja Google Play, como o Spotify, o iHeart Radio e o Skype. O Android Auto foi desenvolvido para ser compatível com os *smartphones* Android versão 5.0 (Lollipop). Os recursos não envolvem a obtenção de dados de diagnóstico, ou

controle do veículo, a menos que dispositivos adicionais, como leitores OBD, sejam utilizados.

A fabricante Volksvagen lançou diversos veículos em 2016, como os modelos Golf R e o Fox, que já possuíam integração ao Apple Car Play e ao Mirrorlink, agora também integrados ao Android Auto e promete que, o restante de sua linha de 2016, também contará com essa tecnologia (LANCASTER, 2015). Ao mesmo tempo, a Hyundai lançou a nova versão do sedã de luxo Sonata também integrada ao Android Auto. Além da central inteligente, o Sonata conta com alguns recursos adicionais de segurança, como a redução automática de velocidade ao perceber freadas do veículo à frente e aviso de esquecimento da chave no interior do veículo (HYUNDAI, 2015). Outra inovação é a possibilidade de se integrar ao aplicativo BlueLink, que chama socorro em caso de acidente, que permite aquecer ou resfriar o interior do carro através do relógio de pulso e possibilita acender as luzes ou abrir a garagem automaticamente ao chegar em casa. O modelo de luxo, no entanto, não está à venda no Brasil.

Apesar da curta trajetória no segmento, o Google possui importante influência no que diz respeito à mobilidade. A Open Automotive Alliance reforça seu posicionamento enquanto empresa de tecnologia e os resutados apresentados em 2015, cerca de dois anos após sua fundação, já demonstram a competitividade que impõe aos seus concorrentes.

2.3.5 Microsoft - Windows Embedded Automotive

A Microsoft é a empresa de tecnologia que está presente a mais tempo no mercado de IVI e pode ser considerada a líder do segmento, até o momento, devido à larga adoção de sua plataforma pelos fabricantes de veículos. Semelhante à GENIVI, a Microsoft possui uma arquitetura de *middleware* bastante robusta e bem definida, cujas APIs podem ser acessadas e estendidas por softwares de terceiros. Baseada no sistema operacional Windows CE for Automotive, a arquitetura da Microsoft pode ser encontrada em uma vasta gama de dispositivos, como navegadores GPS, centrais multimídia e reprodutores de DVD. Além disso, a plataforma promete oferecer maior robustez e desempenho, sem menosprezar a interface visual e a experiência do usuário (MICROSOFT, 2010).

O Windows Embedded Automotive também possui suporte a comandos de voz para a utilização *hands-free* dos recursos do *smartphone*, como realização de chamadas, envio e recebimento de SMS, acesso à lista de contatos, dentre outros. Possui, ainda, suporte para conexão com iPod, múltiplos formatos de mídia, localização de arquivos, controle de *playlist* e rádio, visualização da galeria de imagens e navegação Web.

2.3.6 Fiat Blue&Me

Fruto de uma parceria entre a Microsoft e a Fiat, o sistema Blue&Me é uma vertente do Windows Embedded Automotive estreada no Fiat Punto, em 2007, e que está presente atualmente no Fiat 500. A solução permite controlar funções multimídia, acessar a lista de contatos e realizar ligações por meio de comandos de voz ou dos controles do volante. Além disso, está disponível um sistema de navegação GPS próprio e integrado ao painel de instrumentos do veículo (FIAT, 2015). O Blue&Me também permite a conexão via USB, ou *Bluetooth*, com *smartphones* para a reprodução de músicas MP3 através do aplicativo BlueMe Player disponível para iOS ou BTHeadSet para Android.

O aplicativo Fiat Social Drive é uma solução que permite ao motorista saber o que está ocorrendo na Internet e nas redes sociais enquanto dirige, de maneira mais cômoda e segura. Ao se autenticar no aplicativo, o usuário pode informar quais são seus amigos favoritos. Após isso, é possível telefonar para a central Fiat Social Drive, via comandos de voz, e ser informado sobre as últimas novidades relacionadas à sua rede social preferida.

Além da integração às redes sociais, a Fiat lançou o aplicativo eco:Drive, uma solução que fornece ao motorista informações sobre as emissões de CO2 na atmosfera e sobre o consumo de combustível de seu veículo. A proposta do aplicativo é exibir esses dados em gráficos de fácil interpretação para que o condutor possa analisar e, então, aprimorar seu comportamento obtendo uma redução no consumo. Durante o trajeto, o usuário mantém um *pen drive* conectado ao veículo, no qual serão armazenados os dados de diagnóstico. Após isso, é possível exportar as informações para um computador que possua o eco:Drive instalado. Ao realizar a análise da rota, será atribuída uma pontuação de zero a cem que classificará o condutor no eco:Ranking de acordo com os dados de aceleração, desaceleração, velocidade, trocas de marcha e emissão de CO2. Atualmente o sistema está disponível no Fiat 500 e no Grande Punto, mas, em breve outros modelos equipados com o Blue & Me terão o mesmo sistema (MICROSOFT, 2010).

2.3.7 Ford SYNC e OpenXC

O sistema integrado de comunicação e entretenimento Ford Sync é uma das mais bem sucedidas iniciativas em IVI da Microsoft. Em parceria com a Ford, o software foi instalado em mais de dois milhões de veículos desde seu lançamento, em 2007 (MICROSOFT, 2010). Além dos recursos disponíveis na maioria das alternativas concorrentes, como reconhecimento de fala, navegação GPS, envio e recebimento de SMS e controle de *playlist*, a versão 5 do Ford

Sync possui funcionalidades adicionais, como o ajuste da temperatura do ar-condicionado via comandos de voz, o acionamento do viva voz ao ligar o veículo e a discagem automática para a polícia em caso de acidente (FORD MOTOR COMPANY, 2015a).

Um dos grandes diferenciais apresentados pela Ford é a disponibilização de uma suíte de desenvolvimento de aplicativos para o público geral. Por meio do AppLink SDK, um conjunto de APIs desenvolvido pela Ford, os desenvolvedores podem estender as funcionalidades da central Ford SYNC integrando-a a aplicativos Android e iOS (FORD MOTOR COMPANY, 2015b). Os desenvolvedores cadastrados no *Ford Developer Program* devem seguir padrões rigorosos de usabilidade e de navegação em seus aplicativos, de modo que, eles possam ser utilizados com segurança ao volante. Com o AppLink também é possível obter diversas informações de diagnóstico, como temperatura externa; nível de combustível; consumo instantâneo; odômetro; pressão dos pneus; situação do cinto de segurança; posição do freio de mão; rotações por minuto; localização GPS; velocidade; além de poder exibir notificações no painel de instrumentos e de utilizar a interface de comandos de voz para interagir com o aplicativo. Essa gama de recursos e possibilidades torna o AppLink bastante atrativo frente aos seus concorrentes, pois a variedade de aplicativos nas lojas virtuais GooglePlay e AppStore pode ser determinante para garantir maior aceitação no mercado de IVI.

Além de possuir uma solução proprietária, a Ford também contribui com a comunidade de software livre. O OpenXC (OPENXC, 2015) consiste em um conjunto de especificações e implementações em hardware e software que tem o objetivo de integrar aplicativos móveis aos dados de diagnóstico do veículo. Apesar de ser uma iniciativa da Ford, o OpenXC foi concebido para que seja compatível com qualquer veículo fabricado a partir de 1996. A plataforma opera sobre um adaptador OBD conectado ao barramento de diagnóstico do automóvel associado a um SDK para Android que implementa todas as particularidades da comunicação via *Bluetooth*, Wi-Fi ou USB com esse dispositivo. Sendo assim, o desenvolvimento de aplicações integradas torna-se mais simples. Por meio de uma API bem definida, o desenvolvedor pode obter diversas informações do automóvel, como ângulo do volante, rotações por minuto, posição do câmbio, estado da ignição, estado do pedal do freio, estado do freio de mão, estado do farol, posição do pedal do acelerador, torque da transmissão, velocidade do veículo, consumo médio e instantâneo, abertura das portas, estado do limpador de pára-brisas, odômetro, nível de combustível, posição geográfica, dentre outras. No entanto, a disponibilidade das informações está sujeita à compatibilidade de alguns modelos.

Tanto o Ford Sync quanto o OpenXC possuem simuladores em software para o teste de aplicações. O ambiente de desenvolvimento é o Eclipse ou o Android Studio do Google.

O formato dos dados que trafegam entre a API do OpenXC e o aplicativo é JSON (ECMA INTERNATIONAL, 2013), e portanto, de fácil interpretação e manipulação. A solução também permite que uma trajetória seja armazenada em JSON e reproduzida posteriormente para fins de testes.

2.4 Considerações Finais

As empresas de tecnologia estão lutando pelo domínio do mercado dos veículos e é evidente a corrida tecnológica que se apresenta em meio a sua consolidação. Com o lançamento do Apple CarPlay e do Android Auto, em 2014, a disputa ficou ainda mais acirrada. Mesmo para a Microsoft, que já estava posicionada desde o ano 1998, o peso da ampla adoção de *smartphones* Apple e Android é uma ameaça preocupante (ARTHUR, 2014). Em breve, alguns fabricantes oferecerão suporte aos dispositivos Apple e Android simultaneamente. Outros permitirão escolher suporte a um ou outro dispositivo na compra do veículo. E, também haverá os que darão suporte a apenas uma marca exclusiva. Isso significa que em breve a escolha de um carro novo poderá estar diretamente relacionada à escolha do melhor *smartphone* e vice-versa.

No contexto das cidades inteligentes, os chamados sistemas de transporte inteligentes, ou ITS (*Inteligent Transportation Systems*), são projetos e iniciativas que aplicam tecnologias avançadas de comunicação de dados, logística, informação, eletrônica e computação para resolver problemas de mobilidade como congestionamentos, eficiência no transporte, segurança e conservação ambiental. Como permeam diferentes áreas de conhecimento, os sistemas de transporte inteligentes possuem grandes desafios para que possam ser amplamente adotados. Todos conhecem o impacto de um acidente de trânsito no tráfego, mesmo que este tenha ocorrido em uma via diferente da qual se está dirigindo. Um ITS poderia utilizar, por exemplo, o reconhecimento de imagens das câmeras de monitoramento de uma cidade para notificar a ocorrência de acidentes de trânsito e, então, sugerir aos condutores rotas alternativas. Outro grande desperdício de recursos é que, enquanto um lado da via está completamente congestionado, o sentido oposto está frequentemente sendo subutilizado. Existem projetos que prevêem a mobilidade da divisória entre as pistas, de acordo com o fluxo de veículos, de modo que se possa flexibilizar a largura de uma ou de outra para melhor atender ao tráfego instantâneo.

A integração entre os sistemas IVI e OBD tem um grande potencial para aplicações em cidades inteligentes, especialmente no que diz respeito a sistemas de transportes inteligentes. Com o uso dessas tecnologias poderiam ser criadas soluções sobre redes de veículos interligados, de modo a reduzir o número de congestionamentos no trânsito, sugerindo rotas alternativas

e de menor consumo. Outra aplicação interessante é a identificação das regiões de uma cidade que carecem da plantação de mais árvores, de modo a anular o efeito da emissão de poluentes, oferecendo melhor qualidade de vida aos moradores. Também é possível fornecer serviços de análise preditiva para a redução de acidentes, por exemplo, avisando o motorista sobre freadas bruscas de veículos à sua frente na estrada, ou sobre o risco associado a uma determinada ultrapassagem (TOYOTA, 2015). No âmbito da economia de combustível, a informação sobre o estado dos semáforos pode ser utilizada para estimar a velocidade média na qual se irá surfar uma onda verde[2] proporcionando um trajeto com menos paradas, congestionamentos e custos. Os dados de diagnóstico também poderiam ser utilizados para que as seguradoras de veículos pudessem identificar os condutores mais prudentes. Com essa informação, seria possível oferecer apólices de seguro mais baratas para esses clientes, em comparação às dos motoristas mais agressivos.

Neste capítulo foi realizada uma análise das tecnologias de diagnóstico a bordo presentes nos veículos de passeio fabricados atualmente. Também foi apresentado um estudo sobre o mercado de IVI e a importância desses sistemas para a economia de combustíveis. O capítulo a seguir propõe uma aplicação para *smartphones* Android que contribua para um sistema de transporte mais inteligente. Por meio dessa solução, pretende-se oferecer ao condutor dados relevantes que o permitam relacionar seu estilo de condução ao consumo de combustível.

[2] A expressão *surfar uma onda verde* se refere à sincronização de semáforos para a obtenção de maior fluidez no trânsito, em geral, este evento ocorre quando o motorista mantém sua velocidade média próxima ao limite estipulado para a via.

3 ESPECIFICAÇÃO DO APLICATIVO FUELMAP

A proposta do presente trabalho é desenvolver e avaliar uma aplicação para celulares com sistema operacional Android que forneça ao usuário informações relacionadas ao seu perfil de condução. O objetivo é permitir a identificação e o aprimoramento do comportamento do condutor que possua relação com o consumo excessivo de combustíveis. Devido à predominância no mercado brasileiro de veículos de passeio, o foco da solução proposta deverá ser sobre aqueles abastecidos com etanol e gasolina. Apesar de serem compatíveis com o padrão OBD-II, veículos movidos à diesel possuem uma metodologia de cálculo diferente para a estimativa do consumo instantâneo de combustível (RIBEIRO, 2013), informação que nem sempre é disponibilizada pelos fabricantes.

O presente capítulo contempla a especificação da solução de software apresentada. Inicialmente serão detalhados os requisitos da aplicação, permitindo uma visão clara de seu escopo. Posteriormente, será mostrado o diagrama de casos de uso, o qual ilustrará as principais funcionalidades acessíveis pelo usuário. Por fim, estarão dispostos os protótipos de interface, os quais fornecerão uma visualização mais próxima do resultado final esperado.

3.1 Descrição Geral do Aplicativo FuelMap

O aplicativo FuelMap se propõe a trazer para o condutor, seja amador ou experiente, informações que poderão proporcionar-lhe um melhor entendimento sobre o seu veículo em relação à economia de combustível. Por meio de um sistema inteligente de notificações em tempo real, seu uso cotidiano permite ao usuário aprimorar sua maneira de dirigir, de modo que, até mesmo condutores distraídos, passem a prestar mais atenção nas pequenas atitudes de condução. Além disso, é possível saber o custo aproximado de cada percurso realizado, permitindo um melhor planejamento financeiro. Com o uso do FuelMap, muitas perguntas passam a ter resposta, como, por exemplo: qual o horário mais barato para sair do trabalho de carro? Qual percurso permite ficar o menor tempo possível parado em engarrafamentos? Quais trechos de um percurso são os mais caros e quais são os mais baratos? Qual a relação entre a velocidade do veículo e o consumo de combustível ? Talvez muitas dessas perguntas nunca tenham sido realizadas por grande parte das pessoas, ou tenham sido classificadas como sem resposta. O aplicativo FuelMap, no entanto, com o auxílio da telemetria, visa responder essas e outras perguntas de forma simples e intuitiva.

3.2 Engenharia de Requisitos

Segundo Sommerville (SOMMERVILLE et al., 2008), a engenharia de requisitos compete a definição dos requisitos funcionais e não funcionais de modo que esses sejam compreensíveis pelos usuários do sistema sem um conhecimento técnico detalhado. Esses requisitos só devem especificar o comportamento externo do sistema e deve evitar, na medida do possível, características de concepção do software propriamente dito. Consequentemente, essa formalização deve utilizar-se de linguagem simples e de diagramas intuitivos. A seguir estão detalhados os requisitos funcionais e não funcionais para o aplicativo FuelMap.

