ECONODRIVE: COMPUTADOR DE BORDO AUTOMOTIVO EM PLATAFORMA MÓVEL IOS

Vinícius Guedes Fontes[*]
José Antônio O. de Figueiredo[**]

RESUMO

A economia de combustíveis fósseis é de interesse geral, e os hábitos dos motoristas são um dos principais fatores no desperdício de combustível. O computador de bordo automotivo é uma ferramenta que pode auxiliar na reeducação dos motoristas, porém no Brasil esse equipamento não é encontrado na maioria dos veículos. Este artigo descreve a implementação de um computador de bordo automotivo utilizando um dispositivo iOS, capaz de coletar os dados necessários da *Engine Control Unit* (ECU) do veículo através de um equipamento específico e realizar os cálculos necessários para determinar o consumo instantâneo e médio de combustível. O protótipo desenvolvido como prova de conceito determinou que é possível coletar os dados e realizar o cálculo de consumo em tempo real, porém os resultados preliminares foram diferentes do esperado, o que levou à implementações de melhorias no algoritmo, de forma a determinar o consumo com maior precisão. Os resultados finais obtidos ficaram em torno de 1% em relação ao esperado.

Palavras-chave: iOS. OBD-II. Consumo de combustível. Computador de bordo automotivo.

1 INTRODUÇÃO

Diminuir o consumo de combustíveis em veículos automotores é algo que interessa à maioria das pessoas, seja por razões ambientais, financeiras ou ambas. Apesar de a indústria ter apresentado sucessivos progressos na fabricação de veículos mais eficientes e veículos mais leves, os hábitos do motorista continuam sendo os principais fatores a influenciar negativamente o consumo.

Uma ferramenta capaz de auxiliar na reeducação do motorista é o computador de bordo automotivo, pois ele permite o monitoramento do consumo instantâneo e médio de combustível em tempo real. Infelizmente, no Brasil esse item encarece muito o valor final do automóvel, portanto apenas os veículos de maior valor possuem esse equipamento. Outro fator que agrava ainda mais a situação é a impossibilidade de instalação desse opcional após a compra do veículo, condenando

[*] Graduando em Tecnologia em Sistemas para Internet pelo IFSUL campus Passo Fundo. E-mail: contato@viniciusfontes.com
[**] Orientador, professor do IFSUL. Email: jose.figueiredo@passofundo.ifsul.edu.br

o proprietário à ausência dessa ferramenta útil no aprendizado de hábitos mais eficientes.

Este projeto visa melhorar essa situação através do desenvolvimento de uma aplicação para dispositivos móveis iOS capaz de se conectar à ECU do veículo, ler os dados necessários e apresentar os valores do consumo instantâneo e médio e, dessa forma, efetivamente implementar um computador de bordo automotivo. Dentre os trabalhos a este relacionados, destaca-se o artigo de Lightner, publicado em 2005 pela revista *Circuit Cellar* e intitulado *AVR-Based Fuel Consumption Gauge,* onde o autor descreve o desenvolvimento de um medidor analógico de consumo instantâneo de combustível. O presente trabalho baseia-se nos conceitos já explorados por Lightner, ao mesmo tempo que propõe uma nova abordagem para tentar solucionar o problema, implementando mais recursos.

O restante deste artigo está dividido da seguinte forma: a seção dois aborda o desenvolvimento do trabalho, começando pela apresentação do protocolo *On-Board Diagnostics* versão II (OBD-II). Após, são apresentados os métodos de coleta de dados e cálculo de consumo de combustível, bem como os primeiros resultados obtidos. Em seguida, é abordada a modelagem do aplicativo e o processo de refinamento do seu algoritmo. Finalmente, é realizada a análise dos dados obtidos. As considerações finais, incluindo os trabalhos futuros, estão presentes na seção três.

2 DESENVOLVIMENTO

Nesta seção é detalhado o protocolo e método utilizado para a coleta dos dados e o cálculo de consumo, é definido o que é uma prova de conceito e sua importância, a modelagem do *software* é descrita e é realizada uma análise dos resultados obtidos. Com base nestes, os parâmetros do algoritmo de cálculo de consumo são então ajustados a fim de obter resultados mais precisos.

2.1 Protocolo OBD-II

O OBD-II é definido pela *Environment Protection Agency* (EPA) como um sistema computadorizado embarcado em todos os veículos leves e caminhões fabricados a partir de 1996 (CODE OF FEDERAL REGULATIONS, 2010). Esse sistema é projetado para monitorar a performance dos principais componentes de um motor, incluindo aqueles responsáveis por controlar as emissões gasosas.

Todo veículo fabricado a partir de 1996 deve disponibilizar um conector no interior do compartimento de passageiros que permita a leitura de dados de diagnóstico do motor a uma ferramenta de varredura apropriada para este fim. Estes dados incluem valores medidos em tempo real como velocidade do motor, temperatura do líquido de arrefecimento, velocidade do veículo, posição do pedal do acelerador, além de inúmeros outros (LIGHTNER, 2005, p. 1).

O padrão *Society of Automotive and Aerospacial Engineers* (SAE) J1979 especifica que os dados devem ser obtidos um a um, e apenas através de uma requisição efetuada à ECU; ou seja, o protocolo é unicamente do tipo requisição/resposta. Cada pacote de dados OBD-II, tanto na requisição quanto na resposta, possui um cabeçalho de 3 bytes, onde o primeiro define a prioridade/tipo da requisição, o segundo define o endereço de destino e o terceiro define o endereço de origem. O cabeçalho é imediatamente seguido por um payload de até 7 bytes, e o pacote é finalizado com um byte de *Cyclical Redundance Check* (CRC), calculado de acordo o padrão SAE J1850 (LIGHTNER, 2005, p. 2; ELM ELECTRONICS, 2010, p. 35).