3.2.1 Requisitos Funcionais

Os requisitos funcionais definem o comportamento de um sistema, o que ele deve fazer e os recursos providos por ele. Eventualmente, os requisitos funcionais também podem explicitar o que o sistema não deve fazer. Os requisitos funcionais mapeados para o aplicativo FuelMap são:

- Avisos de Navegação em Tempo Real: Durante a condução, o motorista deverá ser notificado sobre maus hábitos ao volante. Essas notificações terão o propósito de forçar uma mudança de comportamento por meio da repetição de alertas. Os comportamentos mais comuns que causam consumo elevado de combustível são: alta rotação do motor, trajetos e horários com muitas paradas, excesso de tempo com veículo ligado parado (para resfriamento interno, por exemplo), arrancadas e freadas bruscas, aceleradas em trocas de marcha e câmbio em ponto morto quando em movimento. Um subconjunto desses comportamentos pode ser definido de acordo com a disponibilidade de informações da interface de diagnóstico do veículo.
- Histórico de Consumo: Essa funcionalidade terá o objetivo de fornecer ao usuário informações sobre o consumo ao longo do tempo e dos percursos realizados, confrontando com os respectivos hábitos praticados em cada um. Cada trajeto percorrido utilizando o aplicativo, deverá possuir um registro do consumo médio, do número de arrancadas bruscas, do tempo parado, do número de rotações elevadas e do percentual de etanol no tanque.
- Mapa de Consumo: O aplicativo deve permitir a visualização georreferenciada dos percursos realizados. Também deve ser possível de identificar visualmente os pontos de

maior consumo sobre o mapa.

- Análise Gráfica: O usuário deve ser capaz de confrontar graficamente os dados de consumo instantâneo do veículo com os dados de diagnóstico mais pertinentes para o consumo de combustível, como rotações por minuto do motor, velocidade instantânea e posição do pedal do acelerador.
- Configurações: O aplicativo deve permitir a configuração dos parâmetros de operação para uma utilização customizada para cada usuário ou veículo, da forma mais simples e intuitiva possível, sem necessitar de conhecimentos de informática ou de mecânica de automóveis.

3.2.2 Requisitos não Funcionais

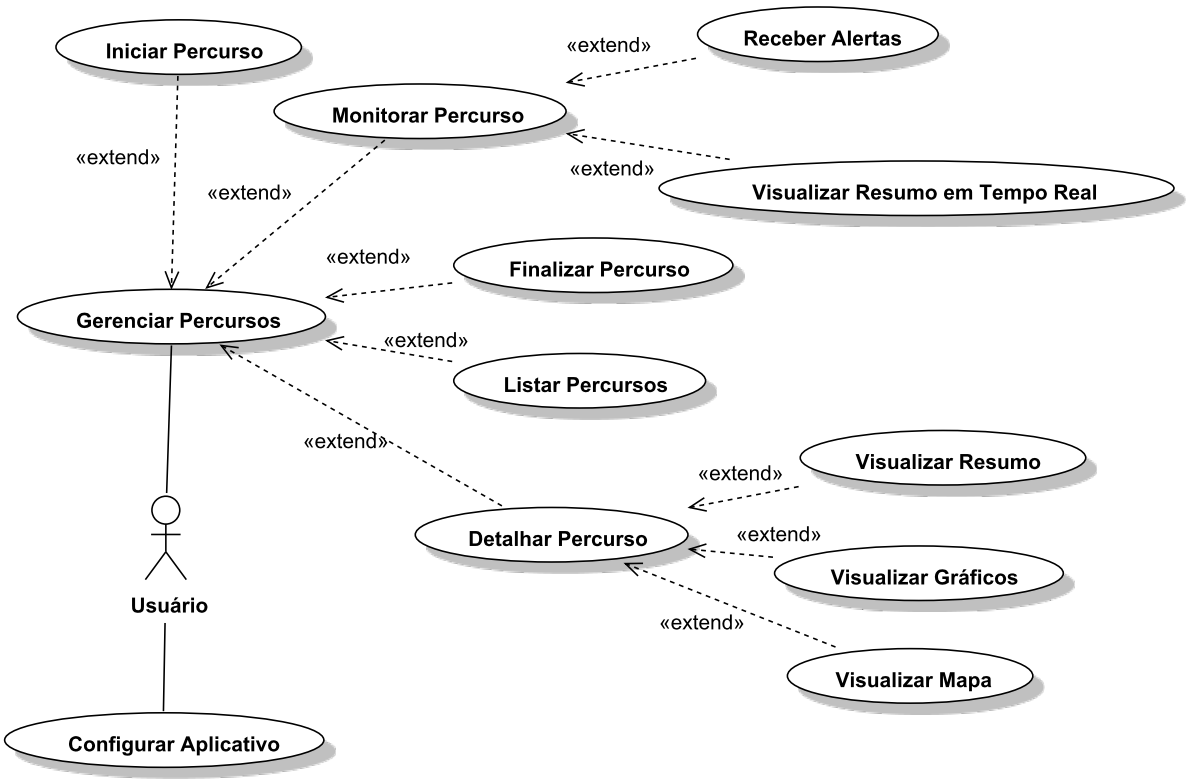
Os requisitos não funcionais são aqueles que dizem respeito às restrições sobre serviços ou funções oferecidos pelo sistema, tais como restrições de tempo, restrições sobre o processo de desenvolvimento, confiabilidade, disponibilidade, ou mesmo sobre padrões de codificação. Esses requisitos, muitas vezes podem ser até mesmo mais importantes que os requisitos funcionais, pois, em alguns casos, se não forem atingidos, o sistema pode se tornar inutilizável. Os requisitos não funcionais mapeados para o aplicativo FuelMap são:

- O aplicativo deve ser desenvolvido empregando soluções de software livre.
- Desconsiderando os *smartphones*, os dispositivos de hardware empregados devem ser de baixo custo.
- O aplicativo deve ser compatível com o sistema operacional Android, versões entre 4.0.4 e 6.0.
- O aplicativo deve ser compatível com adaptadores de hardware ELM 327 - *Bluetooth*.
- O armazenamento de dados deverá ser realizado sobre uma base de dados adequada para aplicações móveis.
- O aplicativo deve poder ser utilizado de maneira *off-line*, ou seja, sem conexão com a Internet.

3.3 Diagrama de Casos de Uso

De acordo com Ian Sommerville (SOMMERVILLE et al., 2008), a transição dos requisitos para o projeto de uma aplicação pode ser realizada por meio da identificação de casos de uso. Os casos de uso podem ser representados por um dos diferentes diagramas previstos na notação UML, os quais permitem ilustrar as ações que podem ser realizadas por agentes, ou atores. Os atores representam as partes interessadas, sejam elas, usuários do sistema ou sistemas externos. Um diagrama de casos de uso documenta o que o sistema faz do ponto de vista de diferentes atores. A Figura 3.1 apresenta o diagrama de casos de uso elaborado sobre os requisitos do aplicativo FuelMap.

Figura 3.1 – Diagrama de casos de uso do aplicativo FuelMap



Fonte: elaborada pelo autor

3.4 Protótipos de Interfaces do Sistema

Segundo a definição de Mcclendon, Regot e Akers (AKERS, 2012), a prototipagem é o processo de construção de um modelo para um sistema. No contexto de soluções de software, os protótipos são empregados para ajudar projetistas a construir um sistema de informação intuitiva e fácil de visualizar para os usuários finais. Por meio da prototipagem, é possível antecipar validações relacionadas à usabilidade e aos aspectos conceituais antes mesmo da etapa de desenvolvimento, economizando custos e atendendo com maior efetividade às expectativas do usuário final. A seguir serão mostrados os protótipos das telas da aplicação FuelMap, os quais representam uma visão inicial sobre o que é esperado. No entanto, a implementação final pode divergir ligeiramente do leiaute a seguir proposto.

Ao abrir o aplicativo, o usuário deverá ser direcionado para a tela de tempo real, onde um menu de opções estará disponível, conforme ilustrado pelo botão (a) na Figura 3.2. Por meio desse menu, o usuário poderá acessar todas as funcionalidades do aplicativo, como a tela de tempo real, o histórico de percursos e as configurações do aplicativo.

Figura 3.2 – Protótipo de Tela Inicial/Menu

Fonte: elaborada pelo autor

A Figura 3.3 mostra a funcionalidade de tempo real do aplicativo. A partir da tela (a) o usuário poderá iniciar um novo percurso, tocando sobre o botão ▶. Após isso, os dados de diagnóstico do veículo passarão a ser monitorados e o usuário receberá alertas visuais e sonoros a cada comportamento seu que contribua para um alto consumo de combustível, conforme tela (b). Ao término do trajeto, o usuário poderá clicar sobre o botão ■ e, então, associar a ele uma descrição textual para facilitar posteriores buscas, conforme ilustrado na tela (c).

Figura 3.3 – Protótipo de Tela de Tempo Real - Iniciar/Monitorar/Finalizar Percurso

Fonte: elaborada pelo autor

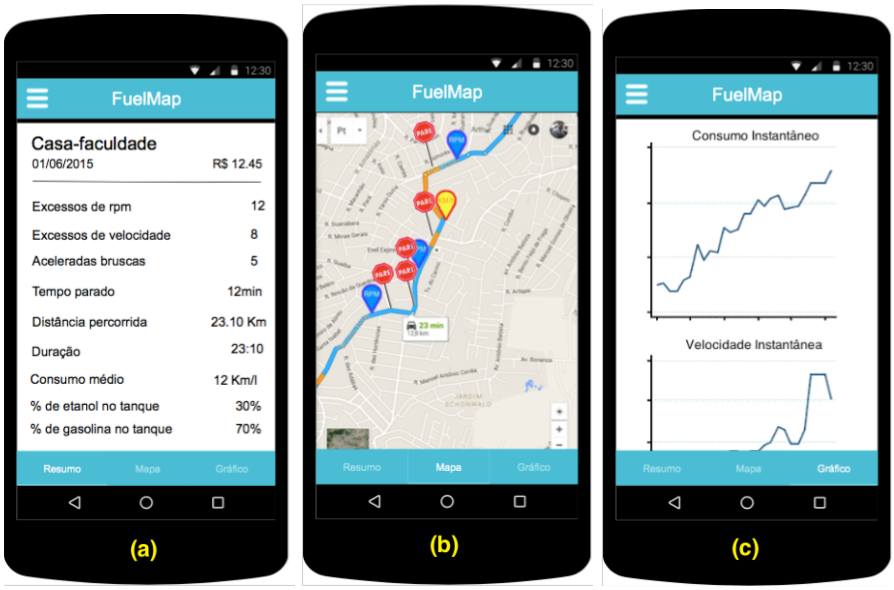
Ao acessar a opção de menu Percursos, será aberta a tela de histórico de percursos, conforme Figura 3.4, a qual permitirá listar todos os trajetos realizados pelo motorista. A lista de registros deverá exibir a data, a descrição textual e o valor total gasto em combustível, além disso, deverá ser possível excluir ou exibir detalhes dos percursos da lista.

Figura 3.4 – Protótipo de Tela de Listagem de Percursos

Fonte: elaborada pelo autor

Ao selecionar um item da lista, será aberta a tela de detalhamento de um percurso específico realizado pelo usuário, conforme Figura 3.5. Essa tela será composta por três abas, que podem ser identificadas, no protótipo, por um menu de opções na região inferior da tela. A aba (a) diz respeito ao resumo do trajeto, nela será possível visualizar o nome do percurso, sua data de realização, o custo total, e informações acerca do comportamento do motorista, como número de excessos de RPM, número de excessos de velocidade, número de aceleradas bruscas, entre outras. Na aba (b), é possível visualizar o percurso em um mapa e identificar, por meio de cores diferenciadas, os trechos de maior consumo de combustível. O mapa mostrará também, os pontos em que o usuário parou o veículo ou excedeu os limites de RPM e de velocidade. A aba (c) permitirá a comparação visual, através de gráficos, do consumo instantâneo de combustível com a velocidade, as rotações por minuto do motor, e a posição do pedal do acelerador.

Figura 3.5 – Protótipo de Tela de Detalhe de Percurso - Resumo/Mapa/Gráficos



Fonte: elaborada pelo autor

A Figura 3.6 mostra a tela de configurações, por meio da qual, o usuário poderá alterar os parâmetros do aplicativo FuelMap, como limite de RPM e de velocidade para receber uma notificação, o nome do adaptador OBD e a motorização do veículo.

Figura 3.6 – Protótipo de Tela de Configurações

Fonte: elaborada pelo autor

Por fim, a Figura 3.7 fornece uma visão geral do fluxo de navegação esperado para o aplicativo. A estruturação de telas de um software em um fluxo de navegação também pode ser chamada de *storyboard* e é uma técnica útil para evitar mal entendidos durante a etapa de concepção das aplicações. Por meio desse mecanismo, é possível especificar o comportamento do software a cada interação do usuário.

Figura 3.7 – Protótipo do Fluxo de Navegação

Fonte: elaborada pelo autor

3.5 Considerações Finais

Neste capítulo apresentou-se a especificação do aplicativo FuelMap e seus requisitos, bem como, os protótipos da interface visual que deverão nortear o desenvolvimento. O próximo capítulo mostrará a implementação do software, as principais decisões de projeto e as tecnologias adotadas, incluindo o dispositivo de hardware utilizado.

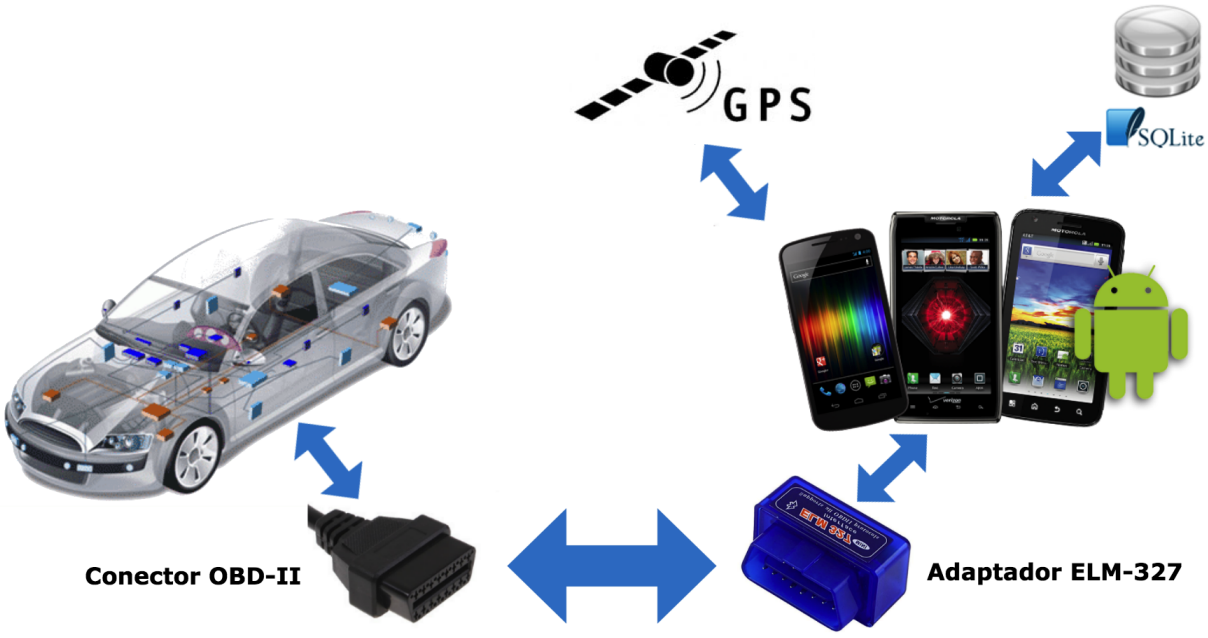
4 IMPLEMENTAÇÃO

Este capítulo apresenta o processo de desenvolvimento da aplicação descrita no capítulo anterior, trazendo detalhes sobre as tecnologias empregadas. Faz parte do processo de desenvolvimento, a execução de testes isolados utilizando simuladores de sistemas OBD, a decisão acerca das tecnologias utilizadas, bem como o estudo sobre a plataforma Android.

4.1 Arquitetura do sistema

A Figura 4.1 ilustra a arquitetura base da solução, que consiste em um veículo com suporte ao padrão OBD-II, um adaptador *Bluetooth* ELM-327, e celulares com sistema operacional Android entre as versões 4.0.4 e 6.0.0. O aplicativo FuelMap estará instalado no dispositivo Android e realizará a comunicação com a ECU do veículo por meio da interface *Bluetooth*. Por meio dessa interface, a aplicação trocará comandos, no protocolo ELM-327, com o adaptador ELM-327, que por sua vez, será responsável por traduzir esses comandos para os respectivos protocolos de enlace e de nível físico do padrão OBD, de acordo com o suporte oferecido pelo modelo do veículo que estiver sendo utilizado.

Figura 4.1 – Arquitetura da solução FuelMap



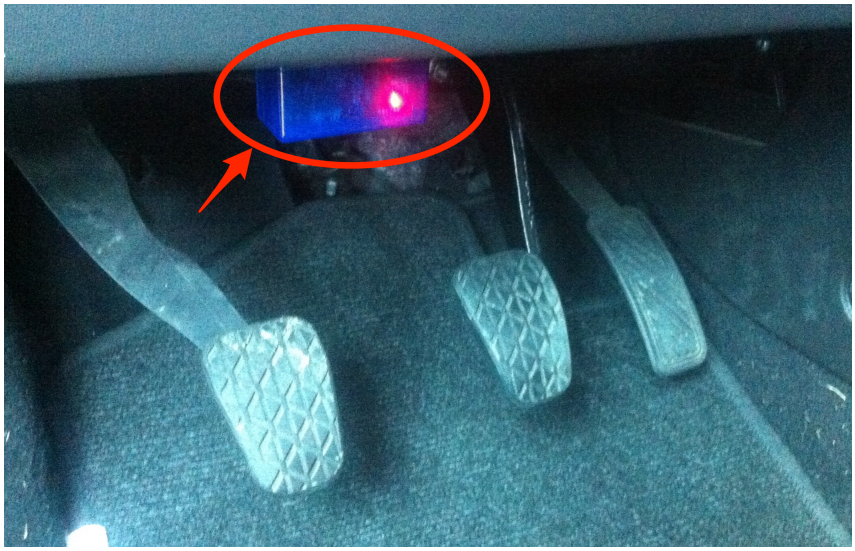
Fonte: elaborada pelo autor

4.2 Hardware empregado

Entre as alternativas tecnológicas estudadas, o OpenXC (seção 2.3.7) apresentou os recursos mais consolidados para desenvolvimento e teste de aplicações veiculares para *smartphones*. Além disso, o OpenXC é completamente *Open Source*, desde o Hardware até a biblioteca de comunicação com celulares Android. Em contrapartida, os adaptadores de hardware OpenXC disponíveis no mercado não implementam o protocolo ELM-327 e custam mais caro que outros leitores OBD comuns, os quais permitem obter um conjunto menor de informações. A Ford oferece dispositivos OpenXC fabricados por ela própria, por aproximadamente $139,00, o que atualmente representa cerca de R$ 490,00, sem considerar frete e impostos.

Devido ao custo da solução OpenXC, o presente trabalho optou pelo uso de adaptadores OBD mais comuns, especificamente, aqueles que implementam o protocolo ELM-327. O adaptador ELM-327 Mini é um dos mais baratos do mercado e pode ser encontrado por cerca de R$ 30,00, já considerando importação e impostos. Assim como seus concorrentes, o ELM-327 Mini suporta todos os protocolos de nível físico previstos no padrão OBD, permite conexão *Bluetooth* com dispositivos Android e identifica automaticamente o protocolo suportado pelo veículo. O veículo utilizado para os testes durante a implementação do aplicativo FuelMap possui os pinos 04, 05, 06, 14 e 16 no conector padrão SAE J1962, portanto, atende ao protocolo ISO15765-4 CAN. A Figura 4.2 ilustra o aspecto visual de um adaptador ELM-327 Mini e sua localização quando conectado ao veículo.

Figura 4.2 – Adaptador ELM-327 Mini conectado ao veículo



Fonte: elaborada pelo autor

No que diz respeito aos aparelhos celulares, foram utilizados durante o desenvolvimento os seguintes modelos: Motorola Moto E, segunda geração, com 1 GB de memória RAM, processador Quad Core com 1,2 GHz, e armazenamento interno de 8 GB; e um *smartphone* Samsung Galaxy S3 Mini, com 512 MB de memória RAM, processador Dual Core 1 GHz, armazenamento interno de 8 GB e Android 4.1.2. O aplicativo foi desenvolvido exclusivamente para a plataforma Android. Ambos os dispositivos contam com sistema de localização GPS e conectividade *Bluetooth*. Não foram utilizados celulares com sistema operacional iOS, pois, segundo Laukkonen (LAUKKONEN, 2015), o adaptador ELM 327 Mini *Bluetooth* não é compatível com esse sistema operacional.

4.3 Software empregado

O aplicativo FuelMap foi desenvolvido para a plataforma Android, portanto, a linguagem de programação utilizada foi o Java sobre a IDE de desenvolvimento Android Studio (GOOGLE, 2016). O Android Studio é uma suíte de desenvolvimento criada pelo Google para substituir a ferramenta até então utilizada, o Eclipse (ECLIPSE, 2016). Com esse avanço, o Google trouxe ganhos de produtividade para os desenvolvedores, dentre eles, pode-se citar: uma plataforma mais leve, pois não necessita de inúmeras estensões que sobrecarreguem o processamento, integração nativa com ferramentas de controle de versão como o Git (CHACON; STRAUB, 2014), gerenciamento de dependências automatizado com a integração à ferramenta Maven (SONATYPE, 2016), automatização do processo de compilação por meio da ferramenta Gradle (GRADLE, 2016), além da integração a um poderoso, e já consagrado, mecanismo de auto completar código, de análise estática e de refatoração da plataforma InteliJ (JETBRAINS, 2016). O Android Studio também facilita a criação das interfaces visuais para aplicativos móveis, algo que, até então apresentava diversos problemas sobre a IDE Eclipse.

Apesar de não ser popularmente conhecida pelas pessoas comuns, a comunicação com o adaptador ELM-327 é um tema bem definido e suficientemente testado pela comunidade de desenvolvimento. O padrão OBD especifica, para cada código de parâmetro (PID), um conjunto de cálculos matemáticos sobre os valores binários lidos da ECU do veículo, a partir dos quais se pode obter a informação desejada. Por conta disso, atualmente, existe um vasto conjunto de bibliotecas *OpenSource*, em diferentes linguagens de programação, que abstraem as particularidades de baixo nível, permitindo que o desenvolvedor dê maior enfoque à lógica de sua aplicação. O presente trabalho utilizou uma biblioteca em Java, denominada OBD Java API (PIRES, 2013), a qual recebe uma referência para um *socket Bluetooth* e possui algumas classes facilitadoras para o envio e recebimento de mensagens do protocolo ELM-327.

Um recurso bastante utilizado em aplicativos analíticos é a visualização de informações. Um dos requisitos do aplicativo FuelMap é a comparação, por meio de gráficos, entre o comportamento do motorista e o consumo instantâneo do veículo, para isso, foi utilizada uma biblioteca gráfica em linguagem Java para Android, chamada *HelloCharts for Android* (WACH, 2013). Apesar de possuir inúmeros tipos de gráficos, foi utilizado apenas o gráfico de linha. Trata-se de uma excelente biblioteca para Android e que ativamente recebe atualizações pelo criador. O aplicativo FuelMap também utiliza o componente de mapas nativo do Android, criado e mantido pela empresa Google. Com o uso desse recurso torna-se mais simples o desenvolvimento do requisito "Mapa de Consumo"(Seção 3.2.1). Também é utilizada a API de geolocalização do Android, *LocationManager*, que permite obter as coordenadas geográficas do usuário e, então, associar trechos de um trajeto a valores de consumo instantâneo para que essa informação seja exibida em um mapa.

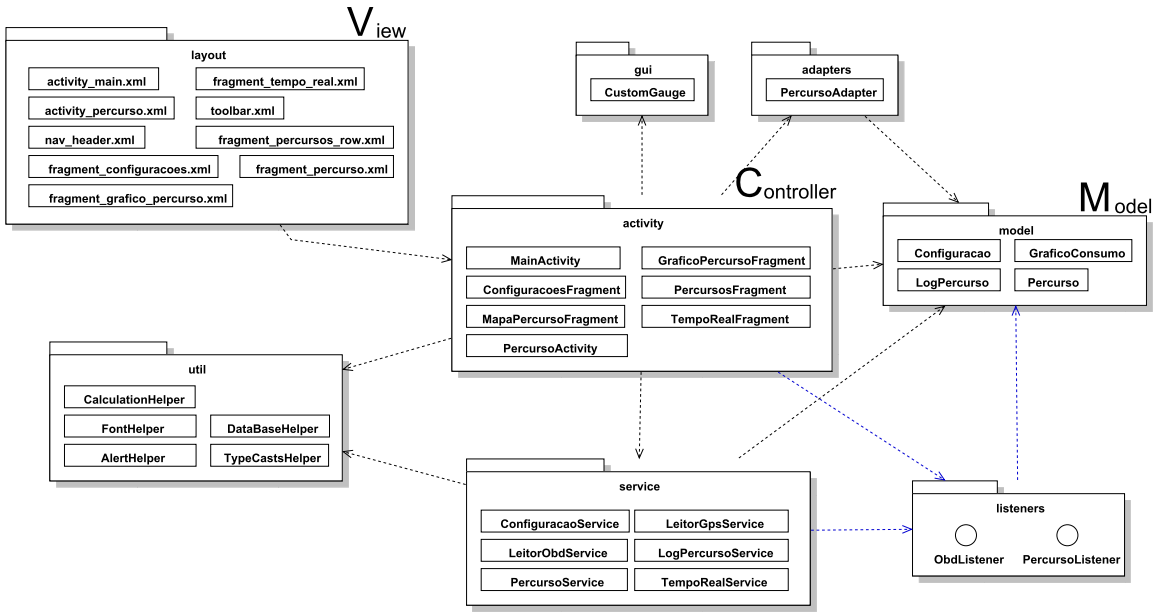
Além das bibliotecas e ferramentas utilizadas no desenvolvimento propriamente dito, se fez necessária a utilização de mecanismos de versionamento de código. Para o desenvolvimento do presente protótipo, foi utilizada a ferramenta Git, que consiste em um sistema de controle de versões distribuído, por meio do qual, é possível gerenciar a evolução do código fonte das aplicações de forma mais simples e robusta. Diferente dos sistemas de controle de versão centralizados, com essa ferramenta, os programadores não precisam bloquear arquivos para edição exclusiva, mas trabalham com a edição de cópias locais e com o conceito de mesclagem de alterações. Além disso, uma das vantagens desse tipo de solução é o controle de ramificações, ou *branches*, as quais permitem que o programador siga trabalhando sobre um código fonte diferente sem interferir na versão principal. Com esses mecanismos, é possível isolar alterações e paralelizar o desenvolvimento, de modo que, seja possível mesclar ramificações a medida que for necessário, de forma leve e facilitada.

4.4 Diagrama de Classes

A Figura 4.3 representa o diagrama de componentes da aplicação desenvolvida, assim como, suas respectivas classes, em um alto nível de abstração. Foram aplicados princípios do padrão de projetos MVC (*Model View Controller*) o qual prevê a separação da camada de apresentação, do modelo de dados e da lógica de negócio. Na aplicação em questão, a camada de apresentação corresponde aos arquivos XML que definem a interface visual em aplicações Android e que estão localizados no diretório de leiautes do projeto, portanto, não estão representados no diagrama de classes. O modelo de dados foi definido no pacote *model* e corresponde às

entidades que o aplicativo FuelMap manipula e transmite como parâmetro entre suas diferentes camadas. Por último, os controladores estão definidos no pacote *activity*, o qual se responsabiliza pelo tratamento de eventos de usuário, como cliques em botões, seleção de registros, montagem de listas, entre outros.

Figura 4.3 – Diagrama de Classes e Componentes da Aplicação FuelMap



Fonte: elaborada pelo autor

No pacote *activity* existem dois tipos distintos de classes, as que herdam da classe base *Activity* e as que herdam da classe *Fragment*. A principal diferença entre ambas é que as classes que herdam de *Activity* são tratadas como uma tela autônoma, ou seja, são carregadas por completo sobrescrevendo ou sobrepondo a tela atual do aplicativo. Já as classes que herdam de *Fragment* tem a possibilidade de serem carregadas em regiões específicas de uma determinada *Activity*, portanto, são componentes menores com leiaute próprio e que podem ser reaproveitados em diferentes situações. Além disso, o ciclo de vida de ambas as classes são diferentes, as *Activities*, por exemplo, não possuem o evento *onAttach* que é chamado toda vez que um *Fragment* é associado a uma *Activity*. Para a implementação de telas de listagem de registros, se fez necessária a criação de um pacote auxiliar, chamado *adapters*, que realiza o mapeamento entre as entidades do modelo de dados e a visualização das células de cada registro em uma lista permitindo a aplicação de estilos e cores personalizados.

Segundo Fowler (FOWLER, 2002), o padrão de projetos *Service Layer*, ou Camada de Serviço, define limites de uma aplicação e seu conjunto de operações disponíveis para camadas clientes. Esse padrão encapsula a lógica de negócios do software, controlando as operações e suas respostas de modo a desacoplar as regras de negócio do serviço oferecido e a aplicação que o consome. No caso do aplicativo FuelMap, estão presentes diversas fontes de dados, como leitores OBD, dispositivos GPS e também bancos de dados locais. De modo a isolar a lógica referente à obtenção e tratamento dessas informações, é possível aplicar o conceito de camada de serviços. Considera-se, portanto, que os controladores da aplicação são clientes de uma camada de serviços, a qual encapsulará a implementação de operações de negócio. No diagrama de componentes mostrado na Figura 4.3, a referida camada de negócios está representada pelo pacote *service*.

O pacote *model* se refere ao modelo de dados da aplicação e contém as classes correspondentes a cada entidade que faz parte do domínio do problema. Por meio dessa camada, é possível aplicar o padrão de implementação *Parameter Object*, que, segundo Beck (BECK, 2008), se propõe a reduzir a quantidade de parâmetros enviados para os métodos substituindo esses por objetos com atributos. Esse padrão ajuda a manter o código limpo, com linhas mais curtas e, portanto, facilita sua compreensão.

Além da estrutura-base MVC, associada a uma camada de serviços, o aplicativo FuelMap agrega dois pacotes adicionais que tem a finalidade de encapsular funções e componentes reutilizáveis. A camada *gui* foi criada com o objetivo de agrupar controles de interface gráfica customizados e que possam ser facilmente reutilizados, por exemplo, os relógios digitais da tela de tempo real do aplicativo. Existe também uma camada de métodos utilitários, *util*, que agrupa classes para conversão entre tipos de dados, cálculos sobre listas de registros, formatação de fontes, exibição de alertas e controle de transações de banco de dados.