Os primeiros 2 bytes do *payload* da requisição correspondem, respectivamente, ao *Diagnostic Test Mode* (DTM) e ao *Parameter Identificator* (PID) solicitado. Os DTMs descrevem o tipo de dado que está sendo solicitado, enquanto que os PIDs especificam o dado específico desejado. A combinação desses dois parâmetros permite selecionar com exatidão o item de dados que se deseja requisitar (ELM ELECTRONICS, 2010, p. 29).

No pacote de resposta da requisição, o *payload* começa com os mesmos 2 bytes referentes ao DTM e ao PID solicitado, com a diferença que o DTM é composto pelo DTM original requisitado acrescido do valor 0x40. O PID da resposta é idêntico ao PID da requisição, e os dados referentes ao PID solicitado vão do terceiro até o sétimo byte. O último *byte*, como na requisição, é um valor de CRC (ELM ELECTRONICS, 2010, p. 35). Neste projeto, é utilizado apenas o DTM 0x01, utilizado para exibir os dados atuais.

2.2 Coleta de dados

Os dados fornecidos pelo sistema OBD-II seguem padrões definidos internacionalmente, tanto pela *International Organization for Standardization* (ISO)

quanto pela SAE, mas esses padrões não preveem a comunicação direta entre o veículo e um sistema computacional, como PCs ou *smartphones*.

Para que isso seja possível, neste projeto será utilizado um *dongle*[1] baseado no circuito integrado ELM327, circuito esse que “é projetado para agir como uma *bridge* entre essas [...] portas OBD e uma interface [ou porta serial] padrão RS-232” (ELM ELECTRONICS, 2010).

Além do ELM327, o *dongle* é dotado também de uma interface de rede sem fios no padrão *Institute of Electrical and Electronics Engineers* (IEEE) 802.11 e dos demais componentes eletrônicos e de *software* necessários ao seu funcionamento. A Figura 1 apresenta o aspecto físico do *dongle*.

Figura 1 - Dongle OBD-II Wi-Fi
Fonte: Do autor

O ELM327 gera automaticamente os cabeçalhos dos pacotes OBD-II para as requisições, bem como os suprime nas respostas. Os *bytes* de CRC também são automaticamente gerados e suprimidos da mesma forma. Assim, é possível requisitar um dado apenas especificando seu DTM e PID e receber a resposta no mesmo formato, sem cabeçalhos e bytes de CRC, diminuindo muito a complexidade da comunicação (ELM ELECTRONICS, 2010, p. 29).

Quando o *dongle* é conectado à porta OBD-II do veículo, uma rede *ad-hoc*, sem criptografia nem autenticação, chamada “CLKDevices” é criada e um *socket*

[1] “Dispositivo conectado a um computador que o permite executar um software específico, ou que pode ser utilizado em outras formas, por exemplo um adaptador sem fio.” (CAMBRIDGE DICTIONARIES ONLINE, tradução nossa)

TCP no endereço IPv4 192.168.0.10 e porta 35000 é criado, aguardando a conexão de clientes. O endereço IPv4 e a porta são fixos e não podem ser alterados. Internamente, o *dongle* realiza a conexão entre a interface serial RS-232 e o *socket* TCP, conforme o diagrama de blocos apresentado na Figura 2.

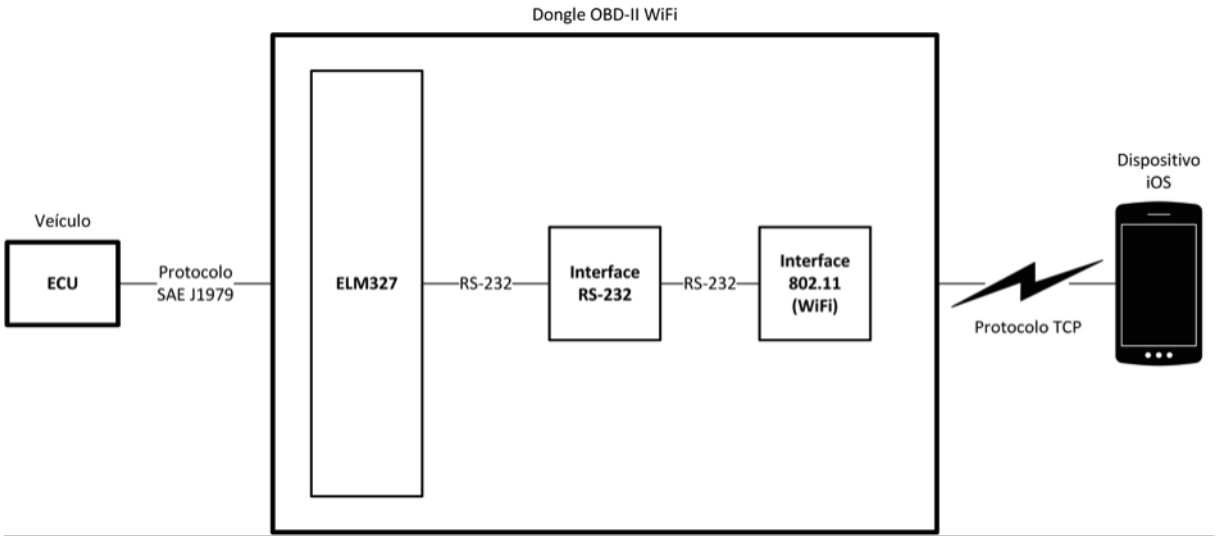

Figura 2 - Diagrama de blocos do *dongle*
Fonte: Do autor

Dessa forma, é possível solicitar dados ao ELM327, como, por exemplo, a temperatura do ar no coletor de admissão. Nesse caso hipotético, envia-se o comando “010F” (DTM 0x01, PID 0x0F) ao ELM327 utilizando-se o *socket* TCP disponibilizado pelo *dongle*. O *dongle* então envia o comando para a interface RS-232 do ELM327, que interpreta os dados recebidos e envia a solicitação para a ECU do veículo. Assim que a ECU responder, o ELM327 envia os dados para o *dongle* utilizando sua interface RS-232. Finalmente, o *dongle* lê os dados recebidos em sua interface RS-232 e os escreve no *socket* TCP. Um exemplo de resposta nesse caso poderia ser “41 0F 4E”, onde o primeiro *byte* é o DTM 0x01 acrescido de 0x40, o segundo *byte* é o próprio PID solicitado e o terceiro corresponde ao valor decimal 78. Dessa forma, a temperatura do ar na admissão informada pela ECU é de 38^{o}C, devido ao *offset* de -40^{o}C que deve ser considerado para esse PID específico.

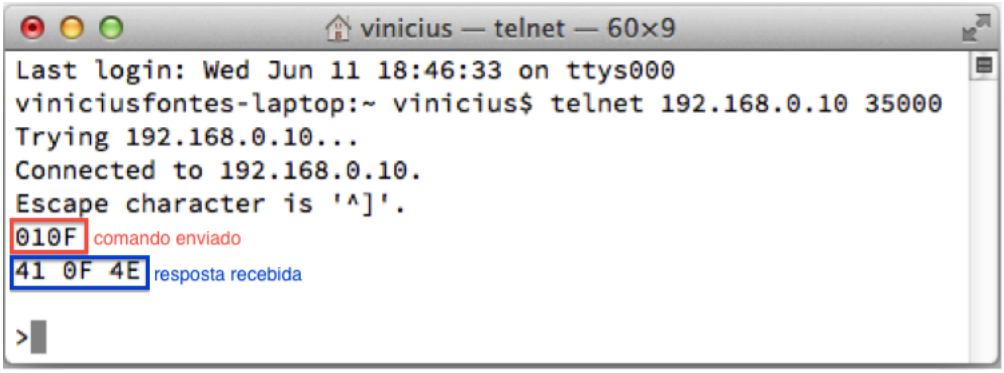

Figura 3 - Comunicação entre um computador e o *dongle*
Fonte: Do autor

O fato da rede *ad-hoc* disponibilizada pelo *dongle* não ter autenticação nem criptografia poderia ser visto como uma falha de segurança. Checkoway et al. (2011) mostraram que é possível "controlar componentes chave (por exemplo, faróis, travas, freios e motor) assim como injetar código em ECUs" [tradução nossa], mas isso só foi possível ao utilizar dispositivos que suportam o padrão SAE J2534. Já o circuito integrado ELM327, utilizado neste projeto, "é capaz apenas de obter informações de veículos e não suporta nenhum tipo de programação de ECU" (ELM ELECTRONICS, 2014, tradução nossa) e, portanto, não deve oferecer risco significativo de segurança.

2.3 Método utilizado para o cálculo de consumo de combustível

Para realizar o cálculo do consumo instantâneo de combustível, os seguintes dados são necessários: velocidade do veículo (VSS, PID 0x0D) e a taxa de massa de ar (MAF, PID 0x10). De acordo com Lightner (2005), apesar de nem todos os veículos possuírem um sensor MAF, é possível aproximar o valor de MAF utilizando outros dados, o que será visto mais adiante.

Sabendo a velocidade do veículo em quilômetros por hora e a MAF é possível calcular o consumo instantâneo, porém antes é preciso saber o valor de alguns outros parâmetros. Uma deles é a taxa da mistura ar/combustível. Ainda segundo Lightner (2005), essa taxa é mantida numa proporção quimicamente ideal de 14,7 gramas de ar para 1 grama de gasolina. Converte-se gramas de ar por segundo em gramas de gasolina por segundo dividindo-se por 14,7.

Outra constante necessária é a densidade da gasolina, que é de 6,17 libras por galão. Como existem 454 gramas em uma libra, divide-se a MAF por 14,7 e por 2.801,18 (valor esse obtido multiplicando-se 6,17 libras por galão por 454 gramas

por libra). Multiplica-se o valor obtido por 3.600 segundos em uma hora e temos a taxa de consumo de combustível em galões por hora. (LIGHTNER, 2005)

De posse da taxa de consumo de combustível, basta dividi-la pela velocidade do veículo em milhas por hora e obtém-se o valor do consumo instantâneo em milhas por galão (MPG). Como o sistema OBD-II retorna a velocidade do veículo em quilômetros por hora, é necessário convertê-la em milhas por hora, multiplicando a VSS por 0,621371. O sistema OBD-II retorna a MAF em gramas por segundo multiplicadas por 100, por isso existe a necessidade de dividir o valor lido por 100. (LIGHTNER, 2005)