Por fim, a camada *listeners* refere-se a uma decisão de projeto para evitar o acoplamento entre os controladores e a camada de serviços do aplicativo. A camada de serviços é responsável pela leitura em tempo real das informações de diagnóstico do adaptador OBD, e essa atividade é realizada por uma tarefa em segundo plano. Dessa forma, na funcionalidade de tempo real, o serviço de leitura de dados necessita informar para o controlador da existência de novas informações para exibição. Assume-se, no entanto, que não é desejável que a camada de serviços referencie explicitamente uma instância de um controlador para realizar essa ação, pois isso criaria um acoplamento desnecessário entre as classes. A solução encontrada para esse probema foi a definição de interfaces que possuem métodos para receberem notificações de mudanças nos dados de diagnóstico. Os controladores que necessitam de atualização em tempo real, portanto,

implementam essas interfaces e sua referência é enviada para os serviços de leitura de dados para que os notifiquem constantemente. Com esse mecanismo, é possível modificar completamente a lógica dos controladores sem afetar a atualização em tempo real realizada pela camada de serviço, desde que os controladores continuem implementando a interface necessária.

4.5 Armazenamento

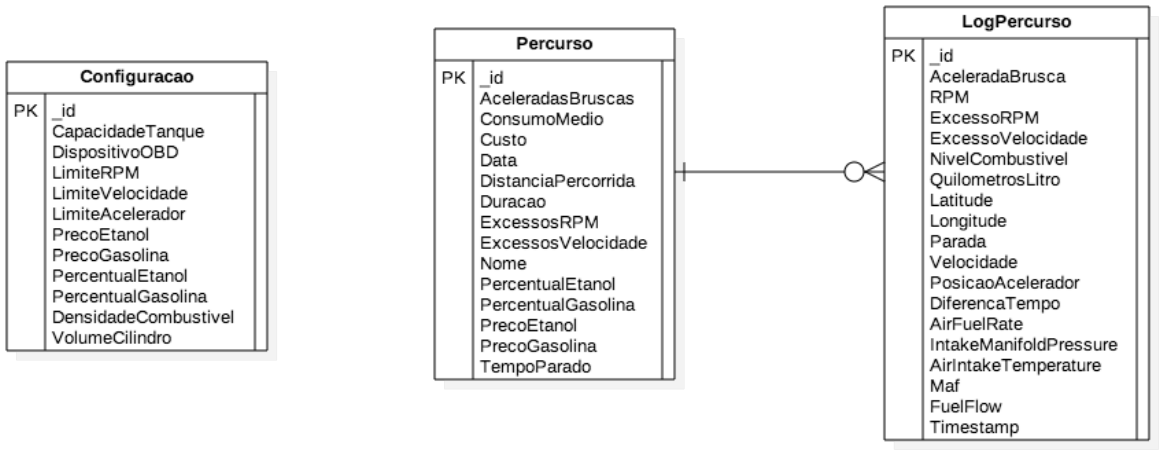
Para o armazenamento local de informações, a aplicação FuelMap utiliza banco de dados SQLite, uma versão simplificada de bancos de dados SQL com o objetivo principal de ser executada em dispositivos com baixo poder de processamento. Esse tipo de banco de dados é largamente utilizado em aplicações para *smartphones*, pois possui boa compatibilidade, é simples e permite o controle de transações, sendo limitado a apenas uma conexão de escrita aberta por vez e sem limitações para transações de leitura.

O código genérico para a manipulação do banco de dados está disposto na classe *DataBaseHelper*, a qual estende a classe padrão da biblioteca do Android, *SQLiteOpenHelper*. Com essa classe é possível criar bancos de dados, abrir e fechar transações, executar comandos SQL para DDL (Linguagem de Definição de Dados) ou DML (Linguagem de Manipulação de Dados). A estrutura do banco de dados seguiu o modelo Entidade-Relacionamento da Figura 4.4, o qual foi projetado durante a etapa de análise de modo a atender a todas as funcionalidades do aplicativo. A base de dados necessitou de três tabelas, uma tabela para o registro dos *logs* de tempo real, uma para o armazenamento do resumo dos percursos realizados e outra para registrar as configurações do aplicativo. A Figura 4.4 representa o modelo Entidade-Relacionamento que descreve a base de dados atual do aplicativo FuelMap, onde as chaves primárias estão identificadas pela anotação *PK*.

4.6 Dificuldades encontradas

Nesta seção serão apresentadas as principais dificuldades enfrentadas durante o desenvolvimento do protótipo da aplicação FuelMap, desde sua concepção até os testes em campo. Além disso, detalha as soluções encontradas para tais problemas e as contribuições de outros trabalhos de pesquisa que estiveram associadas a elas.

Figura 4.4 – Modelo Entidade-Relacionamento da Aplicação FuelMap



Fonte: elaborada pelo autor

Como citado anteriormente, uma das grandes dificuldades em desenvolver aplicações que necessitem de informações de diagnóstico e que sejam compatíveis com diferentes veículos é a disponibilidade de parâmetros de diagnóstico. Apesar de o padrão OBD descrever um conjunto de PIDs, e a forma correta de calcular seu valor a partir de um conjunto de dados binários, não há obrigação legal de que esses parâmetros sejam disponibilizados pelos fabricantes de veículos. Além disso, não existe uma fonte de informação que relacione de forma centralizada todos os modelos de veículos e seus respectivos parâmetros de diagnóstico. Dessa forma, a estratégia que muitas empresas tem adotado para obter tal base de dados é a realização de testes em veículos reais. A página (OUTILS OBD FACILE, 2010) possui uma relação de marcas e modelos de veículos, construída pelos próprios usuários do referido software de diagnóstico, que possibilita identificar a compatibilidade de cerca de 9000 veículos com os respectivos PIDs do padrão OBD. Mesmo assim, essas tabelas não são exaustivas, o que torna difícil a adoção pelo cliente final, dado que, eventualmente, não possui meios de saber se seu veículo é compatível antes de adquirir um software ou dispositivo de hardware.

Para tornar viável a implementação do aplicativo FuelMap, seria necessário consultar os seguintes parâmetros OBD: velocidade, rotações por minuto, posição do pedal do acelerador e consumo, instantâneos, ou seja, referentes ao modo de operação 0x01 que contém informações de tempo real. Haviam disponíveis dois modelos de veículos para a realização dos testes: um Peugeot 207 XS 1.6 ano/modelo 2009 e um Ford Ka+ 1.0 ano/modelo 2015. Mesmo com uma lista relativamente reduzida de parâmetros, nenhum dos veículos possuía o parâmetro de PID 5E, *Engine Fuel Rate*. Como se trata de uma informação essencial para grande parte das funci-

onalidades propostas para o aplicativo, foi necessário investir em pesquisas e testes diretamente nos veículos, consumindo quantidades consideráveis de tempo e de combustível. Felizmente, uma dissertação de mestrado publicada pela Universidade Federal do Porto, realizada pelo estudante Vitor Daniel Ferreira da Cunha Ribeiro (RIBEIRO, 2013), trouxe uma contribuição importante para auxiliar na resolução do problema. A dissertação propõe um modelo de estimativa do consumo instantâneo de combustível para motores baseados no ciclo de Otto, popularmente conhecidos como motores de 4 tempos (SCHULZ, 2009).

A estratégia adotada na dissertação foi a dedução de diferentes equações a partir da lei dos gases ideais e de algumas informações específicas do veículo que fornecessem uma estimativa para o fluxo de combustível, em litros por hora. A partir dessas informações se pode derivar o consumo instantâneo, em quilômetros por litro. A Equação 4.1 é o ponto de partida e representa o fluxo de ar entrando nos pistões em gramas por segundo. Em alguns veículos essa medida pode ser obtida diretamente do leitor OBD, pelo modo de operação e PID 0x01 0x10, respectivamente, no entanto, essa informação não estava disponível em nenhum dos veículos testados neste trabalho.

$$MAF = \frac{(MAP * VE) * ED}{R * (IAT + 273.15)} * MMair * (RPM/60)/2 \tag{4.1}$$

Onde:

MAF = Fluxo de ar em g/s

MAP = Pressão absoluta do ar de admissão em kPa

VE = Eficiência volumétrica em %

ED = Volume somado dos cilindros do veículo em litros

R = Constante ideal dos gases em J/K*mol

IAT = Temperatura do ar de admissão em graus Celsius

MM_{air} = Massa molar do ar em gramas/mol

RPM = Frequência de rotação do motor em rotações por minuto

O objetivo da obtenção do fluxo de ar entrando no motor é a existência de uma relação entre a quantidade de ar e a quantidade de combustível que é injetada nos pistões, portanto, essa informação está diretamente associada com o consumo instantâneo de combustível. Foi necessário, entretanto, obter novos parâmetros de diagnóstico para a aplicação da equação em questão, são eles: *Intake Manifold Pressure* (PID 0x0B) e *Air Intake Temperature* (PID 0x0F). O volume total dos cilindros foi parametrizado e corresponde à motorização do veículo, por exemplo: um veículo com 4 cilindros, de motor 1.6 possui um volume total de 1.6 litros, o

equivalente a 0.4 litros por pistão. Por fim, a eficiência volumétrica é um parâmetro que indica quão próximo do fluxo volumétrico teórico do motor está de fato sendo utilizado em média. Segundo Ribeiro (RIBEIRO, 2013), a eficiência volumétrica representa a proporção entre o volume de mistura de ar e combustível no interior do cilindro em relação ao volume máximo do cilindro. Devido à sua difícil aferição, que depende de diversos fatores, esse parâmetro pode ser considerado aproximadamente 75% do volume total do cilindro.

A Equação 4.1 pode ser utilizada para diferentes modelos de veículos, mesmo que possuam motorizações distintas. Isso se deve ao fato de que a equação considera a proporção entre ar e combustível, o volume total dos pistões e o número de rotações por minuto. A cada rotação completa de um motor baseado no ciclo de Otto, metade dos pistões realiza a etapa de admissão, ou seja, absorve ar e combustível, e a outra metade realiza a etapa de exaustão, ou seja, expele os gases produzidos pela combustão. O mesmo vale para casos excepcionais, como os veículos de 3 cilindros, os quais admitem, em média, 1,5 vezes do volume total do motor, a cada ciclo. A proporção entre ar e combustível obtida na etapa de admissão e o número de rotações por minuto são parâmetros que podem ser obtidos da ECU do veículo por meio dos PIDs 0x44 e 0x0C, respectivamente. Dessa forma, basta que o usuário configure, no aplicativo, o volume total do motor para que o cálculo seja adaptado ao seu veículo.

De posse do fluxo da admissão de ar é possível obter o fluxo de combustível com a Equação 4.2. Para que seja possível sua utilização, foi necessário obter um novo parâmetro de diagnóstico: *Air Fuel Ratio* (PID 0x44). Além disso, foi parametrizada no aplicativo a densidade do combustível presente no tanque que, para gasolina, está entre 0.720 kg/l e 0.775 kg/l e para o etanol, um pouco mais denso, é de aproximadamente 0.789 Kg/l.

$$FF = \frac{MAF * 3600}{AFR_A * FD} \tag{4.2}$$

Onde:

FF = Fluxo de combustível (l/h)

AFR_A = Razão ar/combustível (número de vezes)

FD = Densidade do combustível (g/l)

Uma vez que tenhamos o fluxo de combustível, em litros por hora, torna-se mais fácil calcular o rendimento instantâneo, em Km/l, bastando apenas, dividir a velocidade instantânea pelo valor obtido, conforme a Equação 4.3.

$$Rendimento_{(Km/l)} = \frac{V_{Km/h}}{FF_{l/h}} \tag{4.3}$$

Outra dificuldade encontrada diz respeito às limitações impostas pelo banco de dados SQLite. Por ser um banco de dados minimalista, ele se restringe a apenas uma conexão de escrita aberta por vez, o que dificulta sua manipulação em um ambiente *multithread*. Para solucionar esse problema, foi criada uma classe utilitária, *DataBaseHelper*, que com o uso de monitores Java, compartilha uma única conexão ao banco de dados com as diferentes *threads* da aplicação, gerenciando a exclusão mútua de seus métodos. Outro ponto a destacar é o conjunto limitado de tipos suportados, que não contempla campos de data e booleanos, dessa forma, datas foram armazenadas como *strings* e booleanos foram armazenados como inteiros sendo necessário implementar conversões de tipos na leitura e na escrita das entidades armazenadas.

Além dos aspectos mencionados, a realização de testes funcionais também apresentou dificuldades, pois demandaria estar com o veículo ligado e conectado ao dispositivo celular frequentemente. Esse modelo de trabalho possui um custo, além de poluir o meio ambiente, pois consome combustível desnecessariamente. Para auxiliar nesse processo, foi utilizado um simulador de dados de diagnóstico, o OBDSIM (ICCULUS, 2011), que permite reproduzir o trajeto de um veículo a partir da interface *Bluetooth* de um computador pessoal. Essencialmente, o OBDSIM disponibiliza a mesma interface de um adaptador ELM-327, e suporta os PIDs necessários para o funcionamento do aplicativo FuelMap. Dessa forma, foi possível obter ganhos em produtividade no desenvolvimento e na identificação de falhas de software, além disso, permitiu a reprodução de trajetos específicos que exercitassem cenários de teste selecionados pelo desenvolvedor. Como o objetivo de sua utilização foi apenas de validar detalhes da comunicação utilizando o protocolo ELM-327, não foi necessário avaliar a confiabilidade dos dados produzidos pelo simulador.

4.7 Considerações finais

Neste capítulo apresentou-se o desenvolvimento do aplicativo FuelMap, bem como, os principais artefatos que documentam essa etapa. Também foram mostradas as principais dificuldades encontradas e as contribuições de outros trabalhos de pesquisa que serviram de referência para superá-las. O próximo capítulo consiste na avaliação dos resultados obtidos com a aplicação FuelMap.

5 AVALIAÇÃO

Este capítulo tem por objetivo fazer uma avaliação do trabalho realizado. Inicialmente será apresentada a metodologia adotada, na sequência o funcionamento do aplicativo será validado a partir de alguns cenários previamente estabelecidos. A seguir, será avaliado o consumo de recursos do dispositivo celular. Por fim, será feita uma comparação entre os protótipos de tela inicialmente propostos e o resultado final obtido.

5.1 Plataforma Experimental

Os experimentos foram realizados utilizando um veículo Ford Ka+ 1.0 2015/2015 que possui um conector SAE J1962 e que disponibiliza os seguintes códigos de parâmetros: *Throttle Position* (PID 0x11), *Fuel Level* (PID 0x2F), *RPM* (PID 0x0C), *Speed* (PID 0x0D), *Air Fuel Ratio* (PID 0x44), *Intake Manifold Pressure* (PID 0x0B) e *Air Intake Temperature* (PID 0x0F). Além disso, empregou-se um adaptador ELM 327 Mini com interface *Bluetooth*, o qual conecta o dispositivo celular ao barramento do veículo. Por fim, o aplicativo FuelMap foi instalado em um celular Samsung Galaxy S3 Mini, com sistema operacional Android, GPS e *Bluetooth*. No início dos testes do aplicativo FuelMap, foi utilizado um veículo Peugeot 207 Passion XS 1.6 2009/2009 para validar a compatibilidade da solução. Exceto o PID 0x2F, que indica o nível de combustível presente no tanque, todos os demais parâmetros estavam disponíveis nesse veículo, o que indica que eles são relativamente comuns, mesmo em automóveis populares.

5.2 Metodologia

O primeiro passo realizado foi a obtenção de uma base de dados heterogênea, isso é, com diferentes percursos, em diferentes horários e com perfis de tráfego e de condução distintos. As informações coletadas referem-se a um único modelo de veículo, Ford Ka+ 1.0 2015 e um único modelo de celular, Samsung Galaxy S3 Mini. O objetivo dessa escolha foi fixar a plataforma experimental para que fosse possível avaliar apenas o impacto do comportamento do motorista e das condições do trânsito para a economia de combustível, sem variações causadas por veículos e celulares com características diferentes. Ao todo, foram obtidos 50 trajetos, em um período de 40 dias, totalizando mais de 600.000 leituras de diagnóstico, o equivalente a 28 horas de utilização do aplicativo. Sobre essa base foram realizadas consultas em linguagem SQL e os resultados foram analisados com a ferramenta Microsoft Excel.

Para a avaliação funcional do aplicativo foram definidos três cenários experimentais. O primeiro tem o objetivo de avaliar a operação do sistema de alertas em tempo real, de modo que, eles possam ser percebidos pelo condutor em tempo hábil de tomar uma atitude para melhorar

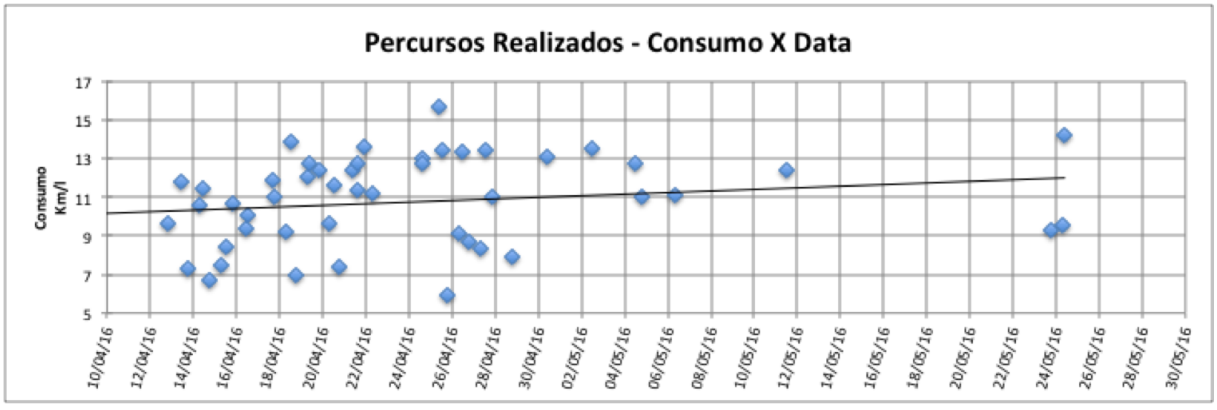
seu comportamento. O segundo tem a finalidade de avaliar se o horário de partida de um determinado percurso influencia no valor gasto com combustíveis devido às flutuações do tráfego. O terceiro, e último cenário, pretende identificar a relação entre os comportamentos do motorista e o consumo do veículo. Algumas análises sobre os dados foram realizadas com o objetivo de comprovar as seguintes hipóteses: (1) o custo de um trajeto está associado ao seu horário de início; (2) um mesmo trajeto consome mais combustível em dias de maior tráfego, pois o veículo permanece parado por mais tempo nesses casos; (3) um trajeto com muitos excessos de RPM e com muitas aceleradas bruscas possui maior custo por quilômetro, pois esses comportamentos aumentam o fluxo de combustível nos pistões.

Além dos testes funcionais, também foi avaliada a aderência da aplicação final aos protótipos definidos inicialmente. Por fim, foi realizada uma análise do uso de recursos do celular, como armazenamento, memória RAM e bateria, os quais são muito importantes, uma vez que, o condutor não deseja que sua bateria acabe logo no início do dia e que ele precisa compartilhar os recursos de processamento e armazenamento do celular com outras aplicações de seu interesse.

5.3 Testes funcionais

Nesta seção são descritos os testes de funcionalidade da exibição de alertas durante a navegação e a análise dos resultados obtidos a partir da base de dados de avaliação, confrontando-os com as hipóteses inicialmente levantadas. A Figura 5.1 mostra a relação de todos os percursos realizados durante os testes, associando-os ao respectivo rendimento do veículo, em quilômetros por litro. O eixo horizontal corresponde à data de realização do trajeto e o eixo vertical mostra o consumo médio praticado.

Figura 5.1 – Gráfico de consumo dos percursos de teste realizados



Fonte: elaborada pelo autor

A linha diagonal que cruza o gráfico foi obtida por meio de uma regressão linear sobre as amostras contempladas. Essa linha, também chamada de *trendline*, indica a tendência associada ao rendimento do veículo ao longo do tempo. Nota-se que, o ponto inicial dessa linha está em, aproximadamente, 10 Km/l, momento em que foram iniciados os testes e em que o condutor ainda apresentava seu estilo de condução habitual. Em 24/05/2016, data do último teste realizado, a reta de tendência está em, aproximadamente 12 Km/l, ou seja, um aumento médio estimado de 20% no rendimento de combustível. No entanto, o índice de correlação calculado entre o número de utilizações do aplicativo e o rendimento do veículo foi de 0,209, o que, de acordo com Portnoi's (PORTNOI'S, 2006), indica que existe uma relação muito fraca entre essas variáveis. Para uma análise estatística mais efetiva, seria necessário obter mais amostras e, eventualmente, avaliar os resuldados de aproximações não lineares.

5.3.1 Cenário A: Verificação dos Alertas

O objetivo do Cenário A é verificar se os alertas de excesso de RPM, velocidade e aceleradas bruscas estão sendo acionados corretamente pelo aplicativo FuelMap. Para isso, o condutor precisou reproduzir maus hábitos, como pisar fundo no acelerador, elevar a frequência de rotação do motor acima do limite configurado e exceder a velocidade máxima definida. O aplicativo foi configurado para alertar o motorista sempre que esse excedesse 3.000 RPM, ou 60 Km/h, ou se o pedal do acelerador fosse pressionado acima de 60% de sua posição máxima.

O aplicativo FuelMap registrou a maior parte dos eventos em que os limites estabelecidos foram violados. No entanto, devido à velocidade de leitura do adaptador OBD, eventualmente, alguns eventos de curta duração podem ser perdidos. Isso ocorre porque a comunicação é realizada de forma serial, e portanto, um PID deve ser lido sequencialmente após o outro. Foi observado que a leitura completa de todos os parâmetros necessários ocorre, tipicamente, em 1 segundo. Se, por exemplo, o condutor exceder 60 Km/h logo após a leitura da velocidade atual, ele precisa permanecer acima do limite até que ocorra uma nova leitura, a qual, poderá levar cerca de 1 segundo. Por outro lado, quando o condutor excede o limite de velocidade imediatamente antes da realização de uma leitura, a exibição do alerta é praticamente imediata. Em todos os casos um sinal sonoro foi emitido simultaneamente à exibição do alerta visual, dessa forma, o condutor pode manter-se olhando para a estrada durante o trajeto e, ainda assim, observar seu estilo de condução. A Figura 5.2 mostra as respectivas telas do aplicativo FuelMap que reproduzem os alertas de navegação. Conclui-se que, no pior caso, é possível capturar todos os eventos com duração maior que 1 segundo. Eventos mais rápidos que isso podem ser perdidos, o que não é desejável, mas não impede a utilização do aplicativo.

Figura 5.2 – Tela de Alertas em Tempo Real - Aplicativo FuelMap

Fonte: elaborada pelo autor

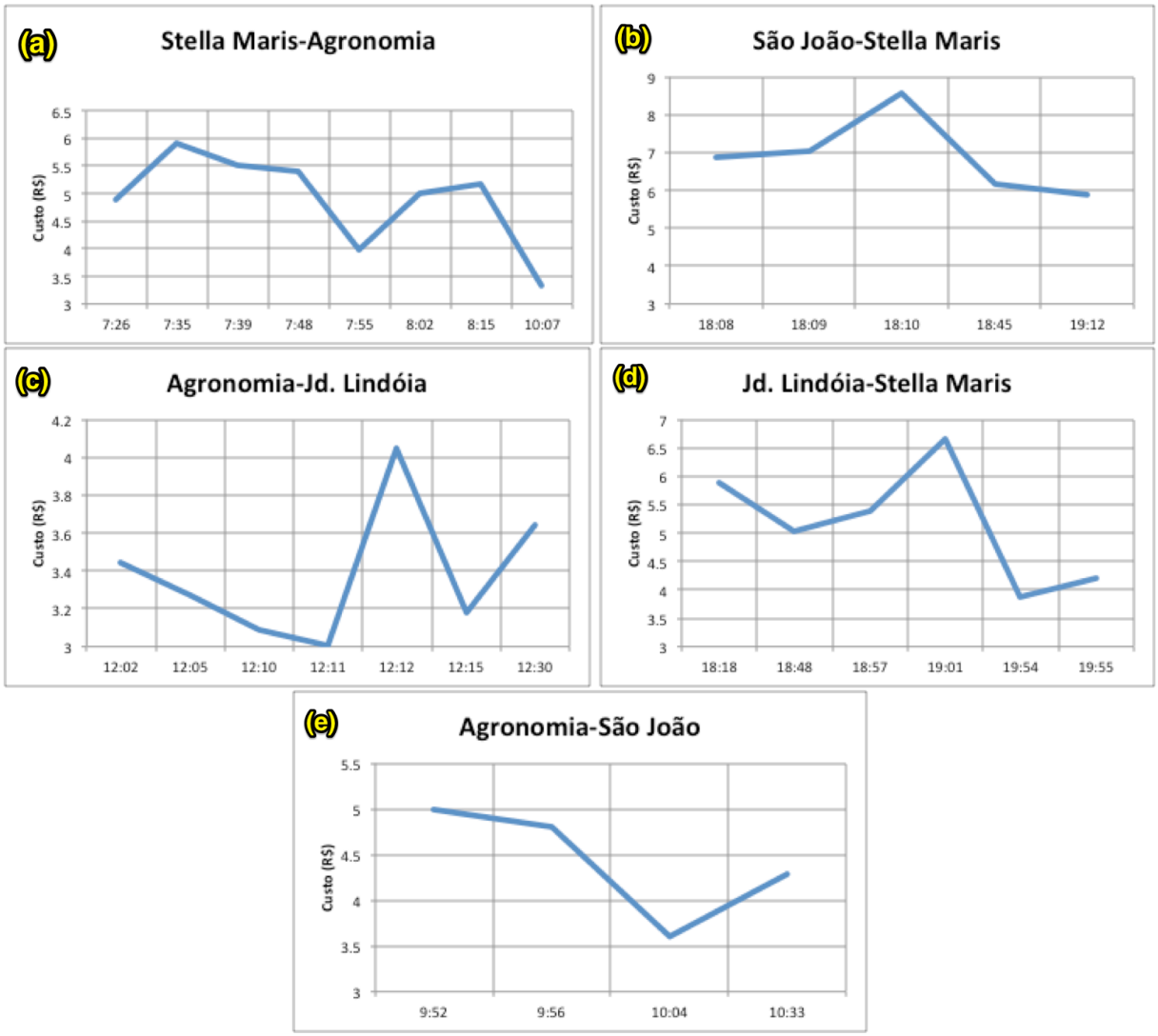
5.3.2 Cenário B: mesmo trajeto, horários diferentes

O Cenário B tem como objetivo mostrar que um mesmo trajeto pode apresentar um custo diferente, dependendo do horário em que é realizado, devido às variações no tráfego. Para isso, foram avaliadas algumas combinações específicas de origem e destino, que tiveram pelo menos duas execuções pelo usuário. Com essa metodologia, foi possível identificar os horários de início que proporcionaram menor custo, conforme mostrado nos gráficos da Figura 5.3.

Com base nos gráficos da Figura 5.3, foi possível identificar pontos de custo mínimo de cada percurso e, então, elaborar a Tabela 5.1. Nessa tabela, é possível identificar a melhor alternativa para o horário de início de cada percurso. As colunas origem e destino identificam cada um dos trajetos; as colunas distância média, rendimento médio e custo médio foram obtidas com base em uma média de valores de todas as realizações de cada percurso; a coluna melhor horário corresponde aos pontos de custo mínimo nos respectivos gráficos da Figura 5.3.

Em alguns casos, como, por exemplo, o trajeto Jd. Lindóia-Stella Maris, Figura 5.3 d, a diferença de custo entre o melhor e o pior horário chegou a R$ 2,79, um valor relativamente pequeno se for observado um único dia. No entanto, se um percurso como esse for realizado ao longo de 30 dias, a economia pode passar de R$ 80,00 mensais, o que seria suficiente para pagar o adaptador OBD ELM-327, necessário para a utilização do aplicativo FuelMap, em menos de um mês.

Figura 5.3 – Gráficos de Custo X Horários



Fonte: elaborada pelo autor

Tabela 5.1 – Tabela de resultados cenário de testes B: mesmo trajeto, horários diferentes

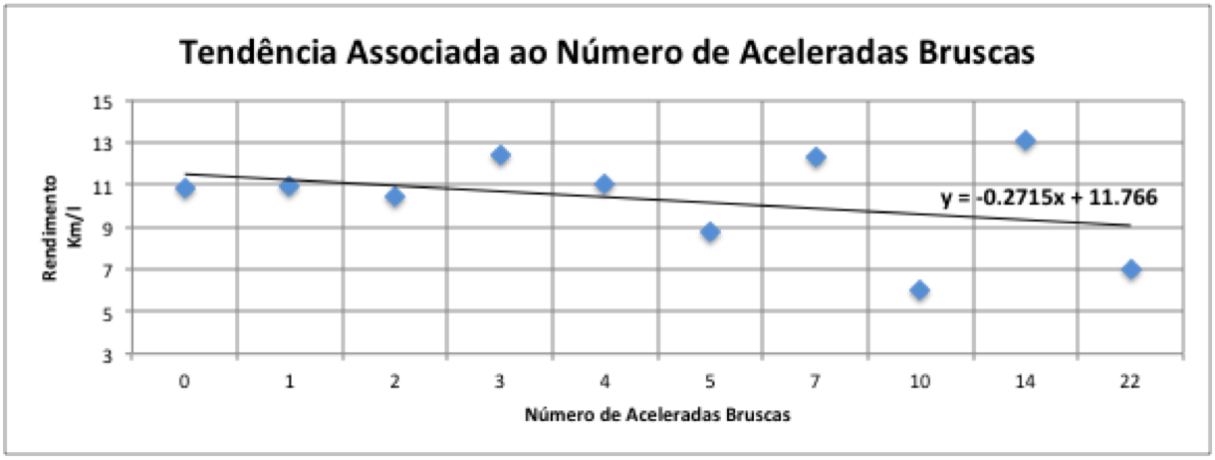
Origem	**Destino**	**Distância Média (Km)**	**Rendimento Médio (Km/l)**	**Custo Médio (R$)**	**Melhor Horário**
Stella Maris	Agronomia	12,46	10,04	4,90	10:07
São João	Stella Maris	15,77	8,45	6,91	19:12
Agronomia	Jd. Lindóia	12,18	12,98	3,38	12:11
Jd. Lindóia	Stella Maris	10,99	8,21	5,17	19:54
Agronomia	São João	16,42	12,93	4,42	10:04

Fonte: elaborada pelo autor

5.3.3 Cenário C: relação entre custo e comportamento

O objetivo do Cenário C é evidenciar a relação entre o perfil de condução do motorista e o consumo do veículo. Para isso, foram realizadas consultas sobre toda a relação de percursos contida no banco de dados de testes. Essas consultas visam comparar a média de rendimento, em Km/l, com o número de excessos de RPM, com o número de aceleradas bruscas, com o número de excessos de velocidade e com o tempo parado durante os percursos realizados. A Figura 5.4 mostra graficamente uma linha de tendência de variação do rendimendo do veículo, em relação ao número de aceleradas bruscas dos trajetos. Foi considerada uma acelerada brusca qualquer leitura de diagnóstico em que a posição do pedal do acelerador estivesse acima de 60% de sua posição máxima.