Portanto, a equação que define o consumo instantâneo de combustível é descrita por Lightner (2005) da seguinte forma:

$$MPG = 7{,}107 \cdot \frac{VSS}{MAF}$$

onde MPG é o consumo instantâneo de combustível, em milhas por galão. A constante 7,107 é obtida a partir da simplificação da equação original:

$$MPG = \frac{14{,}7 \cdot 6{,}17 \cdot 454 \cdot (VSS \cdot 0{,}621371)}{\frac{3600 \cdot MAF}{100}}$$

Caso o veículo não possua um sensor MAF, é possível estimar o valor da taxa de massa de ar usando uma variável sintética chamada IMAP. Porém, nesse caso, torna-se necessário saber o valor do volume do deslocamento do motor (VDM) em litros (também conhecido como “cilindradas”) e a sua eficiência volumétrica (EV) em porcentagem. Lightner recomenda que o valor da EV seja inicialmente definido em 80%, e posteriormente calibrado após verificar a distância percorrida com uma quantia conhecida de combustível. (2011, 2014)

O valor de IMAP é calculado com a seguinte equação:

$$IMAP = \frac{RPM \cdot MAP}{\frac{IAT}{2}}$$

onde RPM é a velocidade do motor em rotações por minutos (PID 0x0C), MAP é a pressão absoluta no coletor de admissão em quilopascáis (PID 0x0B) e IAT é a temperatura do ar na admissão em Kelvin (PID 0x0F). De posse do valor de IMAP, o MAF pode ser aproximado da seguinte forma:

$$MAF = \frac{\frac{IMAP}{60} \cdot EV \cdot VDM \cdot MM}{R}$$

onde MM é a massa molecular média do ar, com valor de 28,97 gramas por mol e R é uma constante representando 8,314 joules por kelvin por mol. (LIGHTNER, 2011)

O consumo médio de combustível é obtido pela média aritmética de todos os valores de consumo instantâneo calculados. Já para se obter o valor em quilômetros por litro, basta multiplicar o valor de MPG por 0,4251. (WOLFRAM ALPHA, 2014).

2.4 Prova de Conceito

Inicialmente foi desenvolvida uma prova de conceito, consistindo em um protótipo de aplicativo iOS capaz de conectar-se ao *dongle* utilizando um *socket* TCP. Através dessa conexão, os dados referentes aos PIDs 0x0B (MAP), 0x0C (RPM), 0x0D (VSS) e 0x0F (IAT) são obtidos de forma contínua, um após o outro, em um laço infinito.

A cada iteração do laço o valor de MPG é calculado utilizando o método da variável sintética IMAP, já que o veículo utilizado nos testes não era provido de sensor MAF. Imediatamente após o cálculo do valor de MPG, seu valor é convertido em Km/l e o *label* referente ao consumo instantâneo da *view* principal do aplicativo é atualizado com o valor atual. Caso a VSS seja zero, o valor de MPG não é calculado, para evitar uma divisão por zero.

Por fim, o valor de MPG já convertido em Km/l é adicionado a uma variável. O valor dessa variável é então dividido pelo número de iterações do laço até o momento, dessa forma obtendo o valor do consumo médio e o exibindo em outro *label* na *view* principal do aplicativo. Caso o valor de MPG não tenha sido calculado nessa iteração, o cálculo do consumo médio não é realizado e o *label* referente ao consumo médio não é alterado.

A prova de conceito demonstrou que é possível coletar os dados e realizar o cálculo de consumo em tempo real. Porém, os resultados preliminares foram

diferentes do esperado. Utilizando um veículo VW Polo 1.6 à gasolina, ano 2003, com potência de 101cv, foram percorridos 127,3Km em rodovias. O consumo médio registrado foi de 17,4Km/L, um valor que difere em 31,03% em relação ao esperado, em torno de 12Km/L.

Algumas possibilidades que acreditamos explicar essa discrepância incluem:

- Valor de 80% para a eficiência volumétrica inadequado;
- Os cálculos não levaram em consideração a adição obrigatória de etanol à gasolina, fator agravado pelo veículo não ser flex;
- Medição da IAT não foi compensada em relação à pressão atmosférica.

Tendo em vista que os resultados obtidos com a prova de conceito não refletiam corretamente a realidade, foi decidido priorizar a precisão dos resultados obtidos. Ou seja, os valores calculados do consumo instantâneo e médio devem estar o mais próximo possível da realidade. Com isso, decidiu-se modelar e desenvolver um novo aplicativo, dessa vez levando em consideração os fatores apontados na seção anterior e realizando as compensações necessárias no cálculo de consumo de combustível.

Também foi decidido que o novo aplicativo deve armazenar os dados coletados em um Sistema Gerenciador de Bancos de Dados (SGBD) ao invés de apenas calcular os consumos instantâneo e médio utilizando valores armazenados em memória. Dessa forma, será possível alterar o algoritmo e recalcular o consumo sem que seja necessário rodar com o veículo, além de permitir comparar as diferenças entre um algoritmo e outro.

2.5 Modelagem

Nesta seção será apresentada a modelagem para o sistema proposto, que foi realizada seguindo os preceitos dos métodos ágeis. O manifesto ágil afirma que deve-se valorizar mais o *software* funcional do que a documentação extensa. (SOMMERVILLE, 2011) Em virtude disso, o processo de modelagem foi abreviado ao máximo, contando com apenas dois diagramas formais em *Unified Modeling Language* (UML).

Como o padrão de projeto adotado com maior frequência é o *delegation*, seu funcionamento básico será explicado antes que a modelagem em si seja abordada.