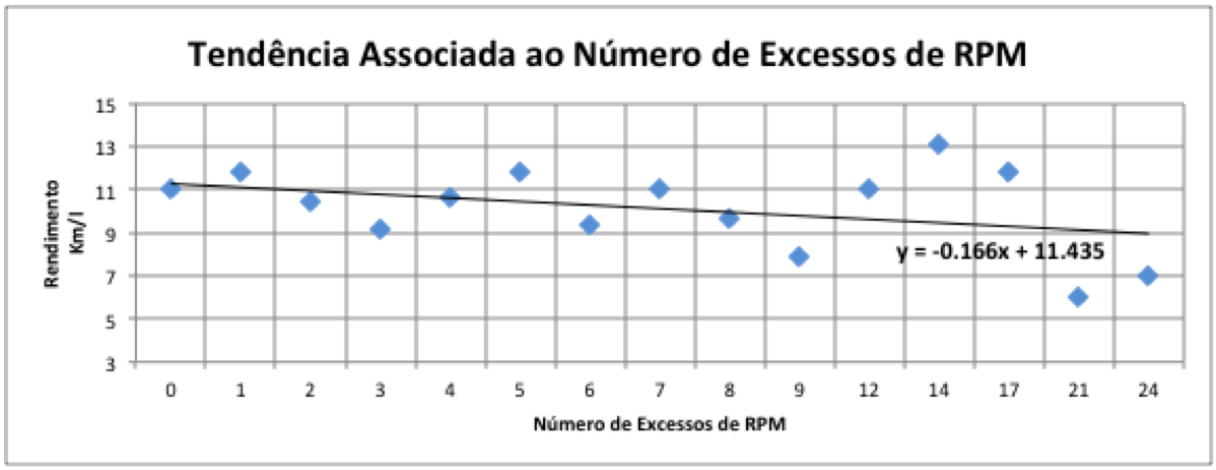
Figura 5.4 – Gráfico de Número de Aceleradas Bruscas X Rendimento Médio (Km/l)



Fonte: elaborada pelo autor

Apesar de existirem flutuações no gráfico, é notável que a linha de tendência possui inclinação negativa, indicando que quanto maior o número de aceleradas bruscas, menor o rendimento do veículo. O índice de correlação observado entre as variáveis foi de -0,41, o que indica que existe uma relação inversa entre elas, mas há uma incerteza associada aos dados, a qual não pode ser desprezada. O mesmo procedimento foi realizado para a avaliação do número de excessos de RPM, conforme mostrado na Figura 5.5. Foram considerados excessos de RPM valores acima de 3.000 rotações por minuto.

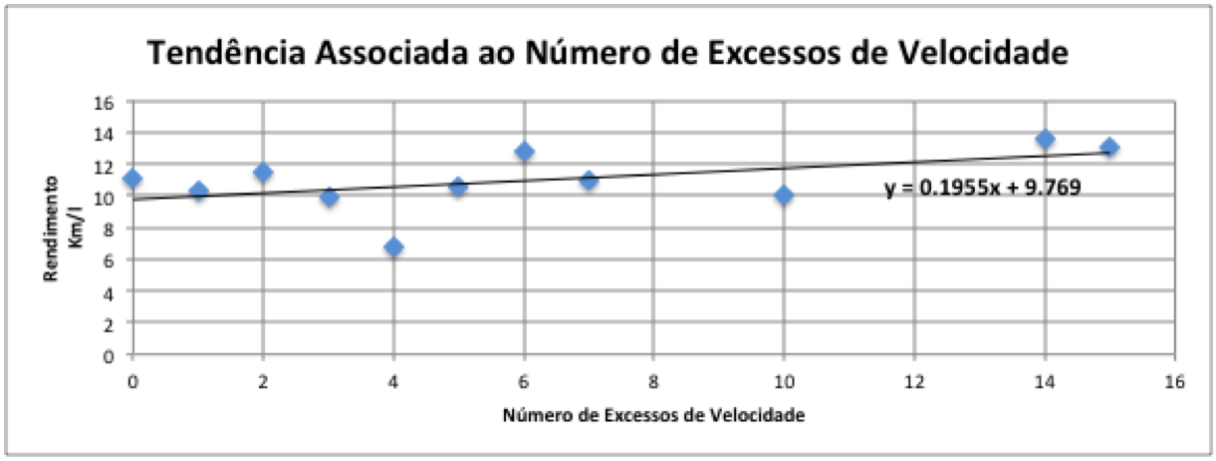
Figura 5.5 – Gráfico de Número de Excessos de RPM X Rendimento Médio (Km/l)



Fonte: elaborada pelo autor

Assim como o número de aceleradas bruscas, a elevação de RPM se mostrou responsável por uma redução no rendimento do veículo, com índice de correlação de -0.456, favorecendo a hipótese de que um trajeto com muitos excessos de RPM possui maior custo por quilômetro. A Figura 5.6 ilustra o mesmo procedimento, dessa vez, sob a perspectiva do número de excessos de velocidade. Foi considerado um excesso de velocidade, qualquer leitura de diagnóstico com valor superior a 60 Km/h.

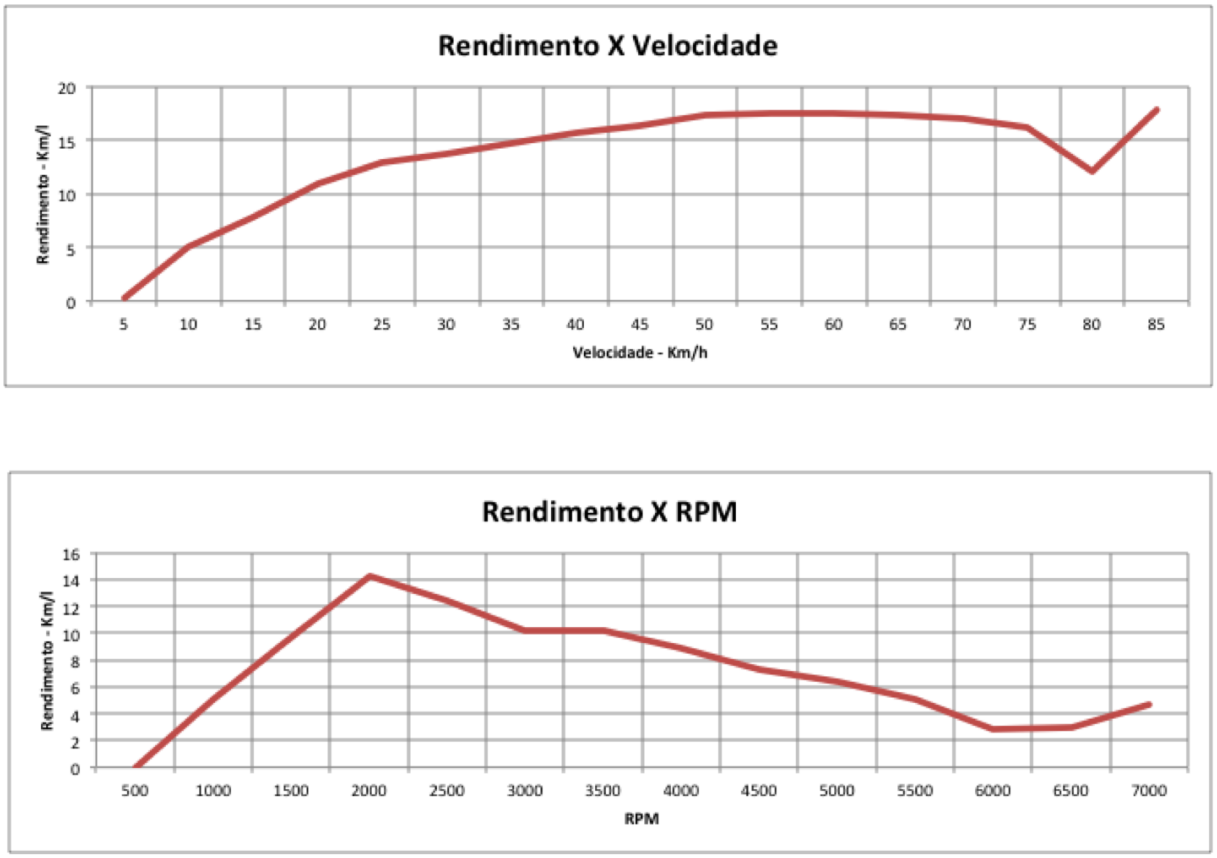
Figura 5.6 – Gráfico de Número de Excessos de Velocidade X Rendimento Médio (Km/l)



Fonte: elaborada pelo autor

Diferentemente dos casos anteriores, velocidades mais altas se mostraram favoráveis à economia de combustível. As variáveis mostradas no gráfico possuem índice de correlação positivo, de 0,521, o que se justifica, devido ao fato de a velocidade estar diretamente relacionada ao rendimento na Equação 4.3. No entanto, vale ressaltar que não basta aumentar cada vez mais a velocidade para obter melhores resultados, pois, o consumo de combustível depende de outras variáveis, como RPM, posição do pedal do acelerador e atrito com o ar. Além disso, o que contribui para a economia de combustível é a manutenção de velocidades relativamente constantes, ou seja, sem variações bruscas. Em complemento a análise baseada na quantidade de excessos cometidos pelo condutor, foram realizadas consultas para comparar o valor absoluto de RPM e de velocidade com o consumo de combustíveis, a Figura 5.7 mostra essa comparação.

Figura 5.7 – Gráfico de RPM e Velocidade X Rendimento Médio (Km/l)



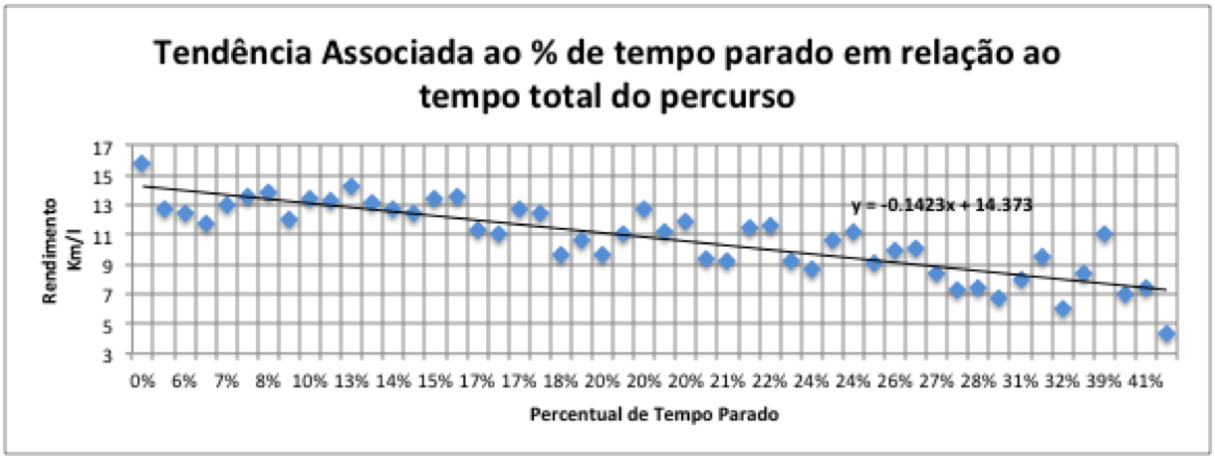
Fonte: elaborada pelo autor

Em relação à velocidade, percebe-se a existência de dois pontos de rendimento máximo, um aos 55 Km/h, equivalente a 17,49 Km/l, e outro aos 85 Km/h, com média de 17,75 Km/l. Conclui-se que, para o veículo testado, a velocidade de 55 Km/h é uma velocidade adequada

para um bom rendimento em circuitos urbanos nas cidades brasileiras, onde o limite típico de velocidade é de 60 Km/h. Os dados coletados não foram suficientes para avaliar velocidades superiores a 85 Km/h. Quanto aos valores de rotações por minuto, nota-se que, existe um único ponto ótimo, em torno de 2.000 RPM, onde o rendimento médio corresponde a 14,21 Km/l. A partir desse valor limite, o rendimento do veículo cai consideravelmente, o que está consistente com a hipótese de que percursos com mais excessos de RPM consomem mais combustível.

Pela mesma razão que os excessos de velocidade contribuíram para um aumento no rendimento, a velocidade zero contribuiu para o gasto excessivo de combustível, pois, nesse caso, o veículo permanece parado com o motor consumindo. Para validar se o aplicativo reage de forma consistente com esse comportamento, a quantidade de tempo parado, em valor absoluto, não pode ser diretamente utilizada, pois, favorece percursos mais curtos. Dessa forma, foi utilizada a proporção de tempo parado em relação à duração total de cada percurso, permitindo uma avaliação mais justa. A Figura 5.8 avalia a relação entre o percentual de tempo parado de cada percurso e o rendimento médio do veículo.

Figura 5.8 – Gráfico de % de Tempo parado X Rendimento Médio (Km/l)



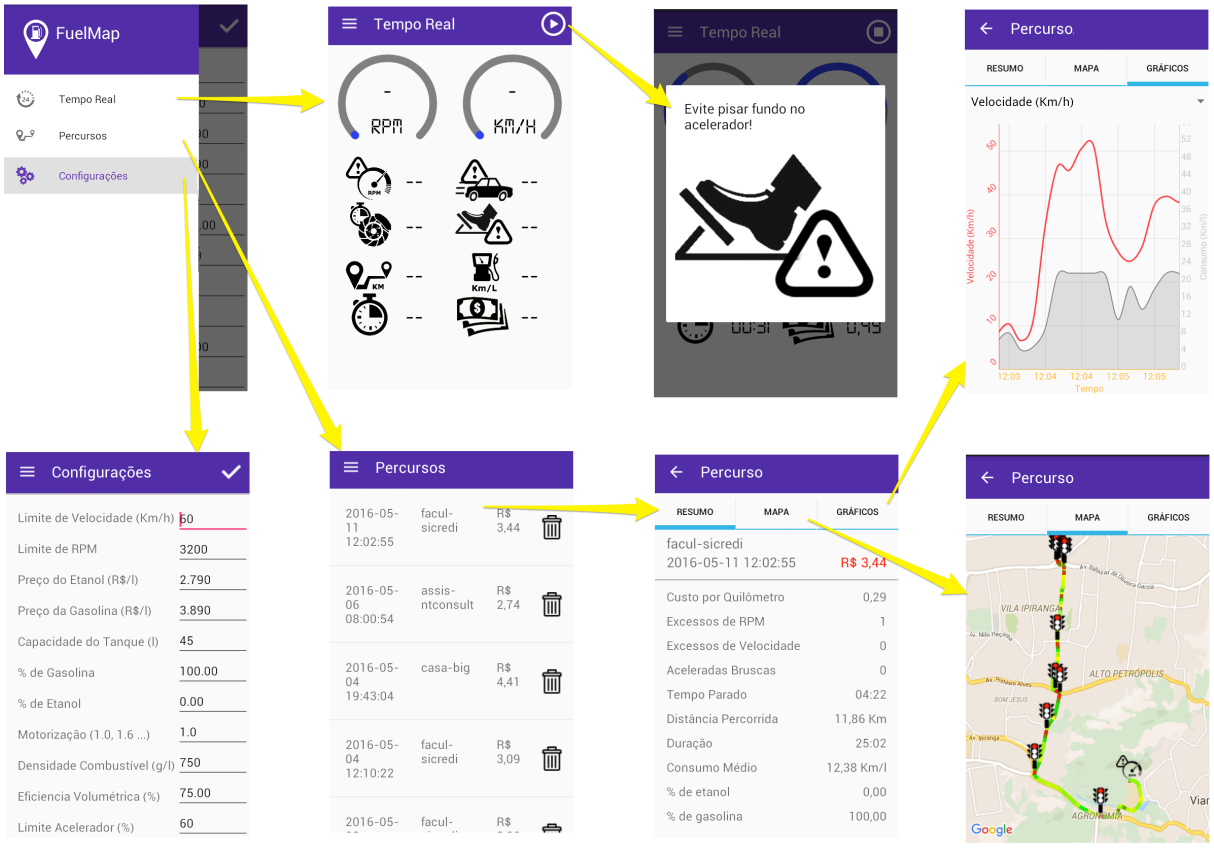
Fonte: elaborada pelo autor

O índice de correlação entre as variáveis indicadas na Figura 5.8 foi de -0,817. Dentre os aspectos analisados no presente trabalho, o tempo parado foi o que mostrou maior confiabilidade em relação à sua contribuição para o consumo excessivo de combustíveis. Isso ficou evidente tanto na frequência de abastecimento do veículo quanto na análise dos dados, o que mostra a importância da identificação de rotas e horários alternativos para se deslocar de um ponto a outro, evitando congestionamentos e horários de pico.