2.5.1. Padrão de projeto *delegation*

Todo o projeto do aplicativo foi realizado utilizando o padrão de projeto *delegation,* que é recomendado pela Apple. Este padrão de projeto consiste basicamente em delegar o processamento a classes diferentes, o que evita o *pooling* e permite trabalhar de forma assíncrona.

Na prática, uma classe é capaz de chamar diretamente um método de outra classe quando um determinado evento ocorrer. Por exemplo, a classe A requisita à classe B o *download* de um arquivo. Assim que o *download* for finalizado, a própria classe A chama o método downloadConcluido da classe B. Se ocorreu um erro no download, a classe A chamaria o método erroNoDownload. Dessa forma, não é necessário que a classe B solicite repetidamente à classe A o andamento do *download:* assim que houver progresso, a classe A poderia chamar o método progressoDoDownload, passando como parâmetro a porcentagem concluída. A referência à instância que receberá os eventos é normalmente armazenada no atributo *delegate* da classe responsável por gerar os eventos.

Os métodos a serem implementados na classe que recebe os eventos são normalmente definidos através de um protocolo, que possui a mesma funcionalidade das interfaces na linguagem Java. Portanto, se uma classe adere a um determinado protocolo, ela deve implementar os métodos nele descritos como obrigatórios. Também é possível ter métodos opcionais, que não geram erros de compilação caso deixem de ser implementados.

2.5.2 Diagrama de casos de uso

De acordo com a UML, atores são aqueles objetos externos ao sistema mas que interagem com ele de alguma forma. Durante a modelagem, foram identificados dois atores:

- Usuário (a pessoa física que interage com o sistema);
- *Scanner*.

A figura do *scanner* representa uma fonte de dados qualquer, que ao contrário do protótipo desenvolvido na prova de conceito, não está limitado apenas ao *dongle*. O *scanner* é responsável por receber as requisições de PIDs e retornar as respectivas respostas, dessa forma definindo um protocolo comum que abstrai a funcionalidade de obtenção dos dados das classes que os requisitam.

Casos de uso, também de acordo com a UML, são descrições das tarefas que o sistema deve realizar. Foram identificados quatro casos de uso:

- Consultar consumo;
- Conectar ao scanner;
- Requisitar PID;
- Calcular consumo.

O usuário apenas interage com o sistema de uma forma, que é através da visualização dos dados apresentados, correspondente ao caso de uso “Consultar consumo”. O sistema precisa calcular o consumo antes de exibi-lo ao usuário, portanto, faz-se necessário um caso de uso para tal operação. No entanto, para que isso seja possível, é indispensável que os dados necessários sejam obtidos. Isso ocorre através das requisições dos PIDs, que também compõem um caso de uso próprio. Da mesma forma, os PIDs só podem ser requisitados após a conexão ao *scanner*, o que justifica um caso de uso para essa funcionalidade.

Os atores, casos de uso e relacionamentos entre estas entidades são detalhados na Figura 4.

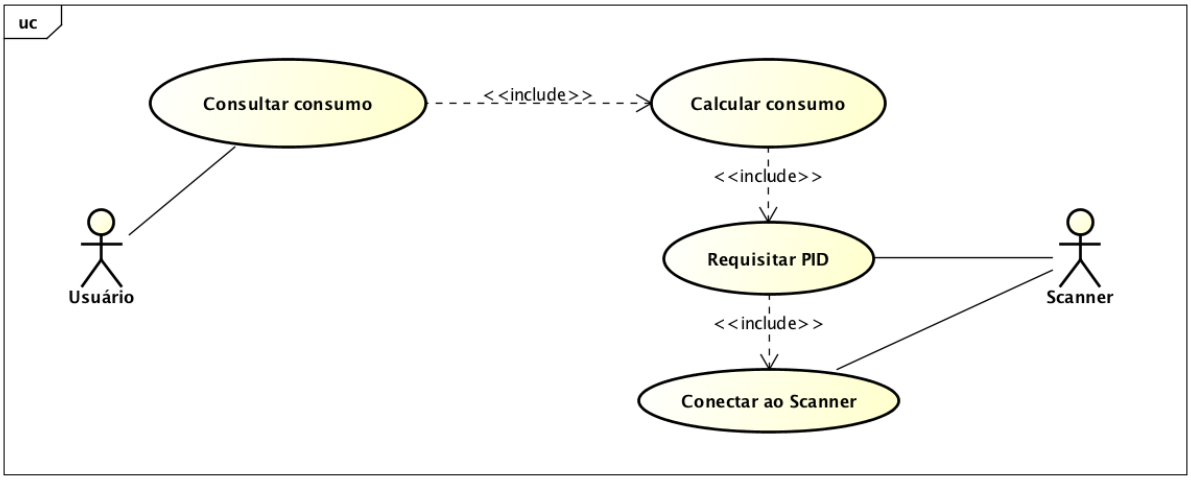

Figura 4 - Diagrama de casos de uso
Fonte: Do autor

2.5.3 Diagrama de classes

Com base nos casos de uso identificados, foi modelado o diagrama de classes da aplicação, apresentado na Figura 5.

O polimorfismo foi utilizado extensamente na modelagem da aplicação. Isso permitiu um grande nível de abstração, especialmente no caso da classe EDFuelEconomy. Esta classe possui um atributo chamado *scanner*, que é um objeto

do tipo EDScanner. Como as classes EDScannerSocket e EDSQLiteReader herdam de EDScanner, o atributo *scanner* da classe EDFuelEconomy pode receber uma instância de qualquer uma dessas classes, sem alterar em nada o seu funcionamento. Especificamente, isso permite que o consumo de combustível seja calculado utilizando o mesmo algoritmo, a partir de múltiplas fontes de dados, de forma totalmente transparente.