5.4 Teste de Funcionalidade da Interface

Esta seção tem o objetivo de avaliar o fluxo de navegação do aplicativo FuelMap, permitindo ter uma visão geral das telas existentes em comparação com a prototipagem realizada na etapa de projeto (Seção 3.4). Conforme ilustrado na Figura 5.9, o fluxo de telas se aproxima muito dos protótipos inicialmente propostos, no entanto, pequenos detalhes visuais foram modificados para uma melhor apresentação no dispositivo celular. Ao abrir o aplicativo o usuário é direcionado para a tela de Tempo Real, onde existe um menu lateral de opções que permite o acesso às diferentes funcionalidades do aplicativo, como as de Tempo Real, Percursos e Configurações. Na tela de tempo real, o usuário pode iniciar um novo percurso, consultar o resumo da viagem e receber notificações em tempo real durante a condução. Ao finalizar o trajeto, esse fica disponível na lista de percursos. É possível excluir um item da lista por meio do botão , ou detalhar um percurso específico, selecionando-o com um toque. Ao visualizar os detalhes de um desses percursos, o usuário pode acessar o mapa ou os gráficos comparativos por meio de diferentes abas.

Figura 5.9 – Fluxo de Telas - Aplicativo FuelMap



Fonte: elaborada pelo autor

A análise gráfica possui um menu *drop down* com três opções distintas para seleção: Velocidade (Km/h), Rotações por Minuto (RPM) e Posição do Acelerador (%). Conforme Figura 5.10, essas opções permitem comparar os dados de telemetria com o consumo instantâneo do veículo, conforme sugerido nos requisitos funcionais do aplicativo (Seção 3.2.1).

Figura 5.10 – Tela de Exibição de Gráficos - Aplicativo FuelMap

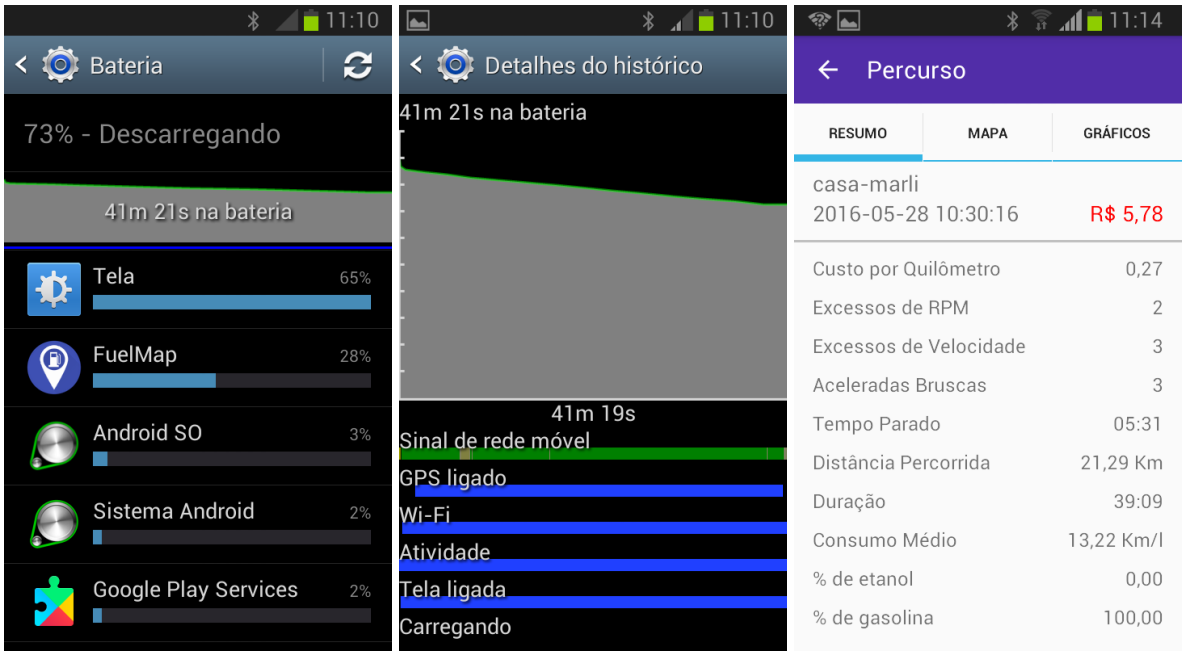
Fonte: elaborada pelo autor

As telas implementadas atenderam a todos os casos de uso definidos na etapa de planejamento. No entanto, não foi realizada uma pesquisa de opinião para avaliar a usabilidade do aplicativo. O motivo para a não realização dessa análise é que os usuários precisariam ter um dispositivo ELM-327 para testar o aplicativo, o que não foi possível.

5.5 Avaliação do consumo de recursos do celular

Nesta seção descreve-se o consumo de recursos do dispositivo celular pelo aplicativo FuelMap. O primeiro ponto a ser observado é o uso de bateria, para isso, foi utilizado o dispositivo celular Samsung Galaxy S3 Mini, cuja bateria foi completamente carregada antes da realização dos testes. O aplicativo foi utilizado em um percurso de 39 minutos, sem estar conectado a uma fonte de alimentação. Ao final do percurso, observou-se uma redução de 27% no nível de carga do celular, sendo que, 28% dessa redução foi ocasionada pelo aplicativo FuelMap. Para evitar o descarregamento da bateria, o usuário pode optar por manter o celular conectado à fonte de alimentação do veículo enquanto estiver utilizando o aplicativo. A Figura 5.11 mostra a tela de consulta do uso da bateria, que pode ser acessada por meio do sistema operacional Android, de onde foram extraídas as estatísticas citadas.

Figura 5.11 – Consumo de bateria - Aplicativo FuelMap



Fonte: elaborada pelo autor

A capacidade da bateria de um dispositivo celular pode ser mensurada utilizando a unidade Watt-hora (Wh), na qual 1 Wh corresponde à quantidade de energia utilizada para alimentar uma carga com potência de um Watt pelo período de uma hora (WIKIPEDIA, 2015). O dispositivo celular testado possui uma bateria de capacidade de 5,7 Wh, portanto, energia total

consumida pelo aplicativo foi de 27% * 28 % * 5,7 Wh, ou 0,43 Wh. A partir da quantidade de energia, é possível calcular a potência dissipada, de acordo com a Equação 5.1.

$$P_{watt} = \frac{E}{t} \tag{5.1}$$

A potência (P_{watt}) corresponde à taxa de energia consumida por unidade de tempo. Na Equação 5.1, a energia está representada pela variável *E* e o tempo está representado pela variável *t*. Substituindo os valores conhecidos de energia, 0,43 Wh, e de tempo, 39 minutos, ajustando as unidades de medida, é possível calcular a potência dissipada pelo uso do aplicativo, conforme Equação 5.2.

$$P_{watt} = \frac{0,43Wh}{39min * \frac{1h}{60min}} = 0,663W \tag{5.2}$$

Para fins de comparação, o mesmo procedimento foi realizado com o aplicativo Waze, que também tem o objetivo de ser utilizado em veículos. Devido à indisponibilidade do mesmo dispositivo celular utilizado no teste do aplicativo FuelMap, o Waze foi testado em um *smartphone* Motorola Moto G3, cuja capacidade da bateria é de 12,597 Wh. Durante um trajeto de 37 minutos, a bateria do celular Motorola Moto G3 teve uma redução de 17 %, sendo que, 4 % foram ocasionados pelo uso do aplicativo Waze. Dessa forma, seu consumo de energia correspondeu a 17 % * 4 % * 12,597 Wh, ou 0,0857 Wh. Substituindo os valores de energia consumida e de duração do percurso na Equação 5.1, temos:

$$P_{watt} = \frac{0,0857Wh}{37min * \frac{1h}{60min}} = 0,139W \tag{5.3}$$

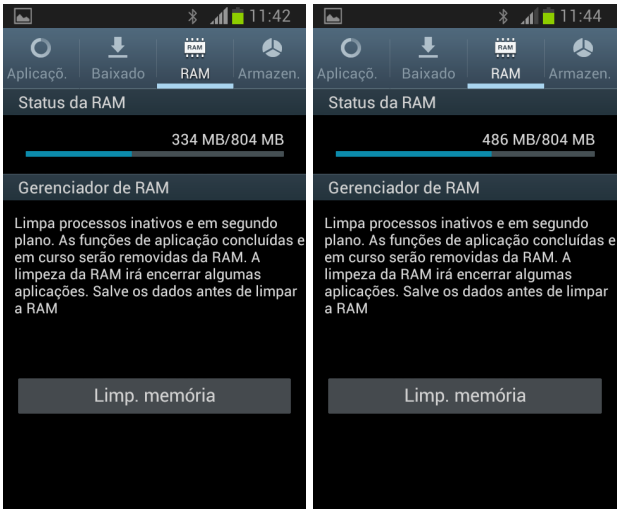
Conclui-se que, o aplicativo FuelMap, consome 4,77 vezes mais bateria que o aplicativo Waze, o que pode ser considerado acima do esperado. Um indício para esse resultado é que a taxa de amostragem dos parâmetros OBD esteja muito alta. Outro fator que pode contribuir para esse consumo é o uso constante de comunicação *Bluetooth*, que não é necessário no aplicativo Waze apesar de empregar o GPS. Além disso, o FuelMap armazena muitas informações em sua base de dados em tempo real, ou seja, utiliza intensivamente o armazenamento físico do dispositivo, consumindo mais energia. A Figura 5.12 ilustra a tela de configurações do sistema operacional Android, do celular Motorola Moto G3, de onde foram extraídos os dados para análise.

Figura 5.12 – Consumo de bateria - Aplicativo Waze

Fonte: elaborada pelo autor

O segundo aspecto a ser observado é o uso de memória RAM, para isso, foi consultado o estado da memória do dispositivo celular antes de executar o aplicativo FuelMap, e após a utilização de suas funcionalidades. Foi percebido um aumento máximo de 152 MB no uso da memória RAM após acessar todas as funcionalidades do aplicativo, como as notificações de tempo real, a consulta pelo histórico de percursos, a exibição do resumo e do mapa de um percurso específico, além da comparação gráfica entre consumo, velocidade, RPM e posição do pedal do acelerador. A Figura 5.13 comprova a variação do uso de memória RAM e foi obtida a partir das configurações do sistema operacional Android.

Figura 5.13 – Consumo de memória - Aplicativo FuelMap



Fonte: elaborada pelo autor

O último aspecto avaliado em relação ao consumo de recursos é a capacidade de armazenamento no cartão de memória SD do dispositivo. Dependendo da frequência de amostragem, dados de telemetria podem gerar conjuntos relativamente grandes de dados, portanto, é preciso tomar cuidado com esse aspecto. Para o armazenamento dos 50 trajetos da base de dados de testes, que somam 28 horas de diagnóstico, foi observado que o aplicativo FuelMap utilizou 73,42 MB, além de 8 MB necessários para o aplicativo propriamente dito, conforme ilustrado pela Figura 5.14. Dividindo o volume de dados total pelo tempo de utilização, pode-se estimar que o aplicativo FuelMap necessita de 2,62 MB por hora de diagnóstico armazenado. Sendo assim, em um dispositivo celular atual, que possua 8 GB de armazenamento interno, seria possível gravar, aproximadamente, 3.000 horas de dados de diagnóstico, o que pode ser considerado aceitável, uma vez que o usuário levará muito tempo para executar o aplicativo por essa quantidade de tempo. Mesmo assim, caso não haja mais memória disponível no dispositivo, o aplicativo permite excluir percursos inteiros do histórico, removendo em cascata todos os registros de diagnóstico associados a ele. Além disso, existe a possibilidade de utilizar apenas a funcionalidade de tempo real, sem armazenar os dados, o que pode ser uma solução alternativa em celulares com pouca capacidade de armazenamento. Em trabalhos futuros, poderia ser disponibilizada a exportação dos dados de diagnóstico para um computador pessoal, ou para repositórios em nuvem. Dessa forma, os requisitos de espaço de armazenamento local no dispositivo podem ser reduzidos a apenas o espaço de armazenamento do aplicativo.

Figura 5.14 – Consumo de armazenamento em 28 horas de utilização - Aplicativo FuelMap

Fonte: elaborada pelo autor

5.6 Considerações Finais

A avaliação do aplicativo FuelMap pode ser considerada satisfatória, uma vez que, o aplicativo funcionou corretamente durante os testes, forneceu informações valiosas em relação aos fatores que contribuem para a economia de combustível, comprovou algumas das hipóteses levantadas inicialmente e detalhou os requisitos de hardware necessários para utilizar a solução. Ao que compete esse último aspecto, pode-se perceber que um parâmetro decisivo para sua determinação é a frequência de amostragem, a qual, se muito alta, pode ser gerar grandes volumes de dados desnecessariamente, além de consumir muita energia. Por outro lado, se for muito baixa, não atenderia aos requisitos funcionais relacionados aos alertas de tempo real. O aplicativo FuelMap realiza, periodicamente, a cada 300ms, a leitura de 7 parâmetros de diagnóstico do veículo. Essa leitura pode durar cerca de 1 segundo, portanto, alguns eventos da funcionalidade de tempo real, com duração inferior a esse tempo, podem ser perdidos, o que não é desejável. Não foi possível aumentar a taxa de amostragem devido a uma limitação no tempo de leitura dos parâmetros de diagnóstico, que pode estar relacionada à qualidade do adaptador ELM-327 Mini, ou a aspectos de implementação da biblioteca utilizada. Mesmo assim, os alertas em tempo real atenderam ao objetivo esperado, que era contribuir para a melhora nos hábitos de condução do motorista.

O uso de memória RAM pode ser considerado relativamente alto, mas se justifica, pois, algumas funcionalidades atuam sobre grandes conjuntos de informações. Por exemplo, a exibição de gráficos e os mapas consomem cerca de 8.000 registros de diagnóstico da base de dados para um percurso de 1 hora. Cada registro consiste em 6 campos do tipo *INTEGER*, 2 campos do tipo *TEXT*, com 20 caracteres cada, e 12 campos do tipo *REAL*. Considerando que o tipo *INTEGER* possui 4 bytes, o tipo *REAL* possui 8 bytes e o campo *TEXT* possui 20 bytes, cada percurso de 1 hora de duração consumiria 1,28 MB de memória RAM para ser exibido no mapa, e essa mesma quantidade seria necessária para a visualização de um dos gráficos. Além disso, as imagens dos mapas e os ícones que indicam os excessos cometidos pelo usuário precisariam ser carregadas na memória.

A tela de tempo real também contribui para essa alocação, pois, mantém em memória os registros de consumo e de velocidade até o término do percurso para o cálculo de valores médios. O aplicativo realiza, aproximadamente, 15 leituras a cada 3,3 segundos, ou seja, com uma frequência de 4,55 amostras por segundo. A cada leitura, são alocados dois valores do tipo *float*, com 32 bits cada, em uma lista. Em um percurso de uma hora de duração, o aplicativo armazenaria, aproximadamente, 32760 valores do tipo *float*, o equivalente a 127,96 KB de

dados. Para fins de comparação, foi avaliado o aplicativo Waze, que é bastante popular e que é utilizado em veículos de passeio. O Waze obteve uma utilização máxima de 147 MB de memória RAM, pouco menos que os 152 MB consumidos pelo aplicativo FuelMap. Com 3.4 % a mais de alocação de memória RAM do que o Waze, conclui-se que esse aspecto não seria um problema para sua utilização nos veículos.