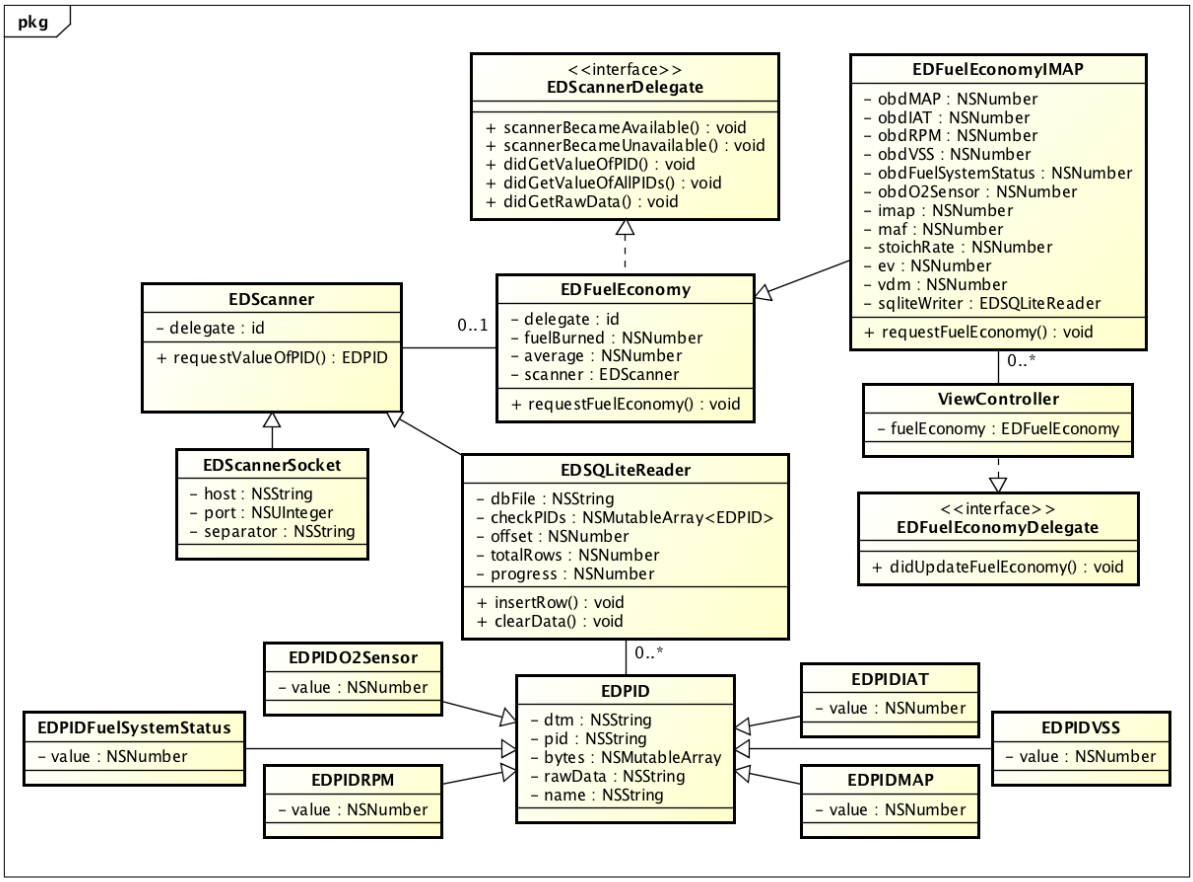

Figura 5 - Diagrama de classes
Fonte: Do autor

2.5.4 Interface com o usuário

A interface foi projetada considerando o fato de que o usuário deve obter a informação que precisa no menor tempo possível, já que sua atenção deve estar voltada para o trânsito. Foi adotado um esquema de alto contraste, branco sobre preto, para facilitar a leitura em qualquer condição de iluminação.

O consumo instantâneo é o item de maior destaque, seguido pelo consumo médio, que são apresentados no centro da interface. No canto inferior esquerdo é

apresentada a distância percorrida e o combustível consumido encontra-se no canto inferior direito, conforme a Figura 6.

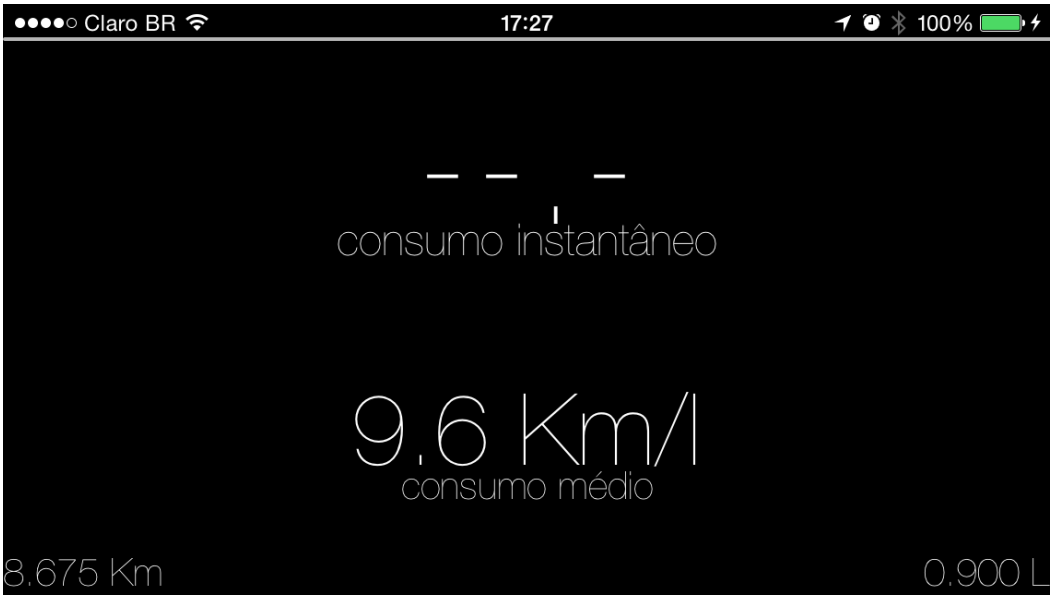

Figura 6 - Interface com o usuário
Fonte: Do autor

2.6 Análise e refinamento do cálculo de consumo de combustível

Como os resultados preliminares obtidos na prova de conceito não foram os esperados, decidiu-se aprimorar o algoritmo utilizado para o cálculo de consumo de combustível.

2.6.1 Metodologia de validação

A metodologia empregada para a validação dos resultados consistiu em encher o tanque de combustível e coletar os dados até que o veículo indicasse que o nível de combustível alcançou a reserva. Como o veículo utilizado tem um tanque de 45 litros, sendo que 7 litros compõem a reserva, a quantidade total de combustível consumido deve ser um valor próximo a 38 litros. Dividindo-se a distância total percorrida por 38 litros, obtém-se o consumo médio em Km/l. O cálculo do combustível consumido, conforme Lightner (2011), é dado pela seguinte equação:

$$galões\ de\ combustível\ = \frac{MAF}{\frac{razão\ ar\ combustível}{\frac{6{,}17}{454}}}$$

Portanto, o método anterior, que consistia em calcular a média aritmética dos consumos instantâneos para calcular o consumo médio, foi abandonado em função do cálculo do combustível consumido, que permite uma melhor validação dos dados.

Ao final da primeira coleta de dados foram percorridos 552,2Km, indicando um valor referencial do consumo médio de 14,53Km/l.

2.6.2 Ajuste dos parâmetros do algoritmo

As alterações no algoritmo compreenderam a coleta dos valores de dois PIDs adicionais: 0x14 (sensor de oxigênio) e 0x03 (status do sistema de combustível). O primeiro PID informa a porcentagem de combustível que está sendo injetada a mais ou a menos dentro dos cilindros, valor este que influencia diretamente a razão da mistura ar/combustível. Já o segundo PID determina se o sistema de combustível está funcionando em modo *closed loop* ou *open loop*.

O modo *closed loop* indica que a leitura do sensor de oxigênio está sendo levada em consideração para alterar a mistura ar/combustível. Já o modo *open loop* indica que o sensor de oxigênio está sendo ignorado. Isso pode ocorrer por motivos como o sensor de oxigênio não estar funcionando corretamente, ou não ter atingido a temperatura necessária para seu correto funcionamento, ou o motor precisar de potência extra naquele instante. Nesse caso, a razão da mistura ar/combustível é fixada em 11:1.

Realizando o cálculo de consumo de combustível considerando a razão estequiométrica[2] da mistura ar/combustível como 14,7:1 e a eficiência volumétrica do motor de 80%, chegou-se ao resultado de 30,385 litros de combustível consumidos e consumo médio de 18,17Km/l, o que representa uma variação de 20,02% em relação ao resultado esperado de 38 litros. Como a diferença entre o valor esperado e o valor calculado foi muito alta, outros fatores devem ser considerados.

Um dos fatores que não foi levado em consideração na prova de conceito foi de que o combustível utilizado no Brasil, chamado de E25, é composto por 25% de etanol e 75% de gasolina. Conforme Hitzemann et al (2010), a razão estequiométrica da mistura ar/combustível do E25 é de 13,2186:1, ao invés de 14,7:1 da gasolina pura. Repetindo o cálculo com o novo valor da razão

[2] Razão ideal que permite aos reagentes serem totalmente consumidos, sem que sobrem ao final da reação química. (BROWN et al., 2012)

estequiométrica, o resultado foi um consumo de 33,232 litros e 16,62Km/l, representando 12,57% de variação em relação ao referencial.

Como Lightner (2011, 2014) recomenda que a eficiência volumétrica do motor seja definida inicialmente em 80% e ajustada caso necessário, o cálculo foi repetido com os mesmos valores anteriores, porém com uma eficiência volumétrica de 90%. Dessa vez o combustível consumido foi calculado como 37,386 litros e 14,77Km/l, representando apenas 1,61% de diferença em relação ao referencial. Estes resultados são apresentados de forma resumida na Tabela 1.

Tabela 1 – Valores de referência e resultados obtidos

	Razão Estequiométrica	**Eficiência Volumétrica (%)**	**Consumo Total (l)**	**Consumo Médio (Km/l)**	**Variação (%)**
Referência (552,2 Km)	**-**	**-**	**38**	**14,53**	**-**
Resultado original	14,7000:1	80	30,385	18,71	20,02
Refinamento 1	13,2186:1	80	33,232	16,62	12,57
Refinamento 2	13,2186:1	90	37,386	14,77	1,61

Fonte: Do autor

2.6.3 Validação dos dados

Para validar os resultados obtidos após o refinamento do algoritmo, foi realizada uma nova coleta de dados utilizando a mesma metodologia. O novo referencial compreendeu um consumo de 38 litros em 372,1Km, resultando em 9,79Km/l. O cálculo do combustível consumido utilizando estes dados coletados foi de 37,668 litros e 9,9Km/l, representando 0,65% de variação em relação à referência, e apenas 0,96% de variação em relação ao cálculo anterior. A análise dos dados da validação é apresentada na Tabela 2.