Uma base de dados com 50 trajetos permitiu a realização de diversas análises interessantes e elucidativas para o presente trabalho. Vale ressaltar que foi analisado um conjunto pequeno de amostras. Não se pode garantir, com certeza, qual horário de realização proporciona o menor custo, ou qual o percentual exato de economia de combustível que está associado ao número de aceleradas bruscas. No entanto, a obtenção de um conjunto de dados suficientemente grande é apenas uma limitação temporal que pode ser reduzida ao analisar informações de diferentes usuários que realizam trajetos semelhantes. Esse aspecto abre possibilidades para alguns trabalhos futuros, os quais, poderiam, por exemplo, permitir exportar os dados de diagnóstico para uma plataforma Web ou para um computador *Desktop*, que consolide-os gerando estatísticas mais confiáveis para os usuários.

O aplicativo FuelMap atingiu os objetivos esperados, isso é, funciona e é capaz de fornecer informações até então desconhecidas pelos motoristas. Essas informações ajudarão a entender e aprimorar o comportamento ao volante para economizar combustível e reduzir a emissão de gases poluentes na atmosfera. Dessa forma, conclui-se que a solução proposta é uma contribuição positiva para os sistemas de transporte das cidades inteligentes.

6 CONCLUSÃO

No contexto das cidades inteligentes, em que existe a intenção de buscar sistemas de transporte eficientes, um dos problemas considerados é o excessivo consumo de combustíveis. Trata-se de um tema de extrema relevância, dado que, cientistas alertam para as graves mudanças climáticas que vem ocorrendo no planeta devido ao efeito estufa, e que sua principal causa é a queima de combustíveis fósseis. O presente trabalho propôs uma solução para esse problema, que consiste em um aplicativo para celulares, baseado em tecnologias de diagnóstico a bordo (OBD-II), com foco na conscientização de condutores em relação aos seus hábitos de condução. O resultado obtido é uma versão completamente funcional do aplicativo, chamado FuelMap, além de uma análise detalhada das tecnologias e das principais empresas existentes no mercado de *In-vehicle Infotainment Systems*. Com base nessas informações, mostrou-se viável e promissora a adoção de aplicações inteligentes para serem utilizadas durante o uso de veículos de passeio para melhorar os hábitos de condução e, com isso, reduzir o consumo de combustível e a poluição.

Ao longo do desenvolvimento foram detalhadas as principais tecnologias que compõem sistemas de diagnóstico a bordo. Com base nessas informações, a solução foi especificada e desenvolvida, seguindo princípios e práticas de engenharia de software. De posse de uma versão funcional do aplicativo FuelMap, foram coletadas informações sobre diversos percursos, as quais subsidiaram uma análise dos aspectos relevantes para a economia de combustíveis. Os resultados se mostraram consistentes com as hipóteses assumidas, que dizem respeito à influência de alguns indicadores como velocidade, RPM, tempo parado e posição do pedal do acelerador para o consumo de combustível. Isso mostra que a escolha dos alertas de condução estava alinhada com os fatores que realmente podem trazer algum benefício para o dia a dia dos usuários.

O aplicativo FuelMap cria uma série de possibilidades para melhorias e aplicações futuras, o que não limita seu uso aos veículos de passeio. Um nicho em que esse tipo de solução pode ser inserido é o controle de frotas, sejam de veículos de aluguel, de caminhões de transportadoras ou de ônibus de agências de turismo. Esses são exemplos de empresas em que o combustível está em sua principal relação de custos variáveis. Para implementar uma solução útil a esses setores, é necessário adequar as equações de cálculo para motores à diesel, os quais possuem funcionamento ligeiramente diferente daqueles movidos à etanol ou à gasolina. Sobre essa plataforma, seria possível criar campanhas corporativas de incentivo à economia de combustíveis, com premiações para os funcionários que atingissem determinadas metas estabelecidas pela empresa, etc.

Ainda no contexto de sistemas de transporte inteligentes, existem diversos aplicativos disponíveis para o auxílio georreferenciado à condução e para a identificação de rotas mais rápidas, ou mais curtas, entre uma origem e um destino, como o Waze e o Google Maps. Essas soluções, no entanto, não levam em consideração que nem sempre os caminhos mais curtos ou mais rápidos são os de menor custo financeiro. A partir dos dados coletados pelo aplicativo FuelMap foi possível perceber que existem trajetos de menor distância que são mais custosos que os demais, devido a diversos fatores, como o tempo parado, o número de subidas e descidas e a velocidade média de cada via. Também há percursos que, apesar de serem mais rápidos exigem mais do motor e, portanto, consomem mais combustível. Nesse contexto, vale destacar que o aplicativo FuelMap pode ser integrado a esses sistemas de navegação, permitindo que o usuário escolha a rota de menor consumo de combustível, independente da duração ou da distância total percorrida. A estensão do aplicativo FuelMap para integrar-se a essas ferramentas é uma possibilidade interessante e útil a ser desenvolvida.

Além da estensão de funcionalidades, um aspecto em que se pode investir futuramente, é a compatibilidade com outros dispositivos celulares. Atualmente, o FuelMap está disponível apenas para o sistema operacional Android, devido à sua implementação em linguagem Java. Para compatibilizar o aplicativo com dispositivos iOS, seria necessário implementar as mesmas funcionalidades existentes na versão atual sobre a plataforma de desenvolvimento da Apple. Além disso, conforme descrito na seção 4.2, os adaptadores ELM 327 *Bluetooth* não são compatíveis com o sistema operacional iOS, portanto, se faz necessário um dispositivo de hardware com conexão Wi-Fi ou USB.

Em relação à interface visual e à usabilidade, existem alguns aspectos que poderiam ser melhorados, como, por exemplo, a tradução para outros idiomas. Além disso, a tela de configurações do aplicativo exige informações técnicas que um usuário leigo pode não ser capaz de informar, como os limites de velocidade, de RPM e de posição do acelerador, a densidade do combustível e a eficiência volumétrica. Uma melhoria interessante seria a seleção de um perfil do motorista, a partir do qual, o aplicativo determinaria automaticamente parte dos parâmetros. Outra facilidade que poderia ser desenvolvida é a identificação automática da proporção entre etanol e gasolina, indicada pela sonda lâmbda. A partir dessa leitura, seria possível calcular a respectiva densidade do combustível presente no tanque.

REFERÊNCIAS

AKERS, C. M. M. L. R. G. **Prototyping**. 2012. <http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html>. (Visitado em 13/05/2016).

ANP. **Anuário Estatístico Brasileiro do Petróleo, Gás Natural e Biocombustíveis 2015**. 2015. <http://www.anp.gov.br>. (Visitado em 30/09/2015).

APPLE INC. **CarPlay**. 2015. <http://www.apple.com/br/ios/carplay/>. (Visitado em 23/09/2015).

ARTHUR, C. **Google, Apple and Microsoft race to bring software to cars**. 2014. <http://www.theguardian.com/technology/2014/jan/06/google-apple-microsoft-software-cars-apps>. (Visitado em 23/09/2015).

BECK, K. **Implementation Patterns**. Addison-Wesley, 2008. (The Addison-Wesley signature series). ISBN 9780321413093. Disponível em: <https://books.google.com.br/books?id=i40hAQAAIAAJ>.

CCC. **MirrorLink**. 2014. <http://mirrorlink.com/cars>. (Visitado em 30/09/2015).

CCC. **Car Connectivity Consortium**. 2015. <http://carconnectivity.org/>. (Visitado em 30/09/2015).

CHACON, S.; STRAUB, B. **Pro git**. [S.l.]: Apress, 2014.

ECLIPSE. **Franca**. 2013. <http://www.eclipse.org/proposals/modeling.franca/>. (Visitado em 27/09/2015).

ECLIPSE. **The Eclipse Foundation open source community website.** 2016. <https://eclipse.org/>. (Visitado em 05/06/2016).

ECMA INTERNATIONAL. **ECMA-404: The JSON Data Interchange Format**. 2013. <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. (Visitado em 27/10/2015).

FIAT. **Blue & Me**. 2015. <http://www.fiat.com.br/conectividade/blue-me.html>. (Visitado em 07/10/2015).

FORD MOTOR COMPANY. **Ford SYNC & MyFord Touch**. 2015. <http://www.ford.com/technology/sync/>. (Visitado em 07/10/2015).

FORD MOTOR COMPANY. **https://developer.ford.com**. 2015. <https://developer.ford.com/>. (Visitado em 07/10/2015).

FOWLER, M. **Patterns of enterprise application architecture**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2002.

GENIVI ALLIANCE. **About GENIVI**. 2015. <http://www.genivi.org>. (Visitado em 24/09/2015).

GOOGLE. **Google Self-Driving Car Project**. 2015. <http://www.google.com/selfdrivingcar/>. (Visitado em 19/09/2015).

GOOGLE. **Open Automotive Alliance**. 2015. <http://www.openautoalliance.net/#about>. (Visitado em 12/10/2015).

GOOGLE. **Android Studio and SDK Tools**. 2016. <https://developer.android.com/studio/index.html>. (Visitado em 05/06/2016).

GRADLE. **Gradle | Modern Open-Source Enterprise Build Automation**. 2016. <http://gradle.org/>. (Visitado em 05/06/2016).

HINES, J. F. **Market Trends: In-Vehicle Infotainment Systems Drive Automotive Semiconductor Growth**. 2014. <https://www.gartner.com/doc/2658826/market-trends-invehicle-infotainment-systems>. (Visitado em 07/10/2015).

HYUNDAI. **2016 Hyundai Sonata Featuring Android Auto | HyundaiUSA**. 2015. <https://www.hyundaiusa.com/sonata/technology.aspx>. (Visitado em 12/10/2015).

ICCULUS. **OBDSim**. 2011. <https://icculus.org/obdgpslogger/obdsim.html>. (Visitado em 22/05/2016).

ISO. **ISO 14230-4: Road vehicles - Diagnostic systems**. 2000.

ISO. **ISO 15765-4: Road vehicles - Diagnostic on Controller Area Networks (CAN)**. 2005.

JETBRAINS. **IntelliJ IDEA the Java IDE**. 2016. <https://www.jetbrains.com/idea/>. (Visitado em 05/06/2016).

KLEINA, N. **Cortana, Siri, Google Now e Tina: qual a melhor assistente pessoal? [vídeo] - TecMundo**. 2014. <http://www.tecmundo.com.br/siri/59361-cortana-siri-google-now-tina-melhor-assistente-pessoal.htm>. (Visitado em 12/10/2015).

LANCASTER, D. T. **Volkswagen's first car with Android Auto starts landing in showrooms | Android Central**. 2015. <http://www.androidcentral.com/volkswagen-announces-android-auto-support-its-2016-lineup>. (Visitado em 12/10/2015).

LAUKKONEN, J. **Why Doesn't My ELM327 iPhone Adapter Work?** 2015. <http://cartech.about.com/od/Connected-Cars/f/ELM327-iPhone.htm>. (Visitado em 11/06/2016).

LUBY, G. **Carros inteligentes**. 2014. <http://pt.slideshare.net/GustavoLuby/carros-inteligentes?related=1>. (Visitado em 19/09/2015).

MERCADODOMECANICO. **Scanner Bosch – Mercado do Mecanico**. 2016. <http://www.mercadodomecanico.com.br/aparelho-diagnostico-scanner--com-software----bosch-kts-470-kts470s/p>. (Visitado em 24/05/2016).

MICROSOFT. **A Technical Companion to Windows Embedded Automotive 7**. 2010. <http://www.microsoft.com/windowsembedded/en-us/windows-embedded-automotive-7.aspx>. (Visitado em 01/10/2015).

NAGEL, S. F. A. **Understanding the communication between automotive mechatronics and electronics for remanufacturing purposes**. 2011. <http://www.lup.uni-bayreuth.de/documents/universitaere_forschung/Understanding_CAN_BUS_Technologies_for_REMAN_Purposes.pdf>. (Visitado em 26/04/2016).

OBD SOLUTIONS. **What is OBD?** 2015. <http://www.obdsol.com/knowledgebase/on-board-diagnostics/what-is-obd/>. (Visitado em 12/10/2015).

OBDTESTER. **OBD2 protocols**. 2015. <http://obdtester.com/obd2_protocols>. (Visitado em 23/04/2016).

OLIVER, S. **CarPlay development characterized as easy, but Apple planning 'slow and steady' rollout**. 2014. <http://appleinsider.com/articles/14/03/06/carplay-development-characterized-as-easy-but-apple-planning-slow-and-steady-rollout>. (Visitado em 30/09/2015).

OPENXC. **OpenXC**. 2015. <http://openxcplatform.com/>. (Visitado em 27/10/2015).

OUTILS OBD FACILE. **Car diagnostic interfaces ELM327 OBD2**. 2010. <http://www.outilsobdfacile.com/diagnostic-interface-elm-327.php>. (Visitado em 30/04/2016).

PIRES, P. **GitHub - pires/obd-java-api: OBD-II Java API**. 2013. <https://github.com/pires/obd-java-api>. (Visitado em 14/05/2016).

PORTNOI'S, M. **aula20.pdf**. 2006. <https://www.eecis.udel.edu/~portnoi/classroom/prob_estatistica/2006_1/lecture_slides/aula20.pdf>. (Visitado em 30/06/2016).

RIBEIRO, V. D. F. da C. Mining geographic data for fuel consumption estimation. 2013.

SAE. **J1979: E/E Diagnostic Test Modes - SAE International**. 1991. <http://standards.sae.org/j1979_201202/>. (Visitado em 27/10/2015).

SAE. **J1962A: Diagnostic Connector - SAE International**. 1992. <http://standards.sae.org/j1962_201509/>. (Visitado em 27/10/2015).

SCANNERS, T. B. O.-I. **OBD2 Scanners - What is Mode 1? « The Best OBD-II Scanners**. 2016. <http://thebestobdiiscanners.com/10-modes-of-operation-for-obd2-scanners/obd2-scanners-what-is-mode-1/>. (Visitado em 22/04/2016).

SCHULZ, D. **Ciclo de Otto**. 2009. <http://www.if.ufrgs.br/~dschulz/web/ciclo_otto.htm>. (Visitado em 29/06/2016).

SINGH, A. **Universal Asynchronous Receiver/Transmiter - Uart**. 2014. <http://pt.slideshare.net/AnkitSingh13/uart-32550652>. (Visitado em 30/04/2016).

SOMMERVILLE, I. et al. **Engenharia de software**. [S.l.]: ADDISON WESLEY BRA, 2008. ISBN 9788588639287.

SONATYPE. **The Central Repository Search Engine**. 2016. <http://search.maven.org/>. (Visitado em 05/06/2016).

SPARKS, K. **VPW J1850 Multiplexing Controller (BDLC) Module**. 2004.

TOYOTA. **Aiming for Safe Vehicles**. 2015. <http://www.toyota-global.com/innovation/safety_technology/>. (Visitado em 22/09/2015).

WACH, L. **GitHub - lecho/hellocharts-android: Charts/graphs library for Android compatible with API 8+, several chart types with support for scaling, scrolling and animations**. 2013. <https://github.com/lecho/hellocharts-android>. (Visitado em 14/05/2016).

WIKIPEDIA. **On-board diagnostics - Wikipedia, the free encyclopedia**. 2014. <https://en.wikipedia.org/wiki/On-board_diagnostics>. (Visitado em 19/09/2015).

WIKIPEDIA. **Watt-hora – Wikipédia, a enciclopédia livre**. 2015. <https://pt.wikipedia.org/wiki/Watt-hora>. (Visitado em 29/05/2016).

ZETTER, K. **Researchers Hacked a Model S, But Tesla's Already Released a Patch | WIRED**. 2015. <https://www.wired.com/2015/08/researchers-hacked-model-s-teslas-already/>. (Visitado em 02/05/2016).