Tabela 2 – Validação dos dados

	Razão Estequiométrica	**Eficiência Volumétrica (%)**	**Consumo Total (l)**	**Consumo Médio (Km/l)**	**Variação (%)**
Referência (372,1 Km)	**-**	**-**	**38**	**9,79**	**-**
Validação	13,2186:1	90	37,668	9,90	0,65

Fonte: Do autor

3 CONSIDERAÇÕES FINAIS

Desde o começo deste trabalho tínhamos em mente que a veracidade e a precisão dos resultados eram de extrema importância. Pouco adiantaria um aplicativo ter sido extensivamente modelado ou implementar inúmeros recursos parcialmente funcionais, se os resultados apresentados estivessem longe da realidade. Por isso, optamos por primeiramente desenvolver um aplicativo que implementa os recursos mais básicos, com ênfase nos testes e na validação dos resultados obtidos. Nossa hipótese inicial era de que, devido à alta complexidade de um veículo automotor e das inúmeras variáveis que afetam seu funcionamento, dificilmente alcançaríamos uma precisão alta. Felizmente, não foi o caso.

Infelizmente, a indústria automobilística brasileira não aparenta dar sinais de que os computadores de bordo deixarão de ser equipamentos presentes quase que exclusivamente em veículos de alto valor. Em virtude disso, e considerando que o *smartphone* é um dispositivo cada vez mais acessível ao público em geral, esperamos que este trabalho possa contribuir na reeducação dos hábitos dos motoristas, auxiliando na redução do consumo de combustíveis fósseis.

Como trabalhos futuros, pretendemos implementar outros métodos de cálculo de consumo de combustível, como por exemplo o método onde a leitura do sensor MAF está disponível diretamente, sem necessidade de aproximação do seu valor através de outros dados. Também planejamos implementar um recurso em que, utilizando as coordenadas geográficas obtidas a partir do sistema de GPS, seja exibido um mapa destacando os locais onde houve maior consumo de combustível, auxiliando ainda mais na reeducação do motorista. Além disso, também pretendemos portar o aplicativo para as plataformas Android e Windows Phone.

ECONODRIVE: AUTOMOTIVE TRIP COMPUTER ON IOS PLATFORM

ABSTRACT

Saving fossil fuels is something of general interest, and the driver's habits are one of the major factors on fuel wasting. The automotive trip computer is a tool that might help on reeducating the drivers, but in Brazil this equipment is not found in most vehicles. This paper describes an implementation of a trip computer using an iOS device capable of fetching the necessary data from the vehicle's ECU using a specific device, and do the needed calculations in order to determine instant and average fuel consumption. The prototype developed as a proof of concept determined that it is indeed possible to fetch the data and calculate fuel consumption in real time, but the results were different from what we expected, which lead to improving the algorithm, in order to achieve greater precision on fuel consumption calculation. The final results obtained were deemed satisfactory.

Keywords: iOS. OBD-II. Fuel consumption. Automotive trip computer.

Referências

BROWN, Theodore L. et al. Chemistry: The Central Science. 12. ed. Boston: Pearson Prentice Hall, 2012.

CAMBRIDGE Dictionaries Online. Dongle noun – definition in the British English Dictionary & Thesaurus. Disponível em <http://dictionary.cambridge.org/dictionary/british/dongle?q=dongle>. Acesso em: 22 jun. 2014.

CHECKOWAY, Stephen et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. 2011. Disponível em: <http://www.autosec.org/pubs/cars-usenixsec2011.pdf>. Acesso em: 01 jun. 2014.

CODE OF FEDERAL REGULATIONS. Title 40, part 85, section 501. 2010. U.S. Government Printing Office.

ELM Electronics Inc. Datasheet ELM327: folha de dados do ELM327 (publicação) ELM327DSH, 2010.

ELM Electronics Inc. OBD Circuit Tips. 2014. Disponível em <http://elmelectronics.com/obdtips.html>. Acesso em: 22 jun. 2014.

HITZEMANN, Hunter et al. Designing a Clean, Quiet, Fuel Efficient High Performance Two- Stroke Flex Fuel Snowmobile. 2010. Disponível em: <http://www.uwplatt.edu/files/sae/CSC_Documents/05_uw-platteville_design_paper.pdf>. Acesso em: 17 nov. 2014.

LIGHTNER, Bruce D. Re: A humble inquiry on your brilliant work on OBD data fetching and computing. [mensagem pessoal]. Mensagem recebida por <contato@viniciusfontes.com>,,em 25 fev. 2014.

LIGHTNER, Bruce D.. AVR-Based Fuel Consumption Gauge. Circuit Cellar, East Hartford, CT, v. 1, n. 183, p.59-67, out. 2005.

LIGHTNER, Bruce D.. MAP- and MAF-Based Air/Fuel Flow Calculator. 2011. Disponível em: <http://www.lightner.net/obd2guru/IMAP_AFcalc.html>. Acesso em: 24 nov. 2014.

SOMMERVILLE, Ian. Software Engineering. 9. ed. Boston: Pearson Education, 2011.

WOLFRAM ALPHA. 1 mpg in km/l. Disponível em <http://www.wolframalpha.com/input/?i=1+mpg+in+km%2Fl>. Acesso em: 22 jun. 2014.